IEC 61970-552

Edition 2.0 2016-09

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Energy management system application program interface (EMS-API) – Part 552: CIMXML Model exchange format

Interface de programmation d'application pour système de gestion d'énergie (EMS-API) – Partie 552: Format d'échange de modèle CIMXML

IEC 61970-552:2016-09(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

IEC 61970-552

Edition 2.0 2016-09

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Energy management system application program interface (EMS-API) – Part 552: CIMXML Model exchange format

Interface de programmation d'application pour système de gestion d'énergie (EMS-API) – Partie 552: Format d'échange de modèle CIMXML

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 33.200

ISBN 978-2-8322-3637-6

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

ENERGY MANAGEMENT SYSTEM APPLICATION PROGRAM INTERFACE (EMS-API) –

Part 552: CIMXML Model exchange format

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61970-552 has been prepared by IEC technical committee 57, Power systems management and associated information exchange.

This second edition cancels and replaces the first edition published in 2013. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) New Clause 4 that defines the versioning of CIMXML format described in this document.
b) Subclause 5.1, the statement on work flow support is removed.
c) Subclause 5.2, Statement about mandatory header added. Rules how to use the header added. The discussion on management of multiple CIMXML documents and archives is removed.

d) Subclause 5.3, FullModelDocumentElement removed, minor version added to profile URI and the meaning of the header is elaborated in Table 2.

e) Subclause 6.2 the description of rdf:ID and rdf:about has been updated.

f) Subclause 6.3 introduce the new urn:uuid form and discuss the backwards compatibility.

g) New Subclause 6.4 added on support of older UUID formats.

h) New Subclause 6.5 discussing object types added.

i) Subclause 7.2.3.3, Position of header described and duplicate rows removed.

j) Document identification and references between documents updated in Table 2 and Subclauses 7.2.3.4 and 7.2.4.6.

k) Subclause 7.2.3.7, A compound element can never be a root element.

l) Subclause 7.2.3.9, description of compound containment added.

m) Subclauses 7.2.3.4 and 7.2.4.7.3, More clarification of cascading delete.

The text of this standard is based on the following documents:

FDIS	Report on voting
57/1752/FDIS	57/1773/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 61970 series, published under the general title *Energy management system application program interface (EMS-API)*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

This part of IEC 61970 is part of the series of standards that define an Application Program Interface (API) for an Energy Management System (EMS).

IEC 61970-301 specifies a Common Information Model (CIM): a logical view of the physical aspects of an electric utility operations. The CIM is described using the Unified Modelling Language (UML), a language used to specify, visualize, and document systems in an object-oriented manner. UML is an analysis and design language; it is not a programming language. In order for software programs to use the CIM, it must be transformed into a schema form that supports a programmable interface.

IEC 61970-501 describes the translation of the CIM in UML form into a machine readable format as expressed in the Extensible Markup Language (XML) representation of that schema using the Resource Description Framework (RDF) Schema specification language.

This part of IEC 61970 specifies how the CIM RDF schema specified in IEC 61970-501 is used to exchange power system models using XML (referred to as CIMXML) defined in the 61970-45x series of profile standards, such as the CIM Transmission Network Model Exchange Profile described in IEC 61970-452.

ENERGY MANAGEMENT SYSTEM APPLICATION PROGRAM INTERFACE (EMS-API) –

Part 552: CIMXML Model exchange format

1 Scope

This part of IEC 61970 specifies the format and rules for exchanging modelling information based upon the CIM. It uses the CIM RDF Schema presented in IEC 61970-501 as the meta-model framework for constructing XML documents of power system modelling information. The style of these documents is called CIMXML format.

Model exchange by file transfer serves many useful purposes. Profile documents such as IEC 61970-452 and other profiles in the 61970-45x series of standards explain the requirements and use cases that set the context for this work. Though the format can be used for general CIM-based information exchange, specific profiles (or subsets) of the CIM are identified in order to address particular exchange requirements. The initial requirement driving the solidification of this specification is the exchange of transmission network modelling information for power system security coordination.

This part of IEC 61970 supports a mechanism for software from independent suppliers to produce and consume CIM described modelling information based on a common format. The proposed solution:

- is both machine readable and human readable, although primarily intended for programmatic access,
- can be accessed using any tool that supports the Document Object Model (DOM) and other standard XML application program interfaces,
- is self-describing,
- takes advantage of current World Wide Web Consortium (W3C) recommendations.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 60050, *International Electrotechnical Vocabulary* (all parts)

IEC TS 61970-2, *Energy management system application program interface (EMS-API) – Part 2: Glossary*

IEC 61970-501:2006, *Energy management system application program interface (EMS-API) – Part 501: Common Information Model Resource Description Framework (CIM RDF) schema*

W3C, *RDF/XML Syntax Specification*

W3C, *XSL Transformations (XSLT)*

W3C, *Document Object Model (DOM)*

3 Terms and definitions

For the purposes of this document, the terms and definitions contained in IEC 60050 (for general glossary) and IEC TS 61970-2 (for EMS-API glossary definitions), as well as the following apply.

3.1
Application Program Interface
API
set of public functions provided by an executable application component for use by other executable application components

3.2
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.

3.3
CIMXML
serialisation format for exchange of XML data as defined in this document

3.4
Document Object Model
DOM
platform- and language-neutral interface defined by the World Wide Web Consortium (W3C) that allows programs and scripts to dynamically access and exchange the content, structure and style of documents

3.5
Document Type Definition
DTD
standard for describing the vocabulary and syntax associated with an XML document

Note 1 to entry: XML Schema and RDF are other forms that can be used.

3.6
Energy Management System
EMS
computer system comprising a software platform providing basic support services and a set of applications providing the functionality needed for the effective operation of electrical generation and transmission facilities so as to assure adequate security of energy supply at minimum cost

3.7
Hypertext Markup Language
HTML
mark-up language used to format and present information on the Web

3.8
Model
collection of data describing instances, objects or entities, real or computed. In the context of CIM the semantics of the data is defined by profiles, see: 4.9. Hence a model can contain equipment data, power flow initial values, power flow results etc.

Note 1 to entry: In power system analysis, a model is a set of static data describing the power system. Examples of Models include the Static Network Model, the Topology Solution, and the Network Solution produced by a power flow or state estimator application.

3.9
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.

3.10
Profile Document
collection of profiles intended to be used together for a particular business purpose

3.11
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.

3.12
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

3.13
Real-World Object
objects that belong to the real world problem domain as distinguished from interface objects and controller objects within the implementation

Note 1 to entry: The real-world objects for the EMS domain are defined as classes in IEC 61970-301 Common Information Model.

Note 2 to entry: Classes and objects model what is in a power system that needs to be represented in a common way to EMS applications. A class is a description of an object found in the real world, such as a PowerTransformer, GeneratingUnit, or Load that needs to be represented as part of the overall power system model in an EMS. Other types of objects include things such as schedules and measurements that EMS applications also need to process, analyze, and store. Such objects need a common representation to achieve the purposes of the EMS-API standard for plug-compatibility and interoperability. A particular object in a power system with a unique identity is modeled as an instance of the class to which it belongs.

3.14
Standard Generalized Markup Language
SGML
international standard for the definition of device-independent, system-independent methods of representing texts in electronic form

Note 1 to entry: HTML and XML are derived from SGML.

3.15
Unified Modelling Language
UML
object-oriented modelling language and methodology for specifying, visualizing, constructing, and documenting the artefacts of a system-intensive process

3.16
Uniform Resource Identifier
URI
Web standard syntax and semantic for identifying (referencing) resources (things, such as files, documents, images)

3.17
eXtensible Markup Language
XML
subset of Standard Generalized Markup Language (SGML), ISO 8879, for putting structured data in a text file

Note 1 to entry: This is an endorsed recommendation from the W3C. It is license-free, platform-independent and well-supported by many readily available software tools.

3.18
eXtensible Stylesheet Language
XSL
language for expressing style sheets for XML documents

4 CIMXML version

The CIMXML version is implemented as an XML processing instruction that appears before the CIMXML document, refer to Table 1.

Table 1 – CIMXML version

XML processing instruction	Version	Revision date
iec61970-552	2.0	2014-03-12

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<?iec61970-552 version="2.0"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:cim="http://iec.ch/TC57/2004/CIM-schema-cim10#"
      …
</rdf:RDF>
```

5 Model exchange

5.1 General

Model exchange typically involves the exchange of a collection of documents, each of which contains instance data, referred to as a model, and a header. The structure and semantics of each model as well as the header are described by a profile, which is not included in the exchanged data. The overall exchange is governed by a collection of profiles in a Profile Document.

A CIMXML document shall consists of a header and a model section.

A header section describes the content of the model section contained in a document e.g. the date the model was created, description etc. The header may also identify other models and their relationship to the present model. Such information is important when the models are part of a work flow where, for example, the models have relations to each other, e.g. a model succeeds and/or depends on another.

5.2 Rules for CIMXML documents and headers

A CIMXML document is described by a single header. Multiple headers in a CIMXML document are not allowed.

The header section shall always be the first element in a CIMXML document. The header section elements are

- FullModel element, refer to 7.2.3.4.
- DifferenceModel element, refer to 7.2.4.6.

The data in the model section is defined by one or more profiles listed within the header.

Elements in a CIMXML document may have references to elements (resources) in other CIMXML documents.

As a single header element is allowed in a CIMXML document the model section may only contain elements that the header can describe. If multiple headers are needed a CIMXML document shall be created for each header.

5.3 Model and header data description

A description of a model is attached as header data to the model. Figure 1 describes the model with header information.

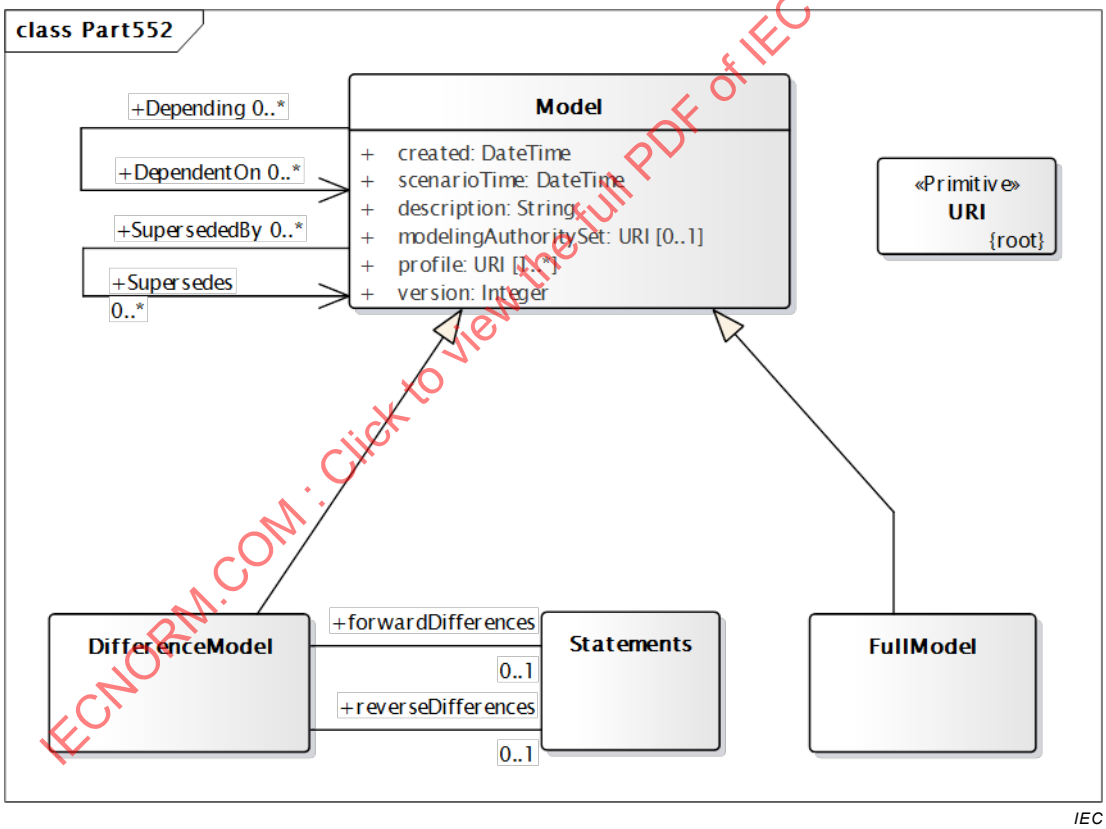

Figure 1 – Model with header

In Figure 1 the classes FullModel, DifferenceModel and Statements describe the model data while the header is described by the class Model. The following is a bottom up description of these classes:

- The Statements class represent a set of Definition (refer to 7.2.3.5) and/or Description (refer to 7.2.3.6) elements.
- The FullModel (refer to 7.2.3.4) class represent the full model header and its contents is described by the Model class.

- The DifferenceModel (refer to 7.2.4.6) class represents the difference model header. The content is described by the Model class, the association role forwardDifferences and association role reverseDifferences. Both association roles may have one set of Statements.
- The Model class describes the header content that is the same for the FullModel and the DifferenceModel. A Model is identified by an rdf:about attribute. The rdf:about attribute uniquely describe the model data and not the CIMXML document A new rdf:about identification is generated for created documents only when the model data has changed. A repeated creation of documents from unchanged model data should have the same rdf:about identification as previous document generated from the same model data.

The ordering of xml elements in the generated document is insignificant except for the FullModel element that always appear first in a document. The Model class attributes are described in Table 2.

Table 2 – Header attributes

Class	Attribute	Description
Model	created	The date when the document was created.
Model	scenarioTime	The date and time that the model represents, e.g. the current time for an operational model, a historical model or a future planned model.
Model	description	A description of the model, e.g. the name of person that created the model and for what purpose. The number of UTF-8 characters is limited to 2000.
Model	modelingAuthoritySet	A urn/uri referring to the organisation or role sourcing the model in the CIMXML document. Models from the same organisation or role but for different profiles shall have the same urn/uri.
Model	profile	A urn describing the Profiles that governs this model. It uniquely identifies the Profile and its version.
Model	version	A description of the version of the model sourcing the data in a CIMXML document. Examples are – Variations of the equipment model for the ModelingAuthoritySet – Different study cases resulting in different solutions. The version attribute is an integer that is changed in synchronisation with the rdf:about identifier, refer to description of the Model class preceding this table.
Model	DependentOn	References to other models that the model in this document depends on, e.g. – A load flow solution depends on the topology model it was computed from – A topology model computed by a topology processor depends on the network model it was computed from. The referenced models are identified by the FullModel rdf:about attribute (see 7.2.3.4) for full model documents and by DifferenceModel rdf:about attribute (see 7.2.4.6) for difference model documents. The references are maintained by the producer of the CIMXML document and the references are valid for the model with version and identifier (see description of Model class preceding this table) for which the document was created. The use of DependentOn by a consumer is optional. If a consumer of a document have also consumed the DependentOn documents it is expected that no dangling references are present in the currently consumed document.
Model	Depending	All documents depending on the model described by this document. This role is not intended to be included in any document exchanging instance data.

Class	Attribute	Description
Model	Supersedes	When a model is updated the resulting model supersedes the models that were used as basis for the update. Hence this is a reference to the models that are superseded by the model in this document. A model can supersede one or more models, for each superseded model one Supersedes reference is included in the header. The referenced models are identified by the FullModel rdf:about attribute (see 7.2.3.4) for full model documents and by DifferenceModel rdf:about attribute (see 7.2.4.6) for difference model documents. The references are maintained by the producer of the CIMXML document and are valid for the model with version and identifier (see description of Model class preceding this table) for which the document was created. The use of Supersedes by a consumer is optional. If a consumer of a document have also consumed the Supersedes documents it is expected that no dangling references are present in the currently consumed document.
Model	SupersededBy	All models superseding this model. This role is not intended to be included in any document exchanging instance data.

If a document is regenerated from a received and unchanged model all attributes in Table 2 shall be the same as in the originally received document. But if a received model is in any way modified none of the attributes may be different depending on the nature of the modifications.

DependentOn, Depending, Supersedes and SupersededBy describe the situation in the system where the document was generated at the time of generation. The use of the references in a receiving system is optional and a receiving system may or may not use the references depending on the use case, refer also to 5.4.

The profile attribute is a URI having the following format for IEC profiles:

- *http://iec.ch/<committee>/<year>/<standard>-<part>/<profile>/<version>/<minor_version>*

where text in *<italic>* is replaced by a describing text, e.g.

- http://iec.ch/TC57/2011/61970-452/Equipment/2/1

The profile URI shall be treated as indivisible where the full string conveys the identification of a profile. Hence software is not supposed to parse and interpret substrings of the profile URI, e.g. year, standard, part etc.

Other organizations using the CIMXML serialization may have other profile URIs.

The UML in Figure 1 translates into CIMXML elements as follows:

1) A leaf class in Figure 1 (DifferenceModel, Statements and FullModel) appears as class elements under the document element (7.2.3.3).
2) Statement elements appear as Definition (7.2.3.5) or Description elements (7.2.3.6).
3) Literal attributes, e.g. Model.created, appears as literal property elements (7.2.3.8).
4) Roles appear, e.g. Model.Supersedes, as resource property elements (7.2.3.11).
5) Inherited attributes and roles appear directly as elements under the leaf class following the rules 3, 4 and 5 above.
6) A CIMXML model document is identified by a Model rdf:about attribute (implicit in the UML). Hence the roles DependentOn and Supersedes are references to the Model rdf:about attribute.
7) A document may be regenerated multiple times from the same model. Documents regenerated from an unchanged model keep the identification (Model rdf:about) unchanged from a previous document generated from the same model.

8) A DifferenceModel or FullModel document generated from the same model will have the same identification and same Supersedes. Hence a DifferenceModel and a FullModel document can be used interchangeably. However, if a receiving system want a full model equivalent with the model in the FullModel document the DifferenceModel document must be applied to a full model corresponding to the superseded.

5.4 Work flow

A work flow is described by a sequence of exchange events. The model description in 5.3 supports work flow events related in time with the Model.Supersedes attribute and events related to profiles with the Model.DependentOn attribute. An example of this is shown in Figure 2.

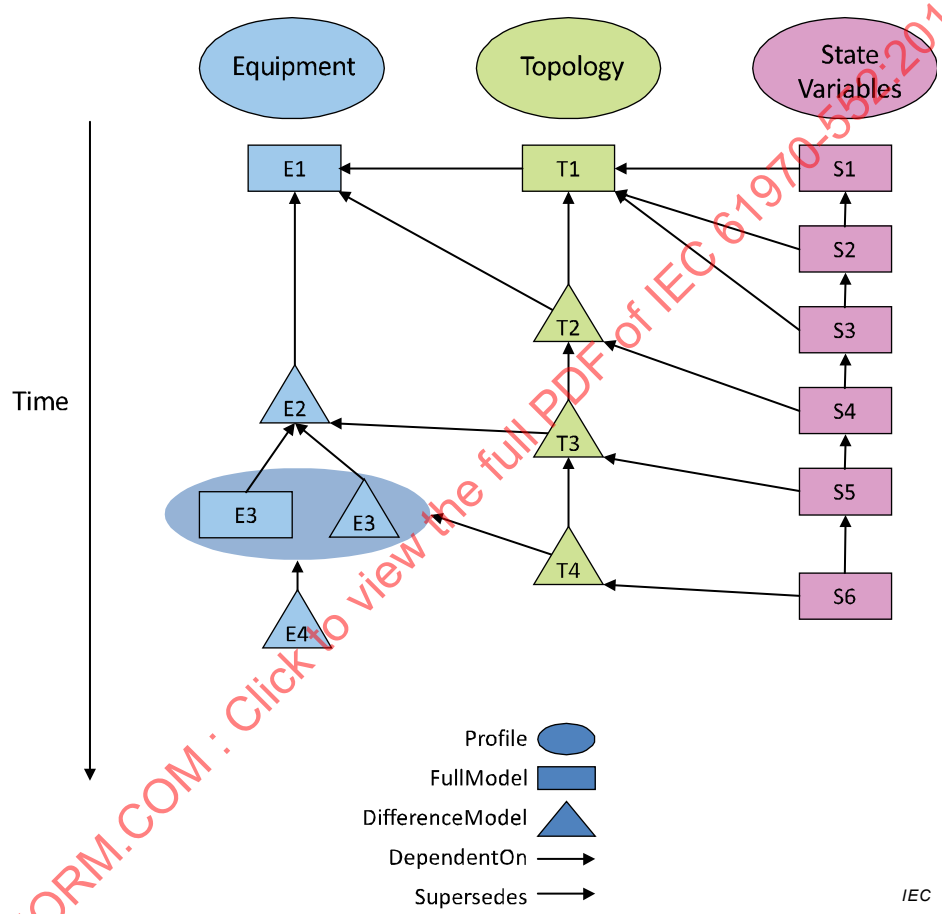

Figure 2 – Example work flow events

In this example, a solved network model is exchanged as a collection of models governed by a Profile Document comprising Equipment, Topology, and State Variables documents. The left time line in Figure 2 represents how the Equipment model document is exchanged over time. The center time line shows how new Topology results are exchanged over time and the Equipment models on which each depends. The right most time line shows how multiple State Variable documents are exchanged and the Topology documents on which they depend. Also note that the equipment model E3 is represented both by a full and a DifferenceModel document. The situation in Figure 2 represents a simple case. A more complex situation is shown in Figure 3.

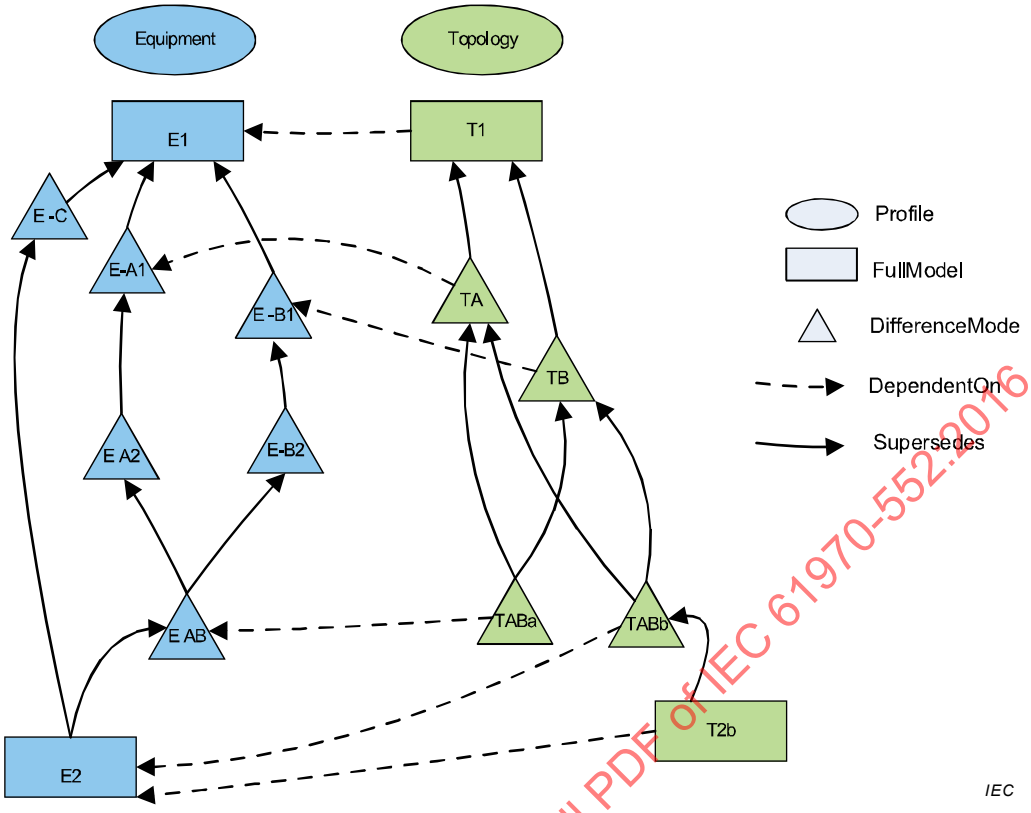

Figure 3 – Example work flow events with more dependencies

The CIMXML documents in Figure 3 may be created from a data modeller environment where multiple change tracks of a model appear in parallel, e.g. the equipment model has three tracks E-Ax, E-Bx and E-C that eventually merge into the full model E2 superseding the equipment model tracks.

A receiver of the CIMXML documents may use any of the topology documents TA, TB, TABa or T2b with the equipment model from E2. As the sender (the data modeller in this example) only verified T2b with E2 this is the only combination that is supposed to fit together. Concerning T2b the receiver may choose to apply TB and TABb to T1 instead of using T2b.

6 Object identification

6.1 URIs as identifiers

UUIDs (Universally Unique IDentifier), also known as GUIDs (Globally Unique IDentifier) can be used to identify resources in such a way that the

- identifiers can be independently and uniquely allocated by different authorities. This is a big advantage with the UUID;
- identifiers are stable over time and across documents.

If, in addition, the UUID is embedded in a Uniform Resource Name (URN) then the document can be simplified by the elimination of XML base namespace declarations (xml:base attributes). The URN is a concise, fixed-length, absolute URI.

The stability of identifiers over time also requires the following

- An identifier shall be created for any object when its existence become know. Note that objects in a model may or may not represent real physical equipment.
- An identifier for an object, e.g. a deleted object, is not allowed to be reused.

The standard for an URN containing a UUID is defined by the Internet Engineering Task Force RFC 4122,

RFC 4122 specifies the syntax of the URN and how the UUID portion following the last colon is allocated. The algorithm is aligned with, and technically compatible with, ISO/IEC 9834-8:2005 Information Technology, "Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components" ITU-T Rec. X.667, 2004.

CIMXML elements are identified by a URI. Also other forms than the URI is allowed but not recommended.

A URI can have two forms:

- URL
- URN

The URL and URN forms have fundamentally different structures, i.e.:

- URL form; *protocol*://*authority*/*path*?*query*#*fragment* where the *protocol* in CIMXML is http
- URN form: urn:*namespace*:*specification* where the *namespace* in CIMXML is uuid.

The URN *specification* format is summarized below

- 8 character hex number
- a dash “-“
- 4 character hex number
- a dash “-“
- 4 character hex number
- a dash “-“
- 4 character hex number
- a dash “-“
- 12 character hex number

where characters are lower case conforming to W3C (ISO 8859/1 8-bit single-byte coded graphic character set known as Latin Alphabet No. 1.

An example of the URN form is shown below

”urn:uuid:26cc8d71-3b7e-4cf8-8c93-8d9d557a4846”.

6.2 About rdf:ID and rdf:about

A CIMXML element can be identified by two different RDF constructs:

- rdf:ID
- rdf:about

In RDF the rdf:ID identification has the specific meaning that the identifier is unique within a document while the rdf:about identification means the identifier is unique within a name space. If the UUID name space urn:uuid is used for the rdf:about identification the identifiers are globally unique. Hence CIMXML promote using rdf:about identification in the UUID name space for all identifiers.

In the past an rdf:ID meant the introduction of a new resource or object while rdf:about meant a reference to an elsewhere introduced resource or object. This distinction is no longer used. As both are UUIDs they have the same meaning. For backwards compatibility both are allowed but the rdf:about form is the preferred.

Identifiers in the FullModel and DifferenceModel elements shall always use the rdf:about identification in the UUID name space, i.e. starting with “urn:uuid:”.

6.3 CIMXML element identification

Resource identification is so central in RDF that all elements representing objects are identified with a rdf:ID or rdf:about XML attribute. All classes in CIM that inherit IdentifiedObject have the UML object identification attribute IdentifiedObject.mRID. The attribute is implicitly mapped to the rdf:ID/rdf:about XML attribute.

A CIMXML document may only use the URN form (see 6.1) as further described below.

CIMXML files contain XML elements describing CIM objects (ACLineSegments, Substations etc.). The CIM has lots of association roles that show up as references in the XML elements (typically as rdf:resource or rdf:about attributes). CIM data is exchanged in different CIMXML documents that depend on each other as described in Clause 4. Some references then cross CIMXML document boundaries. A consequence of this is that the identification of a CIM object must be stable during its life time. Otherwise referencing objects across document boundaries will break.

A common practice in object oriented systems is to assume all objects have an identifier that is unique in space and time which means:

- Different objects are assigned different identifiers.
- Identifiers once assigned are never reused even if the original object having it is gone.

The URN form as described in 6.1 is used as CIMXML element identification with the following differences

- The prefix “urn:uuid:” is replaced by an underscore “_”. The underscore avoids a numeric starting character for the non-base part of the identifier. Starting the non-base part of the identifier with a numeric character is invalid RDF. The underscore is added in all cases to simplify parsers, even if the UUID starts with a non-numeric character.
- The prefix is defined as an xml:base=“urn:uuid:”

Some examples:

- rdf:ID=”_26cc8d71-3b7e-4cf8-8c93-8d9d557a4846” the rdf:ID” form.
- rdf:about=”#_26cc8d71-3b7e-4cf8-8c93-8d9d557a4846” the “hash” form.
- rdf:about=”urn:uuid:26cc8d71-3b7e-4cf8-8c93-8d9d557a4846” the “urn:uuid:” form.

A receiver compatible with Edition 2 is supposed to be able to process all three forms.

A receiver compatible with Edition 1 is supposed to process the rdf:ID and rdf:about forms.

A producer compatible with Edition 2 should preferably only use the urn:uuid: form but is allowed to continue produce the rdf:ID and rdf:about forms.

A producer compatible with Edition 1 will produce the the rdf:ID and rdf:about forms.

6.4 Older ID formats

Over the past different resource identifiers (IDs) have been used, e.g. IDs not complying with 6.3. Hence systems in operation may be using object identifiers that do not comply with the formatting described in 6.3.

Identifiers not complying with 6.3 is not allowed to consist of more than 60 UTF-8 characters.

An ID change means that IDs according to the old format will no longer match with the new format without translation or mapping. As an ID is not self-describing such a translation is difficult to do.

Due to the far reaching consequences of an ID format change existing systems are allowed to continue use the old format for objects.

When such a system is upgraded to support Edition 2 of this document, new objects created in a system should use the ID format as described in 6.1. As a result objects created before the current edition of this document are allowed to keep the old ID format and no translation to the new format is required. This means that the system will support both the old and the new formats.

6.5 Object type

6.5.1 General

Elements in CIMXML are typed by the use of the Definition element (refer to 7.2.3.6). When multiple documents are exchanged it may be the case that the type for CIMXML elements with the same ID may have different types. This specification do not describe how a type change may be communicated.

6.5.2 References to a more generic type than the actual

In an exchange involving multiple profiles data about an object with the same ID may appear in multiple documents. A profile may describe data at a lower level in the inheritance hierarchy than actual type of the object. An example is the PowerSystemResource that has many relations with other classes. PowerSystemResource (PSR) itself is rarely instantiated but rather its subclasses. In a case where data describing a CIM class is split over many profiles it is allowed for each profile to use the level in the inheritance hierarchy that best fits the exchange. The reason for this is to make the profiling simple and only concern about the data of interest. An example of this is exchange locations, see Figure 4.

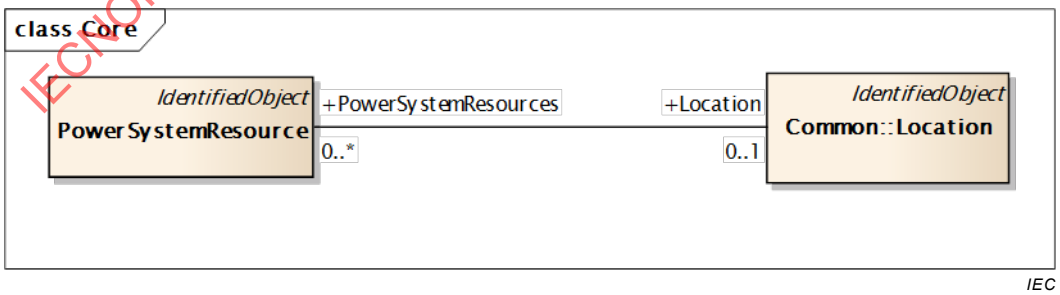

Figure 4 – CIM PSR – Location data model

The PSR class has a Location so a profile about exchange of locations will exchange the attribute PowerSystemResource.Location. But PSR is instantiated as a more specific class, e.g. Breaker, BusbarSystem etc. In a profile for exchange of location information it is impractical to enumerate all possible classes where an exchange of location is allowed also as the number of leaf classes will evolve over time. A stable profile for exchange of location is

just to use the PowerSystemResource class. The profile will then just include PowerSystemResource and a CIMXML element will look as follows:

```
<cim:PowerSystemResource rdf:about="_26cc8d71-12f1-4de9-9e68-125d95073a75" >
     <cim:PowerSystemResource.Location rdf:resource="#_26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
<cim:PowerSystemResource >
```

CIM object data are typically described by several classes in the inheritance hierarchy starting with base classes like cim:IdentifiedObject and ending specific class like cim:Breaker.

Associations may exist at any level in this hierarchy. An example is the class cim:PowerSystemResource (PSR) that has many associations with other classes, e.g. cim:Location.

For an association in a profile one of the sides in the association is chosen, normally the lowest cardinality side. But it is important to select the most natural direction, i.e. it is more natural to state that "a PSR has a Location" than the opposite "a Location may have one or more PSRs". So depending on the level in the inheritance hierarchy where an association is located a resource property element (refer to 7.2.3.11) may refer to an abstract class in the inheritance hierarchy as is the case with the "a PSR has a Location" reference. As specific classes appear in a CIMXML document this can be interpreted so that a profile will have to enumerate all the specific classes inheriting the association, e.g. cim:Breaker, cim:Disconnector etc. This will clutter a profile and make it difficult to maintain.

A resource property element (refer to 7.2.3.11) may appear in two different cases

- A document where the object is defined by a definition element (refer to 7.2.3.6). In this case all properties are listed related to the definition of the CIM object so all classes will be enumerated anyway, i.e. there is no issue to with excessive enumeration.
- A document where an elsewhere defined object is referred to by a description element (refer to paragraph 7.2.3.6). In this case the specific object class does not matter and the object is referred to by an rdf:about attribute.

So elements that relate to elsewhere defined objects shall use the description element instead of the definition element in cases where properties from less specific classes are included. For the "a PSR has a Location" case this means the instead of including all subclasses of PSR just PSR Location is included in the profile and it will show up as a description element instead of a definition element in the CIMXML document. Refer to example in Figure 4.

A profile may describe the relation by referring from the PSR to Location resulting in a CIMXML breaker element having a location:

```
<cim:Breaker rdf:about="_26cc8d71-12f1-4de9-9e68-125d95073a75">
     <cim:PowerSystemResource.Location rdf:resource="#_26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
</cim:Breaker >
```

To avoid including all types of equipment in a profile the reference can go the other way as in the below example:

```
<cim:Location rdf:about="_26cc8d71-3b7e-4cf8-8c93-8d9d557a4847" >
     <cim:Location.PowerSystemResource rdf:resource="#_26cc8d71-12f1-4de9-9e68-
125d95073a75"/>
</cim:Location>
```

To avoid using the same Location for many PSRs the cardinality Location.PowerSystemResource is lowered from 0..* to 1.

Using the solution proposed in this paragraph result in:

```
<cim:PowerSystemResource rdf:about="_26cc8d71-12f1-4de9-9e68-125d95073a75" >
     <cim:PowerSystemResource.Location rdf:resource="#_26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
<cim:PowerSystemResource >
```

7 CIMXML format rules and conventions

7.1 General

Given the CIM RDF Schema described in IEC 61970-501, a power system model can be converted for export as an XML document (see Figure 5). This document is referred to as a CIMXML document. All of the tags (resource descriptions) used in the CIMXML document are supplied by the CIM RDF schema. The resulting CIMXML model exchange document can be parsed and the information imported into a foreign system.

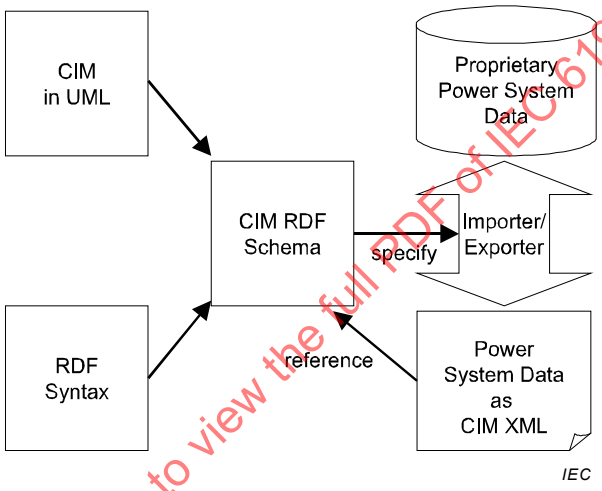

Figure 5 – CIMXML-based power system model exchange mechanism

7.2 Simplified RDF syntax

7.2.1 General

RDF syntax provides many ways to represent the same set of data. For example, an association between two resources can be written with a resource attribute or by nesting one element within another. This could make it difficult to use some XML tools, such as XSL processors, with the CIMXML document.

Therefore, only a subset of the RDF Syntax is to be applied in creating CIMXML documents. This syntax simplifies the work of implementers to construct model serialization and de-serialization software, as well as to improve the effectiveness of general XML tools when used with CIMXML documents. The reduced syntax is a proper subset of the standard RDF syntax; thus, it can be read by available RDF de-serialization software.

Subclauses in 7.2 define a subset of the RDF Syntax. This simplified syntax is for exchanging power system models between utilities. The aim of the specification is to make it easier for implementers to construct de-serialization software for RDF data, to simplify their choices when serializing RDF data, and to improve the effectiveness of general XML tools such as XSLT processors when used with the serialized RDF data.

The reduced syntax is a proper subset of the standard RDF syntax. Thus, it can be read by RDF de-serialization software such as SirPAC [8][1]. In this, it differs from other proposals for a simplified syntax, such as [9], [10].

The reduced syntax does not sacrifice any of the power of the RDF data model. That is, any RDF data can be exchanged using this syntax. Moreover, features of RDF such as the ability to extend a model defined in one document with statements in second document are preserved.

7.2.2 Notation

The simplified syntax is defined in 7.2.3. Each kind of element is defined in a subclause beginning with a model of the element, followed by some defining text, and a reference to the RDF grammar. The semantics of the element are not detailed (refer to the RDF recommendation "W3C: RDF/XML Syntax Specification" for that information). The notation for the element model is as follows:

1) A symbol in *italics* in the position of an element type, attribute name or attribute value indicates the type of name or value required. The symbol will be defined in the text.
2) The symbol *rdf* stands for whatever namespace prefix is chosen by the implementation for the RDF namespace. Similarly the symbol *cim* stands for the chosen CIM namespace prefix.
3) A comment (<*!-- comment text* -->) within the element model indicates the allowed content.
   - A symbol in *italics* stands for a kind of element or other content defined in the text.
   - A construction (a | b) indicates that a and b are alternatives.
   - A construction a* indicates zero or more repetitions of a.
4) All other text in the model is literal.
5) The RDF grammar is described in 6.1, "W3C: RDF/XML Syntax Specification".

7.2.3 Syntax definition (normative)

7.2.3.1 General

The syntax definition is enriched with examples. The examples should help to get a better understanding of the formal syntax definition. The same example is used for several syntax definitions.

UTF-8 is the standard for file encoding. UTF-16 is not supported.

7.2.3.2 Name space URIs defined in this specification

The following name spaces are defined in this specification:

- *cim-model-description_uri* described by xmlns:md
- *difference-model-namespace-uri* described by xmlns:dm

Their values are defined as:

- `xmlns:md="http://iec.ch/TC57/61970-552/ModelDescription/1#"`
- `xmlns:dm="http://iec.ch/TC57/61970-552/DifferenceModel/1#"`

[1] Numbers in square brackets refer to the bibliography.

7.2.3.3 Document element

```
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
     xmlns:cim="cim-namespace-uri"
     xmlns:md="cim-model-description_uri"
     xml:base="urn:uuid:">
     <!-- Content: full-model (definition|description)* -->
</rdf:RDF>
```

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:cim="http://iec.ch/TC57/2004/CIM-schema-cim10#"
     xmlns:md="http://iec.ch/TC57/61970-552/ModelDescription/1#"
     xml:base="urn:uuid:">
     <md:FullModel rdf:about="urn:uuid:26cc8d71-3b7e-4cf8-8c93-8d9d557a4846">
          <md:Model.created>2008-12-24T03:19:13</md:Model.created>
          <md:Model.Supersedes rdf:resource="urn:uuid:26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
          <md:Model.DependentOn rdf:resource="urn:uuid:26cc8d71-3b7e-4cf8-8c93-
8d9d557a4848"/>
          <md:Model.version>32</md:Model.version>
     <md:Model.modelingAuthoritySet>http://polarenergy.com/2008/NorthPoleTSO</md:Mo
del.modelingAuthoritySet>
          <md:Model.description>Santa Claus made a study case peak load summer base
topology solution</md:Model.description>
          <md:Model.profile>http://iec.ch/TC57/61970-
452/Equipment/1</md:Model.profile>
     <md:Model.profile>http://iec.ch/TC57/61970-
452/EquipmentShortCircuit/1</md:Model.profile>
     </md:FullModel>
     …
</rdf:RDF>
```


1) The element type is rdf:RDF.
2) The RDF namespace must be declared as http://www.w3.org/1999/02/22-rdf-syntax-ns#.
3) The CIM namespace must be declared. With newer versions of the CIM schema the version needs to be adjusted in the CIM name space. Parties exchanging documents have to agree on the used version.
4) Other namespaces may be declared.
5) The full model header element appears before the full model definition/description elements.

7.2.3.4 Full-model element

```
<md:FullModel rdf:about=model-uri>
     <!--     Content:     (literal-property|resource-property|compound-property)* -->
</md:FullModel >
```

```
<md:FullModel rdf:about="="urn:uuid:26cc8d71-…">
   <md:Model.created>2008-12-24T03:19:13</md:Model.created>
      <md:Model.Supersedes rdf:resource="urn:uuid:26cc8d71-a002-4c2b-bcf4-7bc97430bf87"/>
      <md:Model.DependentOn rdf:resource=urn:uuid:26cc8d71-a002-4c2b-bcf4-7bc97430bf88"/>
      <md:Model.version>32</md:Model.version>
   <md:Model.modelingAuthoritySet>http://polarenergy.com/2008/NorthPoleTSO</md:Model.modelingAuthoritySet>
      <md:Model.description>Santa Claus made a study case peak load summer base topology solution</md:Model.description>
      <md:Model.profile>http://iec.ch/TC57/61970-456/StateVariables/1</md:Model.profile>
</md:FullModel>
```

1) The full model element introduces a new model.
2) The value of the about attribute, model-uri, is a name chosen by the implementation. The model-uri uniquely identifies a document and is the name referenced by other documents, e.g. by Supersedes or DependentOn, as indicated in Figure 2.
3) The FullModel rdf:about attribute identifies a CIMXML document.

7.2.3.5 Definition element

```
<classname rdf:ID=identity>
   <!-- Content:
   (literal-property|resource-property|compound-property)*
   -->
</classname>


<classname rdf:about=resource-uri>
   <!-- Content:
   (literal-property|resource-property|compound-property)*
   -->
</classname>
```

Example:

```
<cim:SynchronousMachine rdf:about="#_31dcf429-6bfb-4e2e-b2996-42491b3abc1">
      <cim:IdentifiedObject.name>IN-2</cim:IdentifiedObject.name>
      <cim:SynchronousMachine.minimumMVAr>-9999</cim:SynchronousMachine.minimumMVAr>
      <cim:SynchronousMachine.operatingMode rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#SynchronousMachineOperatingMode.generator"/>
      <cim:RegulatingCondEq.RegulationSchedule rdf:resource="#_ca32746f-a002-4c2b-bcf4-7bc97430bf87"/>
      <cim:Equipment.EquipmentContainer rdf:resource="#_6cb8701a-12f1-4de9-9e68-125d95073a75"/>
</cim:SynchronousMachine>
```

1) The definition element introduces a new resource and gives its type. There are two forms: the first as an rdf:ID attribute and the second as an rdf:about attribute.

2) The element type, *classname*, is the XML qualified name of a class from the CIM schema or other schema declared as a namespace in the document element.

3) The value of the id attribute, *identity*, is chosen by the implementation. It must be unique in the document. (It is not necessarily related to the power system resource name.)

4) A Definition element with the same identification may appear in different documents. The meaning of this is that the properties in the different documents describe the same object and that the description is split over multiple documents. This is the case when data for a class is divided in multiple profiles and each profile is exchanged as its own CIMXML document.

7.2.3.6 Description element

```
<rdf:Description rdf:about=resource-uri>
   <!-- Content:
   (literal-property | resource-property | compound-property)*
   -->
</rdf:Description >
```

Example:

```
<rdf:Description rdf:about="#_26cc8d71-a002-4c2b-bcf4-7bc97430bf87">
     <cim:IdentifiedObject.name>TROY</cim:IdentifiedObject.name>
</rdf:Description>
```

1) The description element adds information about a resource introduced elsewhere in this or another document.

2) The resource-uri is a URN-reference that identifies the subject resource.

3) The Description element is used only in difference models (refer to 7.2.4). It is never used in full models.

4) A Description element with the same identification may appear in different documents. The meaning of this is that the properties in the different documents describe the same object and that the description is split over multiple documents. This is the case when data for a class is divided in multiple profiles and each profile is exchanged as its own CIMXML document.

7.2.3.7 Compound element

```
<classname>
   <!-- Content:
   (literal-property | resource-property | compound-property)*
   -->
</classname>
```

Example:

```
<cim:DateTimeInterval>
     <cim:DateTimeInterval.start>2013-02-28</cim:DateTimeInterval.start>
     <cim:DateTimeInterval.end>2013-02-29</cim:DateTimeInterval.end>
</cim:DateTimeInterval>
```

1) The compound element introduces a structured value. The value does not represent a resource nor have any *identity*. It can only appear as the object of a property.

2) The element type, *classname*, is the XML qualified name of a compound class.
3) A compound element is treated as an indivisible unit. Hence a compound element is not supposed to be split in multiple elements having different sets of members. Refer also to 7.2.4.7.4.
4) A compound element is always part of another element and cannot be a root element.

7.2.3.8 Literal-Property element

```
<propname>
<!-- Content: text -->
</propname>
```

Example:

```
<cim:SynchronousMachine rdf:about="#_31dcf429-6Bfb-4e2e-b2996-42491b3abc1">
     <cim:IdentifiedObject.name>IN-2</cim:IdentifiedObject.name>
     <cim:SynchronousMachine.minimumMVAr>-9999</cim:SynchronousMachine.minimumMVAr>
     <cim:SynchronousMachine.operatingMode rdf:resource="http://iec.ch/TC57/2001/CIM-
schema-cim10#SynchronousMachineOperatingMode.generator"/>
     <cim:RegulatingCondEq.RegulationSchedule rdf:resource="#_ca32746f-a002-4c2b-
bcf4-7bc97430bf87"/>
     <cim:Equipment.EquipmentContainer rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a75"/>
</cim:SynchronousMachine>
```

1) The literal-property element introduces a property and a literal value applying to the enclosing resource.
2) The element type, *propname*, is the XML qualified name of a property from the CIM schema or other schema declared as a namespace in the document element.
3) The content *text* is any XML text with <, >, and & escaped representing the value of the property.
4) Floating point numbers may slightly change due to rounding effects when imported and re-exported again. This is allowed and need to be managed by applications, e.g. by use of a dead band in case the values are compared.
5) A Literal-Property element with the same propname may appear multiple times if the cardinality of the corresponding schema attribute is larger than 1.

7.2.3.9 Compound-Property element

```
<propname>
     <!-- Content: (compound-element) -->
</propname>
```

Example:

```
<cim:TimeSchedule>
     <cim:TimeSchedule.scheduleInterval> <!-- the compund property -->
          <cim:DateTimeInterval> <!-- another compund element -->
               <cim:DateTimeInterval.start>2013-02-28</cim:DateTimeInterval.start>
               <cim:DateTimeInterval.end>2013-02-29</cim:DateTimeInterval.end>
          </cim:DateTimeInterval>
     </cim:TimeSchedule.scheduleInterval>
</cim:TimeSchedule>
```

1) The compound property element contains a compound element.

2) As a compound element may contain compound properties an indefinitely deep hierarchy is possible.

3) A Compound-Property element with the same propname may appear multiple times if the cardinality of the corresponding schema attribute is larger than 1.

7.2.3.10 Resource-Property element

```
<propname rdf:resource=resource-uri/>
```

1) The resource-property element introduces a property and a resource as its value applying to the enclosing resource.

2) The element type, *propname*, is the XML qualified name of a property from the CIM schema or other schema declared as a namespace in the document element.

3) The *resource-uri* is an URN-reference that identifies a resource.

4) For relations with roles having cardinality greater than one the resource-property element shall be repeated as many times as there are references

5) An association in a schema is bidirectional and has two roles (ends). Including Resource-Property elements for both roles is allowed but preferably elements for one side only is included. Then the side with the lowest cardinality is preferred but not required.

6) A Resource-Property element with the same propname may appear multiple times if the cardinality of the corresponding schema role is larger than 1.

Example 1 – URN-Reference:

The example contains two references one for a RegulationSchedule and the other to the parent represented as EquipmentContainer.

```
<cim:SynchronousMachine rdf:about="#_31dcf429-6bfb-4e2e-b299-642491b3abc1">
     <cim:IdentifiedObject.name>IN-2</cim:IdentifiedObject.name>
     <cim:SynchronousMachine.minimumMVAr>-9999</cim:SynchronousMachine.minimumMVAr>
     <cim:SynchronousMachine.operatingMode rdf:resource="http://iec.ch/TC57/2001/CIM-
schema-cim10#SynchronousMachineOperatingMode.generator"/>
     <cim:RegulatingCondEq.RegulationSchedule rdf:resource="#_cd32746f-a002-4c2b-
bcf4-7bc97430bf87"/>
     <cim:Equipment.EquipmentContainer rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a75"/>
</cim:SynchronousMachine>
```

Example 2 – Enumeration:

The example defines the attribute value of SynchonousMachine.operatingMode as “generator”. The operatingMode is specified in the CIM schema as the enumeration SynchronousMachineOperatingMode.

```
<cim:SynchronousMachine rdf:about="#_31dcf429-6bfb-4e2e-b2996-42491b3abc1" >
     <cim:IdentifiedObject.name>IN-2</cim:IdentifiedObject.name>
     <cim:SynchronousMachine.minimumMVAr>-9999</cim:SynchronousMachine.minimumMVAr>
     <cim:SynchronousMachine.operatingMode rdf:resource="http://iec.ch/TC57/2001/CIM-
schema-cim10#SynchronousMachineOperatingMode.generator"/>
     <cim:RegulatingCondEq.RegulationSchedule rdf:resource="#_cd32746f-a002-4c2b-
bcf4-7bc97430bf87"/>
     <cim:Equipment.EquipmentContainer rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a75"/>
</cim:SynchronousMachine>
```

The example defines the attribute value of SynchonousMachine.operatingMode as “generator”. The operatingMode is specified in the CIM schema as the enumeration SynchronousMachineOperatingMode.

Example 3 – Role with cardinality greater than one:

```
<cim:SynchronousMachine rdf:about="#_31dcf429-6bfb-4e2e-b299-642491b3abc1">
      <cim:IdentifiedObject.name>IN-2</cim:IdentifiedObject.name>
      <cim:SynchronousMachine.minimumMVAr>-9999</cim:SynchronousMachine.minimumMVAr>
      <cim:SynchronousMachine.operatingMode rdf:resource="http://iec.ch/TC57/2001/CIM-
schema-cim10#SynchronousMachineOperatingMode.generator"/>
      <cim:RegulatingCondEq.RegulationSchedule rdf:resource="#_cd32746f-a002-4c2b-
bcf4-7bc97430bf87"/>
      <cim:Equipment.EquipmentContainer rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a75"/>
      <cim:Equipment.ReactiveCapabilityCurves rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a76"/>
      <cim:Equipment.ReactiveCapabilityCurves rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a77"/>
      <cim:Equipment.ReactiveCapabilityCurves rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a78"/>
</cim:SynchronousMachine>
```

7.2.4 Syntax extension for difference model

7.2.4.1 General

The general syntax definition in the first part of this clause is used for partial and full model data exchange. Once the initial complete set of model data is exchanged, only updates are required to maintain the model as changes occur. In general, those changes can be specified as a set of differences between two models. The difference document is itself an RDF model (a collection of RDF statements) and therefore can be processed by an RDF infrastructure.

7.2.4.2 Example use case

To illustrate the DifferenceModel document approach to handling difference model updates, an example use case is provided. In this example, the participants are Regional Energy Co. and Network Power Co.:

- Each participant has a copy of a power system model, B1.
- Regional Energy Co. updates B1, to reflect forthcoming power system modifications, producing B2.
- Regional Energy Co. sends the differences between B1 and B2 to Network Power Co. as a difference model.
- Network Power Co. reviews and validates the difference model.
- Network Power Co. merges the difference model with its copy of model B1, to produce B2.

An alternative would have been for Regional Energy Co. to simply send Network Power Co. a copy of B2. However, B2 is a very large model and it is not feasible to validate it in any reasonable period of time. Validation is not entirely automated, but involves analysis by experts. Indeed, the best validation strategy for B2 may be to compare it to the previously validated B1. This brings us back to the need for a difference model.

A more complicated use case would involve more than two participants. Several peers of Regional Energy Co. would contribute difference models to Network Power Co. This use case would introduce issues of parallel model changes and concurrency conflict.

7.2.4.3 Requirements

Given two RDF models, B1 and B2, called base models, the requirement is for a difference model that:

- Represents the differences between the two base models.
- Is itself an RDF model (a collection of RDF statements) and therefore can be processed by RDF infrastructure.

- Efficiently represents a small difference between two large base models.
- When an object is deleted, the system applying the differences will not perform any the "cascading deletions", i.e. finding and deleting all other contained objects. Instead it is the responsibility of the system sourcing a deletion to include any cascading deletions.
- Remove operations are not reversible (at least, not from the information in the difference model).
- May contain information about itself such as authorship, purpose and date.
- May contain information to protect against conflicts arising when two difference models are created concurrently from the same base model.

The requirement to treat each difference document as a database commit operation is outside the scope of this service (i.e., a roll back functionality, if desired, is the responsibility of the receiving application, not the sending application). This is in recognition of the fact that the sending application may not be aware of changes made in the B2 model documents by other agents since the last update to B1.

7.2.4.4 Structure of difference document

Given two base RDF models, B1 and B2, the difference model is made up of four groups of statements, each encoded as a sequence of resource description structures:

- Forward difference statements, comprising statements found in B2, but not in B1.
- Reverse difference statements, comprising statements found in B1, but not in B2.
- Precondition statements, comprising statements found in both B1 and B2 and considered to be dependencies of the difference model in an application defined sense.

Any or all of the three groups can be empty.

The difference model itself is represented by a compound element of type dm:DifferenceModel.

The following properties apply to the difference model resource:

- dm:forwardDifferences is a property of the difference model whose value is a collection of statements (i.e., resources of type rdf:Statement) representing the forward difference statements.
- dm:reverseDifferences is a property of the difference model whose value is the collection of reverse difference statements.
- dm:preconditions is a property of the difference model whose value is the collection of precondition statements.

Header properties also apply to the difference model resource. These may indicate authorship, date and purpose. These properties can be drawn from the Dublin Core vocabulary or any other convenient schema.

The namespace for the difference model vocabulary, represented by the prefix dm: in the foregoing, is: http://iec.ch/TC57/61970-552/DifferenceModel/1#.

7.2.4.5 Preconditions and concurrency

The precondition statements are a subset of both B1 and B2 and carry no difference information. In simple, sequential model revision scenarios they can be omitted.

For a large shared model, sequential revision is not always feasible. Revisions are likely to be constructed concurrently by different participants, without reference to each other. Concurrency issues must be handled, but the conventional database-oriented approach of using locks to detect incompatible concurrent transactions is not feasible on a web-scale.

The precondition statements are an alternative to locks. Informally, they represent the information that would have been read-locked in an equivalent database transaction. Software agents that process difference models can check that the preconditions hold and, if not, warn of incompatible model revisions.

The choice of statements to include as preconditions is application-specific (as is the choice of which information to lock in a database transaction). Preconditions should include statements that would affect decisions of the agent that produced the model revision.

7.2.4.6 Difference model template

The following is a template for the conventional syntax of a difference model.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:cim="cim-namespace-uri"
     xmlns:md="cim-model-description_uri"
     xmlns:dm="difference-model-namespace-uri"
     xml:base="urn:uuid:">
     <dm:DifferenceModel rdf:about=model-uri>
          <!--Content:(literal-property|resource-property|compound-property)*
          -->
          <dm:preconditions parseType="Statements">
               <!-- Content: (definition|description)* -->
          </dm:preconditions>
          <dm:forwardDifferences parseType="Statements">
               <!-- Content: (definition|description)* -->
          </dm:forwardDifferences>
          <dm:reverseDifferences parseType="Statements">
               <!-- Content: (definition|description)* -->
          </dm:reverseDifferences>
     </dm:DifferenceModel>
</rdf:RDF>
```

1) Simply for clarification with the namespace “dm” new statements are introduced that are valid extensions to the standard RDF syntax through the new property rdf:parseType, which is called Statements.
2) The content model of an element with rdf:parseType=”Statements” is the same as the content model of the rdf:RDF element.
3) The content generates the same RDF statements as if it appeared in an rdf:RDF element.
4) The DifferenceModel rdf:about attribute identifies a CIMXML document.

7.2.4.7 Difference model usage

7.2.4.7.1 General

The following cases explain the usage of the difference model.

7.2.4.7.2 Add resource

The difference model contains for a given resource only a forward Difference statement if the particular is resource is added.

EXAMPLE:

The following example adds two new ACLineSegments each with its adjacent Terminals. The Terminals are linked to new ConnectivityNodes. Those ConnectivityNodes are assigned to a new VoltageLevel in an existing Substation.

```
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:cim="cim-namespace-uri"
      xmlns:md="cim-model-description_uri "
      xmlns:dm="difference-model-namespace-uri"
      xml:base="urn:uuid:">
      <dm:DifferenceModel rdf:about="#_26cc8d71-3b7e-4cf8-8c93-8d9d557a4846">
            <md:Model.created>2008-12-24</md:Model.created>
            <md:Model.Supersedes rdf:resource="#_26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
            <md:Model.DependentOn rdf:resource="#_26cc8d71-3b7e-4cf8-8c93-
8d9d557a4848"/>
            <md:Model.version>V32</md:Model.version>
      <md:Model.modelingAuthoritySet>http://polarenergy.com/2008/NorthPoleTSO</md:Mo
del.modelingAuthoritySet>
            <md:Model.description>Santa Claus made a study case peak load summer base
topology solution</md:Model.description>
            <md:Model.profile>http://iec.ch/TC57/61970-
452/EquipmentModel/1</md:Model.profile>
            <md:Model.version>179</md:Model.version>
            <dm:forwardDifferences rdf:parseType="Statements">
                  <!-- Add ACLineSegment ACLine_New1 -->
                  <cim:ACLineSegment rdf:about="#_26cc8d71-12f1-4de9-9e68-
125d95073a75">
                        <cim:IdentifiedObject.name>New 1</cim:IdentifiedObject.name>
                        <cim:Conductor.r>0.0646</cim:Conductor.r>
                        <cim:Conductor.x>0.5961</cim:Conductor.x>
                        <cim:Conductor.bch>0.4066</cim:Conductor.bch>
                  </cim:ACLineSegment>
                  <cim:Terminal rdf:about="#_26cc8d71-… ">
                        <cim:IdentifiedObject.name>T1</cim:IdentifiedObject.name>
                        <cim:Terminal.ConnectivityNode rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                        <cim:Terminal.ConductingEquipment rdf:resource="#_26cc8d71-…"/>
                  </cim:Terminal>
                  <cim:Terminal rdf:about="#_26cc8d71-12f1-4de9-9e68-125d95073a756>
                        <cim:IdentifiedObject.name>T2</cim:IdentifiedObject.name>
                        <cim:Terminal.ConnectivityNode rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                        <cim:Terminal.ConductingEquipment rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                  </cim:Terminal>
                  <!-- Add ACLineSegment ACLine_New2 -->
                  <cim:ACLineSegment rdf:about="#_26cc8d71-12f1-4de9-9e68-
125d95073a75">
                        <cim:IdentifiedObject.name>New 2</cim:IdentifiedObject.name>
                        <cim:Conductor.r>0.0646</cim:Conductor.r>
                        <cim:Conductor.x>0.5961</cim:Conductor.x>
                        <cim:Conductor.bch>0.4066</cim:Conductor.bch>
                  </cim:ACLineSegment>
                  <cim:Terminal rdf:about="#_26cc8d71-…">
                        <cim:IdentifiedObject.name>T1</cim:IdentifiedObject.name>
                        <cim:Terminal.ConnectivityNode rdf:resource="#_26cc8d71-…"/>
                        <cim:Terminal.ConductingEquipment rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                  </cim:Terminal>
                  <cim:ConnectivityNode rdf:about="#_26cc8d71-12f1-4de9-9e68-
125d95073a75">
                        <cim:IdentifiedObject.name>ND   New1</cim:IdentifiedObject.name>
                        <cim:ConnectivityNode.EquipmentContainer
rdf:resource="#_26cc8d71-12f1-4de9-9e68-125d95073a75"/>
                        <cim:ConnectivityNode.Terminals rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                  </cim:ConnectivityNode>
                  <cim:Terminal rdf:about="#_26cc8d71-12f1-4de9-9e68-125d95073a75">
```

```
                    <cim:IdentifiedObject.name>T2</cim:IdentifiedObject.name>
                    <cim:Terminal.ConnectivityNode rdf:resource="#_26cc8d71-…"/>
                    <cim:Terminal.ConductingEquipment rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                </cim:Terminal>
                <cim:ConnectivityNode rdf:about="#_26cc8d71-12f1-4de9-9e68-
125d95073a75">
                    <cim:IdentifiedObject.name>ND   New2</cim:IdentifiedObject.name>
                    <cim:ConnectivityNode.EquipmentContainer
rdf:resource="#_26cc8d71-12f1-4de9-9e68-125d95073a75"/>
                    <cim:ConnectivityNode.Terminals rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                </cim:ConnectivityNode>
                <cim:VoltageLevel rdf:about="#_26cc8d71-12f1-4de9-9e68-
125d95073a75">
                    <cim:IdentifiedObject.name>230K</cim:IdentifiedObject.name>
                    <cim:VoltageLevel.Substation rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                    <cim:VoltageLevel.BaseVoltage rdf:resource="#_26cc8d71-12f1-
4de9-9e68-125d95073a75"/>
                </cim:VoltageLevel>
            </dm:forwardDifferences>
        </dm:DifferenceModel>
</rdf:RDF>
```

7.2.4.7.3 Delete resource

The difference model contains for a given resource only a reverseDifference statement if the particular resource is deleted.

Cascading deletes are deletes where an object and its child objects (if any) are deleted. In a cascading delete it would be possible to just include the root or parent object in a CIMXML document. The receiver then has to figure out what child objects to delete. To make clear what objects are included in a cascading delete the creator of the CIMXML document shall include all objects as elements in the cascade. Including only the root or parent object is not allowed.

The EquipmentContainer-Equipment relation is a parent-child relation where deletion of an EquipmentContainer shall also result in a deletion of its child Equipment. Other examples of such parent child relations are

- EquipmentContainers also has a parent child relation, e.g. Station-VoltageLevel.
- PowerTransformer and it's TransformerEnds
- ConductingEquipment and its Terminals

The CIM does not currently specify the containment relations. As this information is missing it is up to an implementer to decide which relation is regarded a containment relation. This spoils interoperability. This is the reason to include all objects in a cascaded delete to indicate the sending systems interpretation of containment.

Delete elements shall have all its property elements included. Reason is that this enables reversing the delete operation and re-creates the object.

EXAMPLE:

The example below contains the deletion of a PowerTransformer with all resources that are hierarchically subordinated.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cim="cim-namespace-uri"
xmlns:dm="difference-model-namespace-uri"
xml:base="urn:uuid:">
     <dm:DifferenceModel rdf:about="#_26cc8d71-12f1-4de9-9e68-125d95073a75">
          <!-- Delete Transformer -->
          <dm:reverseDifferences rdf:parseType="Statements" >
               <cim:PowerTransformer rdf:about="#_41bb4445-6756-43fa-9e5a-
48B6cd71790e">
                    …all properties of the transformer follows here…
               </cim:PowerTransformer>
               …all parts of the transformer follows here….
          </dm:reverseDifferences>
     </dm:DifferenceModel>
</rdf:RDF>
```

7.2.4.7.4 Update resource

The difference model contains for a given resource forwardDifference and reverseDifference statements if the resource is changed.

EXAMPLE:

The example below defines the move of the EnergyConsumer from 115k to 230k through the changed link from its Terminal to a different ConnectivityNode.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:cim="cim-namespace-uri"
     xmlns:dm="difference-model-namespace-uri"
     xml:base="urn:uuid:">
     <dm:DifferenceModel rdf:about="#_26cc8d71-12f1-4de9-9e68-125d95073a75">
          <!-- Move EnergyConsumer load from 115K to 230K -->
          <dm:forwardDifferences rdf:parseType="Statements" >
               <rdf:Description rdf:about="#_39e4e305-1c70-4dcc-a423-45e4812dcd07">
                    <cim:Terminal.ConnectivityNode rdf:resource="#_612fa147-902c-
4f88-be3f-0302b3750b18"/>
               </rdf:Description>
          </dm:forwardDifferences>
          <dm:reverseDifferences rdf:parseType="Statements" >
               <rdf:Description rdf:about="#_39e4e305-1c70-4dcc-a423-45e4812dcd07">
                    <cim:Terminal.ConnectivityNode rdf:resource="#_5d74fc6a-b518-
4a3e-9e72-4827efd197cf"/>
               </rdf:Description>
          </dm:reverseDifferences>
     </dm:DifferenceModel>
</rdf:RDF>
```


For change of compound elements (7.2.3.7) the complete compound is replaced, i.e. the old element and all its members are removed by a reverse difference statement and added back with a forward difference statement.

7.3 CIMXML format style guide

A useful feature of RDF syntax is that it allows an arbitrary subset of a power system model to be serialized in a document. This is a two edged sword, however. A document produced by one party may not be usable by a second party if it does not contain all the properties expected. Moreover, a document containing a partial model may not be usable if the resource URN's do not agree with other documents.

The following guidelines apply to the content of a CIMXML document and help maximize the range of applications that can use it.

1) Include the likely primary key properties of each resource at the point it is introduced. For example, the cim:IdentifiedObject.name and cim:Equipment.EquipmentContainer properties are likely to be required properties.

   Reason: a large class of applications will want to load a database with the model data. Many database schemas will require primary key values on insertion.

2) Include single-valued properties rather than their many-valued inverse. For example, use cim:Equipment.EquipmentContainer and not cim:EquipmentContainer.Equipments.

   Reason: Because these properties are inverses, a statement predicated on one implies the converse statement predicated on the other. It is less error prone to include only one side and makes editing or transforming the document easier.

3) When encountering many to many relationships, there is usually a primary direction of reference. Include the primary reference rather than their many-valued inverse. For example, use cim:SynchronousMachine.MVArCapabilityCurves and not cim:MVArCapabilityCurve.SynchronousMachines, since the primary relationship is from SynchronousMachine to MVArCapabilityCurve.

   Reason: Same reasons as for item 2 above.

4) When encountering a single-valued relationship with a single value inverse, include either one, but not both. Importing software needs to be designed to handle either direction of reference and infer the inverse.

   Reason: Because these properties are inverses, a statement predicated on one implies the converse statement predicated on the other. This is less error prone, and arguably, makes editing or transforming the document easier.

5) Many valued properties, if used, appear as repeated property elements having the same property name.

7.4 Representing new, deleted and changed objects as CIMXML elements

The following cases exists for identification of elements and how they appear in full or difference models

- New objects are represented by the definition element (refer to 7.2.3.5) identified by a rdf:ID or rdf:about attribute in full or difference models.
- Deleted objects are represented by the definition element (refer to 7.2.3.5) identified by a rdf:ID or rdf:about attribute in difference models.
- Changed objects are represented by the description element (refer to 7.2.3.6) identified by a rdf:about attribute in difference models.
- An added property (e.g. internally a null value is changed to a valid value) is a change that appears only in the forward section of a difference model.
- A removed property (e.g. internally a valid value is changed to a null value) is a change that appears only in the backwards section of a difference model.

7.5 CIM RDF schema generation with CIM profile

This part of IEC 61970 discusses the generation of CIM RDF Schema. A CIMXML model exchange document uses a subset of the CIM to address the model exchange needs of a specific use case; see Part 400 series profile documents. A CIM profile defines that portion of the CIM that an importer and exporter of a CIMXML document should be expected to handle. The RDF Schema for a profile then contains only the classes and properties defined for that profile.

A RDF Schema file can be generated from the CIM UML model by an application having a user interface where the subset of the CIM UML model is interactively specified. The RDF Schema file can be used by an application to validate a CIMXML document, refer to Figure 6.

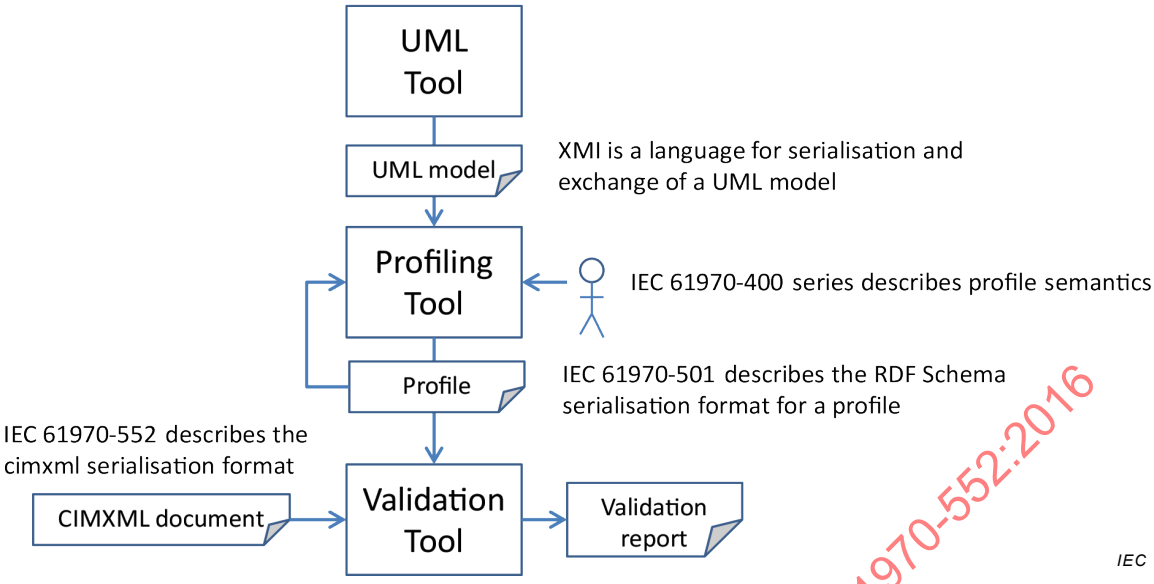

Figure 6 – Relations between UML, profile and CIMXML tools

Figure 6 describe the “UML model” at the top is used in a “Profiling tool” to generate a “Profile”. A “Validation tool” can use an existing “Profile” to validate a “CIMXML document” and generate a “Validation report”.

7.6 CIM extensions

The CIM RDF schema can be extended with new classes and attributes by providing a separate namespace. Because a separate namespace is used, the customized CIMXML documents clearly delineate what is CIM standard and what is custom. Several different custom extensions can exist and be clearly identified within the same XML document. When these customized documents are imported to information systems that know nothing about the extensions, the elements with the unknown tags can be simply ignored. The following declaration identifies an extended namespace “bpa”.

```
xmlns:bpa="http://www.bpa.gov/schema/cim_extension/2001may"
```

For example, we can add a non-CIM attribute, `OriginalPO`, to the `Breaker` class, as shown below. These customized tags for BPA can be simply ignored if a system import program is not interested in such extensions.

```
<cim:SynchronousMachine rdf:about="#_31dcf429-6Bfb-4e2e-b2996-42491b3abc1">
     <cim:IdentifiedObject.name>IN-2</cim:IdentifiedObject.name>
     <cim:SynchronousMachine.minimumMVAr>-9999</cim:SynchronousMachine.minimumMVAr>
     <cim:SynchronousMachine.operatingMode rdf:resource="http://iec.ch/TC57/2001/CIM-
schema-cim10#SynchronousMachineOperatingMode.generator"/>
     <bpa:OriginalPO>PO1234378</bpa:OriginalPO>
     <cim:RegulatingCondEq.RegulationSchedule rdf:resource="#_ca32746
fa0024c2bbcf47bc97430bf87"/>
     <cim:Equipment.EquipmentContainer rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a75"/>
</cim:SynchronousMachine>
```

The RDF schema corresponding to this extension can be added to a separate RDF schema document thereby keeping the CIM RDF schema clearly separate and allowing each to evolve independently.

7.7 RDF simplified syntax design rationale

The following points explain some of the choices made in the simplified syntax.

1) The literal properties could be represented by property attributes (RDF “W3C: RDF/XML Syntax Specification” grammar clause 6.10). This would be more compact. However, property elements were chosen because they are easier to deal with in XSLT expressions. (For example, they can be sorted.) They also make it easier to represent multi-line text.

2) The syntax is flat, with a two-level resource/property structure. More deeply nested structures might be more compact. Moreover, a well-chosen nested structure might permit common queries to be more easily encoded in XSLT expressions. On the other hand, the flat structure was chosen because it is the simplest structure possible and is easy to produce and interpret. By avoiding any application dependency on the details of a nesting structure it should be a more portable syntax.

3) All resources are given a type at the time they are introduced (by the definition element). However, the RDF model allows a resource to be un-typed. In the present application, un-typed resources are not required. However the difference model uses un-typed resources as described in 7.2.3.6.

Bibliography

[1] *XML for CIM Model Exchange*, IEEE PES PICA 2001 Conference Paper, May 2001, A. deVos, S. Widergren, J. Zhu

[2] *Simplified RDF Syntax for Power System Model Exchange*, CIMXML Interoperability Group Paper, 16 November 2000, A. deVos

[3] *Resource Description Framework (RDF) Model and Syntax Specification*, W3C Recommendation 10 February 2004 http://www.w3.org/TR/REC-rdf-syntax

[4] *Resource Description Framework (RDF) Schema Specification*, W3C Proposed Recommendation 03 March 1999 http://www.w3.org/TR/PR-rdf-schema, Dan Brickley, R.V. Guha, Netscape

[5] *Uniform Resource Identifiers (URI): Generic Syntax*; Berners-Lee, Fielding, Masinter, Internet Draft Standard August, 1998; RFC 2396

[6] *Namespaces in XML*; Bray, Hollander, Layman eds, W3C Recommendation; http://www.w3.org/TR/1999/REC-xml-names-20091208

[7] *Extensible Markup Language 1.0* (Second Edition), W3C Recommendation 6 October 2000, http://www.w3.org/TR/REC-xml, Bray, Paoli, Sperberg-McQueen, Maler

[8] *SiRPAC – Simple RDF Parser & Compiler*, http://www.w3.org/RDF/Implementations/SiRPAC

[9] IEC 61968-11, *Application integration at electric utilities – System interfaces for distribution management – Part 11: Common information model (CIM) extensions for distribution*

[10] IEC 61970-1, *Energy management system application program interface (EMS-API) – Part 1: Guidelines and general requirements*

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

INTERFACE DE PROGRAMMATION D'APPLICATION POUR SYSTÈME DE GESTION D'ÉNERGIE (EMS-API) –

Partie 552: Format d'échange de modèle CIMXML

AVANT-PROPOS

1) La Commission Électrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale IEC 61970-552 a été établie par le comité d'études 57 de l'IEC, Gestion des systèmes de puissance et échanges d'informations associés.

Cette deuxième édition annule et remplace la première édition parue en 2013. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

a) Nouvel Article 4 définissant la version du format CIMXML décrit dans le présent document.
b) Paragraphe 5.1, la déclaration sur la prise en charge du flux de travail est supprimée.

c) Paragraphe 5.2, une déclaration sur le caractère obligatoire de l'en-tête a été ajoutée. Des règles régissant l'utilisation de l'en-tête ont été ajoutées. Le débat portant sur la gestion de plusieurs documents et archives CIMXML n'est plus évoqué.

d) Paragraphe 5.3, FullModelDocumentElement est supprimé, une version mineure est ajoutée à l'URI de profil et la signification de l'en-tête est détaillée dans le Tableau 2.

e) Paragraphe 6.2, la description de rdf:ID et rdf:about a été mise à jour.

f) Le Paragraphe 6.3 présente la nouvelle forme urn:uuid et traite de la compatibilité ascendante.

g) Un nouveau Paragraphe 6.4 a été ajouté pour la prise en charge des anciens formats UUID.

h) Un nouveau Paragraphe 6.5 portant sur les types d'objets a été ajouté.

i) Paragraphe 7.2.3.3, description de l'emplacement de l'en-tête et suppression des lignes doubles.

j) Identification du document et références entre les documents mis à jour au Tableau 2 et en 7.2.3.4 et 7.2.4.6.

k) Paragraphe 7.2.3.7, un élément compound ne peut jamais être un élément racine.

l) Paragraphe 7.2.3.9, description de l'emboîtement composé ajoutée.

m) Paragraphes 7.2.3.4 et 7.2.4.7.3, précisions supplémentaires sur la suppression en cascade.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
57/1752/FDIS	57/1773/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 61970, publiées sous le titre général *Interface de programmation d'application pour système de gestion d'énergie (EMS-API)*, peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives à la publication recherchée. À cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

La présente partie de l'IEC 61970 est l'une des différentes parties de la série de normes qui définissent une interface de programmation d'application (API) pour un système de gestion d'énergie (EMS).

L'IEC 61970-301 spécifie un Modèle d'Information Commun (CIM): une vue logique des aspects physiques des opérations d'un service public de distribution d'électricité. Le CIM est décrit à l'aide du Langage de Modélisation Unifié (UML), un langage utilisé pour spécifier, visualiser et documenter les systèmes de façon orientée objet. Le langage UML est un langage d'analyse et de conception; et non un langage de programmation. Pour que les logiciels utilisent le CIM, ce dernier doit être converti en un schéma prenant en charge une interface programmable.

L'IEC 61970-501 décrit la traduction du CIM au format UML en un format lisible par une machine comme exprimé dans la représentation du Langage de balisage extensible (XML) de ce schéma à l'aide du langage de spécification du Schéma du Cadre de Description des Ressources (RDF).

La présente partie de l'IEC 61970 spécifie la manière dont le schéma RDF du CIM spécifié dans l'IEC 61970-501 est utilisé pour échanger des modèles de réseau électrique à l'aide du langage XML (appelé CIMXML) défini dans la série de normes de profils 61970-45x, telles que le Profil d'échange du modèle de réseau de transport du CIM décrit dans l'IEC 61970-452.

INTERFACE DE PROGRAMMATION D'APPLICATION POUR SYSTÈME DE GESTION D'ÉNERGIE (EMS-API) –

Partie 552: Format d'échange de modèle CIMXML

1 Domaine d'application

La présente partie de l'IEC 61970 spécifie le format et les règles pour échanger des informations de modélisation basées sur le CIM. Elle utilise le Schéma RDF du CIM présenté dans l'IEC 61970-501 comme le cadre de métamodèle pour générer les documents XML contenant les informations relatives à la modélisation des réseaux électriques. Le style de ces documents est appelé format CIMXML.

L'échange de modèles par transfert de fichiers répond à plusieurs objectifs utiles. Les documents de profils tels que l'IEC 61970-452 et d'autres profils dans la série de normes 61970-45x spécifient les exigences et les cas d'utilisation posant le contexte pour cette tâche. Bien que le format puisse être utilisé pour l'échange d'informations basé sur le CIM, les profils (ou sous-ensembles) spécifiques du CIM sont identifiés afin d'aborder les exigences d'échange particulières. L'exigence initiale contrôlant la consolidation de la présente spécification est l'échange des informations de modélisation du réseau de transmission pour la coordination de la sécurité des réseaux électriques.

La présente partie de l'IEC 61970 prend en charge un mécanisme pour les logiciels provenant de fournisseurs indépendants afin de produire et d'utiliser les informations de modélisation décrites dans le CIM dans un format commun. La solution proposée:

- est à la fois lisible par l'homme et par la machine, bien qu'elle soit essentiellement destinée à un accès programmatique,
- peut être accessible à l'aide de tout outil prenant en charge le Modèle Objet de Document (DOM) et d'autres interfaces de programmation d'application de XML normalisé,
- est autodescriptive,
- met à profit les recommandations actuelles du World Wide Web Consortium (W3C).

2 Références normatives

Les documents suivants cités dans le texte constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 60050, *Vocabulaire Électrotechnique International* (toutes les parties)

IEC TS 61970-2, *Energy management system application program interface (EMS-API) – Part 2: Glossary* (disponible en anglais seulement)

IEC 61970-501:2006, *Energy management system application program interface (EMS-API) – Part 501: Common Information Model Resource Description Framework (CIM RDF) schema* (disponible en anglais seulement)

W3C, *RDF/XML Syntax Specification (Spécification de la Syntaxe RDF/XML)*

W3C, *XLS Transformations (Transformations XSL) (XSLT)*

W3C, *Document Object Model (Modèle Objet de Document) (DOM)*

3 Termes et définitions

Pour les besoins du présent document, les termes et définitions donnés dans l'IEC 60050 (pour les définitions de glossaire général) et l'IEC TS 61970-2 (pour les définitions de glossaire de l'EMS-API), ainsi que les suivants s'appliquent.

3.1
Interface de Programmation d'Application
API
ensemble des fonctions publiques qu'offre un composant exécutable d'application pour être utilisées par d'autres composants exécutables d'application

Note 1 à l'article: L'abréviation « API » est dérivée du terme anglais développé correspondant « Application Program Interface ».

3.2
Modèle d'Information Commun
CIM
modèle abstrait représentant tous les principaux objets dans une entreprise de service public de distribution d'électricité généralement contenus dans un modèle d'information EMS

Note 1 à l'article: En fournissant une façon normalisée de représenter des ressources de réseau électrique comme des classes et des attributs d'objets ainsi que leurs relations, le CIM facilite l'intégration des applications EMS développées de façon indépendante par différents fournisseurs, entre des systèmes EMS complets développés de façon indépendante ou entre un système EMS et d'autres systèmes concernés par différents aspects de l'exploitation d'un réseau électrique tels que la gestion de la production ou de la distribution.

Note 2 à l'article: L'abréviation « CIM » est dérivée du terme anglais développé correspondant « Common Information Model ».

3.3
CIMXML
format de sérialisation pour l'échange de données XML tel qu'il est défini dans le présent document

3.4
Modèle Objet de Document
DOM
interface de plate-forme et de langage neutre définie par le World Wide Web Consortium (W3C) permettant aux programmes et aux scripts d'accéder dynamiquement au contenu, à la structure et au style des documents et de les échanger

Note 1 à l'article: L'abréviation « DOM » est dérivée du terme anglais développé correspondant « Document Object Model ».

3.5
Définition de Type de Document
DTD
norme pour décrire le vocabulaire et la syntaxe associés à un document XML

Note 1 à l'article: Le Schéma XML et RDF sont d'autres formes pouvant être utilisées.

3.6
Système de Gestion d'Énergie
EMS
système informatique comprenant une plate-forme logicielle offrant des services de soutien de base et un ensemble d'applications offrant les fonctionnalités nécessaires au bon fonctionnement des installations de production et de transport d'électricité afin d'assurer la sécurité adéquate d'approvisionnement énergétique à un coût minimal

Note 1 à l'article: L'abréviation « EMS » est dérivée du terme anglais développé correspondant « Energy Management System ».

3.7
Langage de Balisage Hypertexte
HTML
langage de balisage utilisé pour mettre en forme et présenter les informations sur le Web

Note 1 à l'article: L'abréviation « HTML » est dérivée du terme anglais développé correspondant « Hypertext Markup Language ».

3.8
modèle
ensemble de données décrivant les instances, les objets ou les entités réel(le)s ou calculé(e)s. Dans le contexte du CIM, la sémantique des données est définie par des profils, voir: 4.9. Par conséquent, un modèle peut contenir des données sur les équipements, les valeurs initiales et les résultats des flux de puissance, etc.

Note 1 à l'article: Dans une analyse de réseau électrique, un modèle désigne un ensemble de données statiques décrivant le réseau électrique. Des exemples de Modèles incluent le Static Network Model (Modèle de Réseau Statique), la Topology Solution (Solution de Topologie) et la Network Solution (Solution de Réseau) produits par une application de calcul de répartition des flux de puissance ou un estimateur d'état.

3.9
profil
schéma qui définit la structure et la sémantique d'un modèle pouvant être échangé

Note 1 à l'article: Un Profil est un sous-ensemble restreint du CIM plus général.

3.10
document de profil
ensemble de profils destinés à être utilisés ensemble dans un but commercial particulier

3.11
Cadre de Description des Ressources
RDF
langage recommandé par le W3C pour exprimer des métadonnées que les ordinateurs peuvent traiter simplement

Note 1 à l'article: RDF utilise XML comme syntaxe d'encodage.

Note 2 à l'article: L'abréviation « RDF » est dérivée du terme anglais développé correspondant « Resource Description Framework ».

3.12
schéma RDF
langage de spécification de schéma exprimé à l'aide de RDF pour décrire les ressources et leurs propriétés, y compris la manière dont les ressources sont liées les unes aux autres, utilisé pour spécifier un schéma spécifique à l'application

3.13
objets réels
objets du monde réel différents des objets implémentés dans les interfaces et les contrôleurs

Note 1 à l'article: Les objets réels pour le domaine de l'EMS sont définis comme des classes dans l'IEC 61970-301 Modèle d'Information Commun.

Note 2 à l'article: Les classes et les objets modélisent ce qui se trouve dans un réseau électrique et qui nécessite d'être représenté de la même façon pour les différentes applications EMS. Une classe est une description d'un objet réel, telle que PowerTransformer, GeneratingUnit ou Load qui nécessite d'être représentée comme faisant partie du modèle intégral de réseau électrique dans un EMS. Parmi les autres types d'objets figurent des éléments tels que les programmes et les mesurages que des applications EMS nécessitent également de traiter, d'analyser et d'enregistrer. De tels objets nécessitent une représentation commune pour atteindre les objectifs de compatibilité de connexion et d'interopérabilité de la norme EMS-API. Un objet particulier dans un réseau électrique ayant une identité unique est modélisé comme une instance de la classe à laquelle il appartient.

3.14
Langage Normalisé de Balisage Généralisé
SGML
norme internationale pour la définition de méthodes indépendantes de l'appareil et du système pour représenter les textes au format électronique

Note 1 à l'article: HTML et XML sont issus de SGML.

Note 2 à l'article: L'abréviation « SGML » est dérivée du terme anglais développé correspondant « Standard Generalized Markup Language ».

3.15
Langage de Modélisation Unifié
UML
langage de modélisation orienté objet et méthodologie pour spécifier, visualiser, élaborer et documenter les artefacts d'un processus intensif du système

Note 1 à l'article: L'abréviation « UML » est dérivée du terme anglais développé correspondant « Unified Modeling Language ».

3.16
Identificateur de Ressource Uniforme
URI
syntaxe et sémantique normalisées de Web pour identifier (référencer) les ressources (éléments tels que des fichiers, des documents, des images)

Note 1 à l'article: L'abréviation « URI » est dérivée du terme anglais développé correspondant « Uniform Resource Identifier ».

3.17
Langage de Balisage Extensible
XML
sous-ensemble du Langage Normalisé de Balisage Généralisé (SGML), ISO 8879, pour insérer des données structurées dans un fichier texte

Note 1 à l'article: Il s'agit d'une recommandation approuvée du W3C. Elle est gratuite, indépendante de la plate-forme et adaptée à de nombreux outils logiciels facilement disponibles.

Note 2 à l'article: L'abréviation « XML » est dérivée du terme anglais développé correspondant « eXtensible Markup Language ».

3.18
Langage Extensible de Feuille de Style
XSL
langage pour exprimer les feuilles de style des documents XML

Note 1 à l'article: L'abréviation « XSL » est dérivée du terme anglais développé correspondant « eXtensible Stylesheet Language ».

4 Version CIMXML

La version CIMXML est implémentée sous la forme d'une instruction de traitement XML apparaissant avant le document CIMXML, voir le Tableau 1.

Tableau 1 – Version CIMXML

Instruction de traitement XML	Version	Date de révision
iec61970-552	2.0	2014-03-12

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<?iec61970-552 version="2.0"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:cim="http://iec.ch/TC57/2004/CIM-schema-cim10#"
      …
</rdf:RDF>
```

5 Échange de modèles

5.1 Généralités

L'échange de modèles implique généralement l'échange d'un ensemble de documents, chacun contenant des données d'instance (désignées comme modèle) et un en-tête. La structure et la sémantique de chaque modèle ainsi que de l'en-tête sont décrites par un profil, qui n'est pas inclus dans les données échangées. L'échange global est régi par un ensemble de profils décrit dans un Document de Profil (norme de profils 61970-45x).

Un document CIMXML doit être constitué d'une partie en-tête et d'une partie modèle.

La partie en-tête décrit le contenu de la partie modèle contenue dans le document, par exemple la date de création du modèle, la description, etc. L'en-tête peut également identifier d'autres modèles ainsi que leur relation avec le modèle actuel. Ces informations s'avèrent importantes lorsque les modèles font partie d'un flux de travail comme lorsque les modèles sont liés les uns aux autres, par exemple lorsqu'un modèle succède à et/ou dépend d'un autre.

5.2 Règles applicables aux documents CIMXML et aux en-têtes

Un document CIMXML est décrit par un seul en-tête. Les en-têtes multiples ne sont pas admis dans un document CIMXML.

La partie en-tête doit toujours être le premier élément d'un document CIMXML. Les éléments de la partie en-tête sont

- l'élément FullModel, voir 7.2.3.4.
- l'élément DifferenceModel, voir 7.2.4.6.

Les données contenues dans la partie modèle sont définies par un ou plusieurs profils énumérés à l'intérieur de l'en-tête.

Les éléments d'un document CIMXML peuvent faire référence à des éléments (ressources) contenus dans d'autres documents CIMXML.

Dans la mesure où un seul élément d'en-tête est admis dans un document CIMXML, la partie modèle peut uniquement contenir des éléments pouvant être décrits par l'en-tête. Si plusieurs en-têtes sont nécessaires, un document CIMXML doit être créé pour chaque en-tête.

5.3 Description des modèles et des données d'en-tête

Une description d'un modèle est attachée comme données d'en-tête au modèle. La Figure 1 décrit le modèle avec des informations d'en-tête.

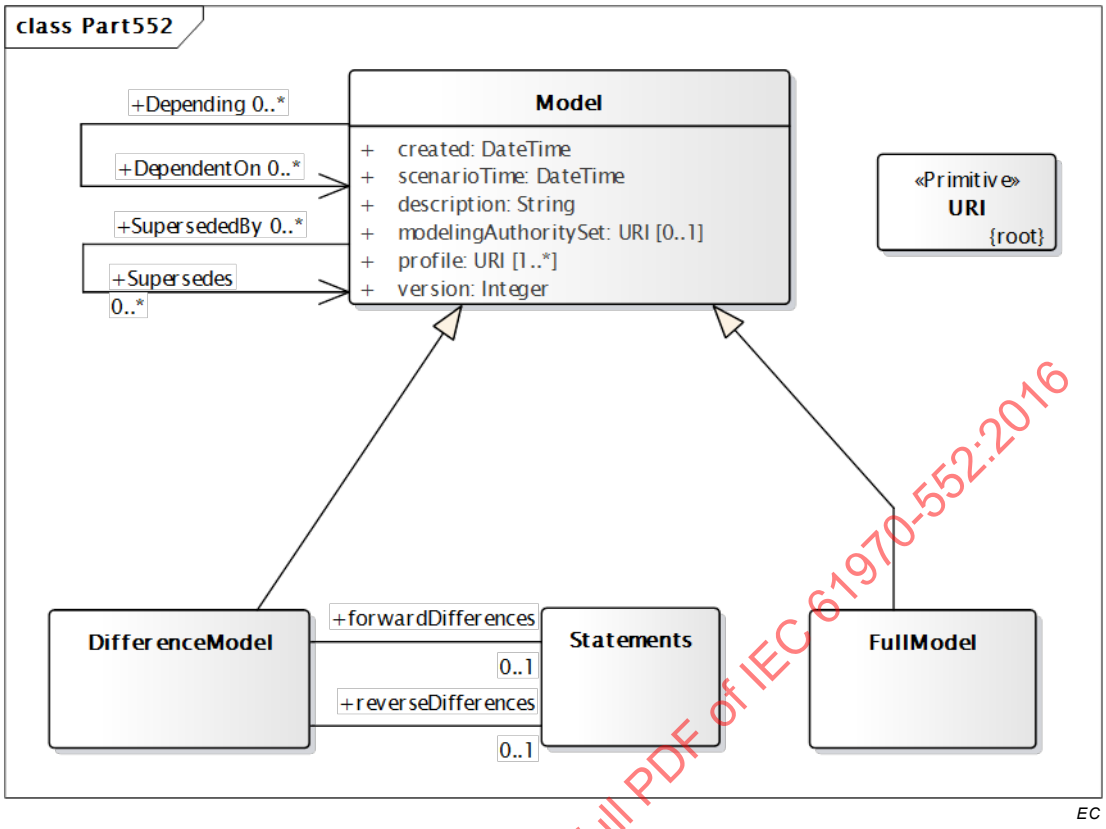

Figure 1 – Modèle avec en-tête

Dans la Figure 1, les classes FullModel, DifferenceModel et Statements décrivent les données du modèle alors que l'en-tête est décrit par la classe Model. Une description de bas en haut de ces classes est donnée ci-dessous:

- La classe Statements représente un ensemble d'éléments Definition (voir 7.2.3.5) et/ou Description (voir 7.2.3.6).
- La classe FullModel (voir 7.2.3.4) représente l'en-tête du modèle intégral et son contenu est décrit par la classe Model.
- La classe DifferenceModel (voir 7.2.4.6) représente l'en-tête du modèle de différence. Le contenu est décrit par la classe Model et les rôles d'association forwardDifferences et reverseDifferences. Les deux rôles d'association peuvent posséder un ensemble de Statements.
- La classe Model décrit le contenu de l'en-tête identique à FullModel et à DifferenceModel. Un Modèle est identifié par un attribut rdf:about. L'attribut rdf:about décrit de façon unique les données du modèle et non le document CIMXML. Une nouvelle identification rdf:about est générée pour les documents créés uniquement lorsque les données du modèle ont été modifiées. Il convient d'attribuer à plusieurs documents créés à partir de données non modifiées du modèle la même identification rdf:about qu'aux documents précédents générés à partir des mêmes données du modèle.

L'ordre des éléments xml dans le document généré est sans importance, sauf pour l'élément FullModel, qui apparaît toujours en premier dans un document. Les attributs de la classe Model sont décrits dans le Tableau 2.

Tableau 2 – Attributs d'en-tête

Classe	Attribut	Description
Model	created	Date de création du document.
Model	scenarioTime	Date et heure que le modèle représente, par exemple l'heure actuelle pour un modèle opérationnel, un modèle historique ou un futur modèle planifié.
Model	description	Une description du modèle, par exemple le nom de la personne ayant créé le modèle ainsi que le but de cette création. Le nombre de caractères en UTF-8 est limité à 2 000.
Model	modelingAuthoritySet	Un urn/uri faisant référence à l'organisation ou au rôle à l'origine du modèle dans le document CIMXML. Les modèles issus de la même organisation ou du même rôle mais destinés à des profils différents doivent avoir le même urn/uri.
Model	profile	Un urn décrivant les Profils régissant ce modèle. Il identifie de façon unique le Profil et sa version.
Model	version	Une description de la version du modèle à l'origine des données dans un document CIMXML. Exemples: – Variations du modèle d'équipement pour le ModelingAuthoritySet – Différents cas d'étude amenant différentes solutions. L'attribut de version est un nombre entier modifié en synchronisation avec l'identificateur rdf:about, se référer à la description de la classe Model précédant ce tableau.
Model	DependentOn	Références aux autres modèles dont dépend le modèle contenu dans le présent document, par exemple – Une solution de calcul de flux de puissance dépend du modèle de topologie à partir duquel elle a été calculée – Un modèle de topologie calculé par un processeur de topologie dépend du modèle de réseau à partir duquel il a été calculé. Les modèles cités en référence sont identifiés par l'attribut rdf:about du FullModel (voir 7.2.3.4) pour les documents de modèle intégral et par l'attribut rdf:about du DifferenceModel (voir 7.2.4.6) pour les documents de modèle de différence. Les références sont maintenues par le producteur du document CIMXML et sont valables pour le modèle avec la version et l'identificateur (voir la description de la classe Model précédant ce tableau) pour lesquels le document a été créé. L'utilisation de DependentOn par un consommateur est facultative. Si un consommateur d'un document a également utilisé les documents DependentOn, il est attendu qu'aucune référence non résolue ne soit présente dans le document actuellement utilisé.
Model	Depending	Tous les documents dépendant du modèle décrit par le présent document. Ce rôle n'est pas destiné à être inclus dans un document échangeant des données d'instance.
Model	Supersedes	Lorsqu'un modèle est mis à jour, le modèle résultant remplace les modèles de base utilisés pour la mise à jour. Par conséquent, il s'agit d'une référence aux modèles qui sont remplacés par le modèle contenu dans le présent document. Un modèle peut remplacer un ou plusieurs modèles, et pour chaque modèle remplacé, une référence Supersedes est incluse dans l'en-tête. Les modèles cités en référence sont identifiés par l'attribut rdf:about du FullModel (voir 7.2.3.4) pour les documents de modèle intégral et par l'attribut rdf:about du DifferenceModel (voir 7.2.4.6) pour les documents de modèle de différence. Les références sont maintenues par le producteur du document CIMXML et sont valables pour le modèle avec la version et l'identificateur (voir la description de la classe Model précédant ce tableau) pour lesquels le document a été créé. L'utilisation de Supersedes par un consommateur est facultative. Si un consommateur d'un document a également utilisé les documents Supersedes, il est attendu qu'aucune référence non résolue ne soit présente dans le document actuellement utilisé.
Model	SupersededBy	Tous les modèles remplaçant ce modèle. Ce rôle n'est pas destiné à être inclus dans un document échangeant des données d'instance.

Si un document est régénéré à partir d'un modèle reçu et non modifié, tous les attributs présentés au Tableau 2 doivent être les mêmes que dans le document initialement reçu.

Toutefois, si un modèle reçu est modifié de quelque manière que ce soit, aucun des attributs ne peut être différent selon la nature des modifications.

Les attributs DependentOn, Depending, Supersedes et SupersededBy décrivent la situation au moment de la génération du document dans le système. L'utilisation des références dans un système récepteur est facultative et un système récepteur peut utiliser ou non les références en fonction du cas d'utilisation, voir aussi 5.4.

L'attribut profile est un URI au format suivant pour les profils IEC:

- *http://iec.ch/<committee>/<year>/<standard>-<part>/<profile>/<version>/<minor_version>*

où le texte en *<italique>* est remplacé par un texte descriptif, par exemple

- http://iec.ch/TC57/2011/61970-452/Equipment/2/1

Cet URI doit être considéré comme indivisible, la chaîne intégrale transmet l'identification d'un profil. Par conséquent, le logiciel n'est pas censé analyser et interpréter les sous-chaînes de l'URI, par exemple l'année, la norme, la partie, etc.

D'autres organisations utilisant la sérialisation CIMXML peuvent avoir d'autres URI.

L'UML à la Figure 1 est converti en éléments CIMXML comme suit:

1) Une sous-classe à la Figure 1 (DifferenceModel, Statements et FullModel) apparaît comme des éléments de classe sous l'élément document (7.2.3.3).
2) Les éléments Statement apparaissent comme des éléments Definition (7.2.3.5) ou Description (7.2.3.6).
3) Les attributs littéraux, par exemple Model.created, apparaissent comme des éléments literal property (7.2.3.8).
4) Les rôles apparaissent, par exemple Model.Supersedes, comme des éléments resource property (7.2.3.11).
5) Les attributs et les rôles hérités apparaissent directement comme des éléments de la sous-classe répondant aux règles 3, 4 et 5 ci-dessus.
6) Un document de modèle CIMXML est identifié par un attribut rdf:about de Model (implicite dans l'UML). Par conséquent, les rôles DependentOn et Supersedes sont des références à l'attribut rdf:about de Model.
7) Un document peut être régénéré à plusieurs reprises à partir du même modèle. Les documents régénérés à partir d'un modèle non modifié conservent l'identification (rdf:about de Model) non modifiée d'un document précédent généré à partir du même modèle.
8) Un document DifferenceModel ou FullModel généré à partir du même modèle a la même identification et le même attribut Supersedes. Par conséquent, les documents DifferenceModel et FullModel peuvent s'utiliser indifféremment. Néanmoins, si un système récepteur requiert un modèle intégral équivalent au modèle contenu dans le document FullModel, le document DifferenceModel doit être appliqué à un modèle intégral correspondant au modèle remplacé.

5.4 Flux de travail

Un flux de travail est décrit par une séquence d'événements d'échanges. La description du modèle en 5.3 prend en charge les événements de flux de travail en fonction du temps avec l'attribut Model.Supersedes et les événements liés aux profils avec l'attribut Model.DependentOn. Un exemple est présenté à la Figure 2.

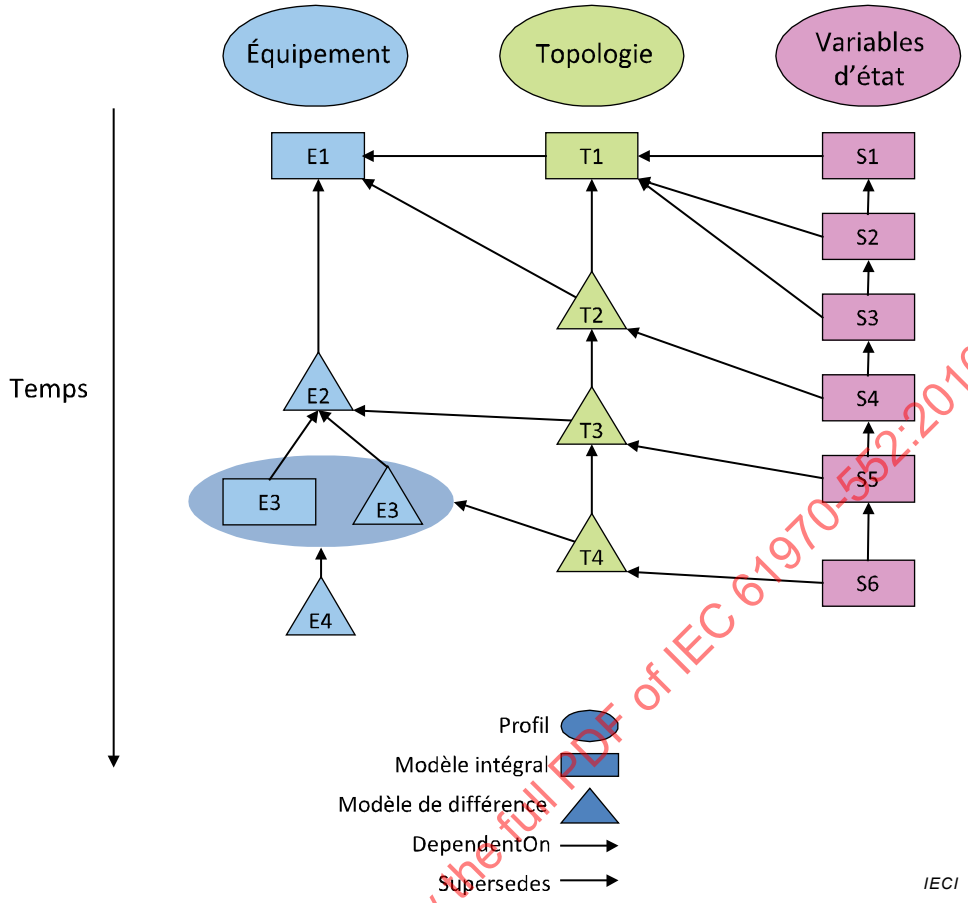

Figure 2 – Exemple d'événements de flux de travail

Dans cet exemple, un modèle de réseau résolu est échangé comme un ensemble de modèles régi par un Document de Profil (norme de profils 61970-45x) comportant les documents Equipment, Topology et State Variables. La ligne des temps à gauche dans la Figure 2 représente la manière dont le document du modèle Equipment est échangé au cours du temps. La ligne des temps au centre représente la manière dont les nouveaux résultats de Topology sont échangés au cours du temps ainsi que les modèles Equipment dont chacun dépend. La ligne des temps la plus à droite représente la manière dont plusieurs documents de State Variables sont échangés ainsi que les documents Topology dont ils dépendent. Noter également que le modèle Equipment E3 est représenté à la fois par un document intégral et un document DifferenceModel. La situation de la Figure 2 représente un cas simple. Une situation plus complexe est représentée à la Figure 3.

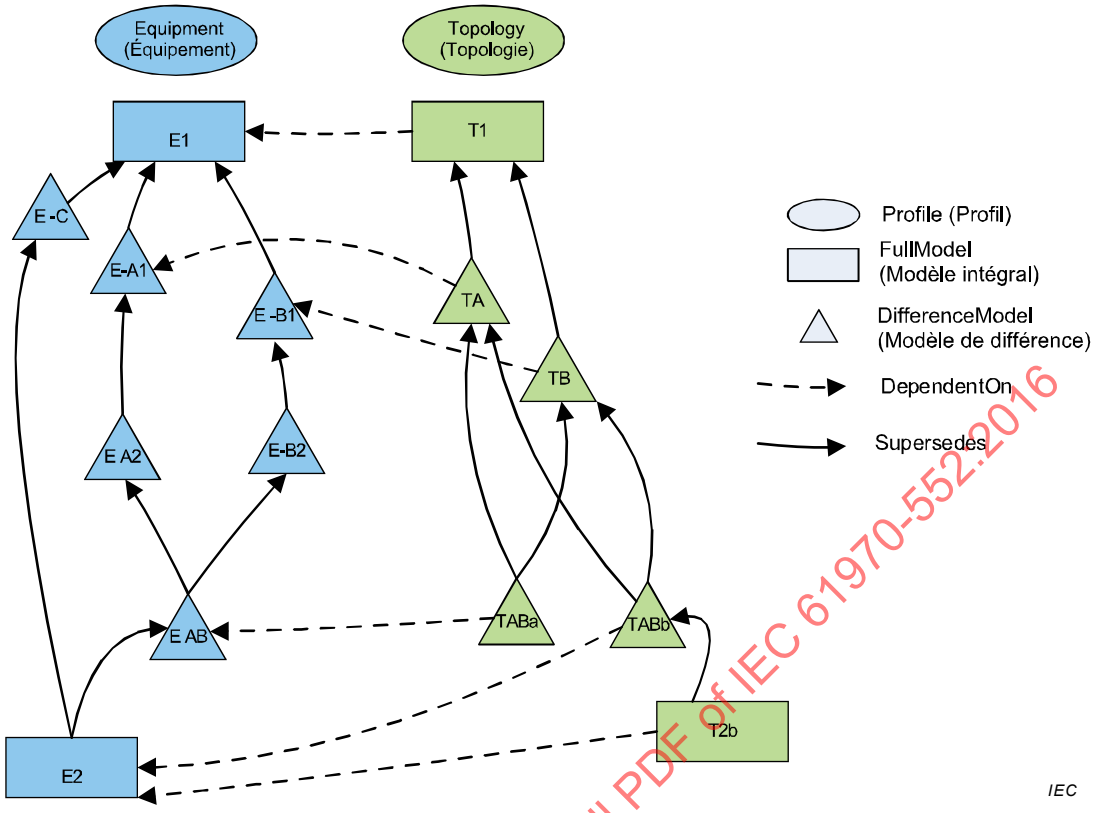

Figure 3 – Exemple d'événements de flux de travail avec plus de dépendances

Les documents CIMXML de la Figure 3 peuvent être créés à partir d'un environnement modélisateur de données comportant plusieurs branches de modification d'un modèle en parallèle, par exemple, le modèle Equipment comporte trois branches E-Ax, E-Bx et E-C qui fusionnent dans le modèle intégral E2 remplaçant les branches du modèle Equipment.

Un récepteur des documents CIMXML peut utiliser n'importe quel document Topology TA, TB, TABa ou T2b avec le modèle Equipment de E2. Étant donné que l'émetteur (ici le modélisateur de données) n'a vérifié que T2b avec E2, il s'agit de la seule combinaison censée fonctionner. En ce qui concerne T2b, le récepteur peut choisir d'appliquer TB et TABb à T1 au lieu d'utiliser T2b.

6 Identification des objets

6.1 URI comme identificateurs

Les UUID (Universally Unique IDentifier – Identificateur Unique Universel), aussi connus sous le nom de GUID (Globally Unique IDentifier – Identificateur Global Unique) peuvent être utilisés pour identifier des ressources de manière à ce que

- les identificateurs puissent être attribués indépendamment et de manière unique par différentes autorités. Ceci représente un avantage conséquent avec l'UUID;
- les identificateurs soient stables dans le temps et dans les documents.

Si, de plus, l'UUID est incorporé dans un Nom Uniforme de Ressources (URN), alors le document peut être simplifié en éliminant les déclarations des espaces de nommage de base XML (attributs xml:base). L'URN est un URI concis, absolu, de longueur fixe.

La stabilité des identificateurs dans le temps impose également les exigences suivantes

- Un identificateur doit être créé pour un objet dès que celui-ci existe. Noter que les objets d'un modèle peuvent ou non représenter un équipement physique réel.
- Il n'est pas permis de réutiliser l'identificateur d'un objet, par exemple, un objet supprimé.

La norme pour un URN contenant un UUID est définie par l'Internet Engineering Task Force RFC 4122.

RFC 4122 spécifie la syntaxe de l'URN et la manière dont la partie de l'UUID suivant les derniers deux-points est attribuée. L'algorithme est aligné sur, et techniquement compatible avec, l'ISO/IEC 9834-8:2005, Technologies de l'information, "Procédures opérationnelles pour les organismes d'enregistrement de l'OSI: Génération et enregistrement des identificateurs uniques universels (UUID) et emploi de ces identificateurs comme composants d'identificateurs d'objet ASN.1" ITU-T Rec. X.667, 2004.

Les éléments CIMXML sont identifiés par un URI. Par ailleurs, des formes différentes de l'URI sont admises mais non recommandées.

Un URI peut prendre deux formes:

- URL
- URN

Les formes URL et URN ont des structures fondamentalement différentes, c'est-à-dire:

- forme URL; *protocol*://*authority*/*path*?*query*#*fragment* où le *protocol* en CIMXML est http
- forme URN: urn:*namespace*:*specification* où *namespace* en CIMXML est uuid.

Le format URN *specification* est résumé ci-dessous

- numération hexadécimale à 8 caractères
- un tiret "-"
- numération hexadécimale à 4 caractères
- un tiret "-"
- numération hexadécimale à 4 caractères
- un tiret "-"
- numération hexadécimale à 4 caractères
- un tiret "-"
- numération hexadécimale à 12 caractères

où les caractères sont en minuscule conformément au W3C (ISO 8859-1, Jeu de caractères graphiques codés sur un seul octet connu sous le nom d'Alphabet latin n° 1.)

Un exemple de forme URN est présenté ci-dessous

"urn:uuid:26cc8d71-3b7e-4cf8-8c93-8d9d557a4846".

6.2 Informations relatives à rdf:ID et à rdf:about

Un élément CIMXML peut être identifié par deux constructions de RDF différentes:

- rdf:ID
- rdf:about

En RDF, l'identification rdf:ID signifie que l'identificateur est unique à l'intérieur d'un document, tandis que l'identification rdf:about signifie que l'identificateur est unique à

l'intérieur d'un espace de nommage. Si l'espace de nommage urn:uuid de l'UUID est utilisé pour l'identification rdf:about, les identificateurs sont globalement uniques. Par conséquent, CIMXML recommande l'emploi de l'identification rdf:about dans l'espace de nommage de l'UUID pour tous les identificateurs.

Dans le passé, un rdf:ID signifiait l'introduction d'une nouvelle ressource ou d'un nouvel objet, tandis que rdf:about faisait référence à une ressource ou un objet introduit(e) à un autre emplacement. Cette distinction n'existe plus. Dans la mesure où ces deux attributs sont des UUID, ils ont la même signification. Bien qu'ils soient tous les deux admis pour la compatibilité ascendante, il est préférable d'utiliser la forme rdf:about.

Les identificateurs dans les éléments FullModel et DifferenceModel doivent toujours utiliser l'identification rdf:about dans l'espace de nommage de l'UUID, c'est-à-dire commençant par "urn:uuid:".

6.3 Identification des éléments CIMXML

L'identification des ressources est tellement essentielle en RDF que tous les éléments représentant des objets sont identifiés par un attribut XML rdf:ID ou rdf:about. Toutes les classes dans le CIM qui héritent de IdentifiedObject ont l'attribut d'identification d'objet UML IdentifiedObject.mRID. L'attribut est mis en correspondance de manière implicite avec l'attribut XML rdf:ID/rdf:about.

Un document CIMXML ne peut utiliser que la forme URN (voir 6.1) comme décrit plus en détail ci-après.

Les fichiers CIMXML contiennent des éléments XML décrivant les objets du CIM (ACLineSegments, Substations, etc.). Le CIM comporte de nombreux rôles d'association qui apparaissent comme des références dans les éléments XML (généralement comme attributs rdf:resource ou rdf:about). Les données du CIM sont échangées dans différents documents CIMXML qui dépendent les uns des autres comme décrit à l'Article 4. Certaines références traversent ensuite les frontières des documents CIMXML. L'une des conséquences est que l'identification d'un objet du CIM doit être stable lors de sa durée de vie. Si tel n'est pas le cas, les références d'objets qui traversent les frontières de documents se brisent.

Une pratique courante dans les systèmes orientés objet consiste à prendre pour hypothèse que tous les objets possèdent un identificateur unique dans l'espace et dans le temps, ce qui signifie que:

- Différents identificateurs sont attribués à différents objets.
- Dès lors qu'ils sont attribués, les identificateurs ne sont jamais réutilisés même si l'objet d'origine les possédant n'est plus présent.

La forme URN décrite en 6.1 est utilisée comme identification d'élément CIMXML avec les différences suivantes

- Le préfixe "urn:uuid:" est remplacé par un trait bas "_". Le trait bas évite d'utiliser un caractère numérique de début pour la partie non de base de l'identificateur. Commencer la partie non de base de l'identificateur avec un caractère numérique est un RDF non valide. Dans tous les cas, le trait bas est ajouté pour simplifier les analyseurs, même si l'UUID commence avec un caractère non numérique.
- Le préfixe est défini comme un xml:base="urn:uuid:"

Quelques exemples:

- rdf:ID="_26cc8d71-3b7e-4cf8-8c93-8d9d557a4846" la forme rdf:ID".
- rdf:about="#_26cc8d71-3b7e-4cf8-8c93-8d9d557a4846" la forme "hash".
- rdf:about="urn:uuid:26cc8d71-3b7e-4cf8-8c93-8d9d557a4846" la forme "urn:uuid:".

Un récepteur compatible avec l'Édition 2 est censé pouvoir traiter les trois formes.

Un récepteur compatible avec l'Édition 1 est censé traiter les formes rdf:ID et rdf:about.

Il convient qu'un producteur compatible avec l'Édition 2 utilise, de préférence, uniquement la forme urn:uuid: Toutefois, il est admis qu'il continue à produire les formes rdf:ID et rdf:about.

Un producteur compatible avec l'Édition 1 produit les formes rdf:ID et rdf:about.

6.4 Anciens formats d'ID

Par le passé, différents identificateurs de ressources (ID) ont été utilisés, par exemple des ID non conformes à 6.3. Par conséquent, les systèmes en place peuvent utiliser des identificateurs d'objets non conformes au formatage décrit en 6.3.

Il n'est pas admis que les identificateurs non conformes à 6.3 soient constitués de plus de 60 caractères en UTF-8.

Dans le cas d'une modification du format d'ID, les ID conformes à l'ancien format ne vont plus être adaptés au nouveau format sans une traduction ou une mise en correspondance préalable. Dans la mesure où un ID n'est pas autodescriptif, une telle traduction est difficile à réaliser.

En raison des lourdes conséquences induites par une modification du format d'ID, il est admis que les systèmes existants continuent à utiliser l'ancien format pour les objets.

Lorsqu'un tel système est mis à niveau pour prendre en charge l'Édition 2 du présent document, il convient que les nouveaux objets créés dans un système utilisent le format d'ID tel qu'il est décrit en 6.1. En conséquence, il est admis que les objets créés avant la présente édition du présent document conservent l'ancien format d'ID sans que ne soit exigée une traduction vers le nouveau format. Cela signifie que le système prend en charge à la fois les anciens et les nouveaux formats.

6.5 Type d'objet

6.5.1 Généralités

Un type est attribué aux éléments du CIMXML au moyen de l'élément Definition (voir 7.2.3.6). Lorsque plusieurs documents sont échangés, il peut arriver que des éléments CIMXML possédant le même ID soient de types différents. La présente spécification ne décrit pas la façon dont une modification de type peut être notifiée.

6.5.2 Références à un type plus générique que le type concret

Lors d'un échange impliquant des profils divers, les données relatives à un objet possédant le même ID peuvent apparaître dans plusieurs documents. Un profil peut décrire les données à un niveau de la hiérarchie d'héritage plus haut que le type concret de l'objet. Un exemple est la PowerSystemResource, qui présente de nombreuses relations avec les autres classes. Toutefois, il est plus courant que les sous-classes de la PowerSystemResource (PSR), et non la PowerSystemResource elle-même, soient instanciées. Dans le cas où les données décrivant une classe du CIM sont divisées en plusieurs profils, il est admis que chaque profil utilise le niveau de la hiérarchie d'héritage le mieux adapté à l'échange. Cela s'explique par la volonté de simplifier le profilage et de cibler uniquement les données pertinentes. À ce titre, les échanges d'emplacement sont donnés en exemple, voir la Figure 4.

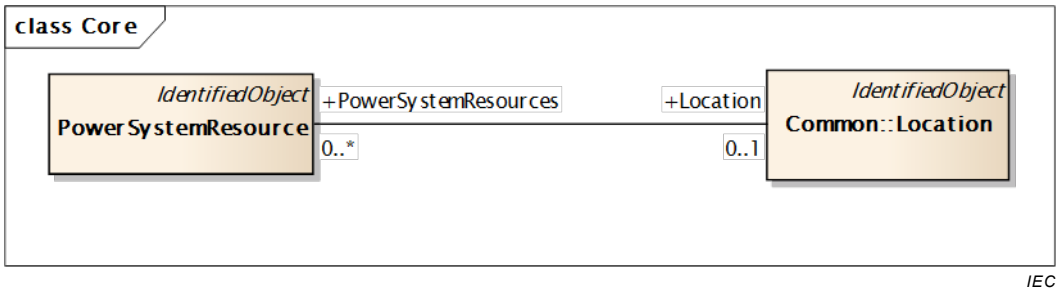

Figure 4 – PSR du CIM – Modèle des données d'emplacement

La classe PSR possède un élément Location, ce qui signifie qu'un profil portant sur l'échange d'emplacements échange l'attribut PowerSystemResource.Location. La PSR est néanmoins instanciée comme une classe plus spécifique, par exemple Breaker, BusbarSystem, etc. Dans un profil prévoyant l'échange d'informations relatives aux emplacements, il est impossible d'énumérer l'ensemble des classes autorisant un échange d'emplacement, d'autant plus que le nombre de sous-classes évolue avec le temps. Un profil stable pour l'échange d'emplacement consiste à utiliser uniquement la classe PowerSystemResource. Le profil inclut alors simplement la PowerSystemResource et un élément CIMXML se présente comme suit:

```
<cim:PowerSystemResource rdf:about="_26cc8d71-12f1-4de9-9e68-125d95073a75" >
     <cim:PowerSystemResource.Location rdf:resource="#_26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
<cim:PowerSystemResource >
```

Les données CIM sont généralement décrites par plusieurs classes dans la hiérarchie d'héritage, depuis des classes de base telles que cim:IdentifiedObject jusqu'aux classes spécifiques telles que cim:Breaker.

Des associations peuvent exister à tous les niveaux de cette hiérarchie. La classe cim:PowerSystemResource (PSR) est notamment associée à de nombreuses autres classes, par exemple cim:Location.

Lorsqu'un profil comprend une association, l'un des côtés de l'association est choisi, généralement celui présentant la cardinalité la moins élevée. Toutefois, il est important de choisir le sens le plus naturel, c'est-à-dire qu'il est plus naturel de déclarer "une PSR a un emplacement" plutôt que "un emplacement peut avoir une ou plusieurs PSR". Par conséquent, en fonction du niveau auquel se situe une association dans la hiérarchie d'héritage, un élément resource property (voir 7.2.3.11) peut faire référence à une classe abstraite de ladite hiérarchie, comme c'est le cas avec la référence "une PSR a un emplacement". Dans la mesure où les classes spécifiques apparaissent dans un document CIMXML, cela peut être interprété de façon à ce qu'un profil doive énumérer toutes les classes spécifiques héritant de l'association, par exemple cim:Breaker, cim:Disconnector etc. Une telle opération a pour conséquence de surcharger le profil, qui devient alors difficile à maintenir.

Un élément resource property (voir 7.2.3.11) peut apparaître dans deux cas différents

- Un document dans lequel l'objet est défini par un élément definition (voir 7.2.3.6). Dans ce cas, toutes les propriétés sont énumérées par rapport à la définition de l'objet CIM. Toutes les classes sont ainsi énumérées, c'est-à-dire qu'une énumération excessive ne pose pas de problème.
- Un document dans lequel un objet défini à un autre emplacement est mentionné par un élément description (voir 7.2.3.6). Dans ce cas, la classe d'objet spécifique n'est pas pertinente et l'objet est mentionné par un attribut rdf:about.

Par conséquent, les éléments se rapportant à des objets définis à un autre emplacement doivent utiliser l'élément description à la place de l'élément definition lorsque des propriétés issues de classes moins spécifiques sont incluses. Appliqué à la déclaration "une PSR a un emplacement", cela signifie que seul l'emplacement de PSR, et non de l'ensemble des sous-classes de PSR, est inclus dans le profil, pour apparaître ensuite comme un élément description et non comme un élément definition dans le document CIMXML. Se reporter à l'exemple de la Figure 4.

Un profil peut décrire la relation en partant de la PSR jusqu'à l'emplacement, ce qui donne lieu à un élément CIMXML Breaker possédant un emplacement:

```
<cim:Breaker rdf:about="_26cc8d71-12f1-4de9-9e68-125d95073a75">
     <cim:PowerSystemResource.Location rdf:resource="#_26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
</cim:Breaker >
```

Pour éviter d'inclure tous les types d'équipements dans un profil, la référence peut se faire dans l'autre sens comme dans l'exemple ci-dessous:

```
<cim:Location rdf:about="_26cc8d71-3b7e-4cf8-8c93-8d9d557a4847" >
     <cim:Location.PowerSystemResource rdf:resource="#_26cc8d71-12f1-4de9-9e68-
125d95073a75"/>
</cim:Location>
```

Pour éviter d'utiliser le même Location pour plusieurs PSR, la cardinalité Location.PowerSystemResource est réduite de 0..* à 1.

La mise en œuvre de la solution proposée dans le présent alinéa donne le résultat suivant:

```
<cim:PowerSystemResource rdf:about="_26cc8d71-12f1-4de9-9e68-125d95073a75" >
     <cim:PowerSystemResource.Location rdf:resource="#_26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
<cim:PowerSystemResource >
```

7 Règles et conventions relatives au format CIMXML

7.1 Généralités

En tenant compte du Schéma RDF du CIM décrit dans l'IEC 61970-501, un modèle de réseau électrique peut être converti pour l'exportation sous forme d'un document XML (voir la Figure 5). Il est fait référence à ce document comme un document CIMXML. Toutes les balises (descriptions de ressources) utilisées dans le document CIMXML sont fournies par le schéma RDF du CIM. Le document d'échange de modèle CIMXML résultant peut être analysé et les informations importées dans un système étranger.

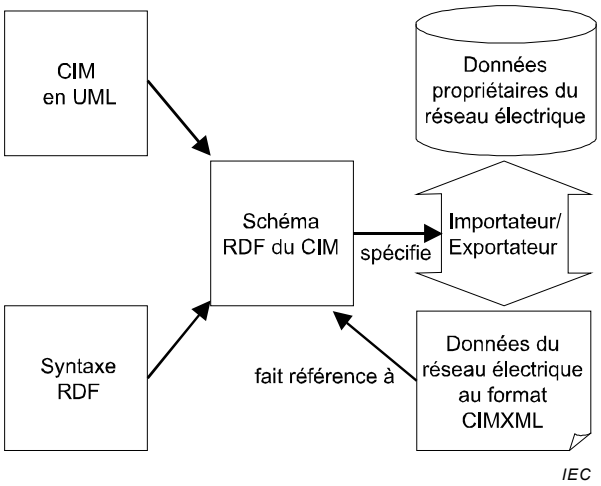

Figure 5 – Mécanisme d'échange de modèles de réseau électrique basé sur le langage CIMXML

7.2 Syntaxe RDF simplifiée

7.2.1 Généralités

La syntaxe RDF fournit de nombreuses façons de représenter le même ensemble de données. Par exemple, une association entre deux ressources peut être écrite avec un attribut ressource ou en imbriquant un élément dans un autre. Cela est susceptible de rendre difficile l'utilisation de certains outils XML, tels que les processeurs XSL, avec le document CIMXML.

Par conséquent, seul un sous-ensemble de la Syntaxe RDF doit être appliqué lors de la création des documents CIMXML. Cette syntaxe simplifie le travail des personnes chargées de la mise en œuvre pour élaborer un logiciel de sérialisation et de désérialisation du modèle, ainsi que pour améliorer l'efficacité des outils XML généraux lorsqu'ils sont utilisés avec des documents CIMXML. La syntaxe réduite est un sous-ensemble adéquat de la syntaxe RDF normalisée; par conséquent, elle peut être lue par le logiciel de désérialisation de RDF disponible.

Les Paragraphes du 7.2 définissent un sous-ensemble de la Syntaxe RDF. Cette syntaxe simplifiée est destinée à l'échange des modèles du réseau électrique entre les entreprises de service public. Le but de la spécification est de faciliter la tâche des personnes chargées de la mise en œuvre pour élaborer le logiciel de désérialisation pour les données RDF, pour simplifier leur choix lors de la sérialisation des données RDF et pour améliorer l'efficacité des outils XML généraux tels que les processeurs XSLT lorsqu'ils sont utilisés avec les données RDF sérialisées.

La syntaxe réduite est un sous-ensemble adéquat de la syntaxe RDF normalisée. Par conséquent, elle peut être lue par un logiciel de désérialisation RDF comme SirPAC [8][1]. Sur ce point, elle se distingue des autres propositions par une syntaxe simplifiée, telle que [9], [10].

La syntaxe réduite n'altère pas la puissance du modèle de données RDF. Cela signifie que n'importe quelle donnée RDF peut être échangée en utilisant cette syntaxe. De plus, les fonctionnalités de RDF sont préservées telles que l'aptitude à étendre un modèle défini dans un document avec déclarations dans un second document.

1 Les chiffres entre crochets se réfèrent à la bibliographie.

7.2.2 Notation

La syntaxe simplifiée est définie en 7.2.3. Chaque type d'élément est défini dans un paragraphe commençant par un modèle de l'élément, suivi de texte de définition et d'une référence à la grammaire RDF. La sémantique de l'élément n'est pas détaillée (voir la recommandation RDF "W3C: Spécification de la Syntaxe RDF/XML" pour cette information). La notation pour le modèle de l'élément est comme suit:

1) Un symbole en *italique* au lieu d'un type d'élément, le nom de l'attribut ou la valeur de l'attribut indique le type de nom exigé ou la valeur exigée. Le symbole est défini dans le texte.
2) Le symbole *rdf* indique le préfixe d'espace de nommage choisi par l'implémentation pour l'espace de nommage RDF. De la même façon, le symbole *cim* indique le préfixe choisi pour l'espace de nommage du CIM.
3) Un commentaire (*<!-- comment text -->*) au sein du modèle d'élément indique le contenu autorisé.
   - Un symbole en *italique* indique un type d'élément ou autre contenu défini dans le texte.
   - Une construction (a | b) indique que a et b sont possibles.
   - Une construction a* indique zéro répétition de a ou plus.
4) Tout le reste du texte dans le modèle est littéral.
5) La grammaire RDF est décrite en 6.1, "W3C: Spécification de la Syntaxe RDF/XML".

7.2.3 Définition de la syntaxe (normative)

7.2.3.1 Généralités

La définition de la syntaxe est enrichie d'exemples. Il convient que les exemples aident à mieux comprendre la définition de la syntaxe formelle. Le même exemple est utilisé pour plusieurs définitions de syntaxe.

L'UTF-8 est la norme pour l'encodage des fichiers. L'UTF-16 n'est pas pris en charge.

7.2.3.2 URI des espaces de nommage définis dans la présente spécification

Les espaces de nommage suivants sont définis dans la présente spécification:

- *cim-model-description_uri* décrit par xmlns:md
- *difference-model-namespace-uri* décrit par xmlns:dm

Leurs valeurs sont définies comme:

- xmlns:md="http://iec.ch/TC57/61970-552/ModelDescription/1#"
- xmlns:dm="http://iec.ch/TC57/61970-552/DifferenceModel/1#"

7.2.3.3 Élément Document

```
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
     xmlns:cim="cim-namespace-uri"
     xmlns:md="cim-model-description_uri"
     xml:base="urn:uuid:">
     <!-- Content: full-model (definition|description)* -->
</rdf:RDF>
```

Exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:cim="http://iec.ch/TC57/2004/CIM-schema-cim10#"
      xmlns:md="http://iec.ch/TC57/61970-552/ModelDescription/1#"
      xml:base="urn:uuid:">
      <md:FullModel rdf:about="urn:uuid:26cc8d71-3b7e-4cf8-8c93-8d9d557a4846">
            <md:Model.created>2008-12-24T03:19:13</md:Model.created>
            <md:Model.Supersedes rdf:resource="urn:uuid:26cc8d71-3b7e-4cf8-8c93-
8d9d557a4847"/>
            <md:Model.DependentOn rdf:resource="urn:uuid:26cc8d71-3b7e-4cf8-8c93-
8d9d557a4848"/>
            <md:Model.version>32</md:Model.version>
      <md:Model.modelingAuthoritySet>http://polarenergy.com/2008/NorthPoleTSO</md:Mo
del.modelingAuthoritySet>
            <md:Model.description>Santa Claus made a study case peak load summer base
topology solution</md:Model.description>
            <md:Model.profile>http://iec.ch/TC57/61970-
452/Equipment/1</md:Model.profile>
      <md:Model.profile>http://iec.ch/TC57/61970-
452/EquipmentShortCircuit/1</md:Model.profile>
      </md:FullModel>
      …
</rdf:RDF>
```

1) Le type d'élément est rdf:RDF.
2) L'espace de nommage RDF doit être déclaré comme http://www.w3.org/1999/02/22-rdf-syntax-ns#.
3) L'espace de nommage du CIM doit être déclaré. Avec les versions les plus récentes du schéma CIM, la version nécessite d'être ajustée dans l'espace de nommage du CIM. Les parties échangeant les documents doivent s'accorder sur la version utilisée.
4) D'autres espaces de nommage peuvent être déclarés.
5) L'élément d'en-tête du modèle intégral apparaît avant les éléments definition/description du modèle intégral.

7.2.3.4 Élément Full-model

```
<md:FullModel rdf:about=model-uri>
      <!--      Content:      (literal-property|resource-property|compound-
property)* -->
</md:FullModel >
```

```
<md:FullModel rdf:about="=”urn:uuid:26cc8d71-…">
   <md:Model.created>2008-12-24T03:19:13</md:Model.created>
      <md:Model.Supersedes rdf:resource="urn:uuid:26cc8d71-a002-4c2b-bcf4-
7bc97430bf87"/>
      <md:Model.DependentOn rdf:resource=urn:uuid:26cc8d71-a002-4c2b-bcf4-
7bc97430bf88"/>
      <md:Model.version>32</md:Model.version>
   <md:Model.modelingAuthoritySet>http://polarenergy.com/2008/NorthPoleTSO</md
:Model.modelingAuthoritySet>
      <md:Model.description>Santa Claus made a study case peak load summer base
topology solution</md:Model.description>
      <md:Model.profile>http://iec.ch/TC57/61970-
456/StateVariables/1</md:Model.profile>
</md:FullModel>
```

1) L'élément full model introduit un nouveau modèle.
2) La valeur de l'attribut about, model-uri, est un nom choisi par l'implémentation. model-uri identifie de façon unique un document et constitue le nom référencé par d'autres documents, par exemple Supersedes ou DependentOn, comme indiqué à la Figure 2.
3) L'attribut rdf:about du FullModel identifie un document CIMXML.

7.2.3.5 Élément Definition

```
<classname rdf:ID=identity>
   <!-- Content:
   (literal-property|resource-property|compound-property)*
   -->
</classname>


<classname rdf:about=resource-uri>
   <!-- Content:
   (literal-property|resource-property|compound-property)*
   -->
</classname>
```

Exemple:

```
<cim:SynchronousMachine rdf:about="#_31dcf429-6bfb-4e2e-b2996-42491b3abc1">
     <cim:IdentifiedObject.name>IN-2</cim:IdentifiedObject.name>
     <cim:SynchronousMachine.minimumMVAr>-9999</cim:SynchronousMachine.minimumMVAr>
     <cim:SynchronousMachine.operatingMode rdf:resource="http://iec.ch/TC57/2001/CIM-
schema-cim10#SynchronousMachineOperatingMode.generator"/>
     <cim:RegulatingCondEq.RegulationSchedule rdf:resource="#_ca32746f-a002-4c2b-
bcf4-7bc97430bf87"/>
     <cim:Equipment.EquipmentContainer rdf:resource="#_6cb8701a-12f1-4de9-9e68-
125d95073a75"/>
</cim:SynchronousMachine>
```

1) L'élément definition introduit une nouvelle ressource et indique son type. Il existe deux formes: la première en tant qu'attribut rdf:ID et la deuxième en tant qu'attribut rdf:about.
2) Le type d'élément, *classname*, est le nom qualifié en XML d'une classe du schéma CIM ou d'un autre schéma déclaré comme un espace de nommage dans l'élément document.
3) La valeur de l'attribut id, *identity*, est choisie par l'implémentation. Elle doit être unique dans le document. (Elle n'est pas nécessairement liée au nom de la ressource du réseau électrique.)
4) Un élément Definition avec la même identification peut apparaître dans différents documents. Cela signifie que les propriétés contenues dans les différents documents décrivent le même objet et que la description est répartie sur plusieurs documents. C'est le cas lorsque les données d'une classe sont divisées en plusieurs profils et que chaque profil est échangé comme son propre document CIMXML.

7.2.3.6 Élément Description

```
<rdf:Description rdf:about=resource-uri>
   <!-- Content:
   (literal-property | resource-property | compound-property)*
   -->
</rdf:Description >
```