BS EN 61970-552:2014

BSI Standards Publication

Energy Management System Application Program Interface (EMS-API)

Part 552: CIMXML Model Exchange Format

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of EN 61970-552:2014. It is identical to IEC 61970-552:2013.

The UK participation in its preparation was entrusted to Technical Committee PEL/57, Power systems management and associated information exchange.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2014.
Published by BSI Standards Limited 2014

ISBN 978 0 580 75930 7
ICS 33.200

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 March 2014.

Amendments/corrigenda issued since publication

Date	Text affected

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 61970-552

March 2014

ICS 33.200

English version

Energy Management System Application Program Interface (EMS-API) - Part 552: CIMXML Model Exchange Format
(IEC 61970-552:2013)

Interface de programmation d'application pour système de gestion d'énergie (EMS-API) -
Partie 552: Format d'échange de modèle CIMXML
(CEI 61970-552:2013)

Schnittstelle für Anwendungsprogramme für Netzführungssysteme (EMS-API) -
Teil 552: CIM-XML-Modell Austauschformat
(IEC 61970-552:2013)

This European Standard was approved by CENELEC on 2013-12-03. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization

Comité Européen de Normalisation Electrotechnique

Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Avenue Marnix 17, B - 1000 Brussels

© 2014 CENELEC - All rights of exploitation in any form and by any means reserved worldwide for CENELEC members.

Foreword

The text of document 57/1386/FDIS, future edition 1 of IEC 61970-552, prepared by IEC/TC 57, "Power systems management and associated information exchange" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN 61970-552:2014.

The following dates are fixed:

• latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2014-09-21
• latest date by which the national standards conflicting with the document have to be withdrawn	(dow)	2016-12-03

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC [and/or CEN] shall not be held responsible for identifying any or all such patent rights.

Endorsement notice

The text of the International Standard IEC 61970-552:2013 was approved by CENELEC as a European Standard without any modification.

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

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

NOTE When an international publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

Publication	Year	Title	EN/HD	Year
IEC 60050	Series	International Electrotechnical Vocabulary (IEV)	-	-
IEC 61968-11	-	Application integration at electric utilities - System interfaces for distribution management - Part 11: Common information model (CIM) extensions for distribution	EN 61968-11	-
IEC/TS 61970-2	-	Energy management system application program interface (EMS-API) - Part 2: Glossary	CLC/TS 61970-2	-
IEC 61970-301	-	Energy management system application program interface (EMS-API) - Part 301: Common information model (CIM) base	EN 61970-301	-
IEC 61970-501	-	Energy management system application program interface (EMS-API) - Part 501: Common Information Model Resource Description Framework (CIM RDF) schema	EN 61970-501	-
W3C	-	Document Object Model (DOM)	-	-
W3C	-	RDF/XML Syntax Specification	-	-
W3C	-	Extensible Markup Language (XML) 1.0	-	-
W3C	-	XSL Transformations (XSLT)	-	-

CONTENTS

INTRODUCTION

This International standard is part of the IEC 61970 series 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.

IEC 61970-552 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 International Standard specifies a Component Interface Specification (CIS) for Energy Management Systems Application Program Interfaces. This part 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 standard 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.

This document is the Level 2 Component Interface Specification document that describes in narrative terms (with text and examples based on the CIM) the detailed definition of the CIMXML format.

2 Normative references

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

IEC 60050 series, *International Electrotechnical Vocabulary*

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

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

IEC 61970-301, *Energy management system application program interface (EMS-API) – Part 301: Common information model (CIM) base*

IEC 61970-501, *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: Extensible Markup Language (XML) 1.0

W3C: XSL Transformations (XSLT)

W3C: Document Object Model (DOM)

3 Terms and definitions

For the purposes of this International Standard, the terms and definitions contained in IEC 60050 (for general glossary) and IEC 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 objects or entities real or computed

Note 1 to entry: In the context of CIM the semantics of the data is defined by profiles; refer to 3.9.

Note 2 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 Modeling Language
UML
object-oriented modeling language and methodology for specifying, visualizing, constructing, and documenting the artifacts 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 stylesheets for XML documents

4 Model exchange header

4.1 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 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 header describes the content of the model 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.

Subclauses 4.2 to 4.4 define the model with header data and work flow it is designed to support.

4.2 CIMXML documents and headers

A CIMXML document is described by a single header. Multiple headers in a CIMXML document are not allowed. Hence instance data in a CIMXML document adhere to a single profile.

In case multiple and possibly related CIMXML documents need to be kept together they shall be collected in an archive, e.g. zip.

4.3 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.

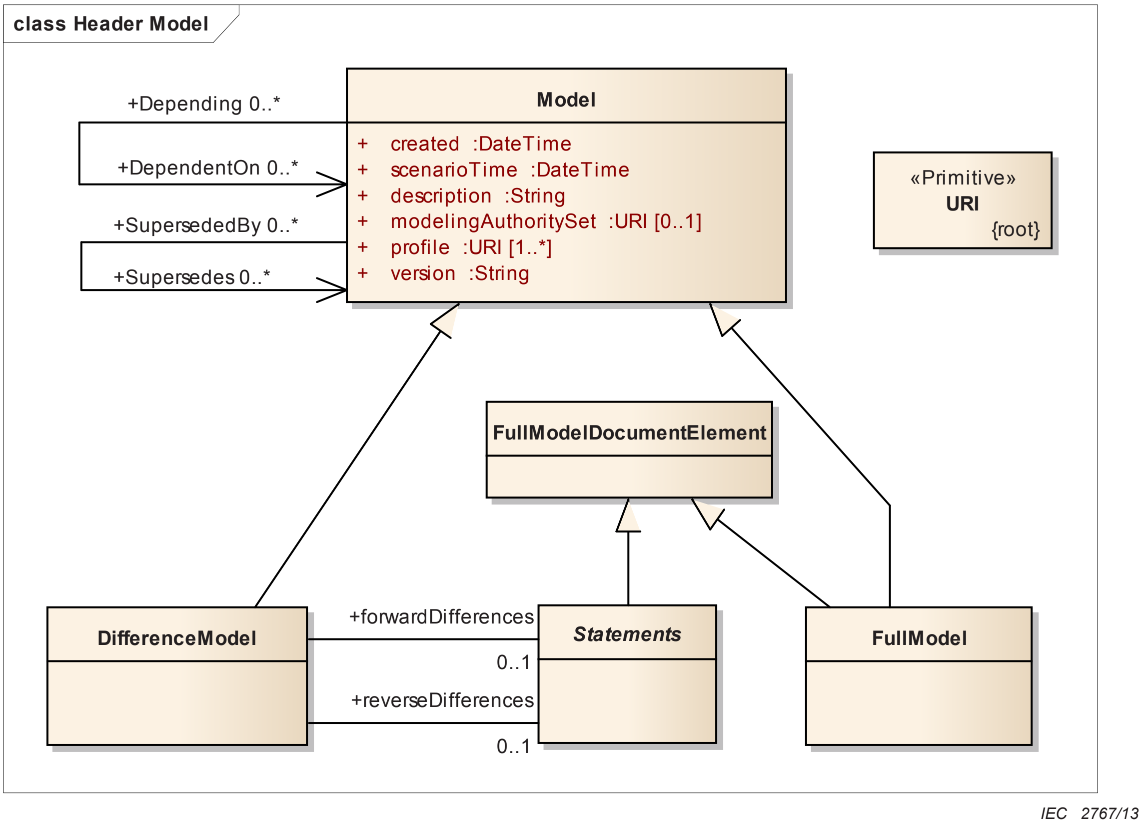

Figure 1 – Model with header

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

- The FullModelDocumentElement class represents any of the elements that may appear in a full model document. It has the two sub types Statements or FullModel, both are further described below. A full model document typically contains one FullModel element and one set of Definition elements.
- The Statements class represent a set of Definition (refer to 6.2.3.5) and/or Description (refer to 6.2.3.6) elements.
- The FullModel (refer to 6.2.3.4) class represent the full model header and its contents is described by the Model class.
- The DifferenceModel (refer to 6.2.4.6) 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 and not the document where the header exists. Hence multiple documents created from the same unchanged data model will have the same rdf:about. This also means that a model change result in a new rdf:about next time a document is created.

The Model class attributes are described in Table 1.

Table 1 – Header attributes

Class	Attribute	Description
Model	created	The date when the model was created (note this is typically not when the CIMXML document was created which is after this time).
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.
Model	modelingAuthoritySet	A URN describing the equipment model sourcing the data in a CIMXML document, e.g. a model for the whole or a part of a country.
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 a custom string that is changed in synchronisation with the rdf:about identifier, refer to description of the Model class above.
Model	DependentOn	A reference to the models that the model described by 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.
Model	Depending	All models depending on this model. This role is not intended to be included in any document exchanging instance data.
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 CIMXML documents describing the updated models.
Model	SupersededBy	All models superseding this model. This role is not intended to be included in any document exchanging instance data.

The profile attribute is a URI having the following format:

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

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

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 (6.2.3.3).
2) Statement elements appear as Definition (6.2.3.5) or Description elements (6.2.3.6).
3) Literal attributes, e.g. Model.created, appears as literal property elements (6.2.3.8).
4) Roles appear, e.g. Model.Supersedes, as resource property elements (6.2.3.10).
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 full model document may be regenerated multiple times from the same source data. Full model documents regenerated from unchanged source data keep the model identification (Model rdf:about) unchanged from the original full model document.

8) When generating a full model document superseding a differential the new full model document will have the same model identification (Model rdf:about) as the differential if the model is unchanged since the differential was created. Hence it is an alternate to the differential.

4.4 Work flow

A work flow is described by a sequence of exchange events. The model description in 4.3 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.

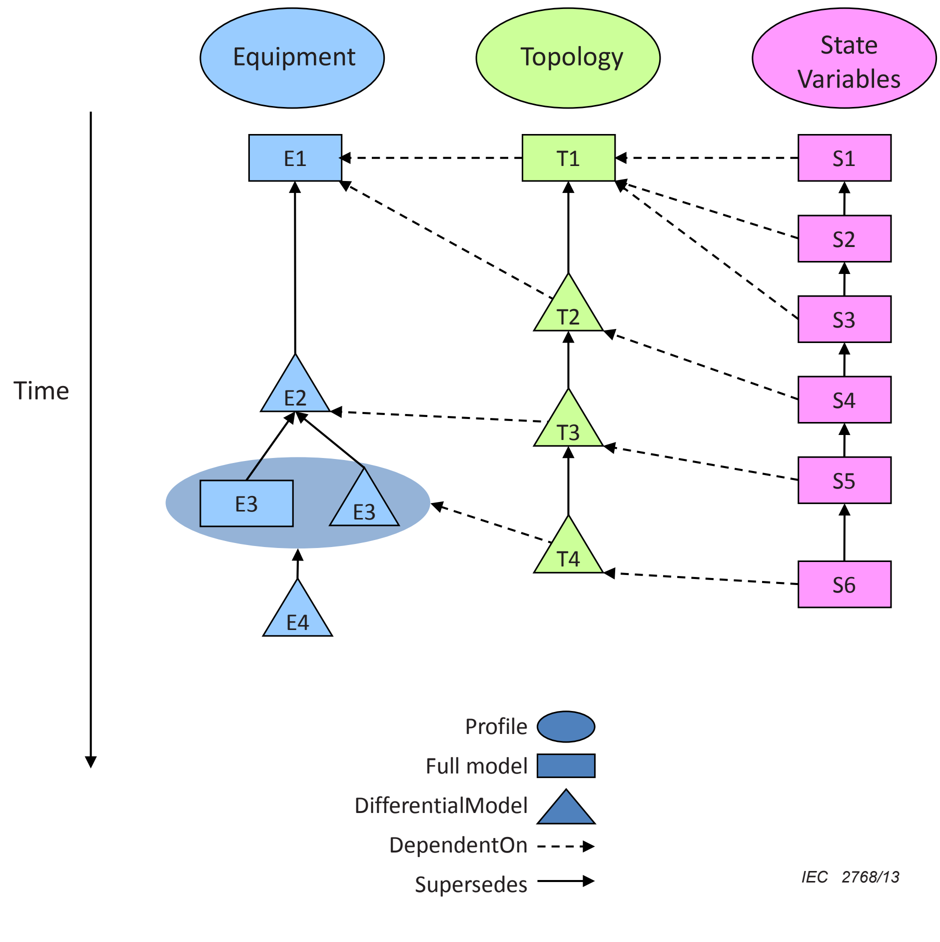

Figure 2 – Example work flow events

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 an incremental document. The situation in Figure 2 represents a simple case. A more complex situation is shown in Figure 3.

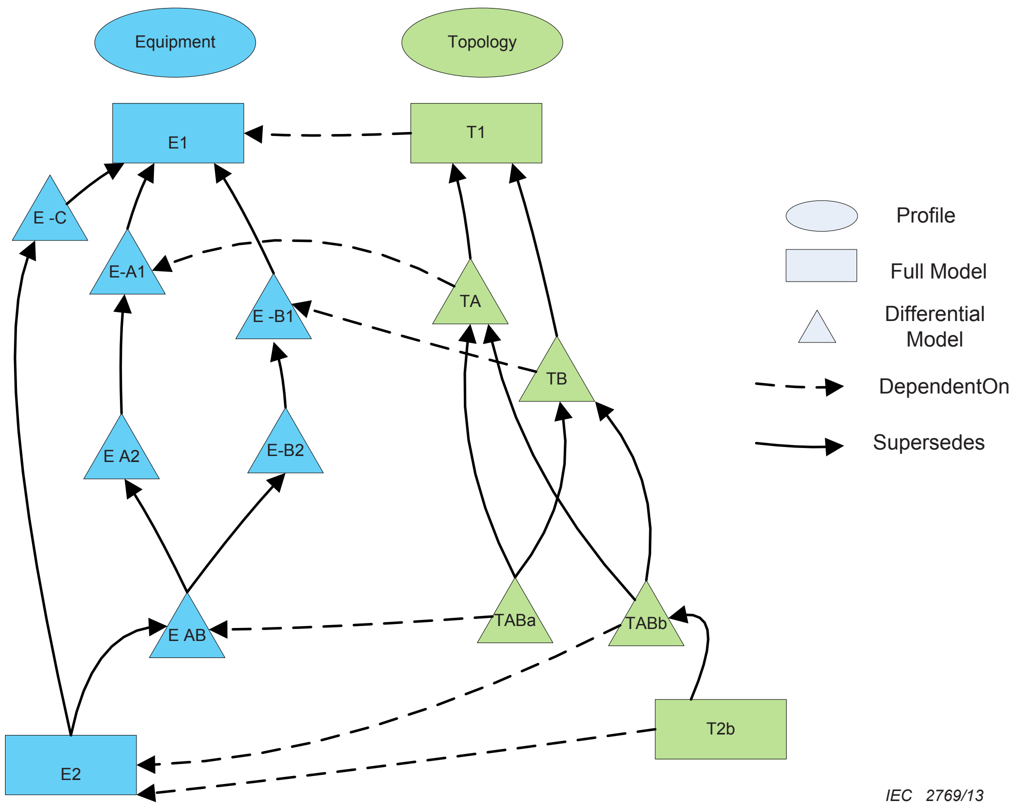

Figure 3 – Example work flow events with more dependencies

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.

5 Object identification

5.1 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 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, IEC 9834-8:2004 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. 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 letters are lower case.

An example of the URN form is shown below

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

5.2 About rdf:ID and rdf:about

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

- rdf:ID
- rdf:about

The use of rdf:ID and rdf:about has a specific meaning that does not align with their definition in RDF. The meanings are:

- an rdf:ID specifies the life of object globally, i.e. its creation or deletion.
- an rdf:about is a reference to an existing object.

5.3 CIMXML element identification

Object 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 5.1) 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 5.1 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”.
- rdf:about=”#_26cc8d71-3b7e-4cf8-8c93-8d9d557a4846”.

6 CIMXML format rules and conventions

6.1 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 3). 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.

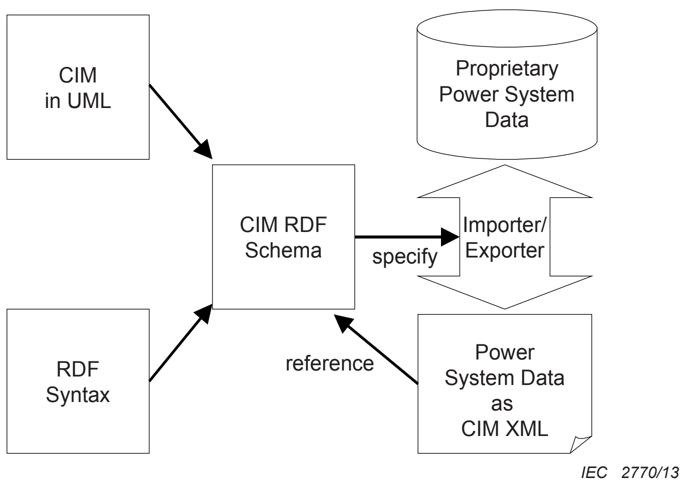

Figure 4 – CIMXML-based power system model exchange mechanism

6.2 Simplified RDF syntax

6.2.1 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.

The following subsections 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.

6.2.2 Notation

The simplified syntax is defined in the following section. Each kind of element is defined in a subsection 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 [3] for that information). The notation for the element model is as follows:

a) 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.
b) 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.
c) A comment 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.
d) All other text in the model is literal.

6.2.3 Syntax definition

6.2.3.1 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. The syntax focused in the example is indicated in bold.

1 Numbers in square brackets refer to the bibliography.

6.2.3.2 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#"`

6.2.3.3 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">
      <!-- Content: full-model (definition|description)* -->
</rdf:RDF>
```

```
<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:">
</rdf:RDF>
```

a) The element type is rdf:RDF.

b) The RDF namespace must be declared as http://www.w3.org/1999/02/22-rdf-syntax-ns#

c) 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.

d) Other namespaces may be declared.

e) The xml:base attribute shall always be present, refer to 5.3.

The RDF [3] grammar clause: 6.1.

6.2.3.4 FullModel element

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

Example:

```
<md:FullModel rdf:about="="#_26cc8d71-…">
   <md:Model.created>2008-12-24</md:Model.created>
      <md:Model.Supersedes rdf:resource="#_26cc8d71-a002-4c2b-bcf4-
7bc97430bf87"/>
```

```
     <md:Model.DependentOn rdf:resource=#_26cc8d71-a002-4c2b-bcf4-
7bc97430bf88"/>
     <md:Model.version>V32</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.

6.2.3.5 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>
```

a) 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 an rdf:about attribute.
b) 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.
c) 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.)

6.2.3.6 Description element

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

Example:

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

```
</rdf:Description>
```

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

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

c) The Description element is used only in difference models (refer to 6.2.4). It is never used in full models.

6.2.3.7 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, *classnam*e, 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 paragraph 6.2.4.7.4.

6.2.3.8 Literal-Property element

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

Example:

```
<cim:SynchronousMachine rdf:ID="_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>
```

a) The literal-property element introduces a property and a literal value applying to the enclosing resource.

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

c) The content *text* is any XML text with <, >, and & escaped representing the value of the property.

d) 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.

6.2.3.9 Compound-Property element

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

Example:

```
<cim:TimeSchedule>
      <cim:TimeSchedule.scheduleInterval>
            <cim:DateTimeInterval>
      <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>
```

6.2.3.10 Resource-Property element

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

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

b) 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.

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

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

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:ID="_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:ID="_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:ID="_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>
```

6.2.4 Syntax extension for difference model

6.2.4.1 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.

6.2.4.2 Example use case

To illustrate the difference document approach to handling incremental 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.

6.2.4.3 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 is responsible for performing the “cascading deletions” ,i.e, finding and deleting all other contained objects. Associations with deleted objects should also be deleted.
- 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.

6.2.4.4 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:

- Header statements, comprising statements about the difference model itself.
- 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 four groups can be empty.

The difference model itself is represented by a resource of type dm:DifferenceModel. It is conventional to use the URN of the model itself for this resource.

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#.

6.2.4.5 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.

6.2.4.6 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>
```

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.

```
<property parseType=”Statements”>
       <!-- Content: (definition|description)* -->
</property>
```

The content model of an element with rdf:parseType=”Statements” is the same as the content model of the rdf:RDF element.

The content generates the same RDF statements as if it appeared in an rdf:RDF element.

6.2.4.7 Difference model usage

6.2.4.7.1 General

The following cases explain the usage of the difference model.

6.2.4.7.2 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:
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-
452/EquipmentModel/1</md:Model.profile>           <dm:forwardDifferences
rdf:parseType="Statements "
       ">
       <!-- Add ACLineSegment ACLine_New1 -->
       <cim:ACLineSegment rdf:ID="_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:ID="_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:ID="_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:ID="_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:ID="_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:ID="_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:ID="_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:ID="_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:ID="_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>
```

6.2.4.7.3 Delete resource

The difference model contains for a given resource only a reverse difference 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 its 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.

Associations not considered a containment relation are cut from objects that that are in a cascading delete, e.g. if a ConnectivityNode is not affected by a delete but a ConductingEquipment connected to it is, then the association Terminal-ConductingEquipment is cut. This means that if the reference to a cut object is from an object that stays the reference from that objects shall be removed which means an update of the object that stays.

Delete elements shall have all its property elements included. The reason is that this enables reversing the delete operation and recreates 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">
```

```
      <!-- header properties omitted for brevity -->
      <!-- Delete Transformer -->
      <dm:reverseDifferences rdf:parseType="Statements" >
        <cim:PowerTransformer rdf:ID="_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>
```

6.2.4.7.4 Update resource

The difference model contains for a given resource forward difference and reverse difference 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">
      <!-- header properties omitted for brevity -->

      <!-- 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 (6.2.3.7) 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.

6.3 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.

a) 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.
b) 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.
c) Include single-valued properties rather than their many-valued inverse. For example, use `cim:Equipment.EquipmentContainer` and not `cim:EquipmentContainer.Contains_Equipments`.

d) 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.

e) 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.

f) Reason: Same reasons as for item c) above.

g) 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.

h) 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.

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

6.4 Representing new, deleted and changed objects as CIMXML elements

The following cases exist for identification of elements and how they appear in full or differential models

- New objects are represented by the definition element (refer to 6.2.3.5) identified by a rdf:ID attribute in full or differential models.
- Deleted objects are represented by the definition element (refer to 6.2.3.5) identified by a rdf:ID attribute in differential models.
- Changed objects are represented by the description element (refer to 6.2.3.6) identified by a rdf:about attribute in differential 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.

6.5 CIM RDF schema generation with CIM profile

IEC 61970-501 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 5.

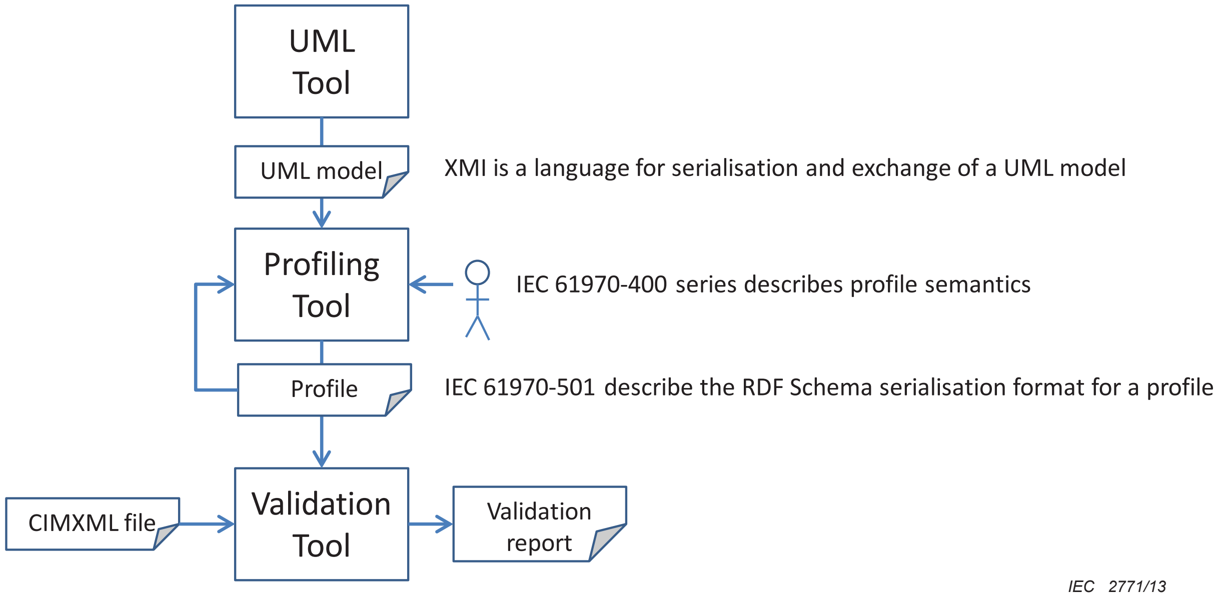

Figure 5 – Relations between UML, profile and CIMXML tools

6.6 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, a non-CIM attribute, *OriginalPO*, can be added 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:ID="_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.

6.7 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 [3] 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 6.2.3.6.

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 22 February 1999 http://www.w3.org/TR/REC-rdf-syntax, Ora Lassila, Ralph R. Swick

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; RFC2396

6. *Namespaces in XML*; Bray, Hollander, Layman eds, W3C Recommendation; http://www.w3.org/TR/1999/REC-xml-names-19990114

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. “A Strawman Unstriped Syntax for RDF in XML”, Tim Berners-Lee, November 1999, http://www.w3.org/DesignIssues/Syntax

10. “Simplified Syntax for RDF”, Sergey Melnik, November 1999, http://www-db.stanford.edu/~melnik/rdf/syntax.html

___

This page deliberately left blank

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

All the data, software and documentation set out in all British Standards and other BSI publications are the property of and copyrighted by BSI, or some person or entity that owns copyright in the information used (such as the international standardization bodies) and has formally licensed such information to BSI for commercial publication and use. Except as permitted under the Copyright, Designs and Patents Act 1988 no extract may be reproduced, stored in a retrieval system or transmitted in any form or by any means – electronic, photocopying, recording or otherwise – without prior written permission from BSI. Details and advice can be obtained from the Copyright & Licensing Department.

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

...making excellence a habit.™