# Dictionary & Library Model Specification

## RosettaNet Dictionary Architecture Program

**Rev. 0.25 — 21 April 2005** **Candidate Specification**

### Document information

| Info | Content |
|---|---|
| **Title** | Dictionary & Library Model Specification |
| **Short title (1 line)** | Dictionary & Library Model Specification |
| **Subtitle** | RosettaNet Dictionary Architecture Program |
| **Short subtitle (1 line)** | RosettaNet Dictionary Architecture Program |
| **Author(s)** | Petter Graff, Alfred Elkerbout |
| **Department** | Dictionary Architecture Foundational Program, RosettaNet |
| **Division** | RosettaNet |
| **Document ID** | RDA CS - April, 2005 |
| **Document type** | Candidate Specification |
| **Revision number** | 0.25 |
| **Status** | CANDIDATE SPECIFICATION |
| **Security status** | |
| **Keywords** | RosettaNet Dictionary Architecture, RDA, Dictionary & Library Model |
| **Abstract** | |

### Distribution information

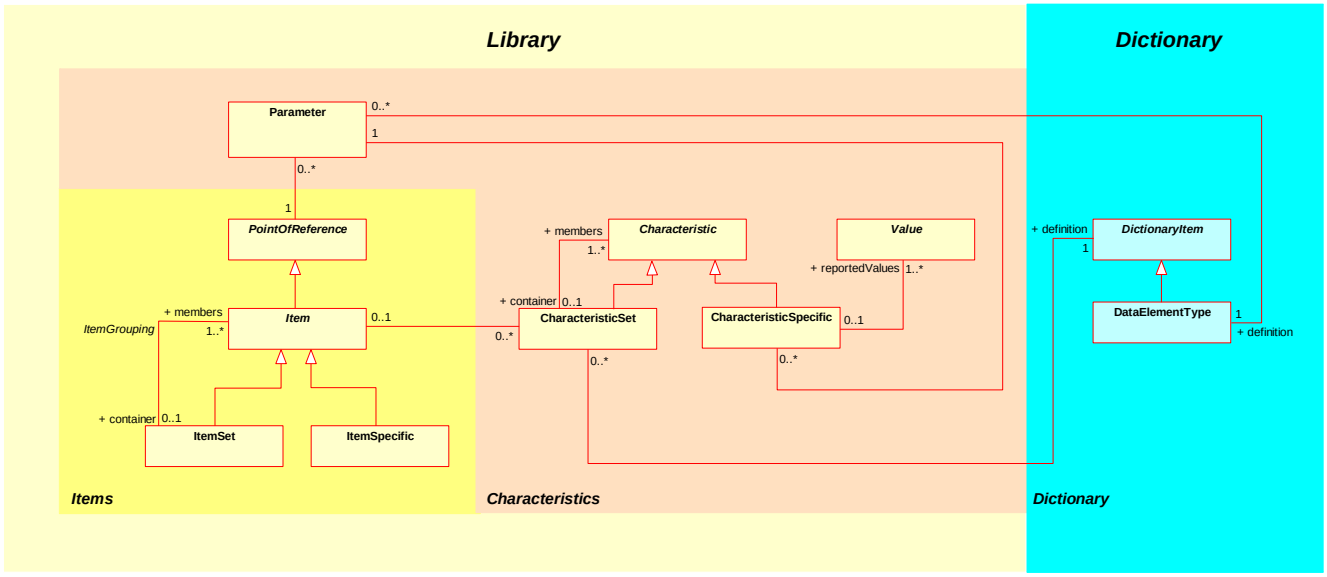
| Name | Department | Address |
|---|---|---|
| RDA Core Team | - | - |
| RDA Candidate Specification non-Core Team reviewers | - | - |
| | | |
| | | |

**Additional information:** This is the Candidate Specification for the RosettaNet Dictionary & Library Model Release 01.00.00

# RosettaNet Dictionary Architecture

## Dictionary & Library Model Specification

**Revision 0.25 — 21 April 2005** **Release 01.00.00**






ROSETTANET
Lingua franca for eBusiness

# 1. Preface

## 1.1 Navigating the PDF version of this specification

This specification is issued in Adobe Portable Document Format. For the convenience of the reader, this document has embedded hyperlinks which come in two flavors:

- **Implicit hyperlinks**: Section 13 “Tables” on page 254, Section 14 “Figures” on page 258 and Section 15 “Contents” on page 259 are fully hyperlinked into the document. However, these hyperlinks are not explicitly marked as such;
- **Explicit hyperlinks**: All other hyperlinks are shown “blue, underlined”. The references to “tables”, “figures” and “contents” shown directly above are examples of such explicit hyperlinks.

In addition to the embedded hyperlinks described above, PDF bookmarks have also been generated and can be used as a fully hyperlinked Table of Contents when viewing the PDF version of this specification in Adobe® Reader®.

## 1.2 Legal disclaimer

RosettaNet, Partner Interface Process, PIP and the RosettaNet logo are trademarks or registered trademarks of “RosettaNet,” a non-profit organization. All other product names and company logos mentioned herein are the trademarks of their respective owners. In the best effort, all terms mentioned in this document that are known to be trademarks or registered trademarks have been appropriately recognized in the first occurrence of the term.

## 1.3 Copyright



## 1.4 Trademarks

RosettaNet™, its members, officers, directors, employees, or agents shall not be liable for any injury, loss, damages, financial or otherwise, arising from, related to, or caused by the use of this document or the specifications herein, as well as associated guidelines and schemas. The use of said specifications shall constitute your express consent to the foregoing exculpation.

## 1.5 Related documents

There are no related documents. Referenced documents can be found in Section 12.4 “References” on page 253

## 1.6 Purpose

This specification is the human-readable counterpart to the computer-sensible UML model for the RosettaNet Dictionary Architecture.

## 1.7 Scope

This document provides the complete specification for the RosettaNet Dictionary Architecture 1.0.0. It is also intended to help the reader understand why the dictionary model was created, and how to read the specification. This document only describes the specification and does not contain any information regarding any future dictionaries that may be implemented from the specification or any PIP that may contain tags or structures based on the specification.

## 1.8 Intended audience

This document is intended to be used by RosettaNet PIP developers and by PIP implementers and backend system developers in RosettaNet partners. It assumes a good understanding of UML, PIP interactions, while a certain knowledge of the Electronic Component domain would be helpful.

## 1.9 Conformance statement

This document does not define any compliance requirements.

## 1.10 Document conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

## 1.11 Reference to normative specifications

The RosettaNet Dictionary Architecture Specification incorporates by reference certain normative standards or specifications from non-RosettaNet sources. These documents are referenced in the text and are listed in Section 12.4 “References” on page 253. This document does not restate material from the referenced document unless this document is changing a part of the referenced document. The reader is expected to refer to the relevant original source document for the text of referred specifications.

## 1.12 Normative content in this specification

The following sections in this specification are normative:

- All constraints expressed in Object Constraint Language (OCL) found in
  - Section 5 “RosettaNet Core” on page 15
  - Section 6 “Primitive Types” on page 52
  - Section 7 “RosettaNet EC Extensions” on page 73
- Section 9 “RosettaNet Dictionary & Library Model Diagrams” on page 97
- Section 10 “Data Dictionary” on page 106

All other content is supplementary and provided solely in support of understanding of the normative content.

## 1.13 Acknowledgements

This specification was developed by the RosettaNet Dictionary Architecture Core Team which at the time of publication consisted of the following individuals:

**Table 1: RosettaNet Dictionary Architecture Core Team**

| Name | Affiliation |
|---|---|
| Bill Breden | Texas Instruments |
| Addie Dijkstra | Philips Semiconductors |
| Alfred Elkerbout | Philips Semiconductors |
| Ted Haas | Independent consultant |
| Sharky Jacaway | Micron Technology [1] |
| John Mackin | RosettaNet Japan |
| Jasper Norris | Freescale Semiconductor |
| Nikola Stojanovic | RosettaNet |
| Mike Young | Agilent Technologies |

[1] Currently at Albertsons

## 1.14 Document version history

**Table 2: Version history**

| Version | Version date | Description |
|---|---|---|
| 01.00.00 | t.b.d. | Final Specification |
| V3 [0.25] | 21 Apr. 2005 | Candidate Specification |
| V2 [0.22] | 08 Apr. 2005 | Team review version |
| V1 [0.8] | 23 Sep. 2004 | Initial draft |

# 2. Introduction

## 2.1 Business context

Businesses have an ever increasing need to improve the efficiency and fitness-for-use of their core processes of which sharing information with trading partners is a key component. The pressure to reduce cycle time and operational costs represents a powerful change agent in the drive to reduce dependency on traditional forms of communication with a focus on human-sensible documents and increase the use of computer-sensible information that may flow directly into Trading Partner systems. RosettaNet was created to address this problem for a wide variety of business processes.

Nowhere is this problem more pressing than in the high-tech industry with its rapidly evolving product portfolios and ever decreasing product life cycles, resulting in a recognition that fundamental changes are needed in the way product and other business information is created, managed and delivered.

The RosettaNet Dictionary Architecture is the culmination - so far - of an effort to address this problem, that started in the early 1990's and as such represents the refinement of ideas and concepts to which many individuals and organizations have contributed. The next section briefly outlines the history of that effort.

## 2.2 The search for a solution

Early attempts to address this problem - by a group of semiconductor suppliers[1] in the early 1990's - were still very much document-centric. Although the standards developed were based on SGML[2], they were initially centered on electronic component product specifications or "datasheets". Later specifications developed by this group and its successor also addressed more computer-sensible types of information such as semiconductor timing and pinning information.

In 1999 it became clear that the document-centric approach was no longer meeting the industry's needs and a more object-centric approach was needed to enable the deployment of seamless, computer-sensible B2B processes. This resulted in a program called ECIX[3] II which in 2001 merged with RosettaNet. The domain model developed by ECIX eventually became the basis for the RosettaNet Dictionary & Library Model. This model builds on a rich legacy of over a decade of developing standards and specifications for computer-sensible product information.

One additional fact is worth mentioning here: the IEC 61360 standard. IEC 61360 (Titled "Standard data element types with associated classification scheme for electric components") has been a major source of knowledge in the development of the RosettaNet Dictionary Architecture. Although the RDA's scope is significantly broader - both content and process wise - than the IEC 61360 scope and both are using very different information modeling languages, many of IEC 61360 basic premises and principles have been instrumental in the development of the RDA.

1. Intel, National Semiconductor, Philips Semiconductors and Texas Instruments formed The Pinnacles Component Information Standard initiative [PCIS]
2. Standard Generalized Markup Language. The Web and XML were still in the future and CD-ROMS were just becoming a mainstream distribution channel.
3. Electronic Component Information Exchange which at the time consisted of Agilent, Hitachi, Intel, Lucent, Mentor, Motorola, Philips Semiconductors, ST Microelectronics and Texas Instruments.

## 2.3 RDA Foundational Program

The RDA Foundational Program was chartered in 2003 to develop an architecture for a scalable, unified, single RosettaNet Dictionary (RND), addressing the needs of the PIP® user community, solution providers and RosettaNet PIP Engineering.

Its value proposition is to increase B2B capabilities and increase trading network size by:

- More rapid deployment to other supply chains, business processes and information domains enabled by this flexible and extensible architecture;
- Improved interoperability via generation of higher-quality, consistent PIPs by referencing the dictionary at PIP-generation time;
- Reduced development and maintenance effort through the increased specificity for computer-sensible, reusable property definitions and other PIP building blocks.

The completion of the RDA specification represents the completion of the RDA Foundational Program's design phase.

## 2.4 RosettaNet Dictionary Architecture intended use

Before we discuss the RDA's use it should be made clear that the use of the term "RDA" in this section is somewhat of a misnomer. The intended use of the RDA is to provide an architecture for the development of dictionaries and libraries. However, it is the deployment and subsequent use of the unified RosettaNet Dictionary to create libraries, message structures and message payloads that will yield actual business benefits. For convenience we'll use the term "RDA" here, realizing this is a placeholder for a deployed, richly populated and well maintained unified RosettaNet Dictionary (RND).

The RDA is broad in scope and can be used in multiple ways. At the highest level of abstraction, the RND will support the following increasingly sophisticated scenarios:

- A human readable repository of shared knowledge. The RDA's structure and precision, supplemented by a maintenance process and organization, make it very suitable to create a unified, authoritative source of information and knowledge.
- A computer-sensible dictionary from which to compile PIPs or other message structures, resulting in very high consistency and avoiding duplication of content.
- A computer-sensible dictionary to be used at interchange time in support of message interpretation and validation (so called "reflective" PIPs such as 2A10)

We'd like to emphasize that the generation of PIPs from a dictionary is not limited to a single message paradigm such as "explicit" (e.g. 3A4) or reflective (e.g. 2A10). RDA generated PIPs may be DTD or schema based.

Finally, the RDA makes it possible to dynamically configure tools from the dictionary. Examples of this include populating pick lists in a user interface and automatically providing correct Units.

# 3. The RosettaNet Dictionary Architecture

This section will introduce the RDA's main features. First form a business and high-level architectural point of view and secondly from a detailed "fitness-for-use" point of view.

## 3.1 Key business and architectural aspects

This section will introduce the main business and architectural aspects that enable the RDA to achieve its value proposition. Some of these have been mentioned in earlier sections and will be expanded upon while others are introduced for the first time. These aspects are divided into the following groups: scalability and extensibility, efficiency, control, unification and interoperability. We'll discuss each in turn.

### 3.1.1 Scalable and extensible

The RDA has been developed with scalability and extensibility in mind. These aspects address multiple dimensions There's the scalability needed to expand to new business domains, enabled by a dictionary that is not limited to a specific domain. There's the scalability resulting from the RDA's ability to support multiple message paradigms, from fully explicit - every dictionary item has its own named tag and metadata in exchanged as well - to fully reflective - only dictionary-ID/value pairs are exchanged, with the dictionary-IDs referring to metadata stored in the dictionary.

The RDA supports private extensions through dictionary addenda. This is dictionary content that for reasons of trading partner information security or speed of development is not (yet) available in the community agreed RND. Using the RDA's structure, trading partners can easily create their own content with minimal impact on existing processes or tools.

### 3.1.2 Efficient

Efficiency, just as with scalability and extensibility also has many faces. Because the well defined structure and precision of the RDA will make it feasible and affordable to generate PIPs, trading partners can look forward to a reduced incremental development effort for addressing a new business process compared to the traditionally hand-crafted PIPs.

Another major contributor to increased efficiency and reduced cycle times is the inherently distributed nature of dictionary content development. Extending the dictionary to address a new business process or even a new business domain can be done by dividing the work across multiple individuals and organizations. Because everything fits into the same framework, there's less need to start a monolithic PIP development effort with its much larger deliverables and the need to go through a more complex governance and approval process.

It is inherently faster, simpler and cheaper to approve new dictionary content vs. approving and validating a new PIP.

### 3.1.3 Controlled

An increased level of control is enabled from a content as well as from a process or governance point of view.

A populated RND represents a controlled vocabulary (content view). This control is enabled by high precision, unambiguous information capture and a high granularity of stored information. Concrete examples are discussed in Section 3.2 “Basic capabilities” on page 8.

From a configuration and life cycle management point of view, the RDA’s built-in structures enable a strictly governed but highly transparent dictionary content development, validation and maintenance process[4].

### 3.1.4 Unified

The RDA will enable deployment of a single, unified dictionary, effectively creating a single repository of shared knowledge for RosettaNet’s trading partners and replacing multiple, disjoint dictionaries. This will remove a major source of inconsistencies and overlapping content.

The RDA has been designed to be domain neutral. Although the emphasis of the developers has been on product information, great care has been taken to ensure applicability in other domains.

The RDA has been designed to be multi-lingual, supporting translated attributes where relevant. This makes it possible to store captured knowledge, optionally translated into other languages, in a single repository.

### 3.1.5 Interoperable

No single dictionary can or should attempt to capture all of the worlds knowledge. The ability to access content (royalty free) from other domain dictionaries is therefore essential. To address this requirement the RDA has been designed with the requirements of the Open Interoperable Domain Dictionary Initiative [OIDDI] in mind. For more information on OIDDI see Section 12.3 “OIDDI” on page 252.



4. A dictionary development, validation and maintenance process it outside the scope of this specification but will be addressed in the implementation phase of the RDA Foundational Program.

## 3.2 Basic capabilities

Whereas the aspects introduced in Section 3.1 on page 6 represent general business and architectural features, success or failure of the RDA lies in its ability to support high-fidelity information capture, meeting the needs of trading partners.

The following sections present a non-exhaustive list of these basic capabilities.

### 3.2.1 A careful separation of two categories of information

The RDA has a strict separation between data and metadata:

- Static information that does not change across multiple instances of dialogs between trading partners (called **metadata** in this specification).
  An example of such information is the definition of “supply voltage” and the fact that “supply voltage” is measured in V(olts).
- Dynamic information that may change with every dialog between trading partners (called **library data** in this specification).
  Example of such information is “Product ABC123 has a supply voltage of 3.3”)

The metadata represents a stable and controlled vocabulary. This allows the trading partners to only interchange the essential information which should enable a more efficient interchange protocol minimizing the required bandwidth.

### 3.2.2 Capture item properties and property values and their definitions

The most basic capability the RDA supports is the capability to capture properties of items and report values for those properties. To do this unambiguously and precisely the dictionary is used to define the metadata for these properties. Items can be many things including:

- Products, services and processes;
- Objects related to products, services or processes such as datasheets, manuals, software, packaging, contractual terms and pricing.

These can be enriched with conditions, units and symbols and further specified with ValueQualifiers such as “min”, “typ”, “max”.

### 3.2.3 Group information in multiple ways

To increase the usability and manageability of both data and metadata it is often useful to create information groups. The RDA supports multiple ways of grouping:

- Groups of items to allow for the reporting of information applicable to a set of items;
- Groups of DictionaryItems for convenience (e.g. improved manageability);
- Groups of DictionaryItems to define information structure or semantics.

Many of these can be nested at arbitrary levels, allowing for the creation of sophisticated structures.

### 3.2.4 Fully define DictionaryItems

To support the RDA’s objectives, each DictionaryItem comes with a rich set of attributes, needed to fully specify a DictionaryItem. A list of categories of attributes includes:

- Globally unique identifiers;
- Definitions, notes and remarks;
- Data types;
- Value lists;
- Dependent conditions;
- Dictionary life cycle information

### 3.2.5 Values and expressions

To allow for a high degree of computer-sensibility, the RDA supports multiple ways of capturing values. In addition to basic data types such as string or integer, the RDA supports:

- Ranges and tolerances;
- Values dependent on the value of another parameter such as "0.5 * $\mathbf{V_{CC}}$"
- Parameters used in expressions such as "$\mathbf{V_I}$ < -0.5 V or $\mathbf{V_I}$ > 6.5 V"

### 3.2.6 Flexible tool support

The dictionary provides the metadata required to map interchange types to backend information systems types local to a trading partner.

The RDA provides a complete language for what can be changed and what is to remain static. It is possible to build software solutions that at most require reconfiguration as the content of the interchange evolves.

### 3.2.7 Local Interchange Rules

The model allows for bilateral extension or restriction of metadata to be used for interchanges. This is an essential feature that allows two or more trading partners to interchange information in a structured and semantically well defined way without having to submit their metadata to a standard organization. This prevents unnecessary delays and enables privacy without loss of semantics.

### 3.2.8 Other capabilities

Other capabilities worth mentioning are:

- Link any item to any other item or group of items and define the semantics of the link in the dictionary. This capability has many uses, e.g. define "kits", show a product's replacement and the linking of legal disclaimers to items.
- Report information on an item that can exist or operate in multiple modes, e.g. a DVD player that has different speeds depending on whether it is in "read" or "write" mode.

The metadata defines what kind of information can be interchanged between the trading partners. It also provides the semantic for the interchanged information. This will minimize the ambiguity of the dialogs between trading partners.

# 4. The RosettaNet Dictionary & Library Model specification

## 4.1 Introducing the Dictionary & Library Model

This document provides a formal model of the business domain as it relates to the interchange of computer-sensible information between trading partners. The model is based on numerous interviews and workshops with business domain experts and standards efforts such as ECIX, IEC and RosettaNet.

The model is prescriptive in terms of what information structure in the domain is relevant for interchanging information. It contains a number of invariant business rules for structure of information that future applications and business processes are likely to support and adhere to.

The model has been primarily tested for the interchange of descriptions of products. However, it is more than likely that the model can be used for other domains. The model is taking into account the variability of product information and is intended to support current and future product specifications. The model provides invariants where the business rules seem immutable and provides structure for defining mutable business rules.

The model contains only structural specification for business information (a.k.a. an information model). No attempt has been made to define applicable business processes or locality of information. We expect current and future business processes to conform to the information model specified. However, it is very likely that new information structure is required to support the temporal stages of creating the information. It would be logical to create another version of the model, extending the model, to support for these upcoming temporal invariants, often called **Analysis Model**.

The analysis model would also support:

- Temporal stages of the creation process
- Approval processes
- Locality of information
- Staging of implementation
- and more

In addition to the missing temporal invariant, the RosettaNet Dictionary & Library Model does not provide any information about locality of data, e.g. where data is stored.

### 4.1.1 Model packages

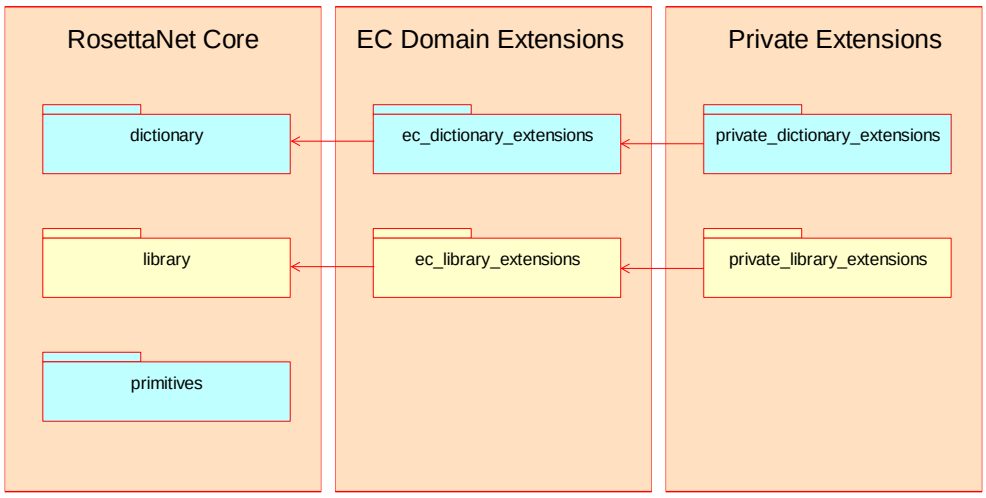



**Fig 1. RDA model packages**

The RosettaNet Dictionary & Library Model is modular and consists of several packages:

- **RosettaNet Core** This module represents the non-business domain specific, immutable core of the model and consists of the following packages:
  - **dictionary** defining metadata
  - **library** defining instance data
  - **primitives** defining basic data types
- **EC Domain extensions**
  - **ec_dictionary extensions** extending the dictionary for the electronic component domain
  - **ec_library extensions** extending the library for the electronic component
- **Private extensions**
  - **private_dictionary extensions** extending the dictionary with private extensions
  - **private_library extensions** extending the library with private extensions

The packages for private extensions are empty placeholders; this specification does not define any private extensions. Figure 1 on page 11 shows the RDA packages and their relations.

## 4.2 Modeling conventions

The model is using standard UML where ever possible. A description of notation and purpose can be found in [UML].

We have selected to use some non-standard modeling notation to increase the readability of the model. These are always easily mapped to standard UML notation.

### 4.2.1 Use of colors

We use colors instead of stereotypes to indicate meta-level object types versus instance level object types.

The colors used:

- **Blue** for meta level object
- **Yellow** for instance level objects
- **Green border** for association types

### 4.2.2 Use of Composition

We use UML composition in some of the class diagrams. The UML composition construct has weak semantics, prompting us to define our use of composition here.

We use the UML composition icon to represent a relationship between two types where the components of the composite have a lifetime dependency on the composite. That is, if the composite is deleted so are the components. This is the only semantic implied. Many of the commonly used interpretations DO NOT apply. Listed here are a few of those

- We do NOT imply that the components can not exist without the presence of the composite
- We do NOT imply that the components are only accessible if navigated to through composite

We only use composite in relationship between primitive types.

### 4.2.3 Use of columns for instance diagrams

To increase the readability of an instance diagram, we arrange objects of the same kind in columns. This extends the UML notation, but does not violate it. Since the location of the object now indicates types, we do not show the type information inside the object.

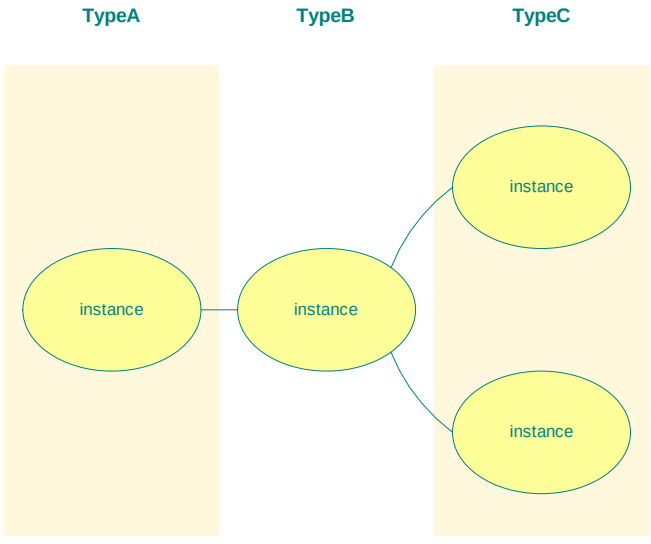



**Fig 2. Example of instance diagram using columns**

## 4.3 Immutable Business Rules

This document contains a description of the type model. The type model describes the information structure that systems must conform to. Consequently, the type model also contains major immutable business rules for how the business information is structured.

## 4.4 Specification walk through

This section will describe how the remainder of this document is structured, providing a high-level walk-through.

### 4.4.1 Overview

Section 5.1 “Overview” on page 15 introduces the three major areas the RosettaNet Dictionary & Library Model is describing:

- **Items** and **ItemSets** - concrete things that a company needs to keep information on;
- **Characteristics** and **CharacteristicSets** - the actual detailed data being kept on those **Items**;
- **DictonaryItems and their sets** - globally agreed standard ways to identify, structure, and type those Items, ItemSets, Characteristics, and CharacteristicSets.

### 4.4.2 Item view

Section 5.2 “Item View” on page 16 provides more extensive information on Items and ItemSets. The section ends with a discussion of the application of the concepts in the section and the constraints on those concepts.

### 4.4.3 Dictionary view

Section 5.3 “Dictionary View” on page 21 provides more extensive information on DictionaryItems and their sets. The section ends with a discussion of the application of the concepts in the section and the constraints on those concepts

### 4.4.4 Characteristic view

Section 5.4 “Characteristic View” on page 46 provides more extensive information on Characteristics and their sets. The section ends with a discussion of the application of the concepts in the section and the constraints on those concepts.

### 4.4.5 Primitive data types

Section 6.1 “Introduction to primitive types” on page 52 provides more extensive information on the primitive data types used in the model. Each section in this section ends with a discussion of the application of the concepts in the section and the constraints on those concepts.

### 4.4.6 RosettaNet Electronic Components (EC) Extensions

Section 7 “RosettaNet EC Extensions” on page 73 explains a modeling issue that seems to apply only to certain sophisticated IC chips, in which the signals applied to specific pins may change their value and meaning depending on the operational mode of the chip. This situation is not common to most manufactured products and hence is seen as a necessary extension of the core RosettaNet data requirements. This section shows how the core concepts can be easily extended to cover this reporting issue and thus suggests

that the model is highly “future proof.” It also may provide a useful methodology for companies facing unique internal requirements to describe data not normally reported on by other companies.

### 4.4.7 Annotated examples

The explanations in the previous sections have been quite abstract and perhaps difficult to apply to actual situations. Section 8 “Annotated examples” on page 81 gives concrete examples of applying the concepts of the previous sections to facilitate understanding of how these concepts can be used in a company.

### 4.4.8 Complete model diagrams

Section 9 “RosettaNet Dictionary & Library Model Diagrams” on page 97 provides the full set of model diagrams.

### 4.4.9 The data dictionary

Section 10 “Data Dictionary” on page 106 describes in detail all classes, attributes and associations depicted in the model.

### 4.4.10 Appendices

Section 12 “Appendix” on page 249 provides design rationale, documents known limitations of the RosettaNet Dictionary & Library Model, explains the Open Interoperable Domain Dictionary Initiative [OIDDI] and provides references to standards and documents used in the development of the model.

## 4.5 OIDDI

For an introduction to the Open Interoperable Domain Dictionary Initiative see Section 12.3 “OIDDI” on page 252. The RDA Specification has been developed with inter-dictionary interoperability as defined by OIDDI. The extent of compliance still has to be documented. The structure, format and representation of **DictionaryItem** identifiers - OIDDI prescribes globally unique identifiers - is important to enable interoperability. For details on the use and definition of **DictionaryItem** identifiers see Section 5.3.3.4 “Uniqueness of DictionaryItems” on page 27, Section 5.3.4.2 “Dictionary Item Uniqueness” on page 44 and the relevant definitions in Section 10 “Data Dictionary” on page 106.

# 5. RosettaNet Core

## 5.1 Overview

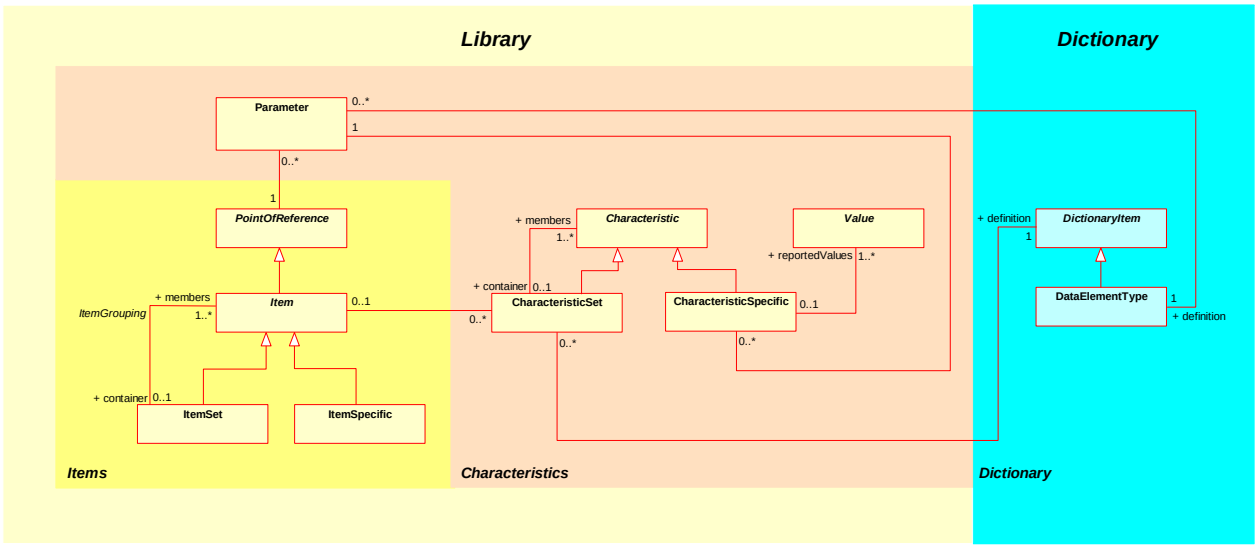



**Fig 3. RosettaNet Dictionary & Library Model overview**

Figure 3 shows an abstract type model for all the structural requirements. The overview model is correct but not complete. There are three major areas - listed at the bottom of Figure 3 - that make up the core of the model.

- **Items**

  A hierarchical structure used to represent artifacts that we want to track information on.

- **Characteristics**

  Characteristics are individual pieces of information about the items.

- **Dictionary**

  The dictionary contains the metadata for characteristics

We often make an even coarser partition and group the characteristics and items together into what we call **Library**.

In addition, a number of EC domain specific extensions are described: **Terminus**, **Mode** and **Connection** and their associations.

Because of the size of the model, we describe the model view by view.

All the model elements found in any of the type models are defined in Section 10 “Data Dictionary” on page 106. A brief summary of the types as defined in the Data Dictionary is repeated as an introduction to each view for convenience.

## 5.2 Item View

### 5.2.1 Type model

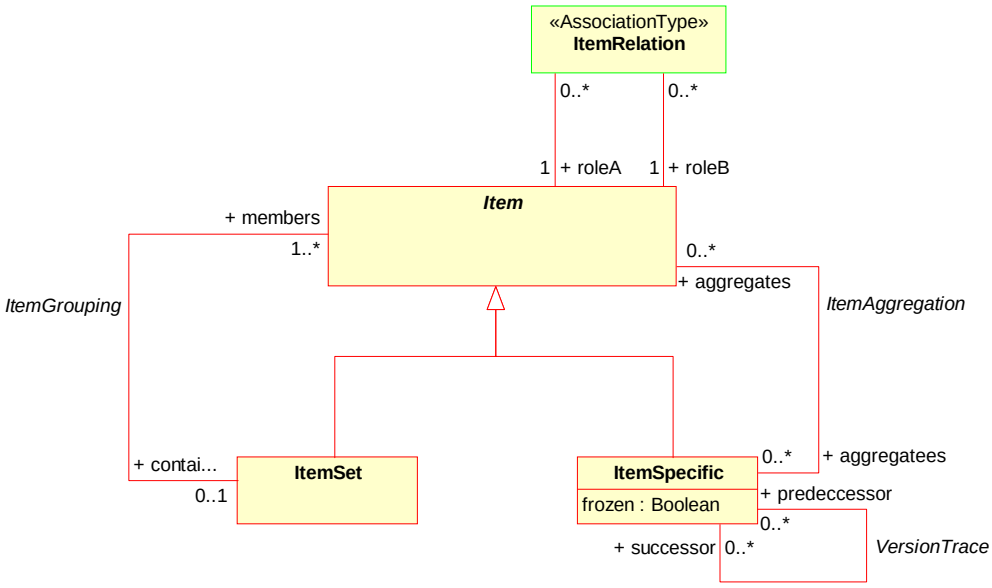



Fig 4. Type model for the “Item” view

### 5.2.2 Definition of CLASSES in the “Item” view

Table 3: CLASS: Item

| Field | Value |
|---|---|
| Name | **Item** |
| Definition | An object type for which **Characteristics** can be specified |
| Note | An **Item** is the main “thing” that information (**Characteristics**, including **Conditions** and **Values**) is reported on |
| Remark | The **Item** type is an abstract model type used to make visible the commonality between **ItemSpecific** and **ItemSet**. Here used to show that instances of both **ItemSpecific** and **ItemSet** can have **ItemRelation**, **Parameters** and **CharacteristicSets**. |

Table 4: CLASS: ItemSet

| Field | Value |
|---|---|
| Name | **ItemSet** |
| Definition | A set of **Items** sharing the same **Characteristics** |
| Note | - |
| Remark | - |

**Table 5: CLASS: ItemSpecific**

| Field | Value |
|---|---|
| Name | **ItemSpecific** |
| Definition | The most atomic object type for which **Characteristics** can be specified |
| Note | - |
| Remark | - |

**Table 6: CLASS: ItemRelation**

| Field | Value |
|---|---|
| Name | **ItemRelation** |
| Definition | A relationship between two **Items** for which the semantics are specified by a **RelationType** |
| Note | - |
| Remark | Every instance of an **ItemRelation** is associated with an instance of a **RelationType** (information stored in dictionaries). |

### 5.2.3 Discussion

**Items** are anything about which one or more characteristics might be collected. For example, a computer, a memory chip, a railroad boxcar, a customer's address, a computer operating system, an element in the periodic table, etc. **Items** can also refer to controlled documents, simulation models or audio-visual content or illustrations.

It must be possible to organize **Items** in a tree structure. We call this the item inheritance hierarchy. The main purpose of the item inheritance tree is to capture the similarities of **Items** in a way that is easy to maintain. The advantages of using an inheritance tree are manyfold:

- An item specifier can attach **Characteristics** to a group of Items by attaching the Characteristic to a parent ItemSet. Attaching a Characteristic to a parent ItemSet means that every Item that is a member of the ItemSet inherits this Characteristic.
- Navigation through the item catalog becomes simple
- Maintenance of the item inheritance tree becomes simpler
- Information redundancy is kept to a minimum

The following diagram shows an example of a product classification tree[5] which is a specific use of an item inheritance tree:

5. Note that although the model allows for constructing product classification trees, it does not mandate any particular classification of products. This is left to dictionary and library implementers.

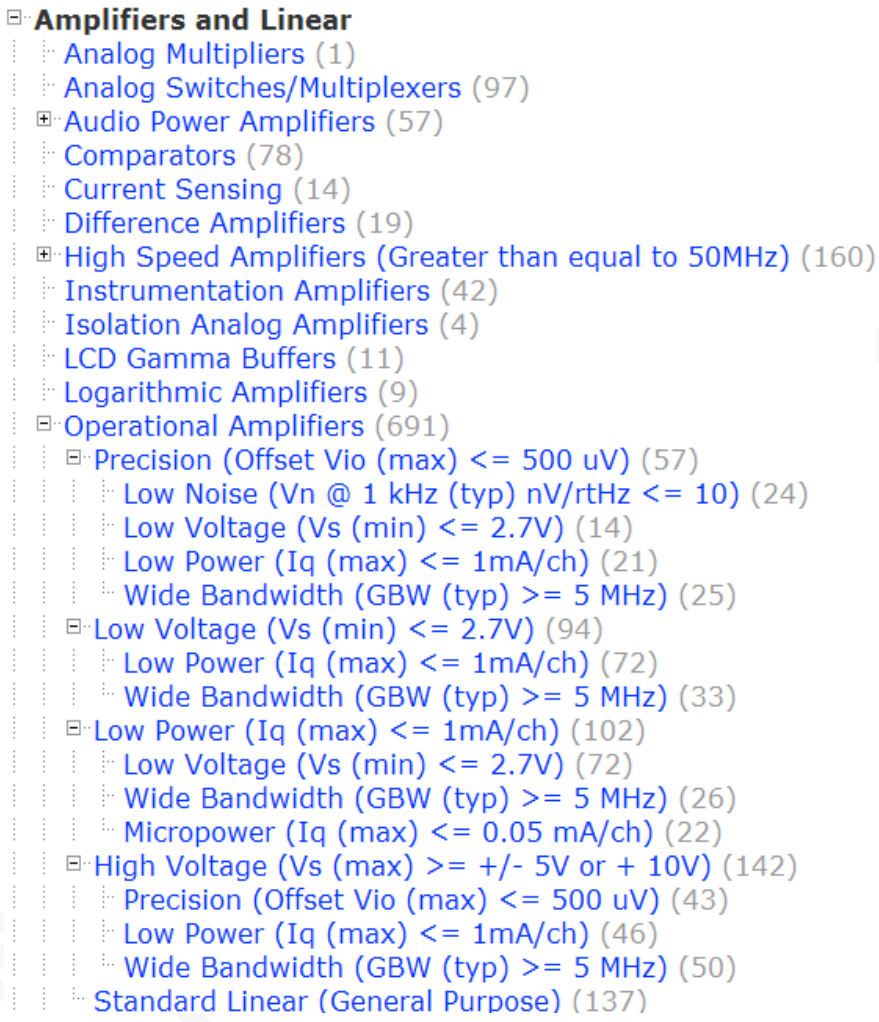



**Fig 5. Example of a product classification tree**

Applications should be able to cross branches of the item inheritance tree by using a concept we call assembly. Assembly enables the item specifier to reuse complete fragments of specifications.

There are two kinds of **Items** in the model:

- **ItemSpecific**. Defines a fully specified item.
- **ItemSet**. Defines a set of items, e.g., transistors, optical storage devices, printer supplies, etc.

Figure 6 “Items, ItemSets and ItemSpecific” on page 19 shows these types in another example of a product classification tree.


**Fig 6. Items, ItemSets and ItemSpecific**

The two kinds have a common supertype **Item**, hence by **Item** we mean either **ItemSpecific** or **ItemSet**.

The purpose of the **ItemSet** is to associate a set of **Characteristics** to a set of related **Items**. For example, both memory and diodes are Electronic Components, have an “operating temperature” characteristic, but not necessarily the **same** temperatures, and have "supply voltage" characteristics, but not necessarily the **same** supply voltages.

The **ItemGrouping** association defines the item hierarchy. The association defines that an instance of an **ItemSet** groups together many **Items**. The grouped **Items** can either be **ItemSpecifics** or **ItemSets**. This allows us an infinite depth classification tree. Notice that the cardinality from **Item** to **ItemSet** is max 1 (one). That means that the classification follows a tree structure (a.k.a., single inheritance).

The **ItemAggregation** association defines a way for the various branches of an item inheritance tree to be used for composition for specification of other **Items**. We foresee this to be an advantage to reuse specifications related to:

- Function
- Packages
- Technology
- Organization
- Test software
- and more...

The **VersionTrace** association is used to keep a chain of versioned specifications of the same specific **Item**. Every time a new version of an item specification is created, we will create a new instance of **ItemSpecific** to store the old specification. We will also link the new and old specification together with use of this association.

The **ItemRelation** defines an arbitrary way of linking various **Items** together. The **RelationType** defines the semantic of this relationship.

### 5.2.4 Constraints

#### 5.2.4.1 Item Relationship For Two Items

An **ItemRelation** must be between two different **Items**.

**Table 7: OCL expression for ItemRelationForTwoItems**

```
context ItemRelation inv ItemRelationForTwoItems:
self.roleA <> self.roleB
```

#### 5.2.4.2 ItemSet Self Exclusion

An instance of **ItemSet** can not have itself as a child (directly or indirectly). In other words, the item inheritance tree must be acyclic.

**Table 8: OCL expression for ItemSetSelfExclusion**

```
function allChildren( is: ItemSet ) =
   is.members>union( is.members->collect( c | allChildren(c) ) )
context ItemSet inv ItemSetSelfExclusion:
   allChildren(self)->excludes( self )
```

#### 5.2.4.3 Item Aggregation Self Exclusion

An instance of an **Item** can not aggregate itself

**Table 9: OCL expression for ItemAggregationSelfExclusion**

```
function allAggregates( is: ItemSpecific) =
  is.aggregates->union(is.aggregates->collect(a|lAggregates(a) ))
context ItemSpecific inv ItemAggregationSelfExclusion:
   allAggregates(self)->excludes( self )
```

#### 5.2.4.4 Item Version Replacement

An specific version of an **Item** can not be replaced by the same version of that **Item**. This applies recursively through the chain of successors/predecessors.

**Table 10: OCL expression for ItemVersionReplacement**

```
function allSuccessors( is: ItemSpecific) =
   is.successors->union (is.successors->collect( s | allSuccessors(s)) :
context ProductSpecific inv ProductVersionReplacement:
   allSuccessors(self)->excludes( self )
```

## 5.3 Dictionary View

### 5.3.1 Type model

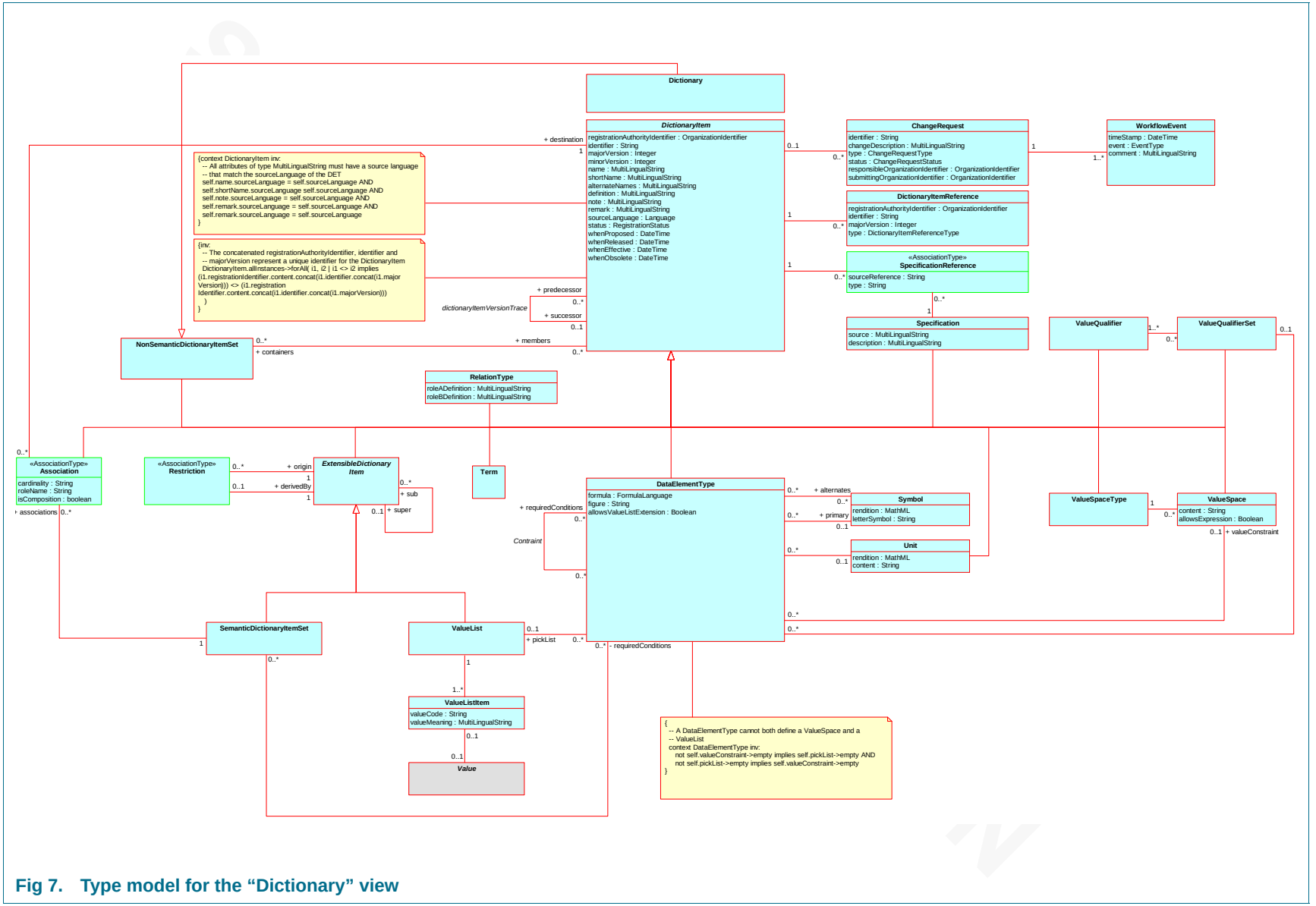



**Fig 7. Type model for the “Dictionary” view**

### 5.3.2 Definition of CLASSES in the “Dictionary” view

**Table 11: CLASS: DictionaryItem**

| Field | Value |
|---|---|
| Name | DictionaryItem |
| Definition | An abstract SuperType defining the attributes common to all entries in a Dictionary |
| Note | - |
| Remark | - |

**Table 12: CLASS: ChangeRequest**

| Field | Value |
|---|---|
| Name | ChangeRequest |
| Definition | Specification of semantic and administrative aspects belonging to a change |
| Note | - |
| Remark | - |

**Table 13: CLASS: WorkflowEvent**

| Field | Value |
|---|---|
| Name | WorkflowEvent |
| Definition | Records the history of a ChangeRequest |
| Note | - |
| Remark | An Event can occur multiple times to report additional information in a comment |

**Table 14: CLASS: DataElementType**

| Field | Value |
|---|---|
| Name | DataElementType |
| Definition | Unit of data for which the identification, description, representation and permissible values are specified by means of a set of attributes |
| Note | The purpose of DataElementTypes are to provide a shared definition of reported Parameters and their associated Characteristics. |
| Remark | - |

**Table 15: CLASS: Association**

| Field | Value |
|---|---|
| Name | Association |
| Definition | A semantic relationship between a SemanticDictionaryItemSet and member DictionaryItems. |
| Note | The Association allows the dictionary specifier to define constraints on the library. |
| Remark | - |

**Table 16: CLASS: ValueListItem**

| Field | Value |
|---|---|
| Name | ValueListItem |
| Definition | Representation of a permissible instance of a DataElementType as an element of a ValueList |
| Note | - |
| Remark | - |

**Table 17: CLASS: SpecificationReference**

| Field | Value |
|---|---|
| Name | SpecificationReference |
| Definition | Reference to a Specification that is a source for or provides context to the semantic parts of a DictionaryItem |
| Note | Allows a DictionaryItem to refer to a Specification and provide context (both the type of reference and a description) for the reference. |
| Remark | - |

**Table 18: CLASS: Specification**

| Field | Value |
|---|---|
| Name | Specification |
| Definition | A Specification external to the Dictionary, providing additional context to a DictionaryItem |
| Note | Defines context for a DictionaryItem's reference to a Specification. This type allows a DictionaryItem to refer to an external specification (Specification) and provide context (both the kind of reference and some description) for the reference. |
| Remark | - |

**Table 19: CLASS: RelationType**

| Field | Value |
|---|---|
| Name | RelationType |
| Definition | An association between two Items |
| Note | - |
| Remark | - |

**Table 20: CLASS: Symbol**

| Field | Value |
|---|---|
| Name | Symbol |
| Definition | Mark or characters used as a sign for representing a DataElementType |
| Note | - |
| Remark | - |

**Table 21: CLASS: ValueSpace**

| Field | Value |
|---|---|
| Name | ValueSpace |
| Definition | A restriction for reported Values on Characteristics for a DataElementType. |
| Note | - |
| Remark | The ValueSpace is defined by a language identified by the associated ValueSpaceType. A typical scenario would be to restrict the Values by XML Schema types. |

**Table 22: CLASS: Unit**

| Field | Value |
|---|---|
| Name | Unit |
| Definition | Prescription of the Unit in which the Value of a quantitative DataElementType shall be expressed |
| Note | - |
| Remark | 1) Preference shall be given to SI Units as defined in ISO 31 and to Units as listed in Annex A of IEC 61360-1:2002-02<br>2) For encoding guidelines see: Units in MathML: http://www.w3.org/TR/mathml-units/<br>3) Only base Units, i.e. Units without prefixes shall be used<br>4) Units shall be specified for quantitative DataElementTypes |

**Table 23: CLASS: ValueList**

| Field | Value |
|---|---|
| Name | ValueList |
| Definition | A set specifying permissible Values of a DataElementType |
| Note | - |
| Remark | - |

**Table 24: CLASS: ExtensibleDictionaryItem**

| Field | Value |
|---|---|
| Name | ExtensibleDictionaryItem |
| Definition | An abstract type allowing extension or restriction of DictionaryItems as well as the definition of an inheritance structure |
| Note | Only subtypes of ExtensibleDictionaryItem can be extended or restricted. |
| Remark | - |

**Table 25: CLASS: NonSemanticDictionaryItemSet**

| Field | Value |
|---|---|
| Name | NonSemanticDictionaryItemSet |
| Definition | Represents an arbitrary grouping of DictionaryItems. |
| Note | - |
| Remark | Allows DictionaryItems to be grouped for navigation or maintenance purpose |

**Table 26: CLASS: SemanticDictionaryItemSet**

| Field | Value |
|---|---|
| Name | SemanticDictionaryItemSet |
| Definition | Represents a semantic grouping of DictionaryItems |
| Note | - |
| Remark | Allows the specifier to provide definition of types and restriction of information by means of Generalization/Specialization or Restriction or Associations; provides context for the definition of DataElementTypes |

**Table 27: CLASS: Restriction**

| Field | Value |
|---|---|
| Name | Restriction |
| Definition | Defines restrictions on two associated ExtensibleDictionaryItems |
| Note | - |
| Remark | Association |

**Table 28: CLASS: Term**

| Field | Value |
|---|---|
| Name | Term |
| Definition | A DictionaryItem defining a word or expression that has a precise meaning in a particular context |
| Note | - |
| Remark | - |

**Table 29: CLASS: ValueSpaceType**

| Field | Value |
|---|---|
| Name | ValueSpaceType |
| Definition | Defines a reference to a language used to define ValueSpaces |
| Note | - |
| Remark | - |

**Table 30: CLASS: ValueQualifier**

| Field | Value |
|---|---|
| Name | ValueQualifier |
| Definition | A DictionaryItem defining a qualification for a Value |
| Note | - |
| Remark | - |

**Table 31: CLASS: ValueQualifierSet**

| Field | Value |
|---|---|
| Name | ValueQualifierSet |

**Table 31: CLASS: ValueQualifierSet**

| Field | Value |
|---|---|
| Definition | A DictionaryItem defining a set with a fixed sequence of ValueQualifiers applicable to the Value of a DataElementType |
| Note | ValueQualifiers in a ValueQualifierSet qualify a single semantic concept as defined in a DataElementType |
| Remark | - |

**Table 32: CLASS: Dictionary**

| Field | Value |
|---|---|
| Name | Dictionary |
| Definition | A computer-sensible set of DictionaryItems defining shared knowledge about one or more domains |
| Note | - |
| Remark | A Dictionary would typically be the root element for all DictionaryItems. |

## 5.3.3 Discussion

### 5.3.3.1 What is metadata?

The dictionary model contains meta information about what we call library instance data. We can exemplify this with an excerpt from an electronic component datasheet.

**Table 33: Characteristics**

| Symbol | Parameter | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{cc}$ | supply voltage | | 2.8 | 3.0 | 3.2 | V |
| $I_{op}$ | operating current | $V_{cc}$ = 3.0 V and $T_{amb}$ = 20°C | | | 3 | mA |

In the table above, the entries in the rows that begin with $V_{cc}$ and $I_{op}$ are library instance data. These are the actual values of interest in describing the electronic component in the datasheet. In our library, the metadata about these instances is that each entry MAY have an entry for Symbol, MUST have an entry for Parameter, MAY have an entry for Condition, MAY have an entry for 'Min', etc. A further constraint (also metadata) might be to say every time the Parameter 'supply voltage' is used, a symbol MUST be specified, and that symbol MUST be $V_{cc}$, and that values for 'Min', 'Typ', and 'Max' MUST be specified, and that the Unit specified MUST be volts.

Examples of metadata:

- The symbols $V_{cc,}$ $T_{amb}$ and $I_{op}$
- The description of a parameter “supply voltage” and “operating current”
- The link between the symbols and their description
- The units V, A and °C
- The value types “Min.”, “Typ.” and “Max.”

Examples of unique data for this product (library information):

- The values for each parameter
- The conditions reported on those parameters

- The link between the conditions and the reported values

The dictionary model captures the structure of the metadata only. The information specific to the product is stored in the library.

### 5.3.3.2 Commonality between various kinds of dictionary items

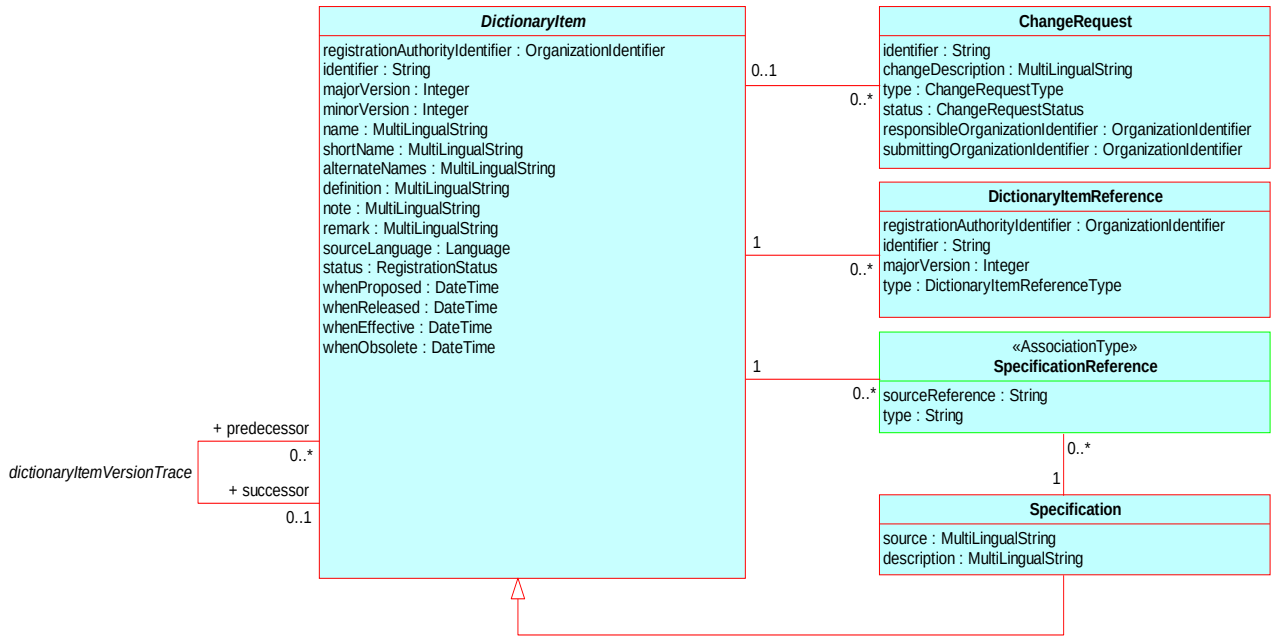



**Fig 8. DictionaryItem**

The **DictionaryItem** is an abstract type that describes the common attributes and associations for all concrete dictionary entries. The **DictionaryItem** plays an important role in the fact that it is guaranteed to be globally unique and that it can be referred to from the library information. In most cases, the reference to **DictionaryItems** provides the definition for the library data.

### 5.3.3.3 Ordering of DictionaryItems

A dictionary contains a set of **DictionaryItems**. The **DictionaryItems** are not explicitly ordered in the dictionary, but (at least) the following ordered rendition should be possible:

- Order by kind (**DataElementType**, **Term**, etc.)
- Order by name
- Order by date created
- Order by date modified

In addition, the **DictionaryItems** can be organized in sets as discussed in Section 5.3.3.10 “DictionaryItem structure” on page 30.

### 5.3.3.4 Uniqueness of DictionaryItems

To ensure unambiguous reference to a **DictionaryItem**, a **DictionaryItem** is guaranteed unique by combining the following attributes:

- identifier
- majorVersion[6]
- RegistrationAuthorityIdentifier

The idea behind this uniqueness scheme is that every registrationAuthorityIdentifier can create their own uniqeness scheme as long as they do not create a **DictionaryItem** with the same major version and identifier.

The uniqueness in formally defined in Section 5.3.4.2 “Dictionary Item Uniqueness” on page 44.

#### 5.3.3.5 Versioning of the DictionaryItems

All dictionary entries are versioned. The version of a **DictionaryItem** is tracked by two version attributes:

- Major version
  Changes every time the **DictionaryItem** changes semantic or in other ways are changed such that it could invalidate library data linked to it.
- Minor version
  Changes every time a minor change has been done to the **Item**. A change that only effects the minor revision can not invalidate the library information. A typical change that merits only a minor version change is the fixing of spelling mistakes.

In addition to the tracking of versions, it is possible keep track of **DictionaryItems** that have been merged into new items using the successor/predecessor association. We allow several predecessors (i.e., the creation of a new **DictionaryItem** as a merge of several other **DictionaryItems**). If a **DictionaryItem** has a successor, then it is no longer the preferred **DictionaryItem** to be used.

#### 5.3.3.6 DictionaryItem attributes

**DictionaryItems** have a set of attributes. The exact semantic of each can be found in Section 10 “Data Dictionary” on page 106. We will discuss a few of them in this section.

Every **DictionaryItem** has a lifecycle. The lifecycle is tracked by the **status** attribute. Notice that the **status** attribute is tracked by an enumeration type called **RegistrationStatus**. The possible values for the **RegistrationStatus** are defined in Section 6.2.3 “Definition of ENUMERATIONS in the “Enumerations” view” on page 55.

The various descriptions and names of **DictionaryItems** (shortName, description, name, alternateNames, etc.) may be tracked in multiple languages, hence the attribute type **MultiLingualString**.

The **DictionaryItem** has several attributes that report the dates of the most relevant **status** changes.

6. The uniqueness here is really referring to what is required to uniquely identify an item. An implementer may select to keep minor versions around (e.g., to enable rollback). In those situations we also need to add minor version to make a dictionary item unique. However, any one referring to a dictionary item by use of a (foreign) key, would always refer to the item with the latest minor version.

### 5.3.3.7 Data Element Type

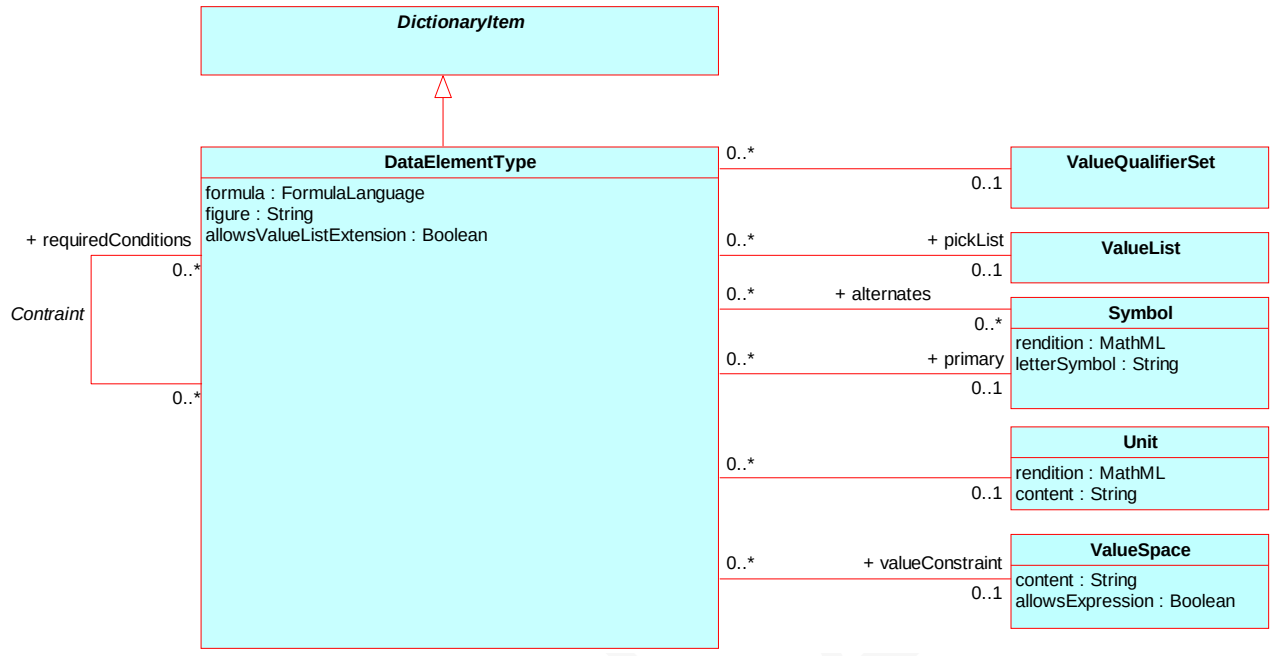



**Fig 9. DataElementType**

The **DataElementTypes** are special kinds of **DictionaryItems**. These pertain to the parametric information. The data element types capture metadata for parametric information.

The **DataElementType** is a subtype of **DictionaryItem**, hence it inherits all the attributes and structure defined in Section 5.3.3.6 “DictionaryItem attributes” on page 28.

The **DataElementType** may enforce certain restrictions on the reported values in a library. These restrictions may be defined in two ways.

- **Values** restricted to a **ValueList**
- **Values** restricted by a **ValueSpace**

### 5.3.3.8 ValueList restriction

If a **DataElementType** is linked to a **ValueList**, it requires that the values reported are to be found as a value of one of the **ValueListItems** in the **ValueList**. A specifier may, however, extend the **ValueList** if (and only if) the attribute **allowsValueListExtension** is true.

### 5.3.3.9 ValueSpace restriction

If a **DataElementType** is linked to a **ValueSpace**, then the reported values are restricted to values constrained by this **ValueSpace**. Typically, this is the “type” of the **Value**, e.g. “real”, “string” or “boolean”.

A **DataElementType** may define a set of **requiredConditions**. If a **Characteristic** of an **Item** is reported based on this **DataElementType**, then it is only valid when the list of requiredConditions are present.

It is possible to define **Symbols** along with the **DataElementType**. There may be one primary symbol and if so, one or more alternate symbols. A **Symbol** is a standard shorthand for representing the **DataElementType**. E.g., “supply voltage” can be represented as $V_{cc}$.

A **DataElementType** may also define a **Unit**. The Unit represents a unit of measure implicitly used when reporting values for this **DataElementType**. E.g., “supply voltage” is measured in volt, hence all reported values for “supply voltage” are implicitly reported in volt.

### 5.3.3.10 DictionaryItem structure

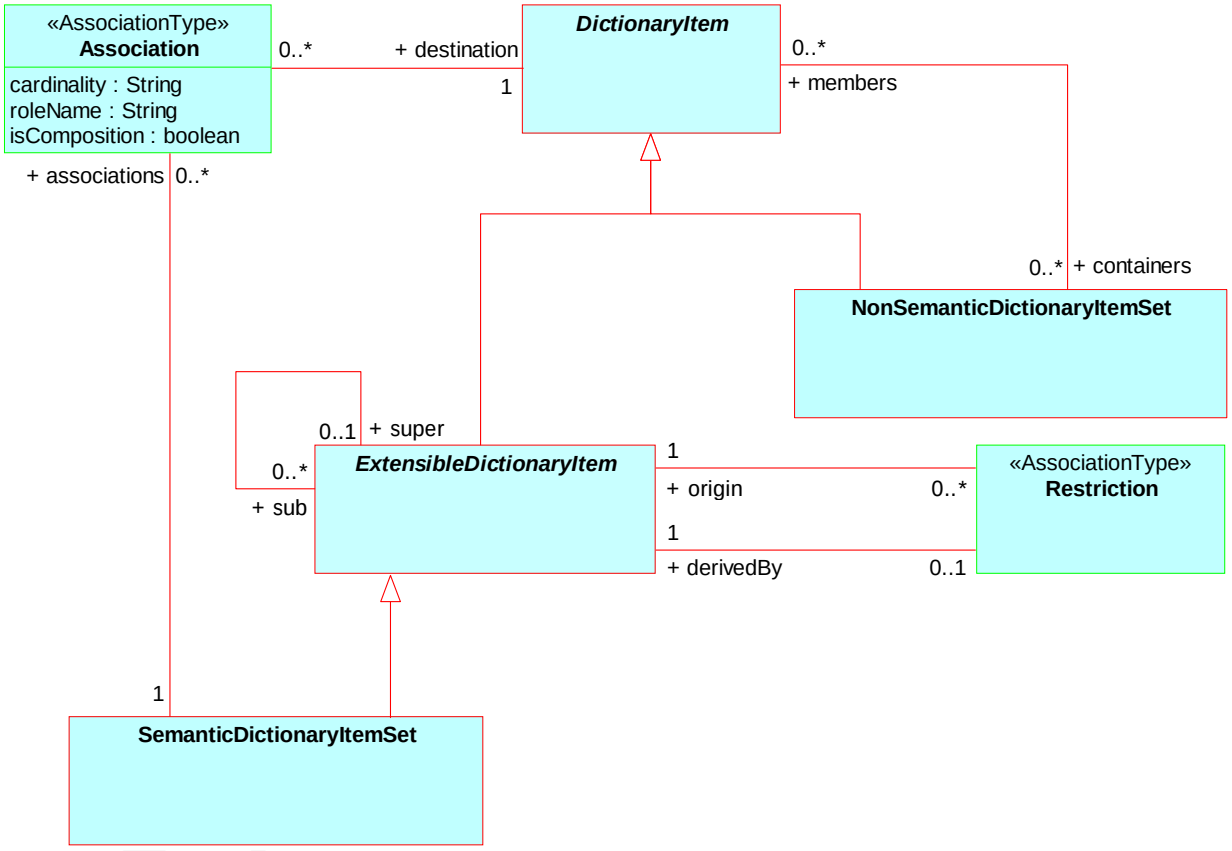



**Fig 10. DictionaryItem structure**

The **DictionaryItems** can be grouped together for various reasons.

A group is represented by an instance of **SemanticDictionaryItemSet**. or a **NonSemanticDictionaryItemSet**. The two kinds of groups are:

- **SemanticDictionaryItemSet**.
  Represents a set of **DictionaryItems** grouped together to provide structural rules for library data.
- **NonSemanticDictionaryItemSet**.
  Represents a set of **DictionaryItems** grouped together for convenience. The set itself has no formal semantic.

A **DictionaryItem** can be member of multiple sets.

When maintaining the dictionary, one may decide to organize the **DictionaryItems** based on various aspects. Examples of such aspects are:

- Responsibility. Grouping **DictionaryItems** based on who maintains the items.
- Source. Grouping **DictionaryItems** based on their origin.
- Functionality. Grouping related **DictionaryItems** together based on the functionality of the item it describes

If the reason for the grouping is to ease the maintenance or navigation to the **DictionaryItems**, the means of doing so is by use of **NonSemanticDictionaryItemSets**.

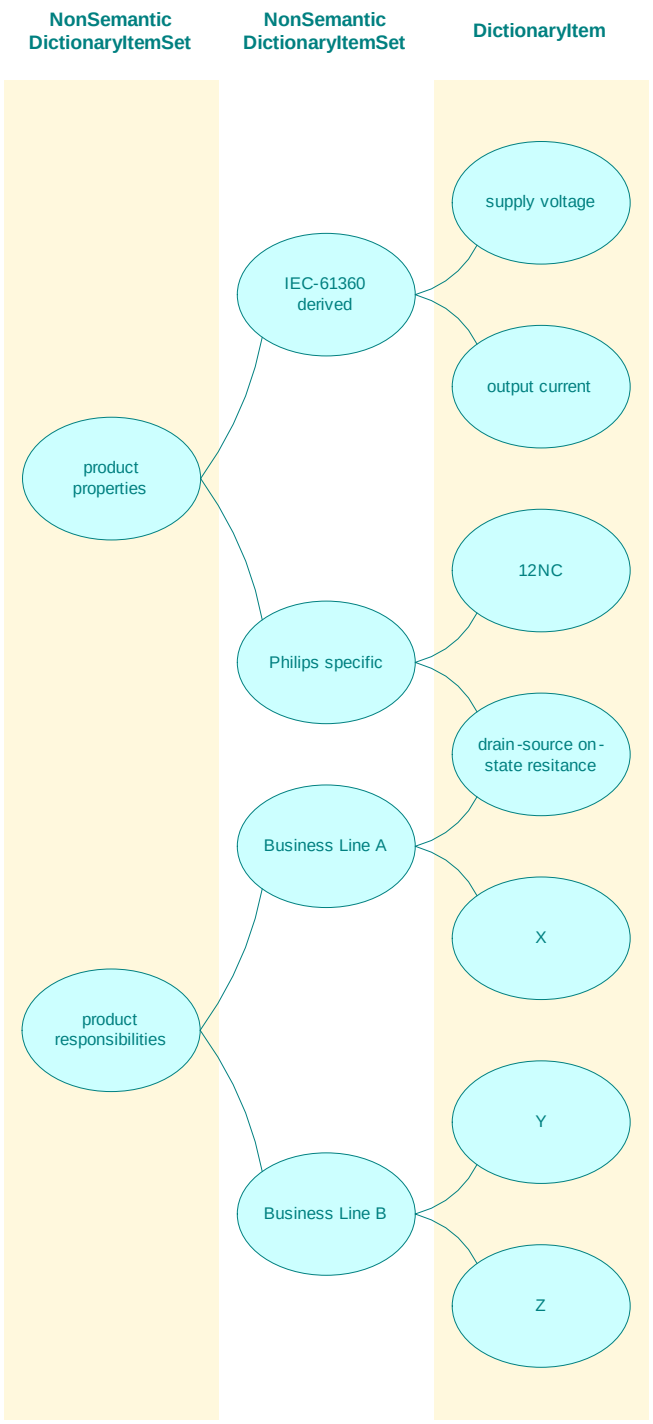



**Fig 11. Use of Non SemanticDictionaryItemSet**

The example above shows the use of **NonSematicDictionaryItemSets**. In this example the various **DictionaryItems** have been organized into convenience groupings. Note the following points from the example

- An **item** may be in more than one set.
  E.g, “drain-source on-state resistance” is both in the “Philips Specific” and in the “Business line B” set
- The groupings form tree structures
- We may have sets of sets.
  E.g. the set “Product responsibility” contains two other sets; “Business line A” and “Business line B”

The grouping of **DictionaryItems** may also have semantics. The means for this is the use of **SemanticDictionaryItemSets**.

**SemanticDictionaryItemSets** provide three different possibilities for semantics:

- Use of **Associations**
- Use of **Generalization/Specialization**
- Use of **Restriction**

**Associations** are useful when we require a particular grouping of **Characteristics** to be reported on an item. This is best shown with examples.

Let’s say we want to exchange addresses. We know the structure of an address to be:

- Addressee
- Street address
- Postcode
- City
- An optional state or province
- Country

We want to make sure that anyone reporting an address includes values for all of these elements. We would then have to define a **SemanticDictionaryItemSet** that defines the structure of an address[7].



7. Please note that the structure of the address shown here and in Section 8.5 “Business address” on page 91 is provided as an example only and is non-normative. It is expected that a fully normative structure or structures for “address” will be developed as a part of a future RosettaNet Dictionary content development effort.

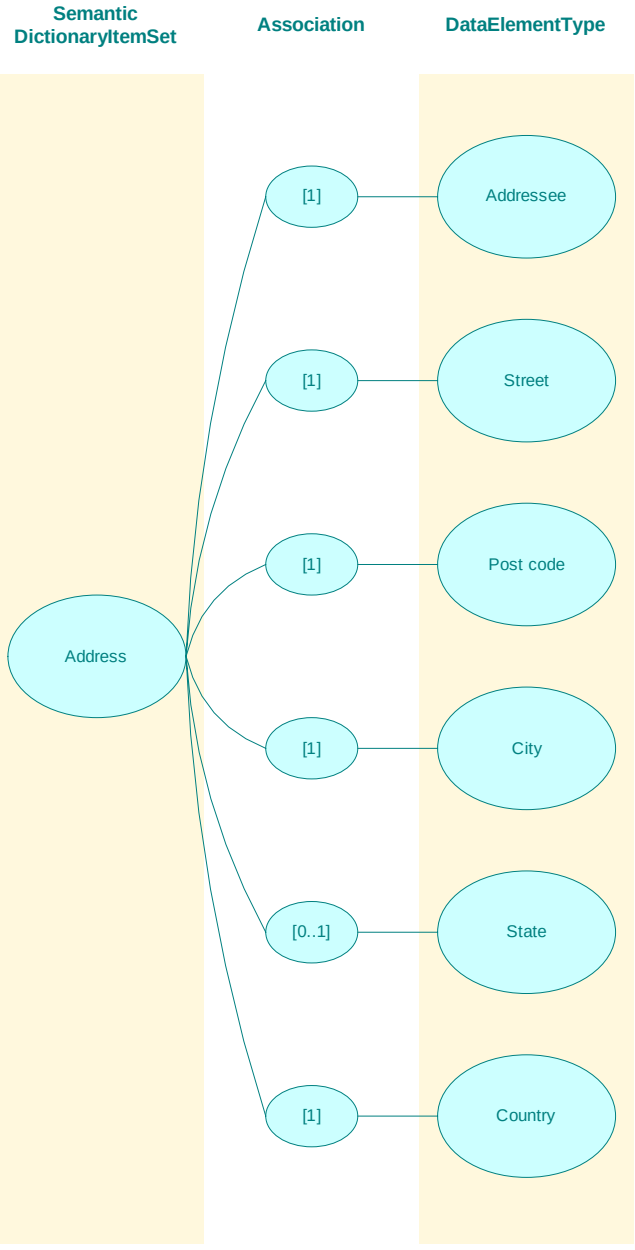



Fig 12. Address Example

Notice that in this example the cardinality (or multiplicity) of the elements has been defined in the instances of **Association**, the set is defined as a **SemanticDictionaryItemSet** and that the individual fields are defined as **DataElementTypes**.

The semantic of such a structure is that, if a **CharacteristicSet** is reported on an **Item** which defines an address, the address should include **CharacteristicSpecifics** providing values for addressee, street, postcode, city, country and optionally state.

We may find that some **SemanticDictionaryItemSet** have common constraints. To handle these cases, we may use the concept of generalization/specialization.

Let's assume we want a special kind of address that has temporary constraints. We want to know from when and to when the address is valid. We call this address "Temporary Address". We want to reuse the constraints specified for address, but we need two more **Characteristics** to be reported, "begin date" and "and date".

The proper way of specifying this in the dictionary would be to instantiate a new **SemanticDictionaryItemSet** that extends the Address

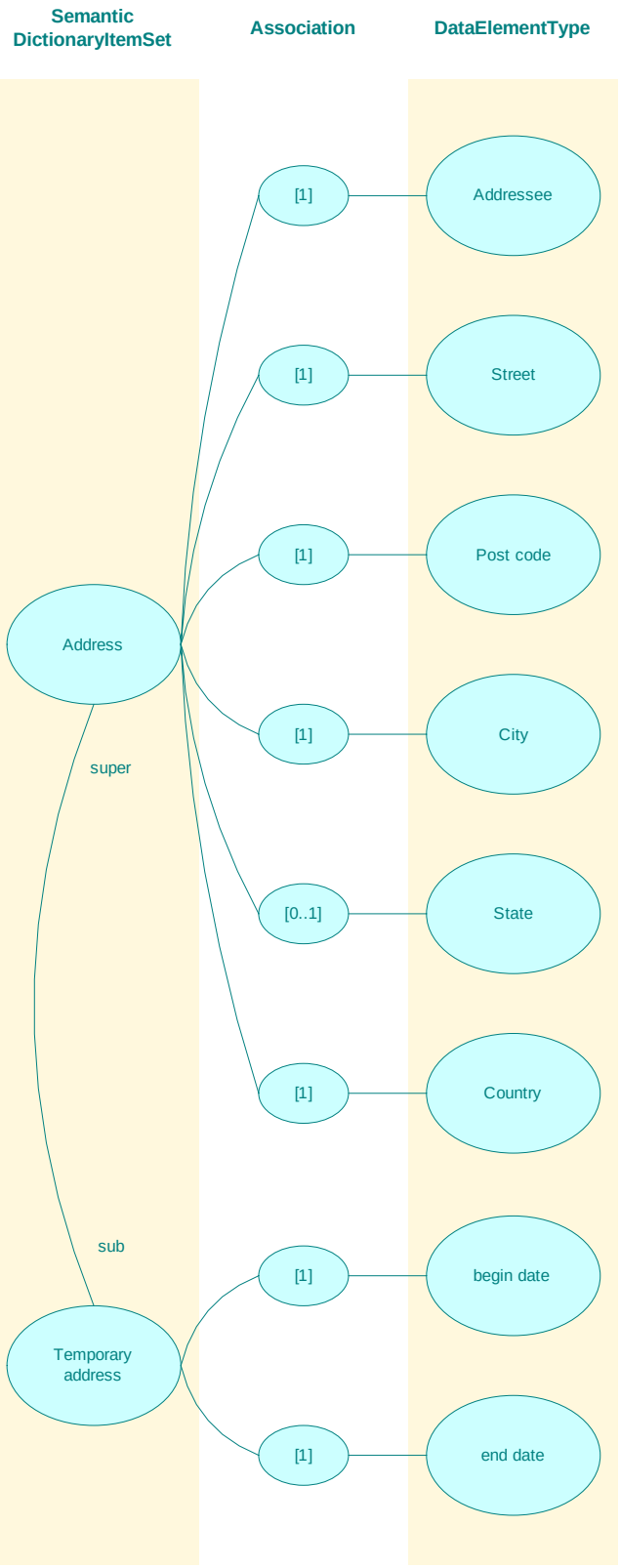



Fig 13. Specialization example

The example above shows how the dictionary can be instantiated to achieve our goals. Notice the following points:

- The “Temporary Address” is defined as a sub type of “Address”
- The “Temporary Address” defines two new data element types

If a library now is defining a new **SemanticDictionaryItemSet** “Temporary Address”, this set must contain values for addressee, street, postcode, city, (optionally) state, country, valid from and valid to.

The last option for defining constraints on the library is to use restriction. The idea here is to be able to say “I want to define a **SemanticDictionaryItemSet** that is like this other one but excluding these characteristics”. In the model we’ve left the semantics for doing this unspecified.

#### 5.3.3.11 Terms

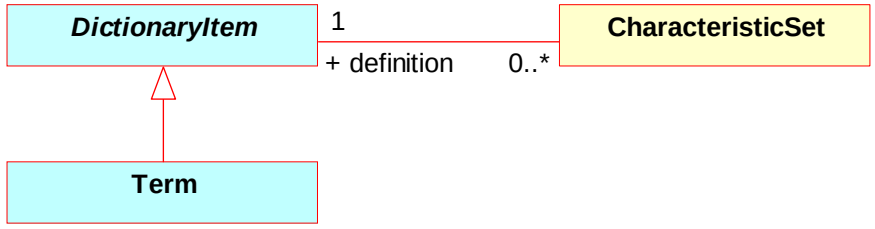



Fig 14. Term

**Terms** are primarily used for two things:

- To define the semantic of a **CharacteristicSet**
- To define the semantic of a commonly used term linked to from a **MultiLingualString**

Before a **CharacteristicSet** can be reported, it must have a definition from the dictionary. The **CharacteristicSet** may link to a **SemanticDictionaryItemSet** (as shown in Section 5.3.3.10 “DictionaryItem structure” on page 30) or if it has no semantic, it may link to a **Term**. Example of these kinds of **CharacteristicSets** may be:

- Optional features
- Electrical characteristics
- Dynamic characteristics
- Mechanical characteristics
- Limiting values

One of the **Terms** is going to have the definition “No Semantic”. This is to define the **CharacteristicSets** that do not have semantics, but are used to group together common conditions for a set of **Characteristics**.

The use of **Terms** from a **MultiLingualString** is yet to be defined, but we know that we want to provide common **Terms** that are referred to from any description inside the dictionary and potentially also in the library.

### 5.3.3.12 ValueQualifiers and ValueQualifierSets

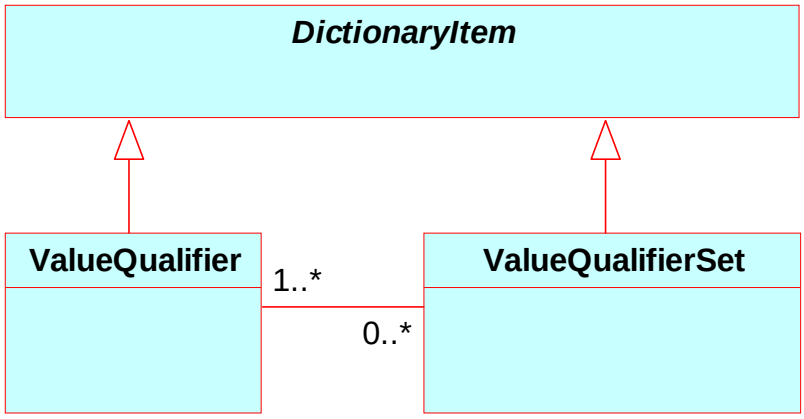



Fig 15. ValueQualifierSet

An instance of a **ValueQualifier** defines partial semantics for a value. Examples of known **ValueQualifiers** are[8]:

- **Min**. The minimum value in a reported set of values.
- **Typ**. The typical value in a reported set of values.
- **Max**. The maximum value in a reported set of values.
- **Nom**. The nominal value in a reported set of values.
- **Spec**. The specified value in a reported set of values.

The **ValueQualifierSet** defines a set of **ValueQualifiers** as recommended or required to be reported together. E.g., “Min,Typ,Max”. The list of known **ValueQualifiers** can easily be extended should the need arise.



8. These ValueQualifiers are examples and as such represent dictionary content and not a part of the RDA specification.

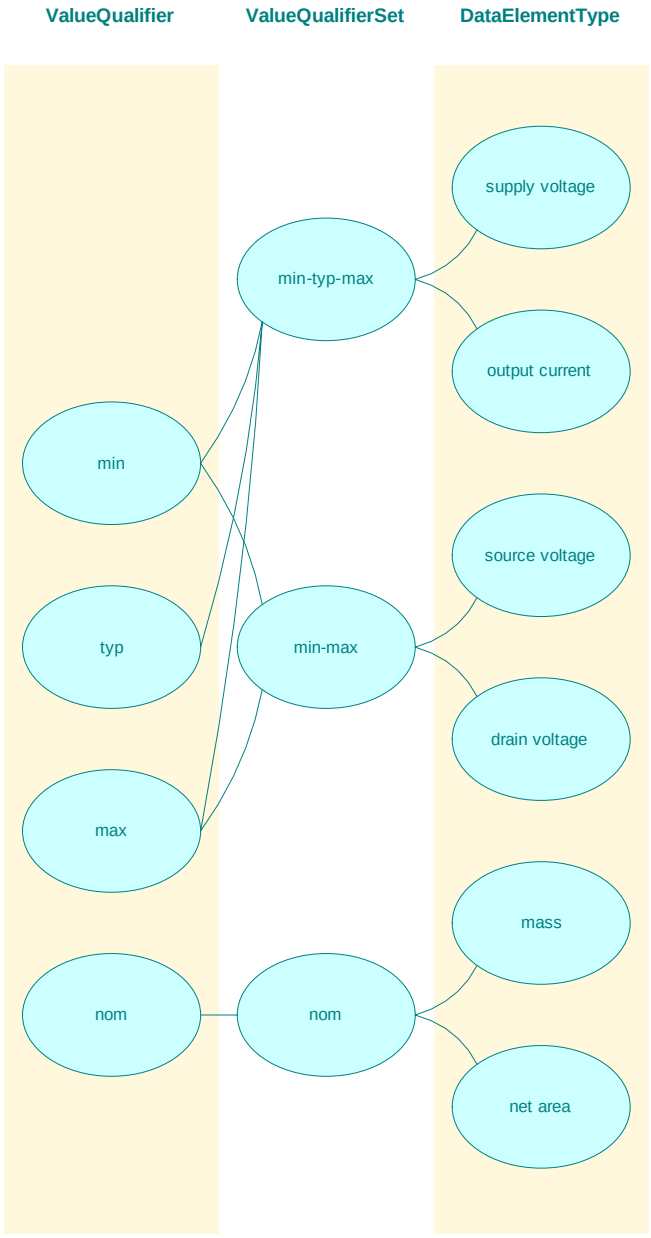



**Fig 16. Example of use of ValueQualifiers and ValueQualifierSets**

Above we see an example of the use of **ValueQualifierSets** and **ValueQualifiers**. Note in this example:

- The **ValueQualifiers** can be used in multiple sets
- The **DataElementTypes** refer to a **ValueQualifierSet** and not to the individual **ValueQualifiers** (even when there is only a single element in the set as in the case of **Nom**)

### 5.3.3.13 Value List

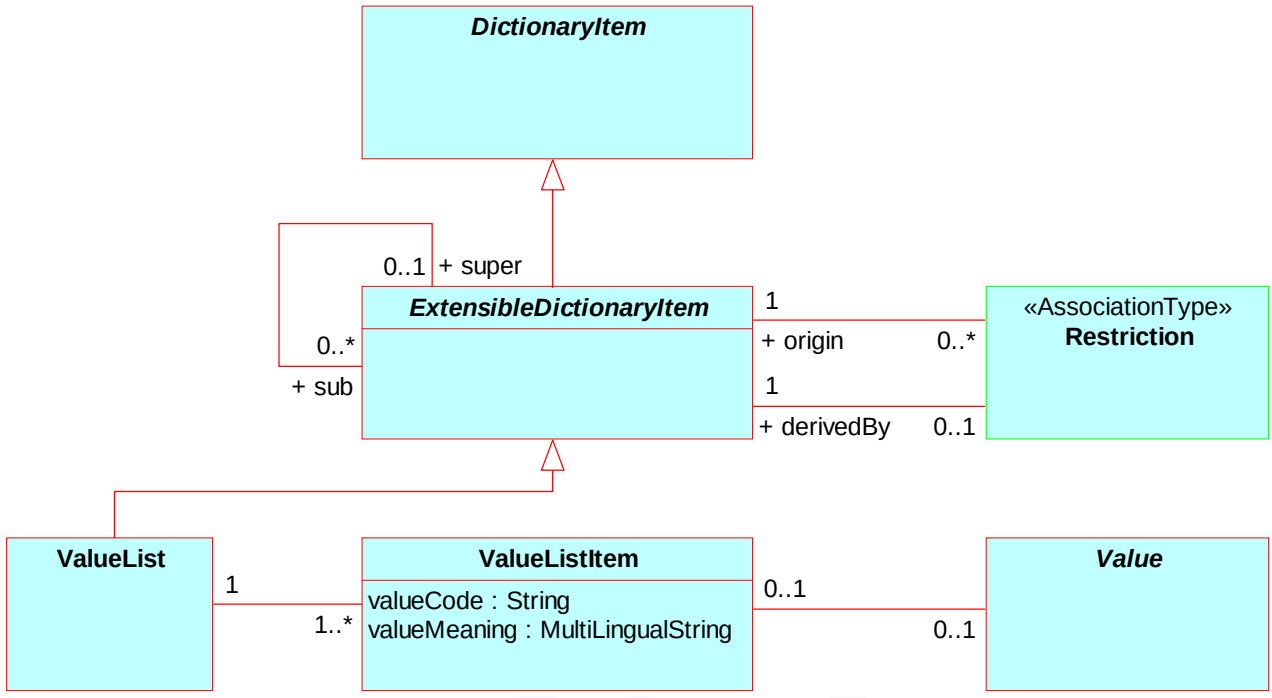



Fig 17. ValueList

A **ValueList** groups together a set of **ValueListItems** with their associated **Values**. The **ValueListItem** may have an associated definition The **ValueList** is used to restrict the possible values reported on a **CharacteristicSpecific**.

Below is an example of such a **ValueList**, based on IEC 61360, AAE634 - terminal material (for leads on electronic components).

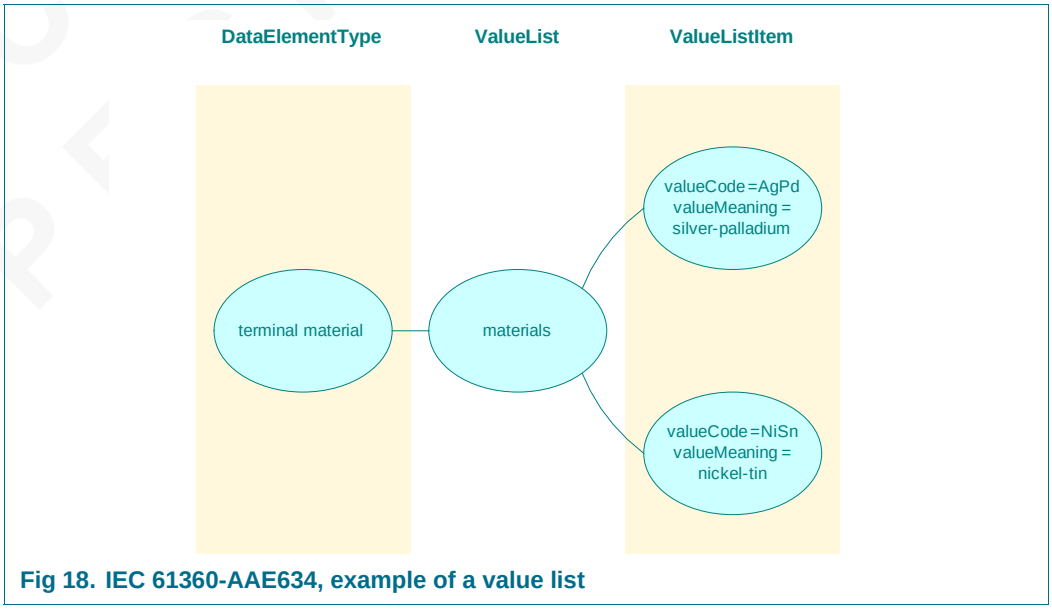



Fig 18. IEC 61360-AAE634, example of a value list

In this example, the terminal material has been restricted to two values:

- silver-palladium
- nickel-tin

These values have been defined as **valueMeanings** on the **ValueListItem**.

Sometimes we also want a **Value** to be associated with the **ValueListItem**, as shown in the example below

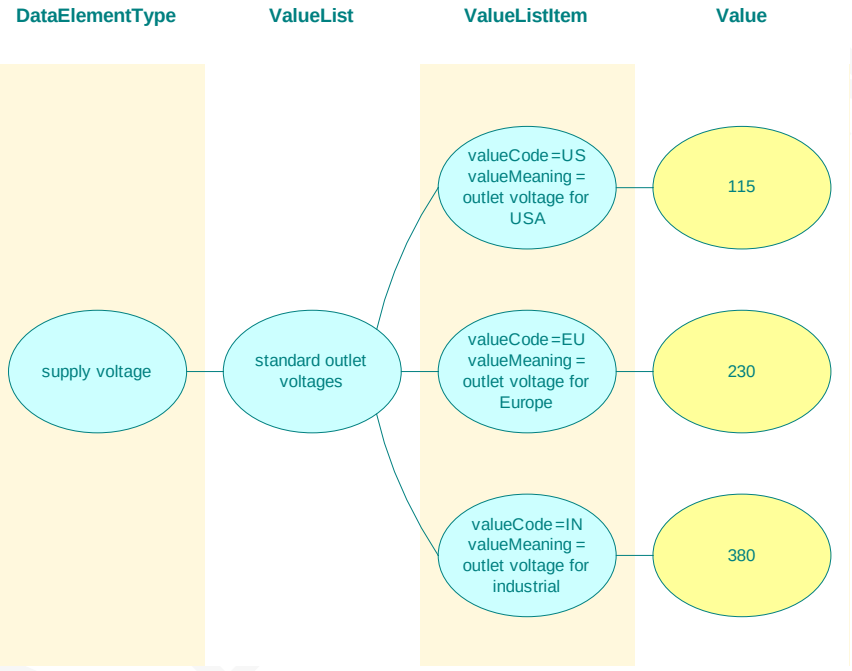



**Fig 19. Standard outlet voltages**

Notice in this example that we also have provided computer-sensible **Values**: 115, 230, 380.

### 5.3.3.14 Value Space

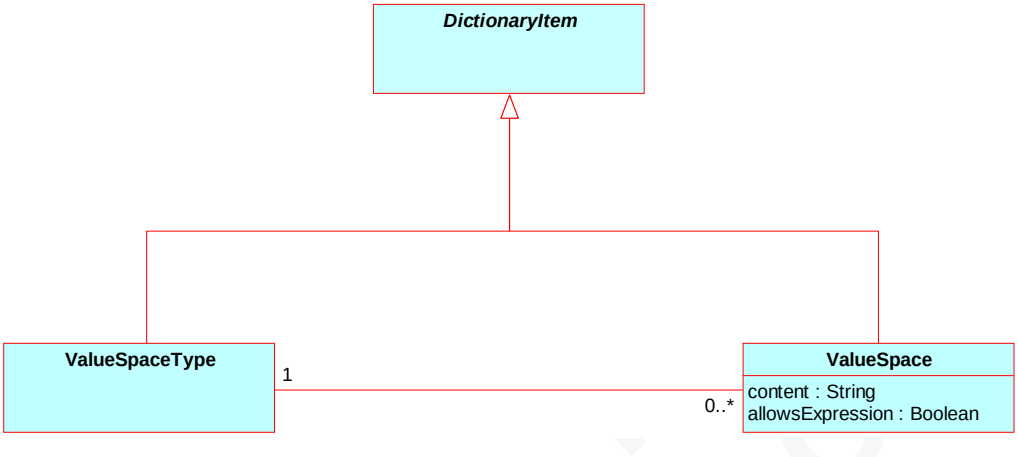



Fig 20. ValueSpace and ValueSpaceType

A **ValueSpace** defines constraints on format or precision of **Values**. If a **DataElementType** is linked to a **ValueSpace**, then all **Values** reported for **Parameters** defined by the **DataElementType** must conform to the rules defined by the **ValueSpace**.

There may be many languages allowed for defining **ValueSpaces**. This is expressed in the model by **ValueSpaceType**. A **ValueSpaceType** defines a language for the **ValueSpace**.

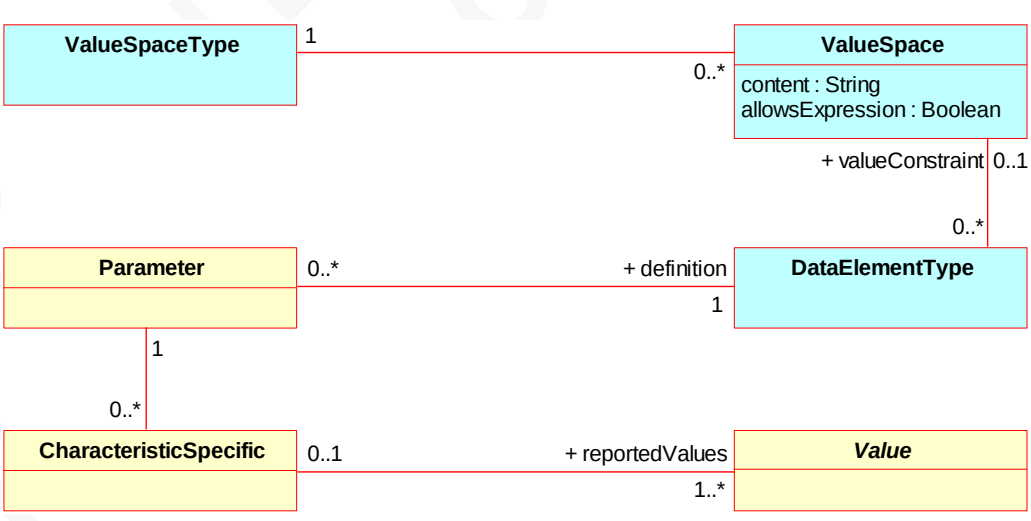



Fig 21. Path from Value to ValueSpace

The figure above shows how **Values** are evaluated against a **ValueSpace**. Every value is linked to a **CharacteristicSpecific**. Each **CharacteristicSpecific** is linked to a Parameter. A Parameter is linked to a **DataElementType**. A **DataElementType** is linked to a **ValueSpace**. Hence, a **Value** can be evaluated against a corresponding **ValueSpace**.

The figure below shows an example of the use of **ValueSpaces**:

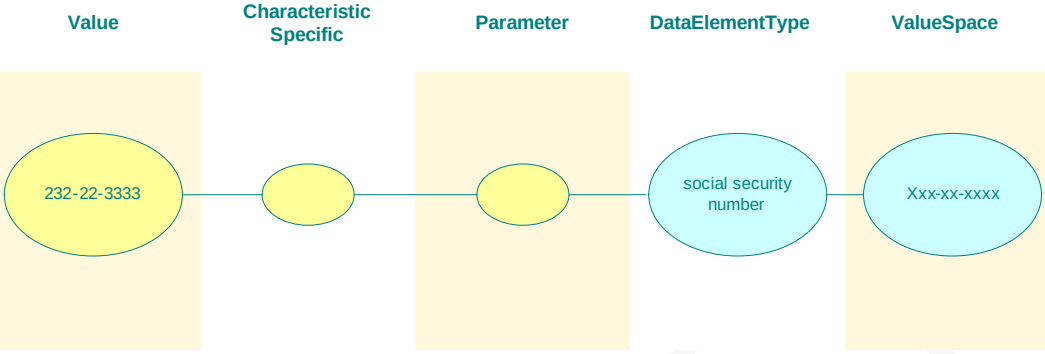



**Fig 22. Example of use of Value Spaces**

In this example we see a reported value for a US social security number. The value is evaluated against a **ValueSpace** defined in the dictionary.

### 5.3.4 Constraints

#### 5.3.4.1 Dictionary Item Deactivation

When a **DictionaryItem** is active (attribute **status** = “RELEASED”) then there is no **DictionaryItem** which replaces this one.

**Table 34: OCL expression for DictionaryItemDeactivation**

```
context DictionaryItem inv DictionaryItemDeactivation:
   self.active implies self.replacedBy->isEmpty
```

#### 5.3.4.2 Dictionary Item Uniqueness

The registrationAuthorityIdentifier, the identifier and the major version makes up the unique identifier for a **DictionaryItem**.

**Table 35: OCL expression for DictionaryItemUniqueness**

```
inv DictionaryItemUniqueness:
   DictionaryItem.allInstances->forAll( pd1, pd2 |
      (pd1 <> pd2) implies
         (pd1.registrationAuthorityIdentifier.concat(
          pd1.identifier.concat(
          pd1.majorVersion))) <>
         (pd2.registrationAuthorityIdentifier.concat(
          pd2.identifier.concat(
          pd2.majorVersion))) <>
   )
```

#### 5.3.4.3 NonSemanticDictionaryItemSet Self Exclusion

A **NonSemanticDictionaryItemSet** can not have itself as a member (directly or indirectly through one of its descendants). Note that this constraint guarantees a non-cyclic graph of **NonSemanticDictionaryItemSets**.

**Table 36: OCL expression for NonSemanticDictionaryItemSet_SelfExclusion**

```
function allChildren( s : NonSemanticDictionaryItemSet ) =
   s.members->union( s.members->collect( m | allChildren( m ) )
context NonSemanticDictionaryItemSet inv NonSemanticDictionaryItemSet_SelfExclusion:
    allChildren(self)->excludes( self )‘
```

#### 5.3.4.4 SemanticDictionaryItemSet Self Exclusion

A **SemanticDictionaryItemSet** can not have itself as a member (directly or indirectly through one of its descendants). Note that this constraint guarantees a non-cyclic graph of **SemanticDictionaryItemSets**.

**Table 37: OCL expression for SemanticDictionaryItemSet_SelfExclusion**

```
function allChildren( s : SemanticDictionaryItemSet ) =
   s.members->union( s.associations.destination->collect( m | allChildren( m ) )
context SemanticDictionaryItemSet inv SemanticDictionaryItemSet_SelfExclusion:
   NOT allChildren(self)->includes( self )‘
```

#### 5.3.4.5 SemanticDictionaryItemSet Member Types

A **SemanticDictionaryItemSet** shall only contain other **SemanticDictionaryItemSets** or **DataElementTypes**.

**Table 38: OCL expression for SemanticDictionaryItemSet_MemberTypes**

```
context SemanticDictionaryItemSet inv SemanticDictionaryItemSet_MemberTypes:
   self.association.destination->forAll( m |
      m.oclType = SemanticDictionaryItemSet or m.oclType = DataElementType
   )
```

#### 5.3.4.6 ExtensibleDictionaryItem Generalization Constraint

A **ExtensibleDictionaryItem** can not subtype itself. Also, a subtype must be of the same concrete kind. E.g., it is not legal to make a **DataElementType** a subtype of **SemanticDictionaryItemSet**.

**Table 39: OCL expression for ExtensibleDictionaryItem_GeneralizationConstraint:**

```
function allSupers(e : ExtensibleDictionaryItemSet ) =
   e.super->union( allSuper( e ) )
SemanticDictionaryItemSet inv ExtensibleDictionaryItem_GeneralizationConstraint:
   self.super.oclType = self.oclType and
   allSuper(self)->excludes( self )
```

#### 5.3.4.7 Symbol Constraint

Only if a **DataElementType** has a preferred **Symbol** can it have alternate **Symbols**.

The alternate **Symbols** can not include the preferred **Symbol**.

**Table 40: OCL expression for SymbolConstraint**

```
context DataElementType inv SymbolConstraint:
   self.primary->isEmpty implies
      self.alternates->isEmpty AND
     self.alternates->excludes( self.primary )
```

#### 5.3.4.8 Specifications Constraint

Specifications can not refer to other specifications.

**Table 41: OCL expression for SpecificationConstraint**

```
context Specification inv SpecificationConstraint:
   self.specificationReferences->isEmpty
```

## 5.4 Characteristic View

### 5.4.1 Type model

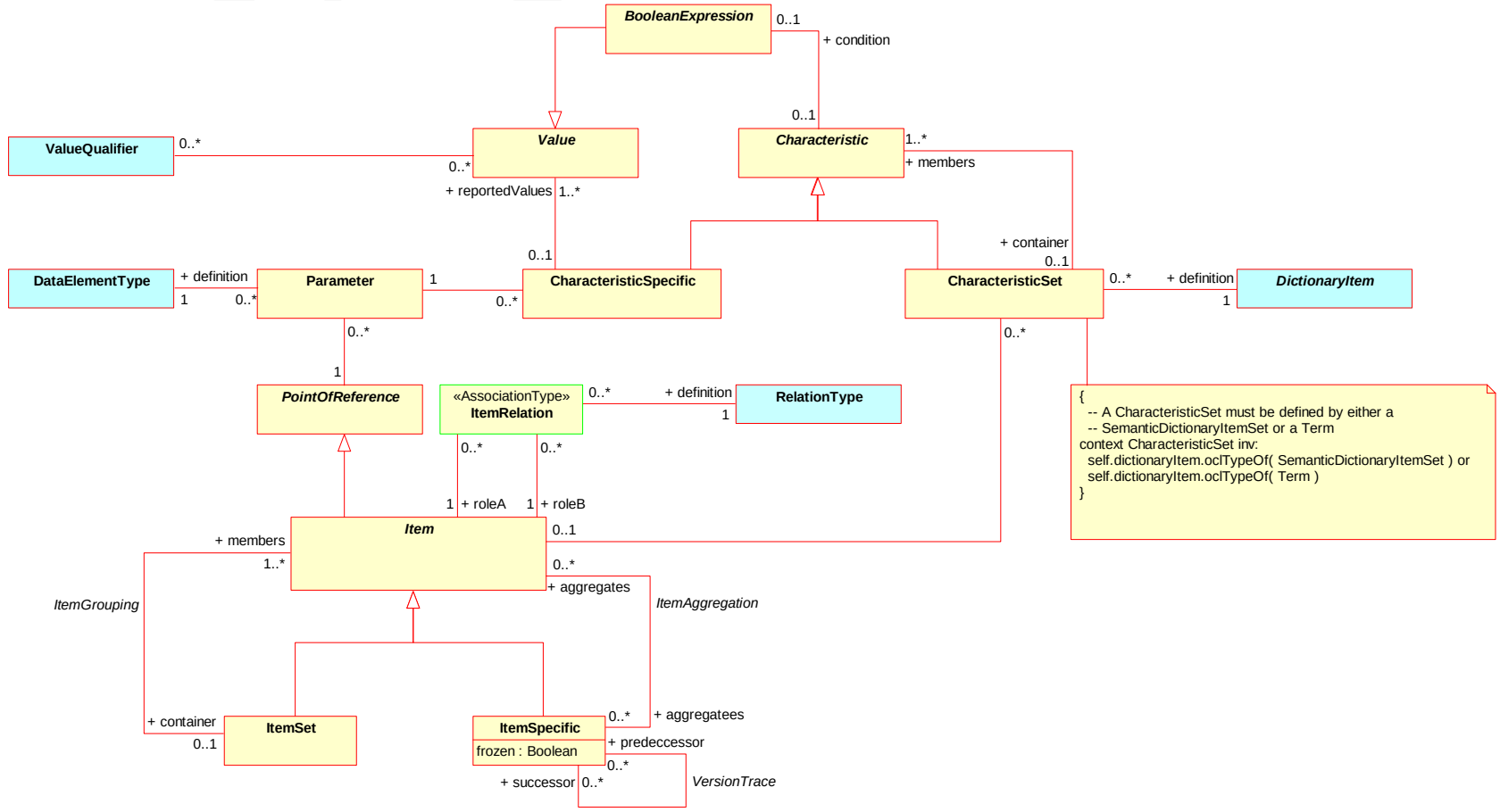



**Fig 23. Type model for “Characteristic”**

### 5.4.2 Definition of CLASSES in the “Characteristic” view

**Table 42: CLASS: Characteristic**

| Field | Value |
|---|---|
| Name | Characteristic |
| Definition | Property of an Item as defined by one or more DataElementTypes |
| Note | - |
| Remark | - |

**Table 43: CLASS: CharacteristicSet**

| Field | Value |
|---|---|
| Name | CharacteristicSet |
| Definition | A set of Characteristics |
| Note | - |
| Remark | If a condition is reported on a CharacteristicSet, the semantic is that the conditions implicitly applies to all its members. If a CharacteristicSet’s definition is a SemanticDictionaryItemSet, its members are constrained by the rules in this SemanticDictionaryItemSet |

**Table 44: CLASS: CharacteristicSpecific**

| Field | Value |
|---|---|
| Name | CharacteristicSpecific |
| Definition | Represents a set of Values reported on a Parameter under a Condition as specified by a DataElementType |
| Note | - |
| Remark | - |

**Table 45: CLASS: Parameter**

| Field | Value |
|---|---|
| Name | Parameter |
| Definition | A concept that can be reported-on for a PointOfReference associated with an Item |
| Note | A Parameter is defined in the Dictionary by a DataElementType |
| Remark | - |

**Table 46: CLASS: PointOfReference**

| Field | Value |
|---|---|
| Name | PointOfReference |
| Definition | A point of reference associated with an Item for which a Parameter can be specified |
| Note | Can be Item, Terminus, Mode, Connection or TerminusModeConnectionAssociation [TMCA] |
| Remark | - |

### 5.4.3 Discussion

An **Item** specification contains a set of properties. These properties are structured as shown in the type model.

The model is best explained through examples. Let's look at the excerpt from a data sheet below.

Table 6: Electrical characteristics
$T_j$ = 25 °C unless otherwise specified.

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Per diode | | | | | | |
| $V_F$ | forward voltage | $I_F$ = 50 mA | - | 0.95 | 1.1 | V |
| $I_R$ | reverse current | $V_R$ = 50 V | - | - | 20 | nA |
| $C_d$ | diode capacitance | f = 1 MHz; see Figure 1 | | | | |
| | | $V_R$ = 0 V | - | 600 | - | fF |
| | | $V_R$ = 1 V | - | 430 | - | fF |
| | | $V_R$ = 20 V | - | 250 | 300 | fF |

**Fig 24. Example extract from BAP70-05**

This datasheet shows a set of properties for the product BAP70-05. When combining all the properties of a product we have a complete product specification.

The properties are built up by use of **Characteristics**, **Parameters**, **DataElementTypes**, **ValueQualifiers**, and **Values**

The data sheet fragment above can be instantiated in our model as shown below:

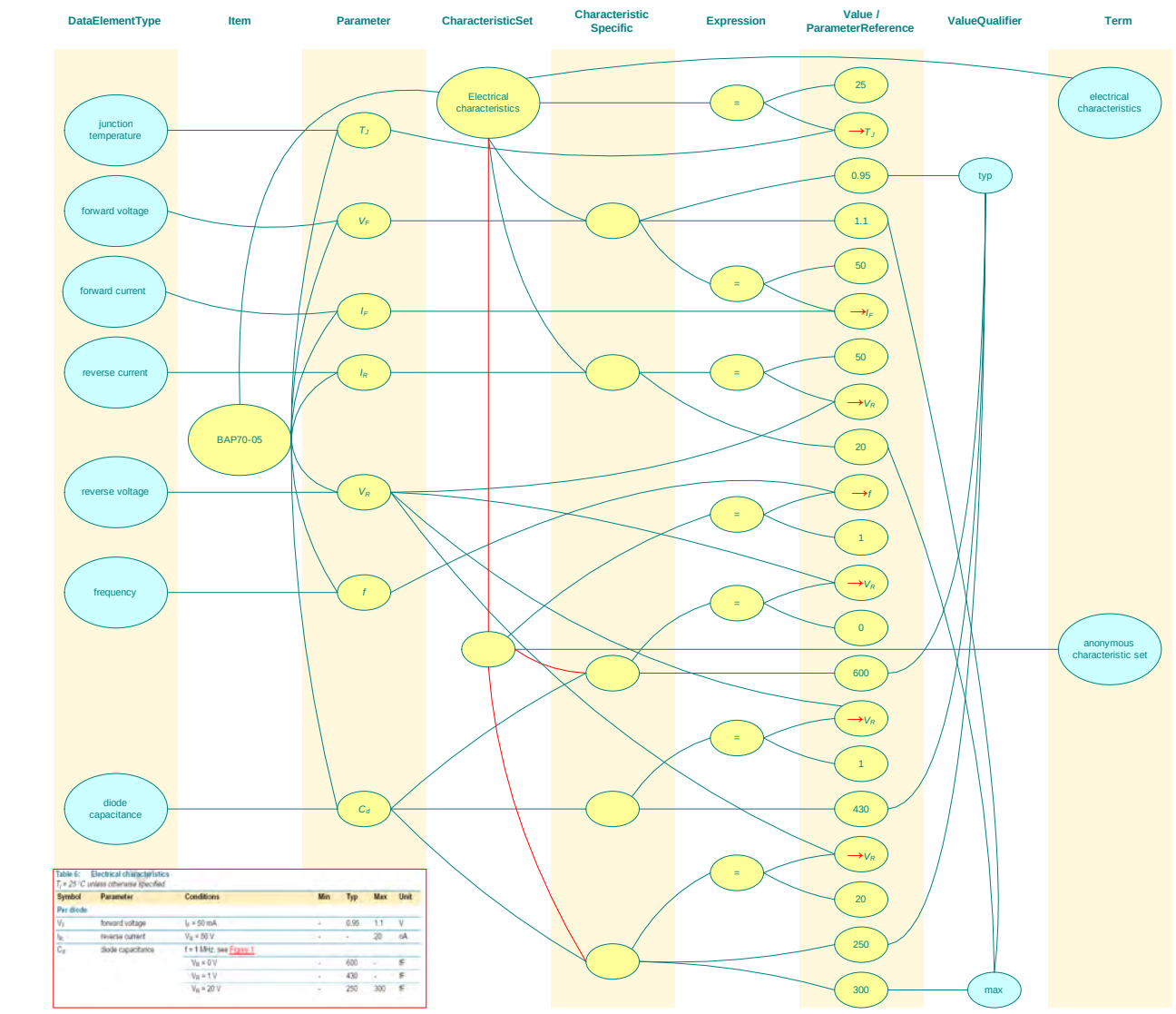



**Fig 25. Example extract from BAP70-05 expressed as a UML instance diagram**

Notice in this example that:

- The example has simplified the expressions.
  E.g. $T_j$ =25 should really have been an instance of **Eq** linked to an instance of **ParameterReference** (referring to the **Parameter** $T_j$) and an instance of an **IntegerValue** (the value 25)
- The **Symbols** and the name of the parameter is picked up from **DataElementTypes**
- Each row in the table corresponds to one **CharacteristicSpecific**
- The values in the cells are instances of **Value** in our model
- Each value is associated with a **ValueQualifier**
- Some parameters do not have characteristics (e.g. $T_j$ is only used in expressions forming conditions)

### 5.4.4 Constraints

#### 5.4.4.1 Kind of Definition For CharacteristicSet

The definition of a **CharacteristicSet** must be a **Term** or a **SemanticDictionaryItemSet**.

**Table 47: OCL expression for KindOfDefinitionForCharacteristicSet**

```
context CharacteristicSet inv KindOfDefinitionForCharacteristicSet:
   self.definition.oclIsKindOf( Term) or
   self.definition.oclIsKindOf( SemanticDictionaryItemSet )
```

#### 5.4.4.2 CharacteristicSet Self Exclusion

A **CharacteristicSet** should never include itself (directly or as a child's child). The **Characteristics** therefore can never form a cyclic graph.

**Table 48: OCL expression for CharacteristicSetSelfExclusion**

```
function allChildren( cs: CharacteristicSet) =
   cs.members->union( s.members.collect ( m | allChildren( m ) ) )
context CharacteristicSet inv CharacteristicSetSelfExclusion:
   allChildren(self)->excludes( self )
```

#### 5.4.4.3 Use of Values as Reported or Constraint

A **Value** can not constrain and report at the same time.

**Table 49: OCL expression for ValueCanNotBeBothConstrainingAndReporting**

```
context Value inv ValueCanNotBeBothConstrainingAndReporting:
   self.constrains->~reportedValues implies self.~condition->isEmpty and
   self.reportedValues->notEmpty implies self.condition->isEmpty
```

#### 5.4.4.4 Characteristics and parameters must link to the same item

A single item should be the root for parameters and characteristics.

**Table 50: OCL expression for Item_CharacteristicParameterConstraint**

```
context Item inv Item_CharacteristicParameterConstraint:
   self.parameter.characteristicSpecific.container.item = self
```

#### 5.4.4.5 SemanticDictionaryItemSet is Prescriptive

When a **CharacteristicSet** is defined by a **SemanticDictionaryItemSet**, all its members are prescribed in the dictionary. That is:

- Only hold **CharacteristicSpecifics** that are defined through **Associations** of the specified **SemanticDictionaryItemSet**.
- Confirm the cardinality specified by the **Association**

The formal constraint below is deliberately weak. It states that one is not allowed to define **CharacteristicSpecific** directly or indirectly in a **CharacteristicSet** on **Parameters** for which the definition is one of the elements in the **SemanticDictionaryItemSet**. This does

not take into account the **Association's** cardinality constraint. The reason for this is to support the temporal stages of creation (e.g., one may be in a state where the **CharacteristicSpecifics** have not been defined YET).

**Table 51: OCL expression for CharacteristicSet_SemanticDictionaryItemSet**

```
function allCharacteristicSpecific( cs: CharacteristicSet) =
   cs.members->select( c1 | c1.oclKind=CharacteristicSpecific)->union(
      s.members.select (c2 | c2.oclKind=CharacteristicSet )->collect( m | allCharacteristicSpecifics(
m )))
context CharacteristicSet inv CharacteristicSet_SemanticDictionaryItemSet:
   self.definition.oclKind=SemanticDictionaryItemSet implies (
     self.definition.association.destination->includesAll(
        allCharacteristicSpecific(self)..parameter.definition )
```

# 6. Primitive Types

## 6.1 Introduction to primitive types

The model uses a set of primitive types (a.k.a. “data types”). By default, RDA primitive types do not have individuality (that is, they are considered equal if and only if they have identical values).

We implicitly support the primitive types specified in UML and MOF: In addition, we have added a few enumerated types and primitive structure.

## 6.2 Enumerations

### 6.2.1 Type model

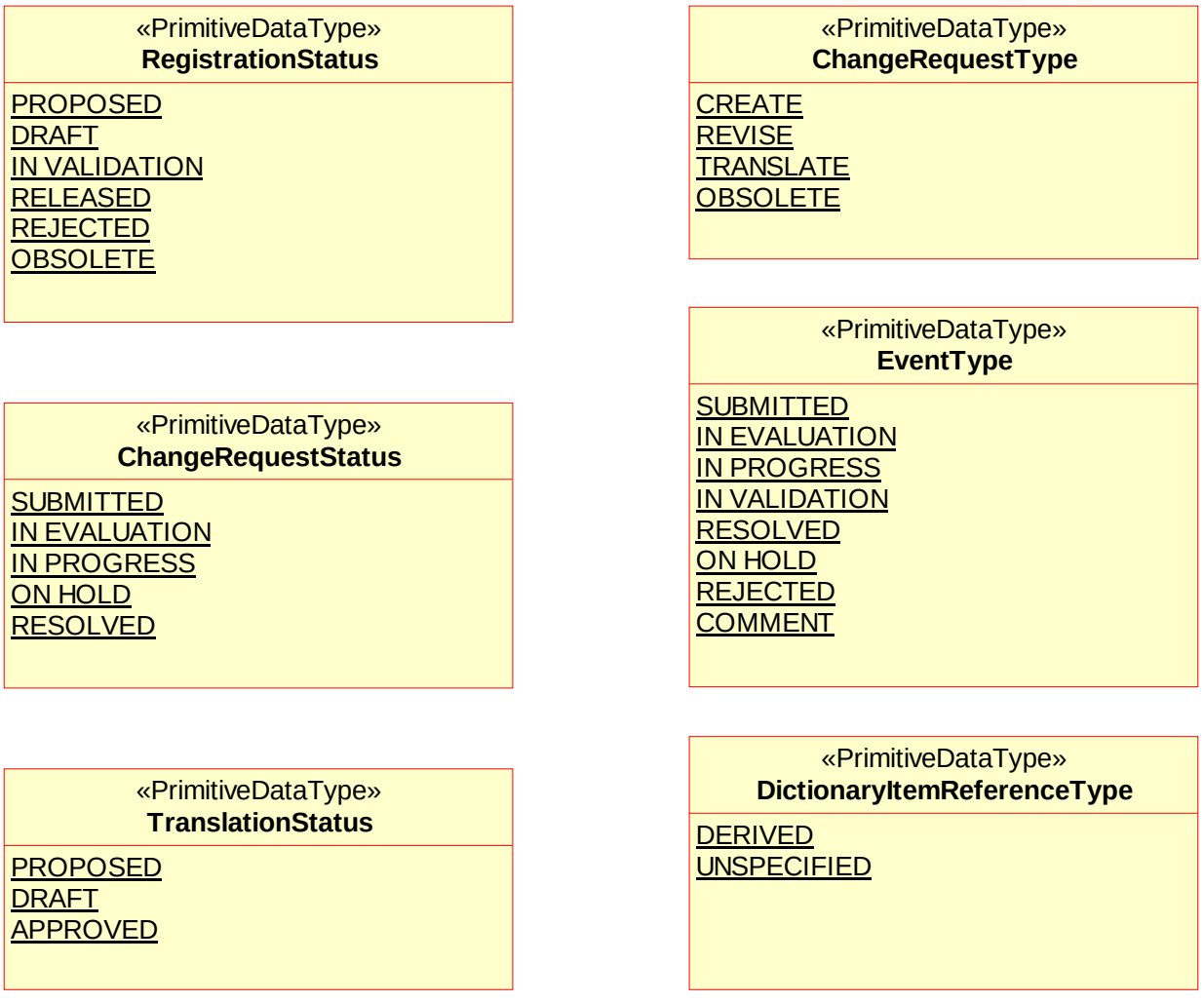



**Fig 26. Type model for “Enumerations”**

### 6.2.2 State diagrams

Three of the enumerations described above represent states depicted in the state charts in this section.

#### 6.2.2.1 DictionaryItem state chart

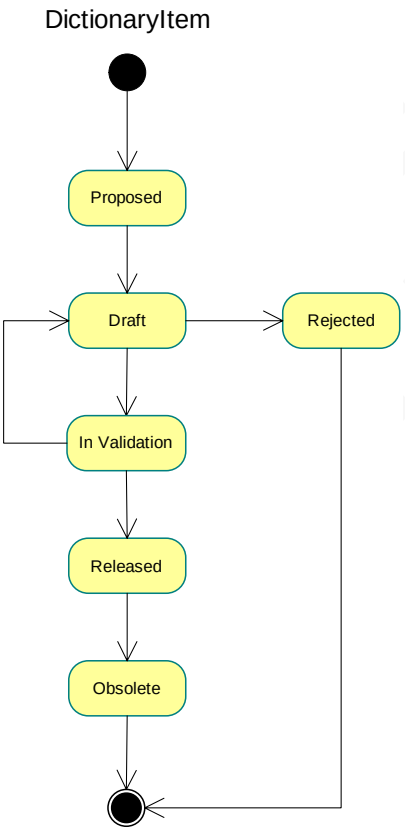



**Fig 27. DictionaryItem state chart**

### 6.2.2.2 ChangeRequest state chart

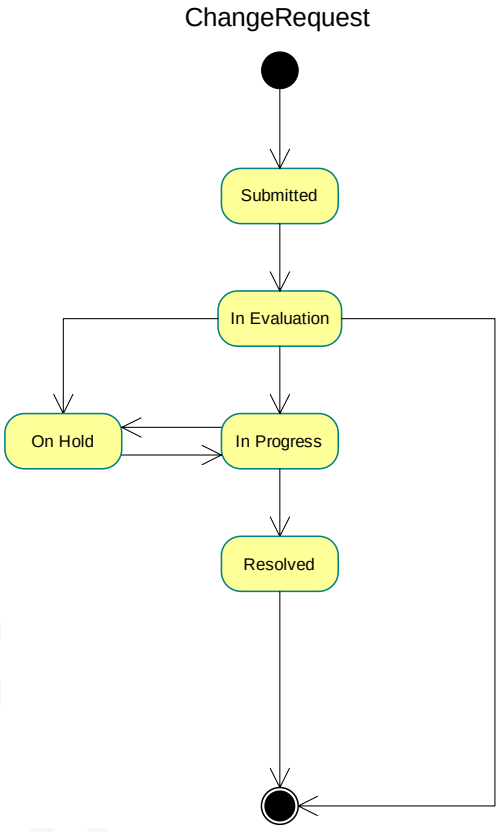



Fig 28. ChangeRequest state chart

#### 6.2.2.3 Translation state chart

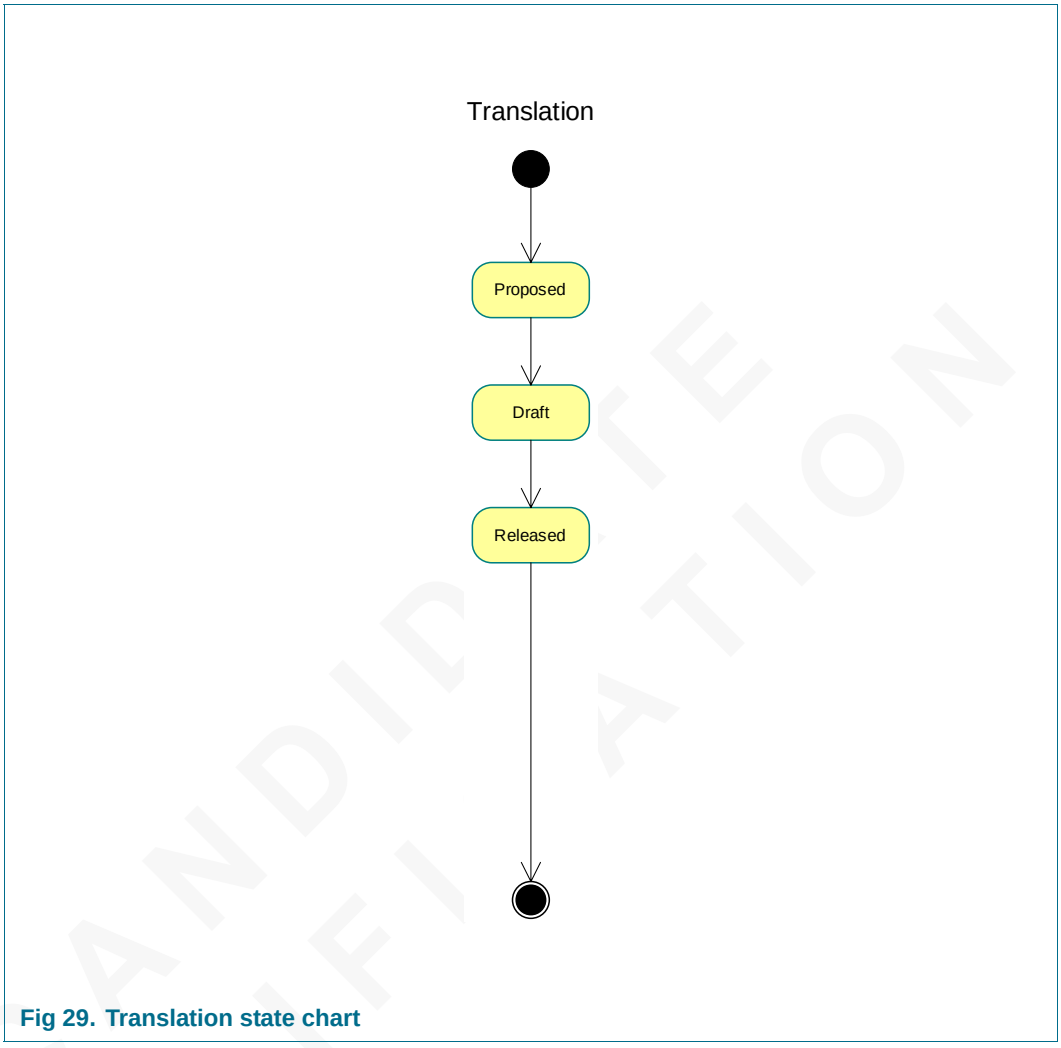



Fig 29. Translation state chart

### 6.2.3 Definition of ENUMERATIONS in the “Enumerations” view

Table 52: Definition of ENUMERATION types

| Enumeration | Definition |
|---|---|
| RegistrationStatus | The registration status projects the possible states of a DictionaryItem with respect to maturity or acceptance. |
| ChangeRequestStatus | Defines the possible states of a request for change. |
| TranslationStatus | Defines the possible states for a translation from the source language to another language. |
| ChangeRequestType | Defines the possible kinds of change requests. |
| EventType | A designator describing a specific step in the evolution of a ChangeRequest |
| SpecificationReference Type | Defines context for a DictionaryItem’s reference to a Specification. This type allows a DictionaryItem to refer to an external specification (Specification) and provide context (both the kind of reference and some description) for the reference. |

**Table 53: ENUMERATION: RegistrationStatus**

| Value | Definition |
|---|---|
| PROPOSED | Status of a DictionaryItem from the time a DictionaryItem has been assigned an identifier until a draft DictionaryItem is available for review and validation |
| DRAFT | Status of a DictionaryItem from the time a draft DictionaryItem is available for review and validation until the status of the DictionaryItem becomes either RELEASED or REJECTED |
| IN VALIDATION | Status of a DictionaryItem while it is being validated |
| RELEASED | Status of a DictionaryItem that will be released for use on the effectiveDate |
| REJECTED | Status of a DictionaryItem that will be not be released for use |
| OBSOLETE | Status of a DictionaryItem that is no longer recommended for use but is still available for reference only |

**Table 54: ENUMERATION: ChangeRequestStatus**

| Value | Definition |
|---|---|
| SUBMITTED | Status of a ChangeRequest from the time it was submitted to the time it has been assigned for evaluation |
| IN EVALUATION | Status of a ChangeRequest during the time it is being evaluated for disposition |
| IN PROGRESS | Status of a ChangeRequest while it is being resolved |
| ON HOLD | Status of a ChangeRequest that is no longer in PROGRESS but not yet RESOLVED |
| RESOLVED | Status of a ChangeRequest for which the final disposition is known |

**Table 55: ENUMERATION: TranslationStatus**

| Value | Definition |
|---|---|
| PROPOSED | Status of a Translation from the time a Translation has been submitted until the time a draft Translation is available for review and validation |
| DRAFT | Status of a Translation from the time a draft Translation is available for review and validation until the status of the Translation becomes RELEASED |
| APPROVED | Status of a Translation that has been approved |

**Table 56: ENUMERATION: ChangeRequestType**

| Value | Definition |
|---|---|
| CREATE | Designator for a ChangeRequest which is submitted with the intent to CREATE one or more DictionaryItems |
| REVISE | Designator for a ChangeRequest which is submitted with the intent to REVISE one or more DictionaryItems |
| TRANSLATE | Designator for a ChangeRequest which is submitted with the intent to TRANSLATE one or more DictionaryItems |
| OBSOLETE | Designator for a ChangeRequest which is submitted with the intent to OBSOLETE one or more DictionaryItems |

**Table 57: ENUMERATION: EventType**

| Value | Definition |
|---|---|
| SUBMITTED | Status of a ChangeRequest from the time it was submitted to the time it has been assigned for evaluation |
| IN EVALUATION | Status of a ChangeRequest during the time it is being evaluated for disposition |
| IN PROGRESS | Status of a ChangeRequest while it is being resolved |
| IN VALIDATION | Status of a ChangeRequest while it is being validated |
| RESOLVED | Status of a ChangeRequest for which the final disposition is known |
| ON HOLD | Status of a ChangeRequest that is no longer in PROGRESS but not yet RESOLVED |
| REJECTED | Status of a ChangeRequest that will not be implemented |
| COMMENT | Event that records a comment at a date and time specified by a DateTimeStamp |

**Table 58: ENUMERATION: SpecificationReferenceType**

| Value | Definition |
|---|---|
| DERIVED | A relation between a SpecificationReference and a DictionaryItem meaning that the DictionaryItem is based upon a DictionaryItem defined in the SpecificationReference |
| UNSPECIFIED | A relation between a SpecificationReference and a DictionaryItem meaning that is of unknown nature |

### 6.2.4 Discussion

The enumerations defined represent attribute types that can hold a fixed set of values where the possible value set is immutable through the lifetime of the specification. The semantic of the values can be defined at specification time and we do not foresee a need for an extendable or mutable semantic.

### 6.2.5 Constraints

There are no known constraints with respect to the enumerated value types.

## 6.3 Text and language support

### 6.3.1 Type model

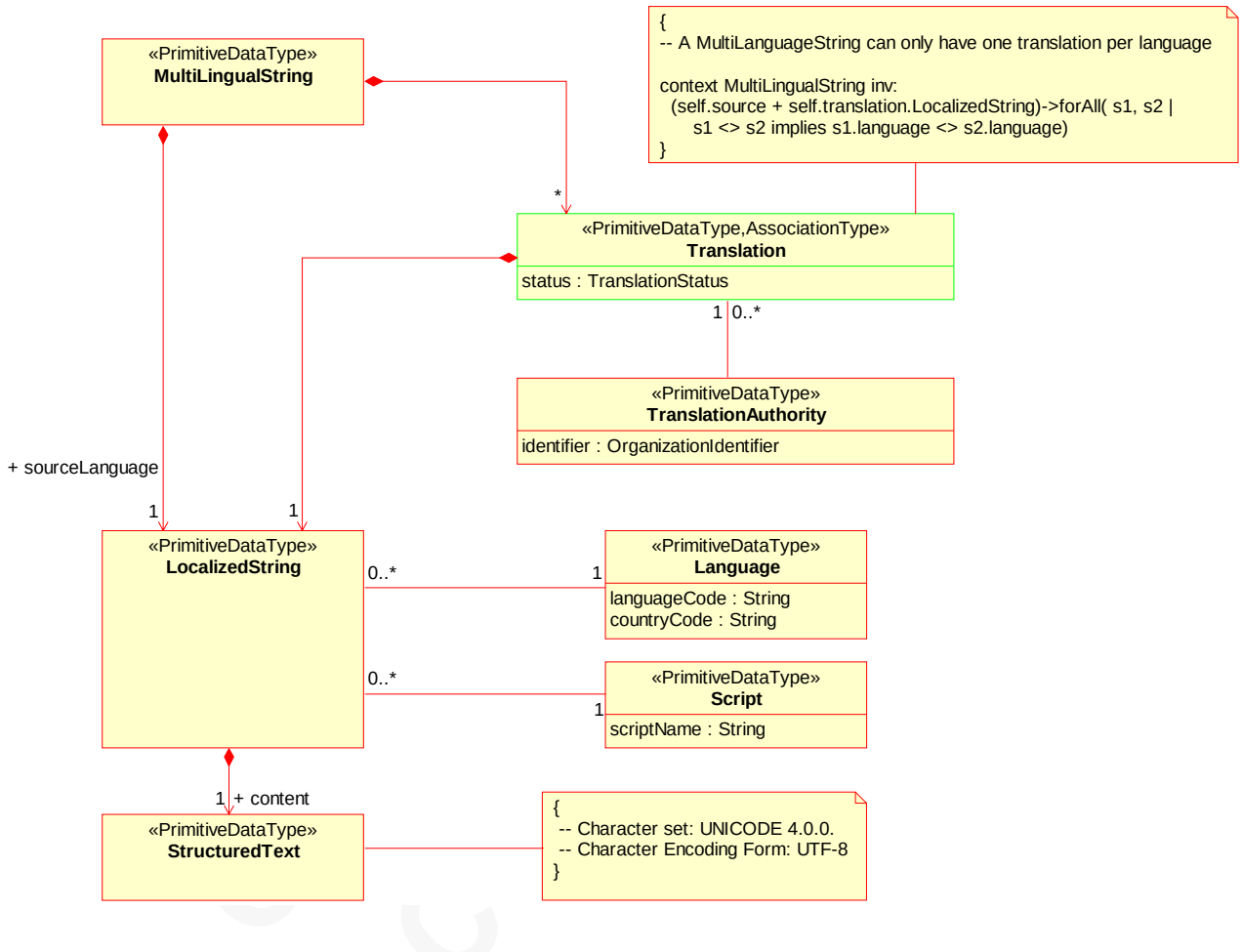



**Fig 30. Type model for “Text & language support”**

### 6.3.2 Definition of CLASSES in the “Text & language support” view

**Table 59: CLASS: MultiLingualString**

| Field | Value |
|---|---|
| Name | MultiLingualString |
| Definition | A primitive type enabling and managing language variants for attributes |
| Note | - |
| Remark | - |

**Table 60: CLASS: LocalizedString**

| Field | Value |
|---|---|
| Name | LocalizedString |
| Definition | A human-readable text expressed in a locale-specific language and script |
| Note | - |
| Remark | - |

**Table 61: CLASS: RichText**

| Field | Value |
|---|---|
| Name | RichText |
| Definition | Text that may have an structure as commonly found in XHTML documents |
| Note | - |
| Remark | RichText is limited to XHTML modules 5.2, 5.4.3, 5.6.1, 5.7 |

**Table 62: CLASS: Translation**

| Field | Value |
|---|---|
| Name | Translation |
| Definition | A rendering from one language into another |
| Note | - |
| Remark | - |

**Table 63: CLASS: TranslationAuthority**

| Field | Value |
|---|---|
| Name | TranslationAuthority |
| Definition | The organizational entity responsible for translating a LocalizedString |
| Note | - |
| Remark | - |

**Table 64: CLASS: Language**

| Field | Value |
|---|---|
| Name | Language |
| Definition | The words, their pronunciation, and the methods of combining them used and understood by a community |
| Note | - |
| Remark | - |

**Table 65: CLASS: Script**

| Field | Value |
|---|---|
| Name | Script |

**Table 65: CLASS: Script**

| Field | Value |
|---|---|
| Definition | A collection of characters for displaying written text, all of which have a common characteristic that justifies their consideration as a distinct set. |
| Note | - |
| Remark | One script can be used for several different languages (for example, Latin script, which covers all of Western Europe). Some written languages require multiple scripts (for example, Japanese, which requires at least three scripts: the hiragana and katakana syllabaries and the kanji ideographs imported from China). |

### 6.3.3 Discussion

The MultiLingualString represents the a datatype that contains some amount of text in at least one language. The MultiLingualString specifies that attributes of this type will contain some text in a specified source language and multiple translations to various other languages. For each translation, we need to know who made the translation (a TranslationAuthority).

This effectively means that every place where we have specified an attribute to be of type MultiLingualString, we expect to track the content with support for internationalization. Each one of the texts (LocalizedText) tracks its language and its script.

### 6.3.4 Constraints

#### 6.3.4.1 Only One Localized String per Language

Every translation of some text must be in a different language and their languages must be different than the source language.

**Table 66: OCL expression for OnlyOneLocalizedStringPerLanguage**

```
function allChildren( cs: CharacteristicSet) =
   cs.members->union( s.members.collect ( m | allChildren( m ) ) )
context MultiLingualString inv OnlyOneLocalizedStringPerLanguage:
   (self.sourceLanguage + self.translation.localizedString)->forAll( s1, s2 |
      s1 <> s2 implies s1.language <> s2.language )
```

## 6.4 Specialized primitives

### 6.4.1 Type model

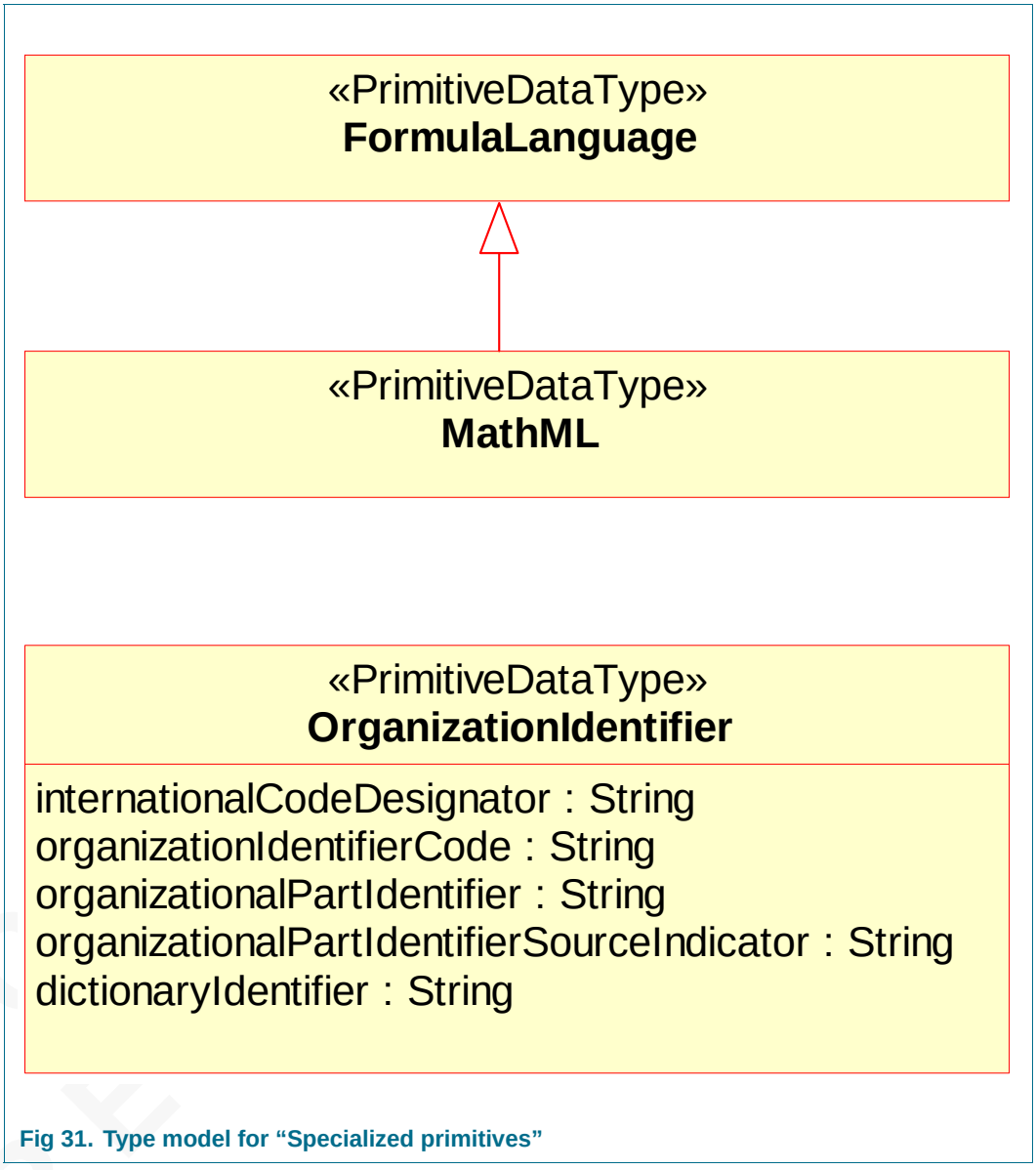



Fig 31. Type model for “Specialized primitives”

### 6.4.2 Definition of CLASSES in the “Specialized primitives” view

Table 67: CLASS: FormulaLanguage

| Field | Value |
|---|---|
| Name | FormulaLanguage |
| Definition | A formally defined method to encode Formulas |
| Note | - |
| Remark | - |

**Table 68: CLASS: MathML**

| Field | Value |
|---|---|
| Name | MathML |
| Definition | Mathematical Markup Language |
| Note | The markup must follow Presentation Markup rules as defined in http://www.w3.org/TR/2003/REC-MathML2-20031021/ |
| Remark | - |

**Table 69: CLASS: OrganizationIdentifier**

| Field | Value |
|---|---|
| Name | OrganizationIdentifier |
| Definition | A Primitive type for uniquely identifying an organization |
| Note | The OrganizationIdentifier is globally unique |
| Remark | NOTE: delimiter between the next 5 attributes is COLON ":" |

### 6.4.3 Discussion

This section contains a few primitives that are simple in structure and that are not enumerated types.

These are simple structures of attributes that appear together as structure or types that we know exist where we want an early designation and let the implementers make decisions within the boundaries of the designation.

The **OrganizationIdentifier** represents a simple structure of simple attributes that together makes up sufficient information to identify an organization.

The **FormulaLanguage** and **MathML** are early designation types.

When using **FormulaLanguage** in the specification, we have identified a need to express a formula; however no preference as to the exact implementation has been identified. Implementers should refer to the definition of the individual attributes for further guidance

A special case of a **FormulaLanguage** is **MathML**. This attribute type has been used when there is a strong preference for using MathML. Again, the individual attributes provide further context.

### 6.4.4 Constraints

There are no known constraints in this section.

## 6.5 Values

### 6.5.1 Type model

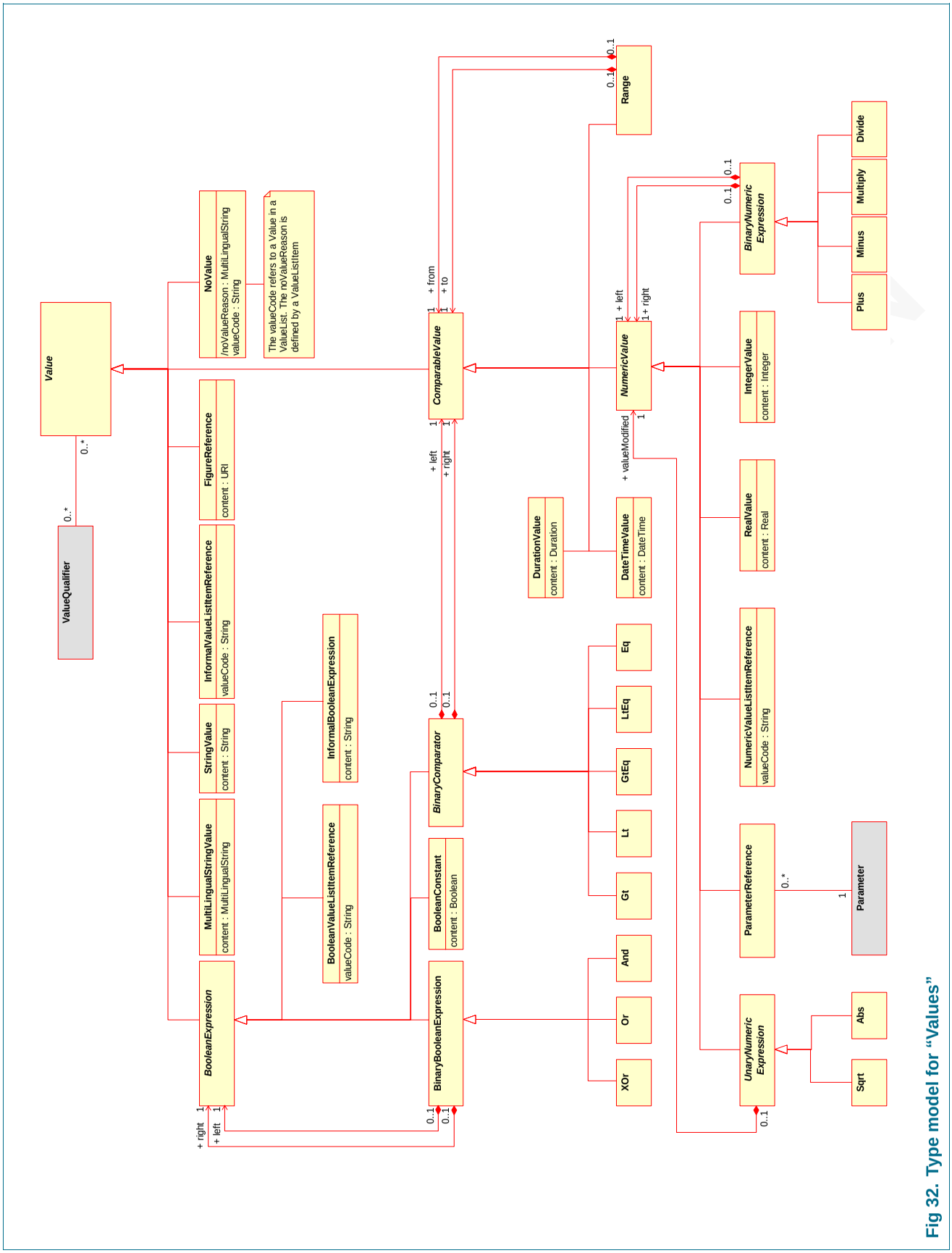



Fig 32. Type model for "Values"

### 6.5.2 Definition of CLASSES in the “Values” view

**Table 70: CLASS: ParameterReference**

| Field | Value |
|---|---|
| Name | ParameterReference |
| Definition | Allows a Parameter to be used as a variable in an expression |
| Note | - |
| Remark | - |

**Table 71: CLASS: FigureReference**

| Field | Value |
|---|---|
| Name | FigureReference |
| Definition | Allows a Figure to be used as a Value |
| Note | - |
| Remark | May be used to point to any external object |

**Table 72: CLASS: Value**

| Field | Value |
|---|---|
| Name | Value |
| Definition | A quantity represented by a symbol or set of symbols |
| Note | - |
| Remark | The Value type is an modeling artifact capturing commonality between all concrete values |

**Table 73: CLASS: NoValue**

| Field | Value |
|---|---|
| Name | NoValue |
| Definition | A NULL Value for which the reason is explicitly stated |
| Note | - |
| Remark | The Value type is an modeling artifact capturing commonality between all concrete values |

**Table 74: CLASS: NumericValue**

| Field | Value |
|---|---|
| Name | NumericValue |
| Definition | A Value of the type Numeric |
| Note | - |
| Remark | - |

**Table 75: CLASS: StringValue**

| Field | Value |
|---|---|
| Name | StringValue |
| Definition | A Value of the type String |
| Note | - |
| Remark | - |

**Table 76: CLASS: DurationValue**

| Field | Value |
|---|---|
| Name | DurationValue |
| Definition | A Value of the type DateTime |
| Note | - |
| Remark | - |

**Table 77: CLASS: DateTimeValue**

| Field | Value |
|---|---|
| Name | DateTimeValue |
| Definition | A Value of the type Duration |
| Note | - |
| Remark | - |

**Table 78: CLASS: RealValue**

| Field | Value |
|---|---|
| Name | RealValue |
| Definition | A Value of the type Real |
| Note | - |
| Remark | - |

**Table 79: CLASS: IntegerValue**

| Field | Value |
|---|---|
| Name | IntegerValue |
| Definition | A Value of the type Integer |
| Note | - |
| Remark | - |

**Table 80: CLASS: Range**

| Field | Value |
|---|---|
| Name | Range |
| Definition | A Value of the type Range |
| Note | - |
| Remark | - |

**Table 81: CLASS: BooleanConstant**

| Field | Value |
|---|---|
| Name | BooleanConstant |
| Definition | A Constant of the type Boolean |
| Note | - |
| Remark | - |

**Table 82: CLASS: BooleanExpression**

| Field | Value |
|---|---|
| Name | BooleanExpression |
| Definition | A mathematical expression in which all variables involved are either 0 or 1; it evaluates to either 0 or 1 |
| Note | - |
| Remark | - |

**Table 83: CLASS: InformalBooleanExpression**

| Field | Value |
|---|---|
| Name | InformalBooleanExpression |
| Definition | Represents a value that requires a human to interpret but that conceptually yields a Boolean |
| Note | - |
| Remark | - |

**Table 84: CLASS: BinaryBooleanExpression**

| Field | Value |
|---|---|
| Name | BinaryBooleanExpression |
| Definition | An abstract type for a Boolean expression with two operands |
| Note | - |
| Remark | Supported functions are AND, OR, XOR |

**Table 85: CLASS: BinaryNumericExpression**

| Field | Value |
|---|---|
| Name | BinaryNumericExpression |
| Definition | An abstract type for a Numeric expression with two operands |
| Note | - |
| Remark | Supported functions are PLUS, MINUS, MULTIPLY, DIVIDE |

**Table 86: CLASS: UnaryNumericExpression**

| Field | Value |
|---|---|
| Name | UnaryNumericExpression |
| Definition | An abstract type for a Numeric expression with one operand |
| Note | - |
| Remark | Supported functions are ABS, SQRT |

**Table 87: CLASS: ComparableValue**

| Field | Value |
|---|---|
| Name | ComparableValue |
| Definition | An abstract SuperType representing a value that can be compared to another value of the same type. |
| Note | - |
| Remark | - |

**Table 88: CLASS: BinaryComparator**

| Field | Value |
|---|---|
| Name | BinaryComparator |
| Definition | An abstract type for a Boolean expression that compares two operands |
| Note | - |
| Remark | Supported functions are LT, GT, LTEQ, GTEQ, EQ |

**Table 89: CLASS: Xor**

| Field | Value |
|---|---|
| Name | Xor |
| Definition | The binary operator Exclusive OR |
| Note | - |
| Remark | - |

**Table 90: CLASS: Or**

| Field | Value |
|---|---|
| Name | Or |
| Definition | The binary operator OR |
| Note | - |
| Remark | - |

**Table 91: CLASS: And**

| Field | Value |
|---|---|
| Name | And |
| Definition | The binary operator AND |
| Note | - |
| Remark | - |

**Table 92: CLASS: Gt**

| Field | Value |
|---|---|
| Name | Gt |
| Definition | The binary operator GREATER THAN |
| Note | - |
| Remark | - |

**Table 93: CLASS: Lt**

| Field | Value |
|---|---|
| Name | Lt |
| Definition | The binary operator LESS THAN |
| Note | - |
| Remark | - |

**Table 94: CLASS: GtEq**

| Field | Value |
|---|---|
| Name | GtEq |
| Definition | The binary operator GREATER OR EQUAL THAN |
| Note | - |
| Remark | - |

**Table 95: CLASS: LtEq**

| Field | Value |
|---|---|
| Name | LtEq |
| Definition | The binary operator LESS OR EQUAL THAN |
| Note | - |
| Remark | - |

**Table 96: CLASS: Eq**

| Field | Value |
|---|---|
| Name | Eq |
| Definition | The binary operator EQUAL |
| Note | - |
| Remark | - |

**Table 97: CLASS: Plus**

| Field | Value |
|---|---|
| Name | Plus |
| Definition | The arithmetic operator PLUS |
| Note | - |
| Remark | - |

**Table 98: CLASS: Minus**

| Field | Value |
|---|---|
| Name | Minus |
| Definition | The arithmetic operator MINUS |
| Note | - |
| Remark | - |

**Table 99: CLASS: Multiply**

| Field | Value |
|---|---|
| Name | Multiply |
| Definition | The arithmetic operator MULTIPLY |
| Note | - |
| Remark | - |

**Table 100: CLASS: Divide**

| Field | Value |
|---|---|
| Name | Divide |
| Definition | The arithmetic operator DIVIDE |
| Note | - |
| Remark | - |

**Table 101: CLASS: Sqrt**

| Field | Value |
|---|---|
| Name | Sqrt |
| Definition | The arithmetic operator SQUARE ROOT |
| Note | - |
| Remark | - |

**Table 102: CLASS: Abs**

| Field | Value |
|---|---|
| Name | Abs |
| Definition | The arithmetic operator ABOLUTE VALUE |
| Note | - |
| Remark | - |

**Table 103: CLASS: BooleanValueListItemReference**

| Field | Value |
|---|---|
| Name | BooleanValueListItemReference |
| Definition | Reference to a ValueListItem of type Boolean |
| Note | - |
| Remark | - |

**Table 104: CLASS: InformalValueListItemReference**

| Field | Value |
|---|---|
| Name | InformalValueListItemReference |
| Definition | Reference to a ValueListItem that represents a Value unsuitable for use in an expression |
| Note | - |
| Remark | - |

**Table 105: CLASS: NumericValueListItemReference**

| Field | Value |
|---|---|
| Name | NumericValueListItemReference |
| Definition | Reference to a ValueListItem of type Numeric |
| Note | - |
| Remark | - |

### 6.5.3 Discussion

#### 6.5.3.1 Introduction

The value model captures the structure for the parametric values reported in item specifications or provided with a **ValueListItem** in the dictionary.

The **Value** model captures expressiveness required by an expression language to provide the required computer sensible semantic of item specifications

A likely implementation is the use of a standard expression language (e.g., MathML). The model captures what part of a selected expression language is required.

**Justification for the Value model:** During the development of this specification, questions have been asked of why we have elected to model the values explicitly. The main discussion has been around the question: **Could we not just state that we'll use MathML?** After all, we are pretty sure we'll end up using MathML in the implementation..

**The problem with MathML is that it is more expressive than desirable and it does not allow us to express everything we want.**

**Why model the Values?:** MathML is an XML schema that allows expression of any mathematical construct. We want a very small subset of this expression power. It is essential that we agree upon this subset such that the trading partners know exactly what to expect when exchanging information.

We may, of course, decide to use MathML in an implementation and write a document specifying what part of MathML to use. The Value model then provides the requirements for this document.

Another problem with MathML is that it does not provide two particular features that we need.

- Express values relative to a **Parameter**
- Express values stored in the dictionary as **ValueList**

The implementers may select to extend MathML with these capabilities. Again, the Value model is now essential to provide the requirements for the extension.

#### 6.5.3.2 Simple values

The model enumerates the simple kind of values required. What we consider as simple **Values** are:

- BooleanValue
- StringValue
- RealValue
- IntegerValue

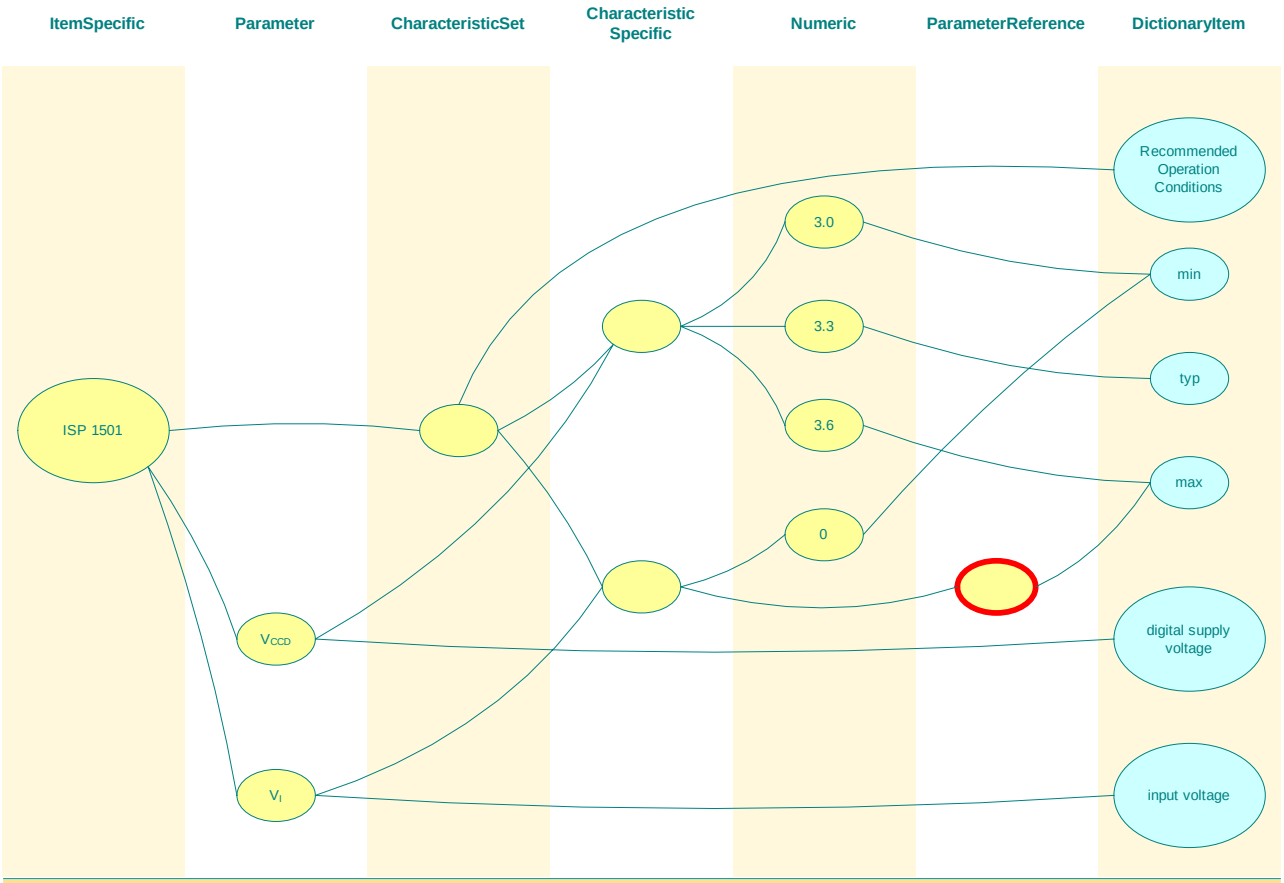
These simple values can be used in isolation, or may be used as building blocks for more complex expressions.

#### 6.5.3.3 Parameter Reference

A **Value** is sometimes reported to be the same as some **Parameter** on an **Item**. To enable this kind of reporting, we allow for a special kind of **Numeric**; **ParameterReference**.

To understand the need for this, let's look at an example taken from a product; Universal Serial Bus 2.0 peripheral transceiver, ISP1501

In this example we see that the max value of $V_I$ is reported to be that of $V_{CCD}$. In an UML instance diagram conforming to the model, this would be reported as follows




| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $V_{CCA}$ | analog supply voltage | | 3.0 | 3.3 | 3.6 | V |
| $V_{CCD}$ | digital supply voltage | | 3.0 | 3.3 | 3.6 | V |
| $V_I$ | input voltage | | 0 | - | $V_{CCD}$ | V |

**Fig 33. Example of ParameterReference**

### 6.5.3.4 Expressions

In reporting parametric data one often has to report values as expressions. An expression often use references and simple values combined in arbitrary complex ways.

We will not go into great details of expressions here. We assume that the selected expression language in the implementation may bring about some compromises. However, we will show a couple of examples of where expressions are used.

Table 8: Static characteristics
*At recommended operating conditions; voltages are referenced to GND (ground = 0 V).*

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $T_{amb}$ = −40 °C to +85 °C [1] | | | | | | |
| $V_{IH}$ | HIGH-level input voltage | $V_{CC}$ = 1.65 V to 1.95 V | 0.65 × $V_{CC}$ | - | - | V |
| | | $V_{CC}$ = 2.3 V to 2.7 V | 1.7 | - | - | V |
| | | $V_{CC}$ = 2.7 V to 3.6 V | 2.0 | - | - | V |
| | | $V_{CC}$ = 4.5 V to 5.5 V | 0.7 × $V_{CC}$ | - | - | V |

**Fig 34. Static characteristics for product 74LVC2G66**

In the above example expressions are used for two purposes:

- To define conditions.
  E.g. $T_{amb}$ is greater than -40 C and lower than 125 C
- To report values relative to some **Parameter**.
  E.g., $V_{IH}$ min is 0.6 x $V_{CC}$

## 6.5.4 Constraints

### 6.5.4.1 Values Can Have Only One Parent

A value is linked to only one parent.

**Table 106: OCL expression for ValueOnlyOneParent**

```
context Value inv ValueOnlyOneParent:
   self.oclKindOf=BooleanExpression implies (
      self.~left->size + self.~right.size+
      self.~reportedValues->size+self.~condition= 1 ) AND
   self.oclKindOf=Comparable implies (
      self.~left->size + self.~right.size+self.~from+self.~to+
      self.~reportedValues+self..~condition->size <=1)
```

### 6.5.4.2 Range Between Two Different Comparable

A range must be between two different values.

**Table 107: OCL expression for RangeBetweenTwoDifferentValues**

```
context Range inv RangeBetweenTwoDifferentValues:
   self.from <> self.to
```

### 6.5.4.3 BinaryComparator Between Two Different Comparable

A comparison must compare two different values.

**Table 108: OCL expression for BinaryComparator_LeftDifferentFromRight**

```
context BinaryComparitor inv BinaryComparator_LeftDifferentFromRight:
   self.left <> self.right
```

# 7. RosettaNet EC Extensions

## 7.1 Electronic Component (EC) Dictionary Extensions

### 7.1.1 Type model

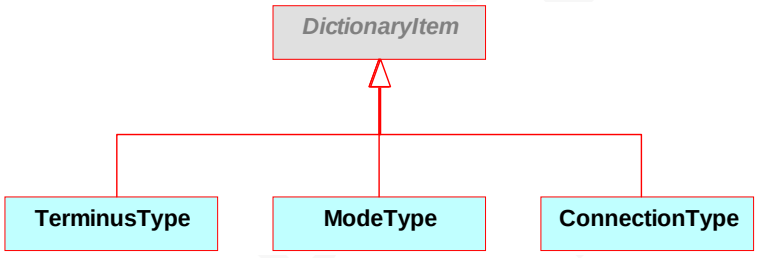



Fig 35. Type model for EC Dictionary Extensions

### 7.1.2 Definition of CLASSES in the “EC Dictionary Extensions” view

**Table 109: CLASS: TerminusType**

| Field | Value |
|---|---|
| Name | TerminusType |
| Definition | A DictionaryItem defining metadata for a Terminus |
| Note | - |
| Remark | - |

**Table 110: CLASS: ModeType**

| Field | Value |
|---|---|
| Name | ModeType |
| Definition | A DictionaryItem defining metadata for a Mode |
| Note | - |
| Remark | - |

**Table 111: CLASS: ConnectionType**

| Field | Value |
|---|---|
| Name | ConnectionType |
| Definition | A DictionaryItem defining metadata for a Connection |
| Note | - |
| Remark | - |

### 7.1.3 Discussion

The EC domain requires three new metadata types that were not defined in the core RDA. These types organize the metadata for terminus, mode and connection information specified in Section 7.2 “Electronic Component (EC) Library Extensions” on page 75.

### 7.1.4 Constraints

There are no know constraints in this section.

## 7.2 Electronic Component (EC) Library Extensions

### 7.2.1 Type model

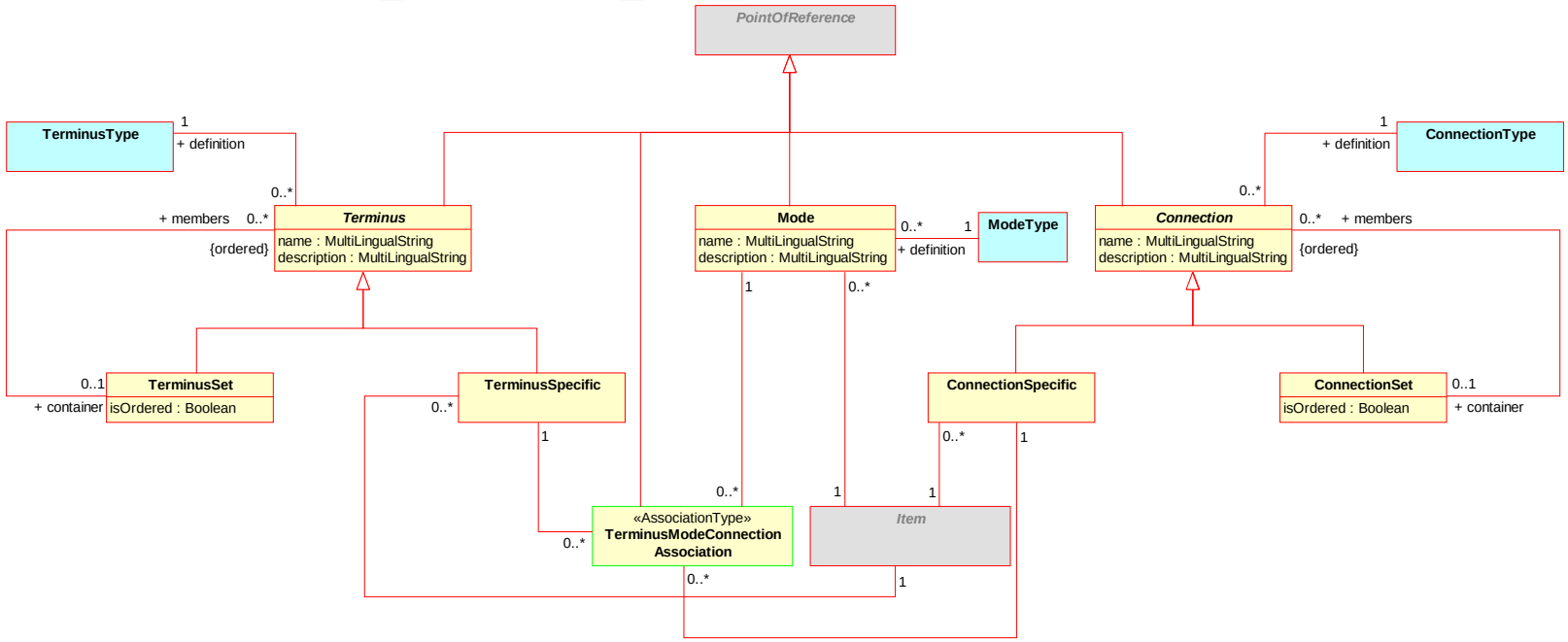



Fig 36. Type model for “EC Domain library extensions”

### 7.2.2 Definition of CLASSES in the “EC Library Extensions” view

**Table 112: CLASS: TerminusSpecific**

| Field | Value |
|---|---|
| Name | TerminusSpecific |
| Definition | A physical connection to an Item |
| Note | - |
| Remark | - |

**Table 113: CLASS: TerminusSet**

| Field | Value |
|---|---|
| Name | TerminusSet |
| Definition | A group of Termini |
| Note | - |
| Remark | - |

**Table 114: CLASS: Terminus**

| Field | Value |
|---|---|
| Name | Terminus |
| Definition | An abstract SuperType defining the commonality between TerminusSpecific and TerminusSet |
| Note | - |
| Remark | In the Electronic Component domain common alternate names are pin, ball and lead |

**Table 115: CLASS: ConnectionSpecific**

| Field | Value |
|---|---|
| Name | ConnectionSpecific |
| Definition | A logical connection to an Item |
| Note | - |
| Remark | - |

**Table 116: CLASS: ConnectionSet**

| Field | Value |
|---|---|
| Name | ConnectionSet |
| Definition | A group of Connections |
| Note | - |
| Remark | - |

**Table 117: CLASS: Connection**

| Field | Value |
|---|---|
| Name | Connection |

**Table 117: CLASS: Connection**

| Field | Value |
|---|---|
| Definition | An abstract SuperType defining the commonality between ConnectionSpecific and ConnectionSet |
| Note | - |
| Remark | - |

**Table 118: CLASS: Mode**

| Field | Value |
|---|---|
| Name | Mode |
| Definition | The state of an Item |
| Note | - |
| Remark | - |

**Table 119: CLASS: TerminusModeConnectionAssociation**

| Field | Value |
|---|---|
| Name | TerminusModeConnectionAssociation |
| Definition | An association class linking a TerminusSpecific, a ConnectionSpecific and a Mode |
| Note | - |
| Remark | - |

## 7.2.3 Discussion

### 7.2.3.1 Introduction

The terminus, modes and connection view shows a model for capturing knowledge which is reported on a particular **Terminus**[9], or a particular **Connection**, or in a particular **Mode** or on the combination of these called **TerminusModeConnectionAssociation** or **TMCA**.

To understand this model, the reader must understand the difference between **Terminus** and **Connection**. The objects are described in detail in Section 10.4 “RosettaNet EC Domain Extensions” on page 234

In short, the difference between a **Terminus** and a **Connection** is that the **Terminus** represents a physical location on a device, while the **Connection** represents the signal connected to this physical location. The **Mode** represents a state of operation of a device.

The **TMCA** represents a combination of one **Mode**, one **Connection**, and one **Terminus**, e.g., a signal on a particular pin when a device is in a particular **Mode**.

A classical example of this kind of structure can be found in pinning diagrams for semiconductor devices. The device is represented as an **Item**. The connections may change dynamically as the device changes mode (a.k.a. a state change).

An information provider may want to specify parameters localized to either of these type of objects (**Terminus**, **Mode**, **Connection**, **TerminusModeConnectionAssociation**).

9. A frequently asked question why this object is not simply called “pin”. The answer is that although semiconductor devices frequently have pins, this is not always an appropriate name as the Terminus can also manifest itself as a ball, pad, ring or other artifact. Hence the more generic name “terminus”.

### 7.2.3.2 A TMCA Example

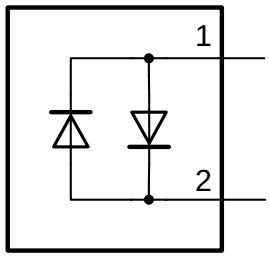



Fig 37. Informal schematic of a bi-color LED

Table 120: Pin map for a bi-color LED

| Mode | Pin 1 | Pin 2 |
|---|---|---|
| Red | **Anode** | **Cathode** |
| Green | **Cathode** | **Anode** |

This example is modeled with the following objects:

- **Terminus**:

  “Pin 1” and “Pin 2” are termini. They represent the physical connection points for the device. A terminus is represented as column 1 or 2 in the above table.

- **Connections**:

  “Anode” and “Cathode” are connections. They represent the signal on the terminuses. A Connection is represented as a unique name in the header of the above table.

- **Mode**:

  “Red” and “Green” are modes. They represent modes of operation for the device. A Mode is represented as row in table above

- **TerminusModeConnecctionAssociation**:

  There are four TMCAs. These objects represents a link between the Connection, Mode, and Terminus. In above table the TMCAs are represented by one of the four highlighted cells.

This example can be represented in a UML instance diagram (simplified to only involve the objects from this view) as follows:

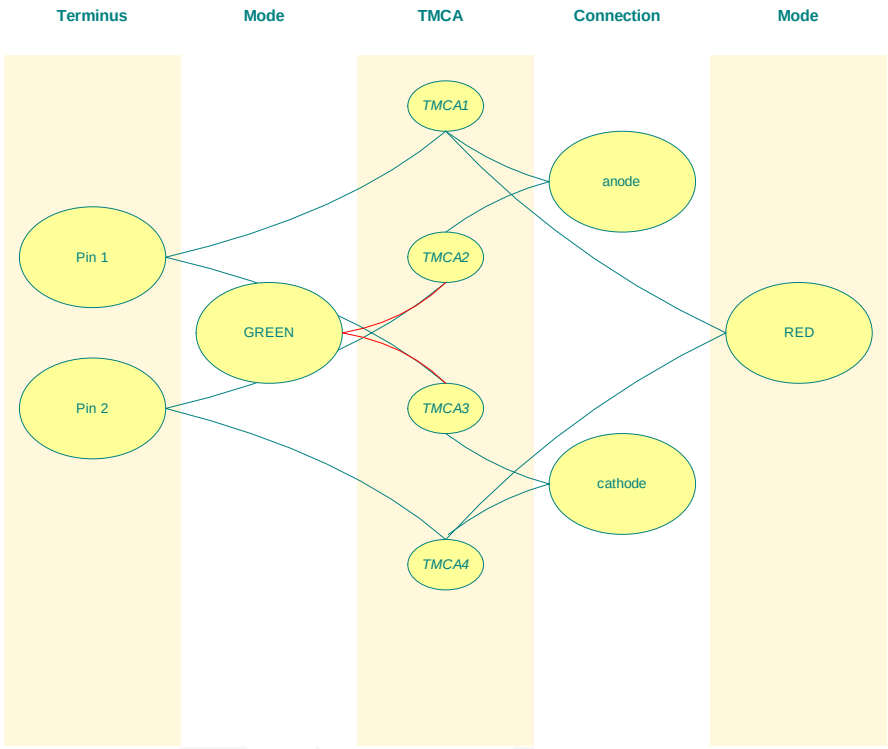



Fig 38. UML Instance Diagram for a bi-color LED

## 7.2.4 Constraints

### 7.2.4.1 Terminus Set Self Exclusion

An instance of **TerminusSet** can not have itself as a child (directly or indirectly). In other words, the item inheritance tree must be acyclic.

Table 121: OCL expression for TerminusSetSelfExclusion

```
function allChildren( ts: TerminusSet ) =
   is.members>union( ts.members->collect( c | allChildren(c) ) )
context TerminusSet inv TerminusSetSelfExclusion:
   allChildren(self)->excludes( self )
```

### 7.2.4.2 Connection Set Self Exclusion

An instance of **ConnectionSet** can not have itself as a child (directly or indirectly). In other words, the item inheritance tree must be acyclic.

Table 122: OCL expression for ConnectionSetSelfExclusion

```
function allChildren( cs: ConnectionSet ) =
   is.members>union( cs.members->collect( c | allChildren(c) ) )
context ConnectionSet inv TerminusSetSelfExclusion:
   allChildren(self)->excludes( self )
```

#### 7.2.4.3 TerminusModeConnectionAssociation Must Connect Same Item

A TerminusModeConnectionAssocaition must connect up termini, modes and connections on the same item.

**Table 123: OCL expression for TerminusModeConnectionAssociationMustConnectSameItem**

```
context TerminusModeConnectionAssociation inv
                    TerminusModeConnectionAssociationMustConnectSameItem:
  self.connectionSpecific.item = self.terminusSpecific.item = self.mode.item
```

#### 7.2.4.4 ConnectionSet must be for Same Item

All connections in a connection set must be linked to the same item.

**Table 124: OCL expression for ConnectionSetSameItem**

```
context ConnectionSet inv ConnectionSetSameItem:
  self.members->forAll( i1, i2 | NOT(i1.item->empty or i2.item->empty) implies i1.item = i2.item )
```

#### 7.2.4.5 TerminusSet must be for Same Item

All termini in a terminus set must be linked to the same item.

**Table 125: OCL expression for TerminusSetSameItem**

```
context TerminusSet inv TerminusSetSameItem:
  self.members->forAll( i1, i2 | NOT(i1.item->empty or i2.item->empty) implies i1.item = i2.item )
```

# 8. Annotated examples

## 8.1 Product Information Object (PIO)

One specific use of **Item** would be to describe semiconductor product **datasheets**. Most of this specification has been focused on the use of **Items** to describe products. This section aims at explaining how the same information structure can be used to specify documents related to products - a PIO or Product Information Object. As a specific example, we'll use a datasheet.

The example chosen is based on the product 74LVC1G57, "Low-power configurable multiple function gate". We want to report that the product 74LVC1G57 has a datasheet. The datasheet has an document number and a URL where the datasheet can be downloaded.

Notice that both the product 74LVC1G57 and its datasheet are instances of **Items**. The have been linked together by an instance of **ItemRelation**. The details of the **RelationType** are omitted from the instance diagram, but for semantic correctness, the **RelationType** called "Data Sheet" should define two roles:

- **Role A**: The owning product
- **Role B**: The datasheet

The datasheet has three parameters each with a corresponding **Characteristic** and **Value**:

- The date of release = September 6, 2004
- The document number = 9397 750 13722
- The download URL = "http://www.semiconductors.philips.com/acrobat_download/datasheets/74LVC1G57_1.pdf"

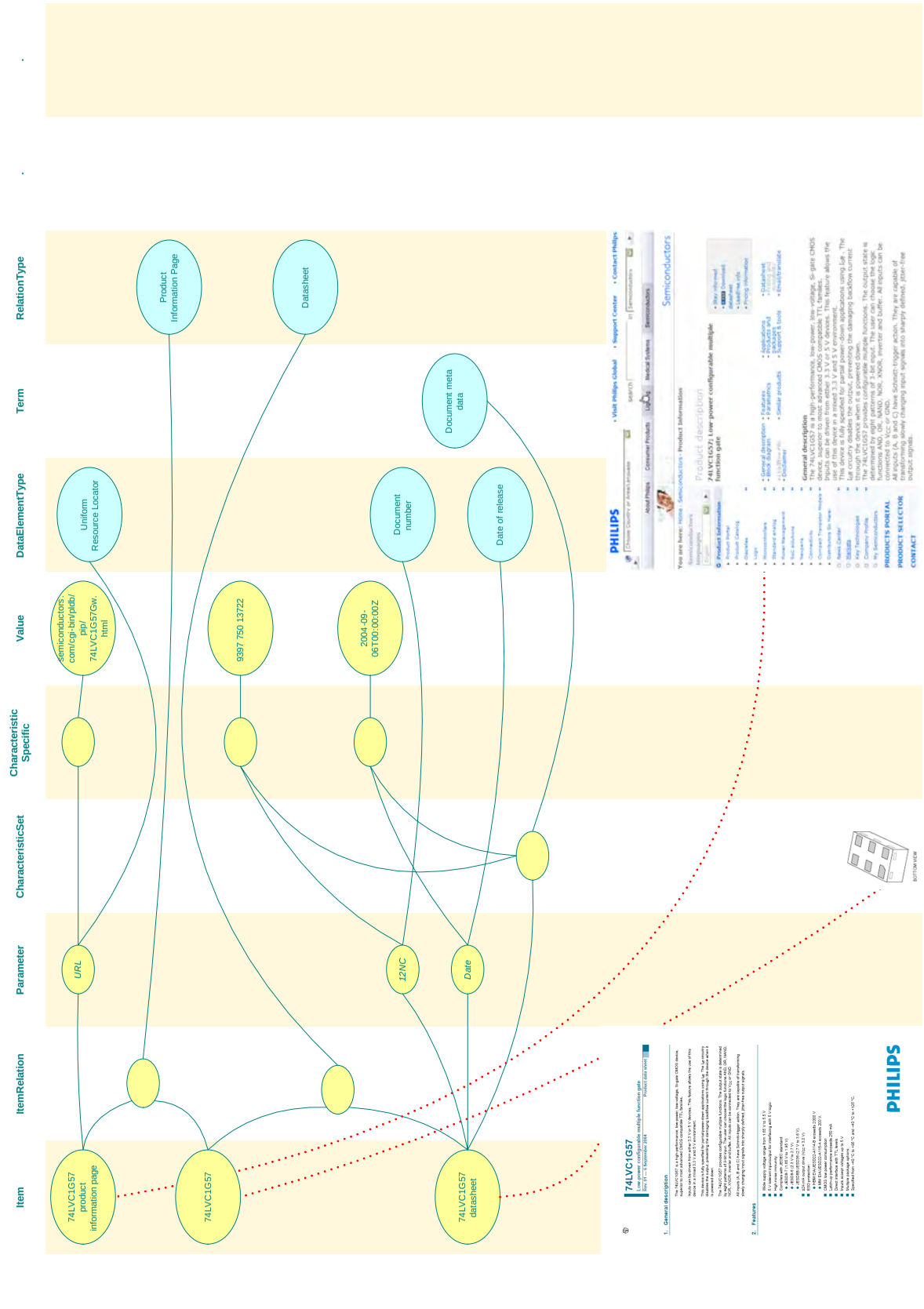



Fig 39. Product Information Objects: example of "ItemRelation"

Reporting datasheets may well require a formal structure. For example, a previous more explicit model[10] required all exchanged data sheets to report the following values:

- Name of document
- File size
- Last modification date
- Version
- When effective
- File format
- Content type

To enforce such structure, we would have to define a **SemanticDictionaryItemSet** and report the **Characteristics** based on the rules defined there.

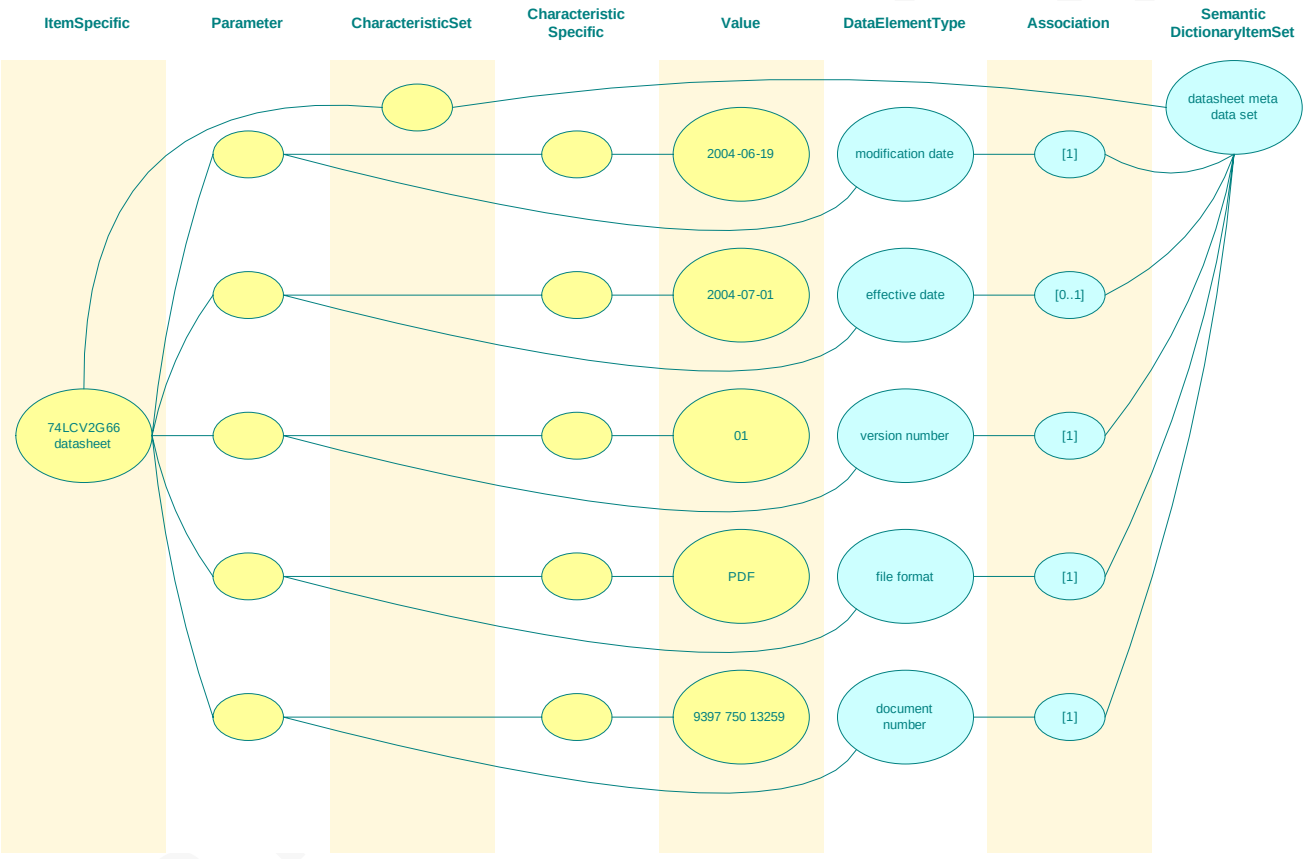



**Fig 40. Use of SemanticDictionaryItemSet for reporting datasheet metadata**

10. An earlier ECIX and RosettaNet model had the datasheets modeled as Product Information Objects. This required a set of attributes to be tracked for all such objects. For continuity, we've elected to specify how this structure would be modeled with the current model

## 8.2 Export control information

In this example the dictionary contains an instance of **SemanticDictionaryItemSet** defining the structure for “export control information” The dictionary specifies the semantic of the structure and what the rule for its composite parts. The rules are:

- Must contain exactly one “Harmonized Tariff Scheduled Code”
- Must contain exactly one “Export Classification Control Number”
- May have a disclaimer

Notice that the conforming instance provided do specify “Harmonized Tariff Scheduled Code” and “Export Classification Control Number”, but never provide a disclaimer. However, a disclaimer is provided for an attached item. This use of disclaimer is unrelated to the use of the “Export control information”

This example shows how a dictionary item may be used as part of a composite and by itself.

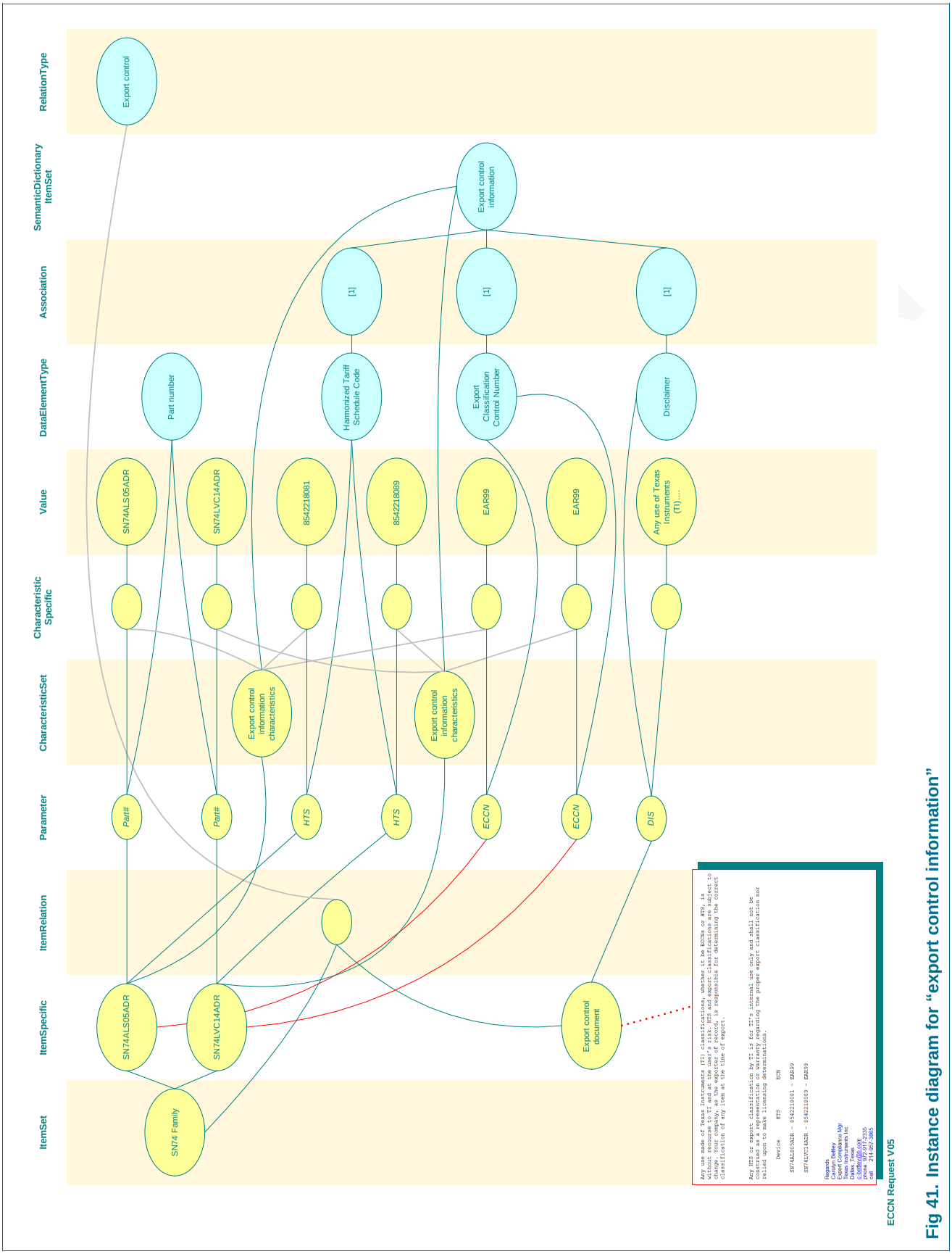



Fig 41. Instance diagram for “export control information”

## 8.3 ParameterReference

The example below shows the use of Parameter Reference. The example is best understood by studying the inserted production information table provided as an insert. In the example, we see that the product has a few characteristics with corresponding values. Some of the values have conditions. The condition is an expression comparing parameters to other parameters or constants. E.g., there is a condition for the values reported for supply current. The condition is that $V_{DRAIN}$ is greater than 60V.

$V_{DRAIN}$ is a reference to a parameter on the Item that may or may not have its own reported characteristic.

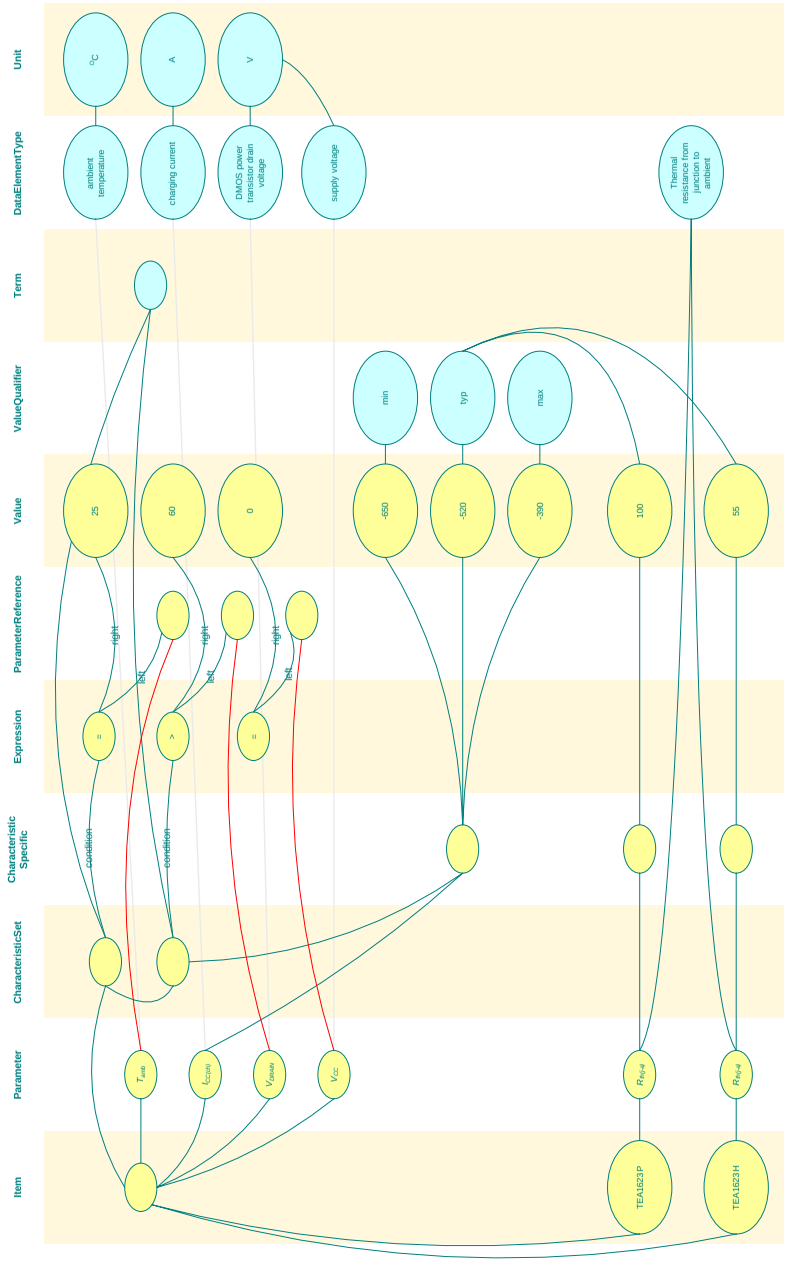



**Table 5: Thermal characteristics**

| Symbol | Parameter | Conditions | Typ | Unit |
|---|---|---|---|---|
| $R_{th(j-a)}$ | thermal resistance from junction to ambient | in free air | | |
| | TEA1623P | | 100 | K/W |
| | TEA1623PH | | 55 | K/W |

**Table 6: Characteristics**

*$T_{amb}$ = 25 °C; no overtemperature; all voltages are measured with respect to ground; currents are positive when flowing into the IC; unless otherwise specified.*

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $I_{CC(ch)}$ | charging current | $V_{DRAIN}$ > 60 V | | | | |
| | | $V_{CC}$ = 0 V | −650 | −520 | −390 | µA |
| | | $V_{CC}$ = 8.5 V | −375 | −275 | −175 | µA |



**Fig 42. Example of the use of "ParameterReference"**

## 8.4 Materials composition information

In the following two instance diagrams we see the same information represented in two different ways. The information being expressed is shown in a table in the instance diagrams.

In the first example the metadata declares a single SemanticDictionaryItemSet that contains the elements required to express the complete row in the table. The library instance is now expressed in a comparably flat structure.

In the second instance diagram the same information is described using nesting. The metadata is declared using a nested structure. The nested structure declared in the metadata is reflected in the library data.

Both examples provide the same information. Although semantically correct from an RDA model point of view, we (arguably) prefer the second example. The advantages to the nesting are:

- The definitions of % and ppm are reusable (used for both component-level and material-level);
- A better continuity from the data expressed in the table to the structure of the library data (the table is actually hierarchical).

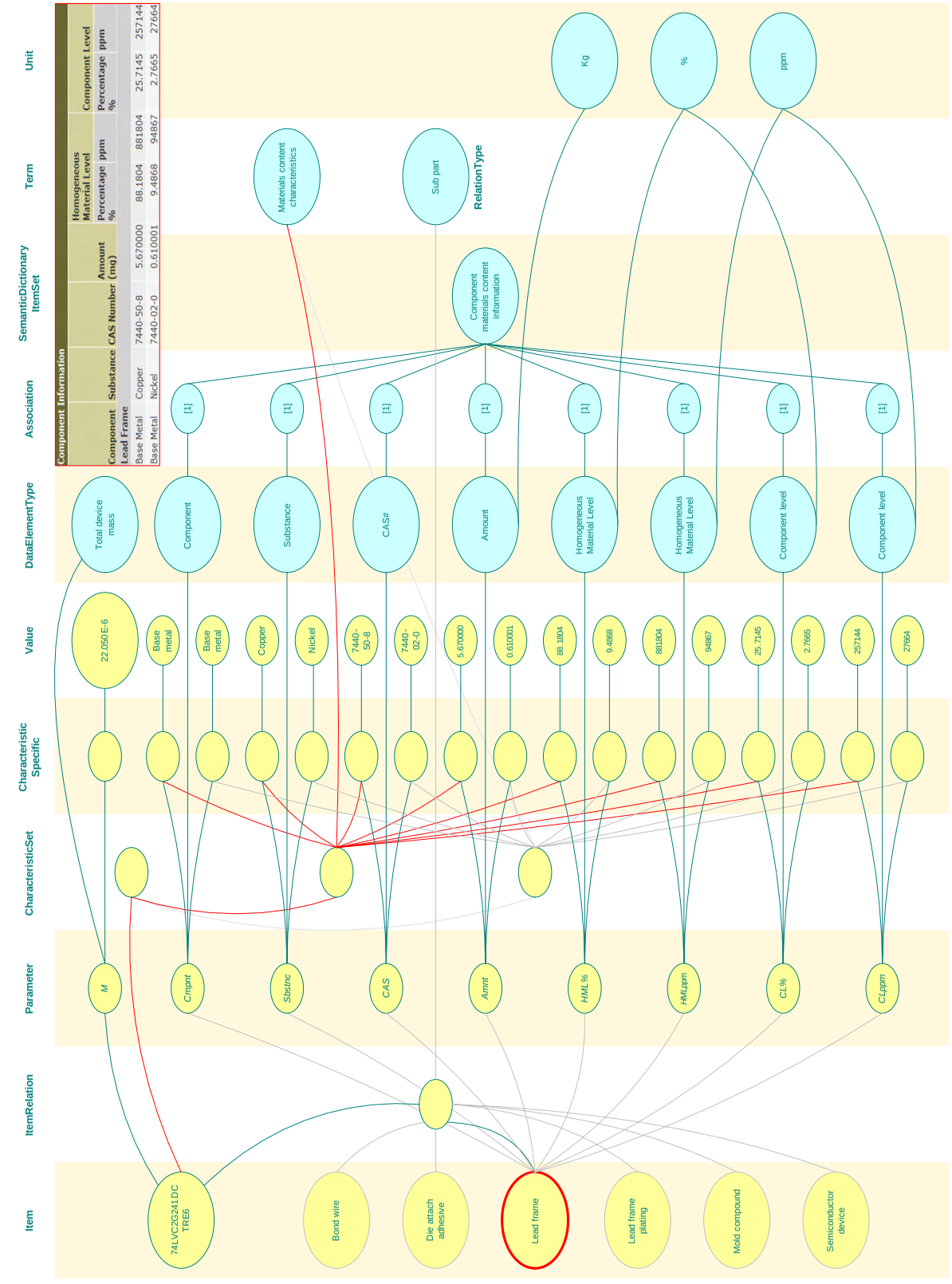



**Fig 43. Materials composition information: Variant "A"**

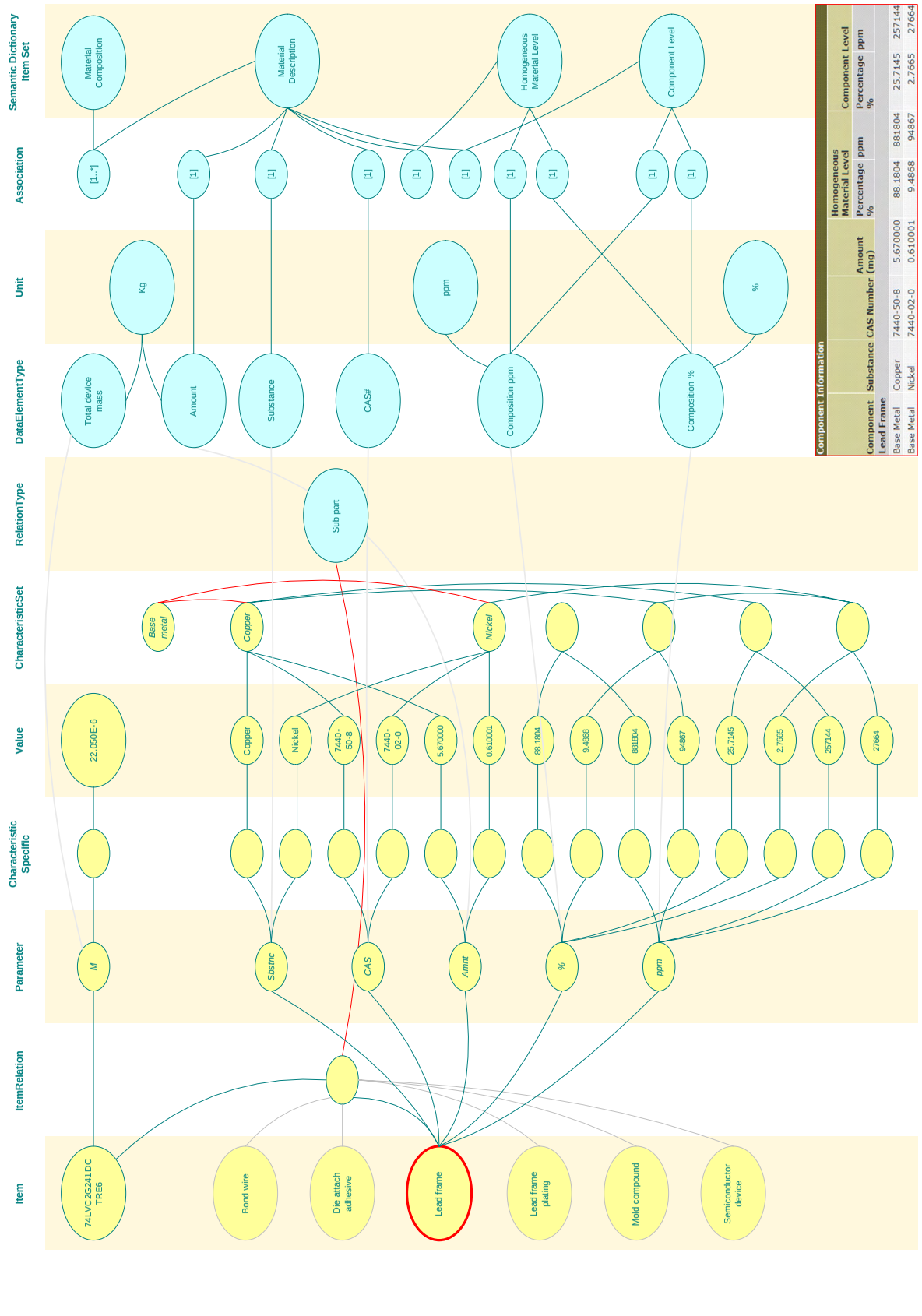



**Fig 44. Materials composition information: Variant "B"**

## 8.5 Business address

In the example below we see a simple example for defining a business address. The dictionary defines the required structure as follows:

- Requires at least one, max three address lines
- Requires one city name
- Requires one global country code
- Requires one national post code
- Allows for (but does not require) one Post Office Box
- Requires a region name

The dictionary defines these rules by defining a separate DataElementType for each of the field types (that is, address line, region name, etc.). The dictionary also defines a SemanticDictionaryItemSet representing the structure type BusinessAddress.

The SemanticDictionaryItemSet defines a separate association fore each of the field types allowed or required. The cardinality of the association defines how many are required/allowed.

The example also shows a conforming instantiation of a library based on the dictionary rules. In this case we have an item “Jane Doe” that has a business address (reported as an instance of CharacteristicSet). The business address has a single address line (allowed, because the dictionary defines that an address can have between 1 and 3 address lines). It continues defining every one of the required fields. There is no Post Office Box (allowed because PostOfficeBox is defined as optional in the dictionary).

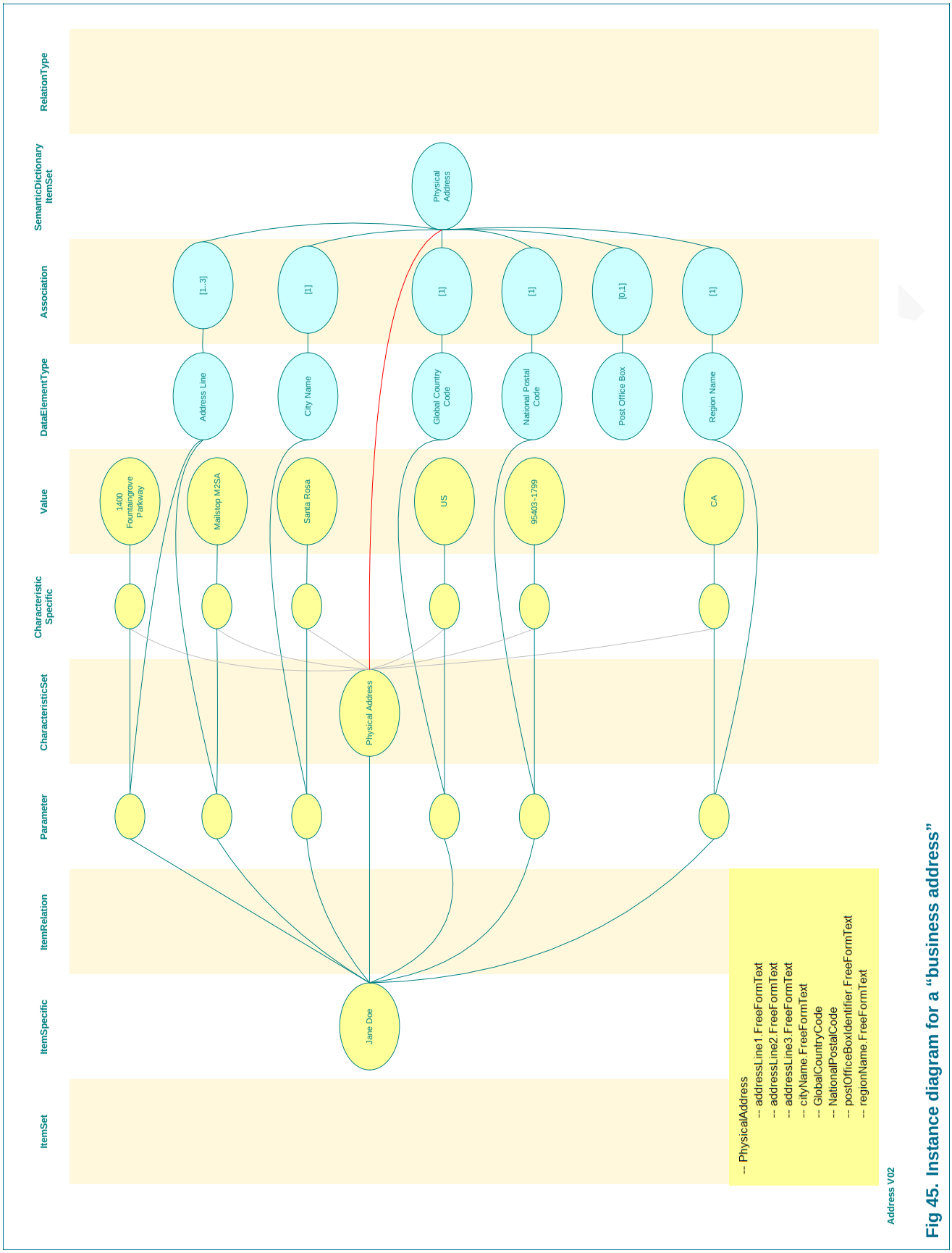



**Fig 45. Instance diagram for a "business address"**

## 8.6 Confectionery

In this example we see the description of a confectionery, a chocolate bar. The dictionary contains a set of **DataElementTypes** that can be used to describe the confectionery:

- Sugar level claim
- Saturated fat
- ... and more

The dictionary also contains one **SemanticDictionaryItemSet** defining Packaging. The Packaging defines that if a specifier provides a CharacteristicSet with the definition Packaging it **must** include “packaging unit” and “pieces” in this set.

The following table shows - highlighted in yellow - selected confectionery data, expressed in Figure 46 on page 93 as an instance diagram.

| GPC Class | Chocolate and Chocolate Substitutes Confectionery | Value | Type |
|---|---|---|---|
| | Chocolate and Chocolate Substitutes Confectionery Brick Variant | HAND HELD/SOLID | enum |
| | Diabetic Claim | NO | enum |
| | Special Occasion Claim | NO | enum |
| | Sugar Level Claim | CONTAINS SUGAR | enum |
| Nutritional Facts | | | |
| | Calories | 230 | integer |
| | Calories from Fat | 120 | integer |
| | Serving size | 1 | integer |
| | Total Fat | 13g | number of grams |
| | Saturated fat | 9g | number of grams |
| | Cholesterol | 10mg | number of milligrams |
| | Sodium | 40mg | number of milligrams |
| | Total Carbohydrates | 25g | number of grams |
| | Dietary Fiber | 1g | number of grams |
| | Sugars | 22g | number of grams |
| | Protein | 3g | number of grams |
| | Vitamin A | 0% | % of daily requirement set buy US Government |
| | Vitamin C | 0% | % of daily requirement set buy US Government |
| | Calcium | 6% | % of daily requirement set buy US Government |
| | Iron | 2% | % of daily requirement set buy US Government |
| Size | | 1.55 | ounces |
| Packaging | | 18/box | |

**Fig 46. Sample data for confectionery example (yellow= “shown in instance diagram”)**

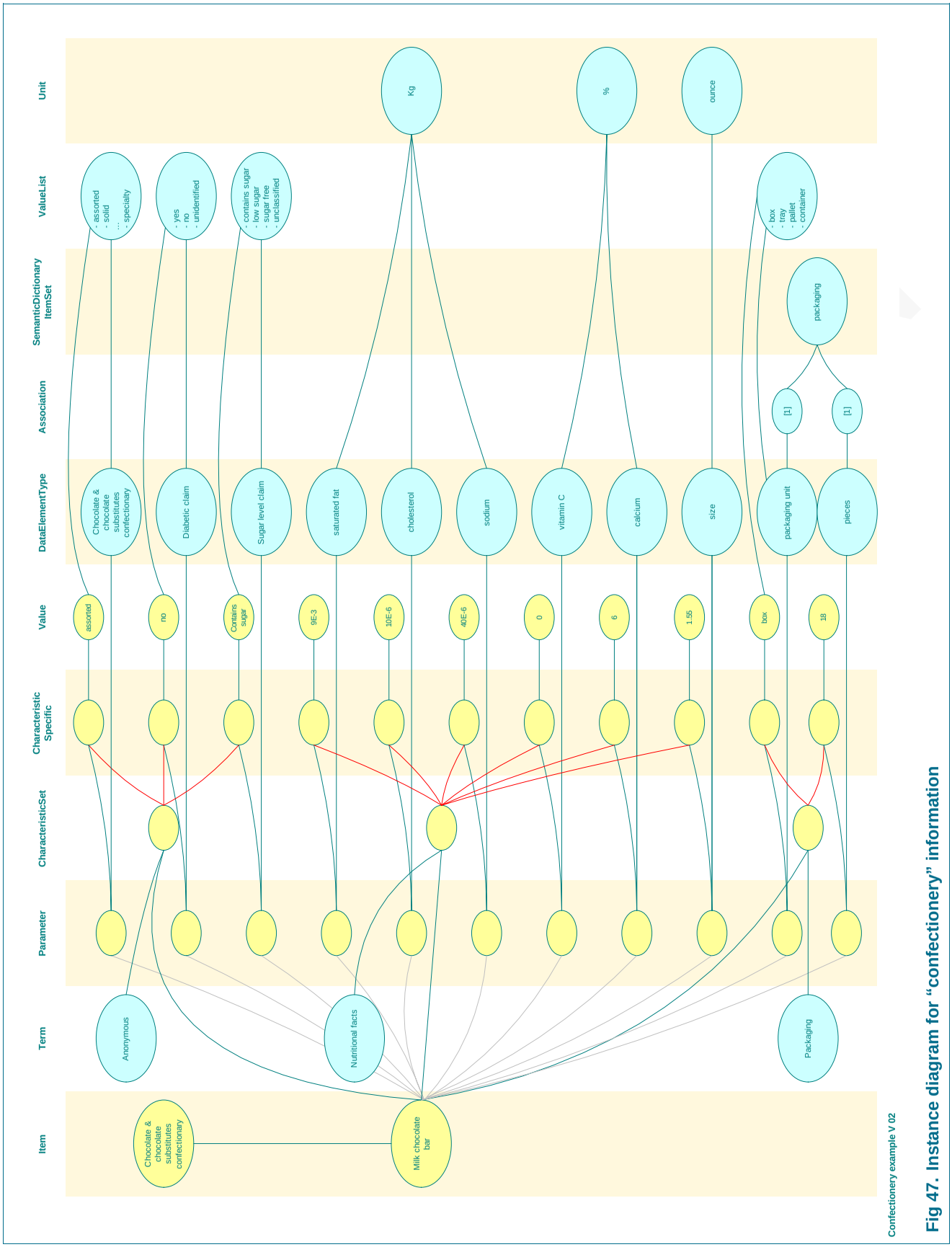



Fig 47. Instance diagram for "confectionery" information

## 8.7 Multi-function device

This example depicts the read/write data for a multi-mode optical drive. The drive's read and write speed are reported per **Mode**, making use of the ValueQualifiers "read" and "write".

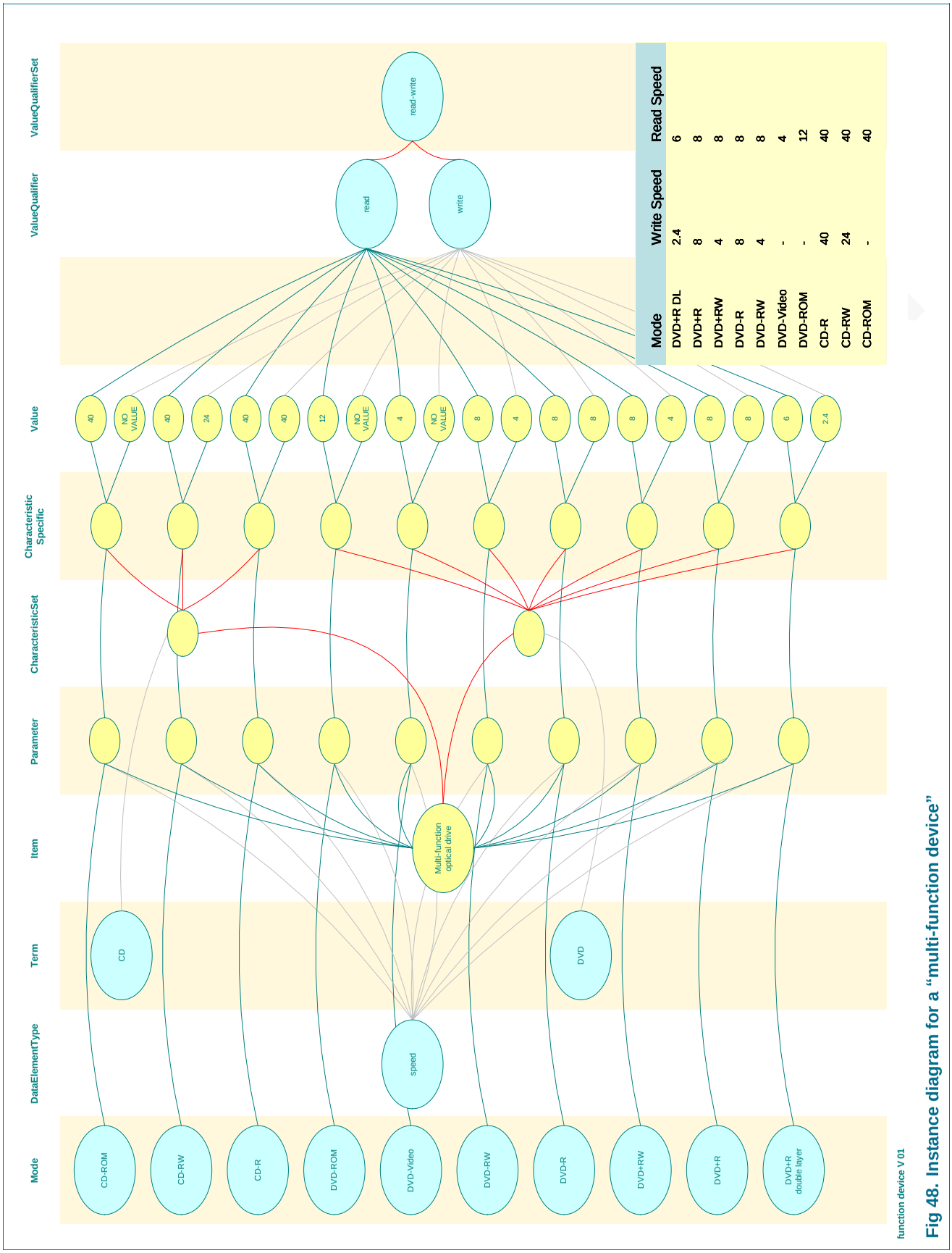



Fig 48. Instance diagram for a “multi-function device”

# 9. RosettaNet Dictionary & Library Model Diagrams

The following section is normative and contains a complete set of model diagrams.

## 9.1 RosettaNet Core

## 9.1.1 Dictionary

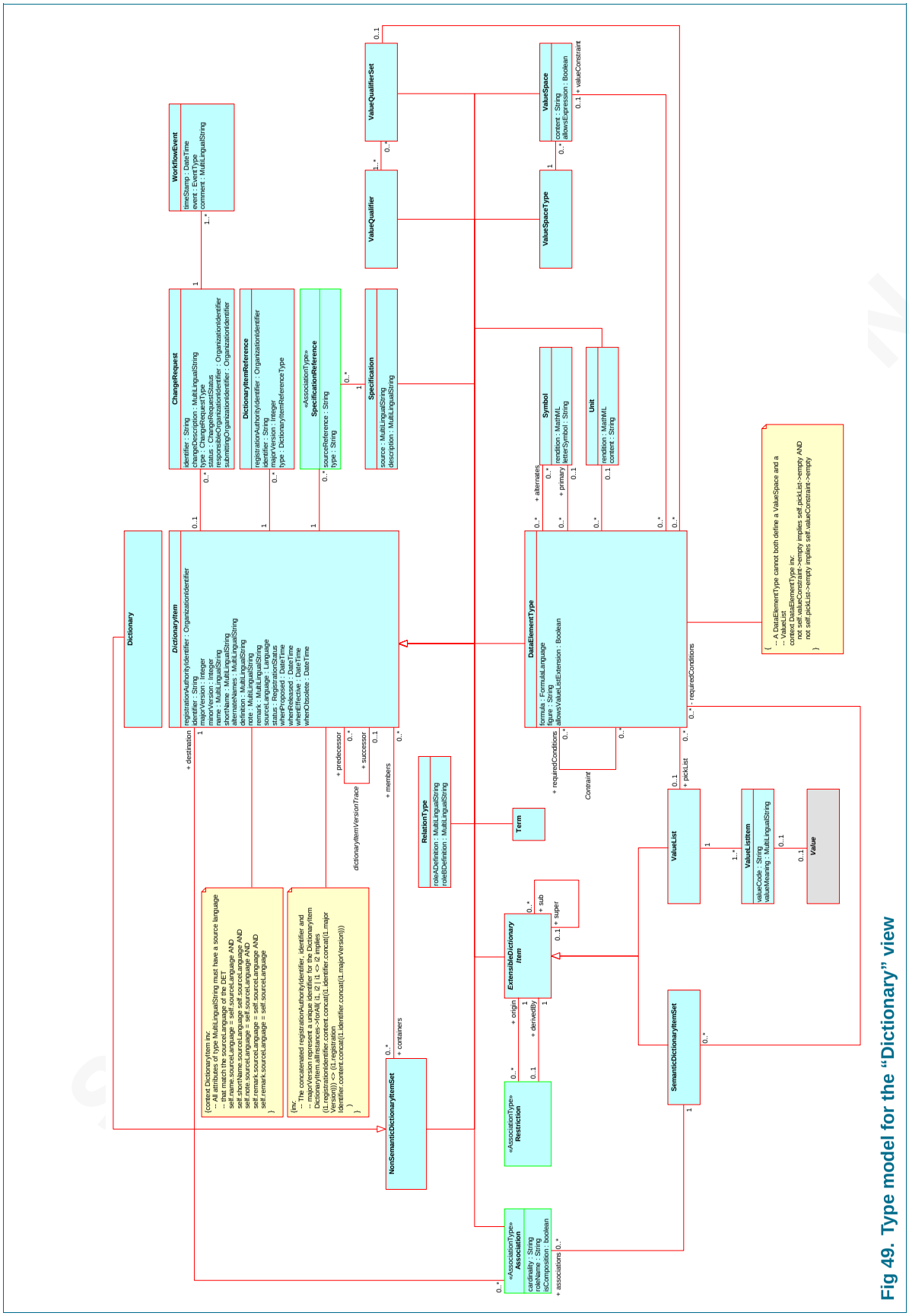



Fig 49. Type model for the "Dictionary" view

## 9.1.2 Library

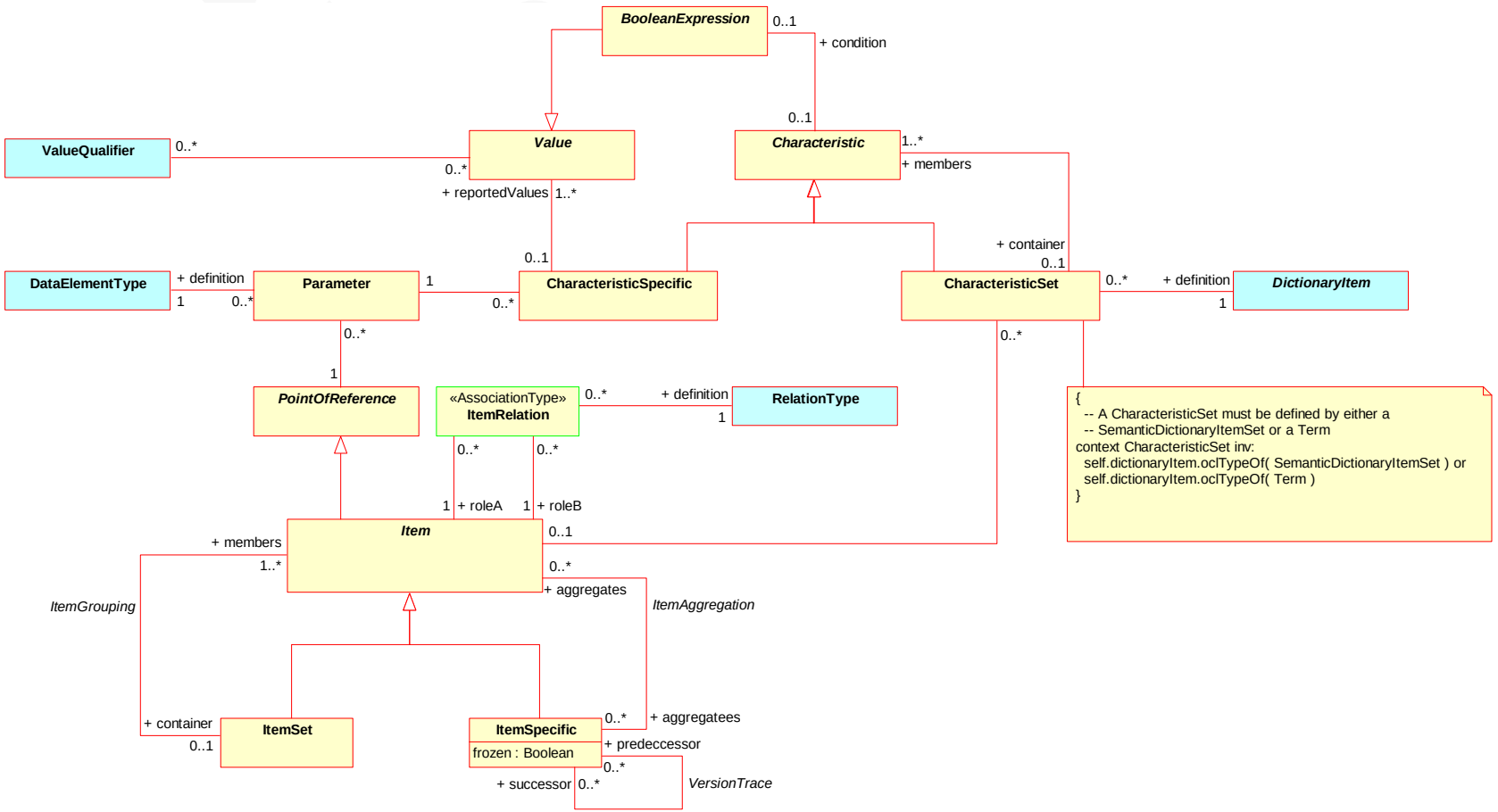



**Fig 50. Type model for “Library”**

### 9.1.3 Primitives

#### 9.1.3.1 Enumerations

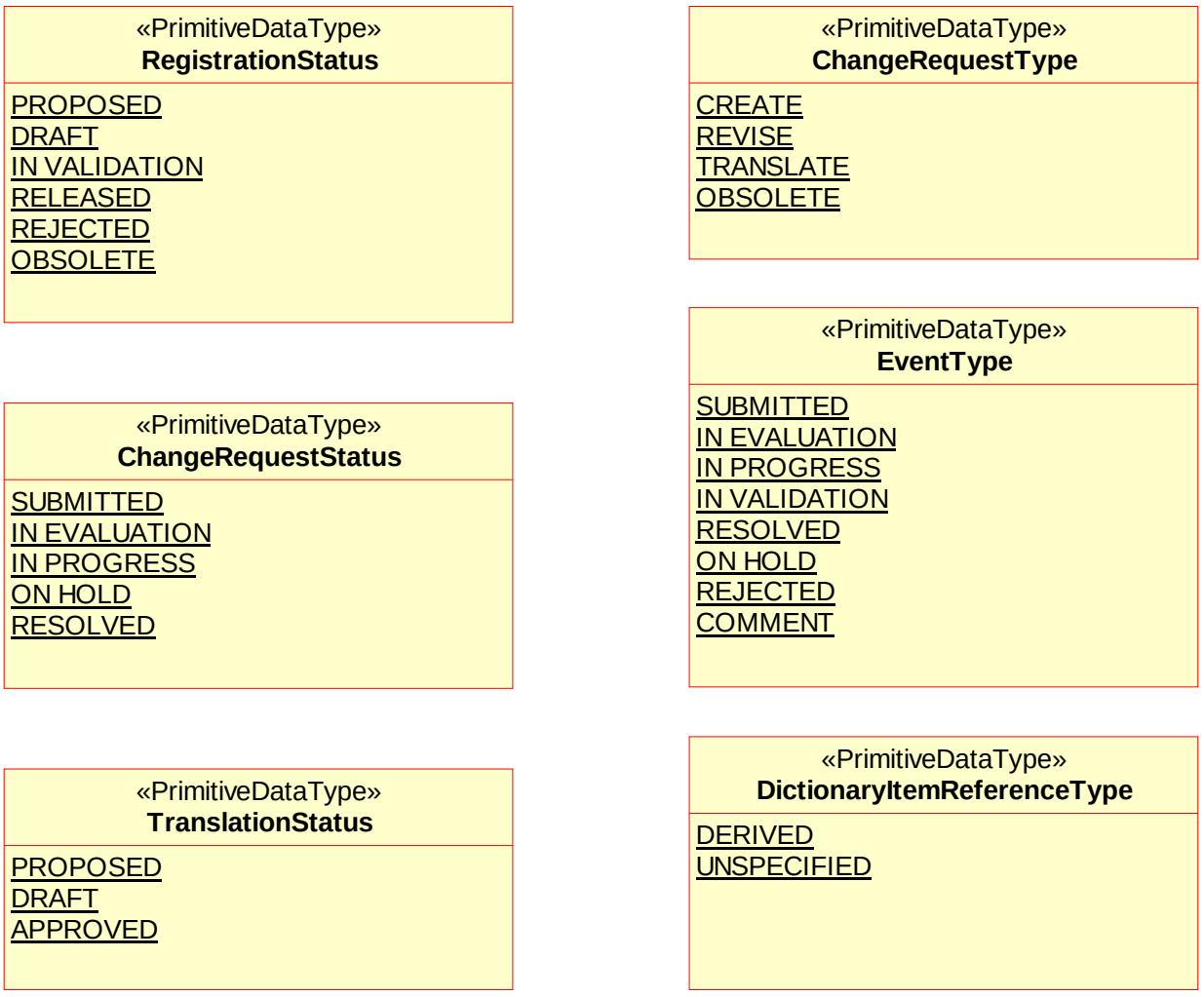



Fig 51. Type model for “Enumerations”

### 9.1.3.2 Text and language support

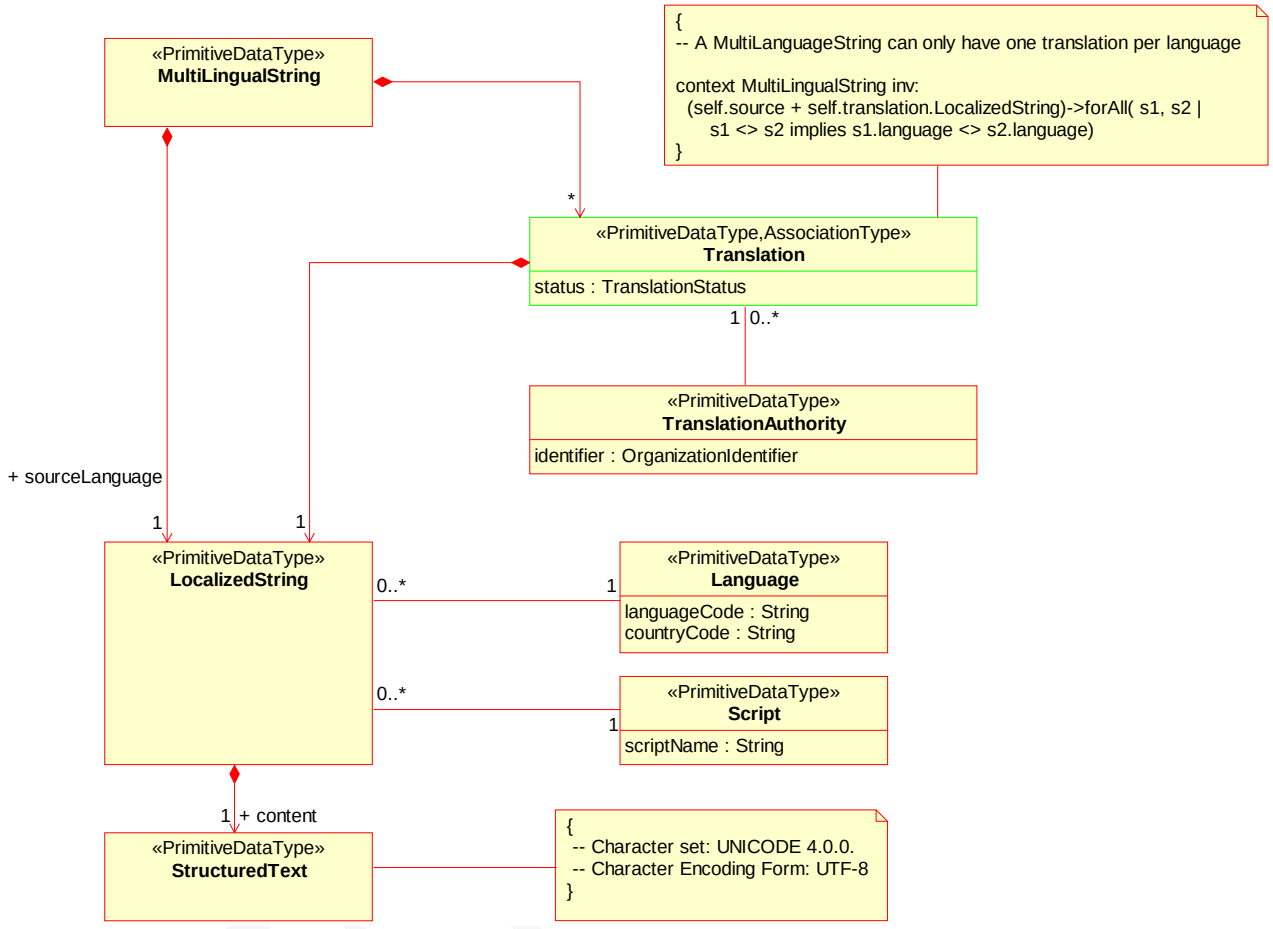



**Fig 52. Type model for "Text & language support"**

#### 9.1.3.3 Specialized primitives

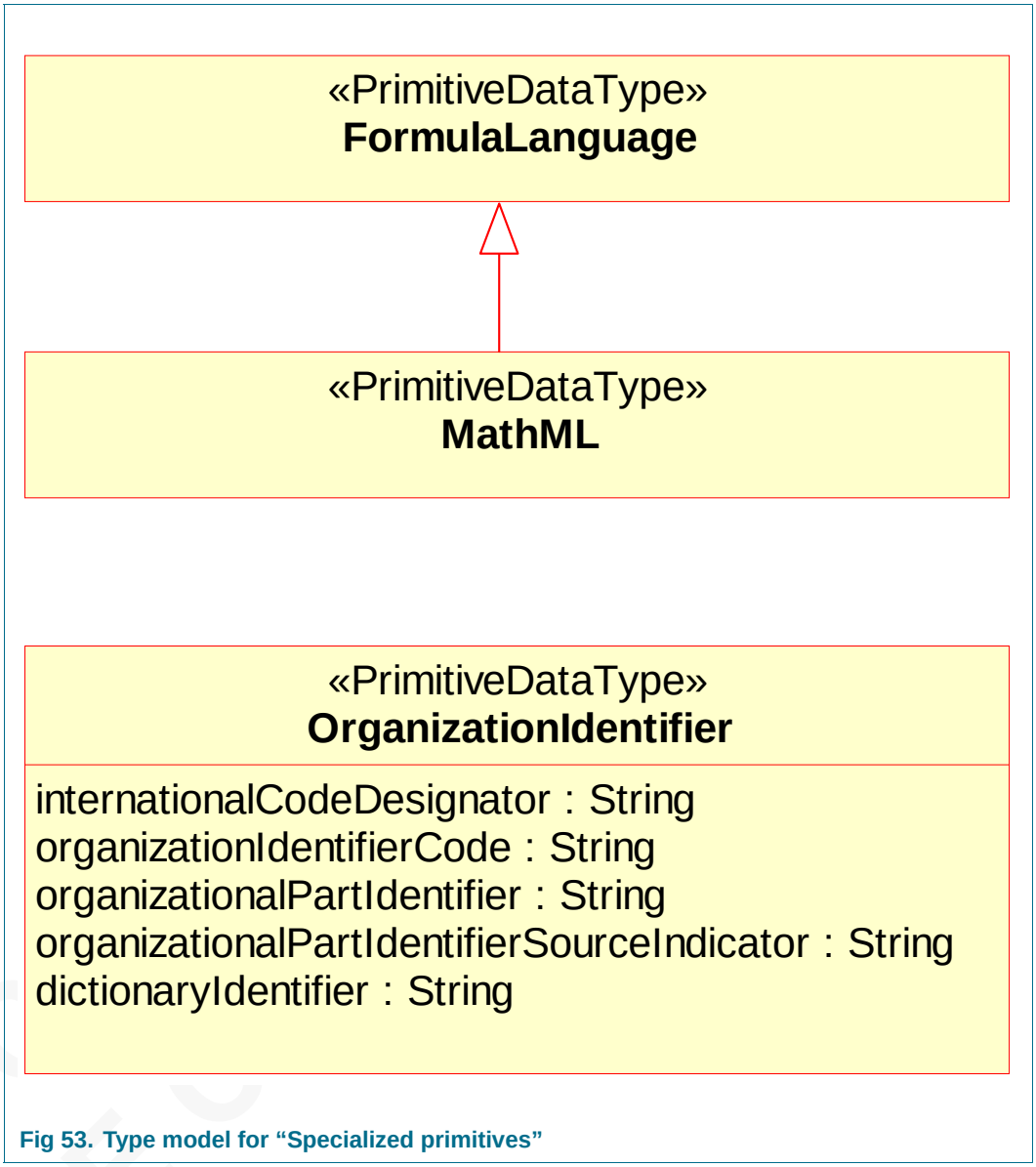



Fig 53. Type model for “Specialized primitives”

### 9.1.3.4 Values

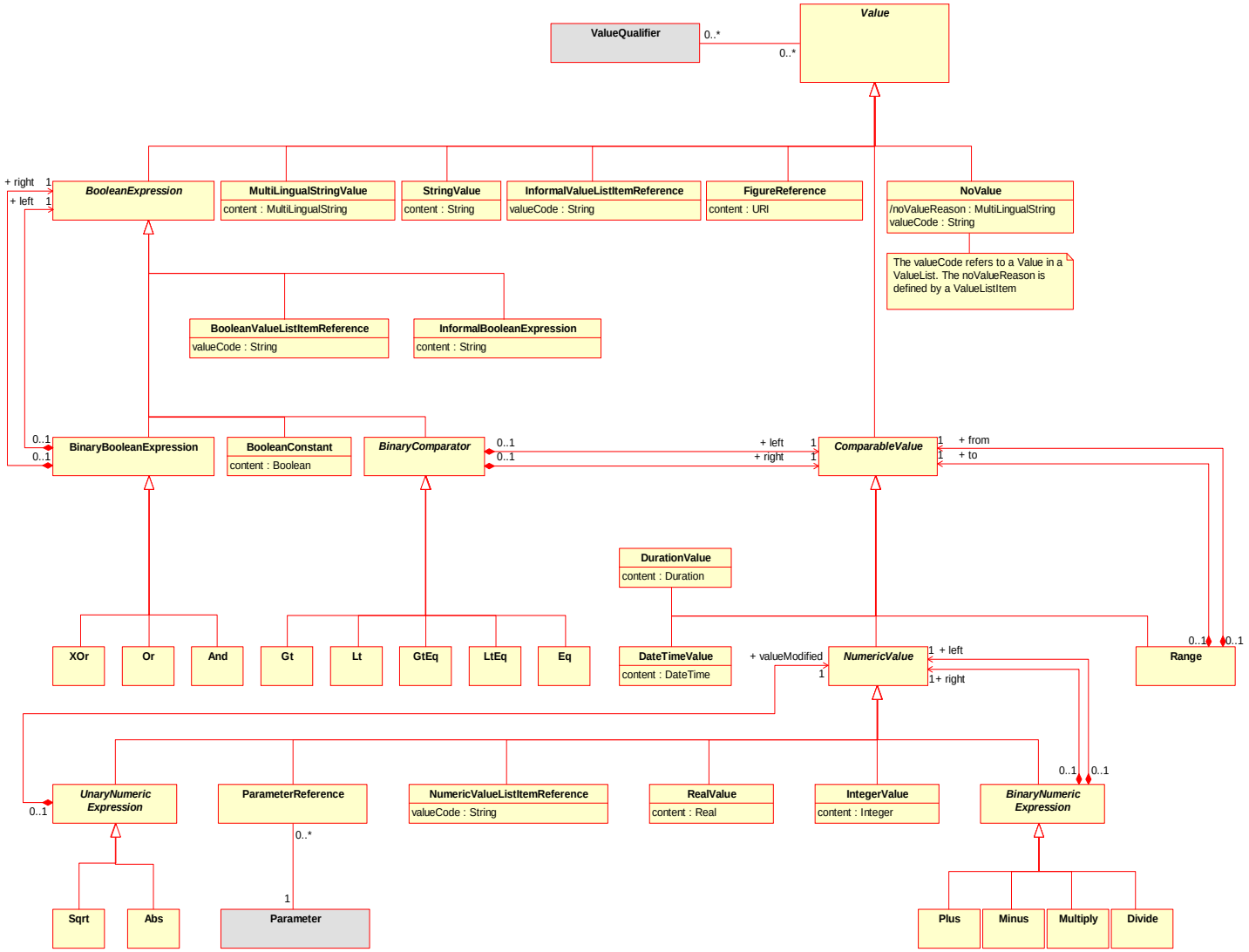



Fig 54. Type model for “Values”

## 9.2 RosettaNet EC Domain extensions

### 9.2.1 EC Domain Dictionary Extensions

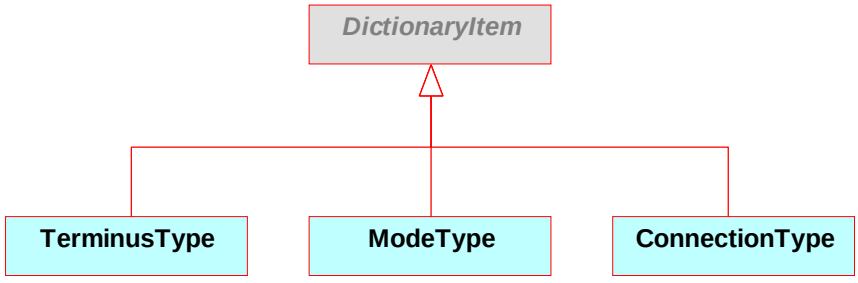



Fig 55. Type model for “EC Domain dictionary extensions”

### 9.2.2 EC Domain Library Extensions

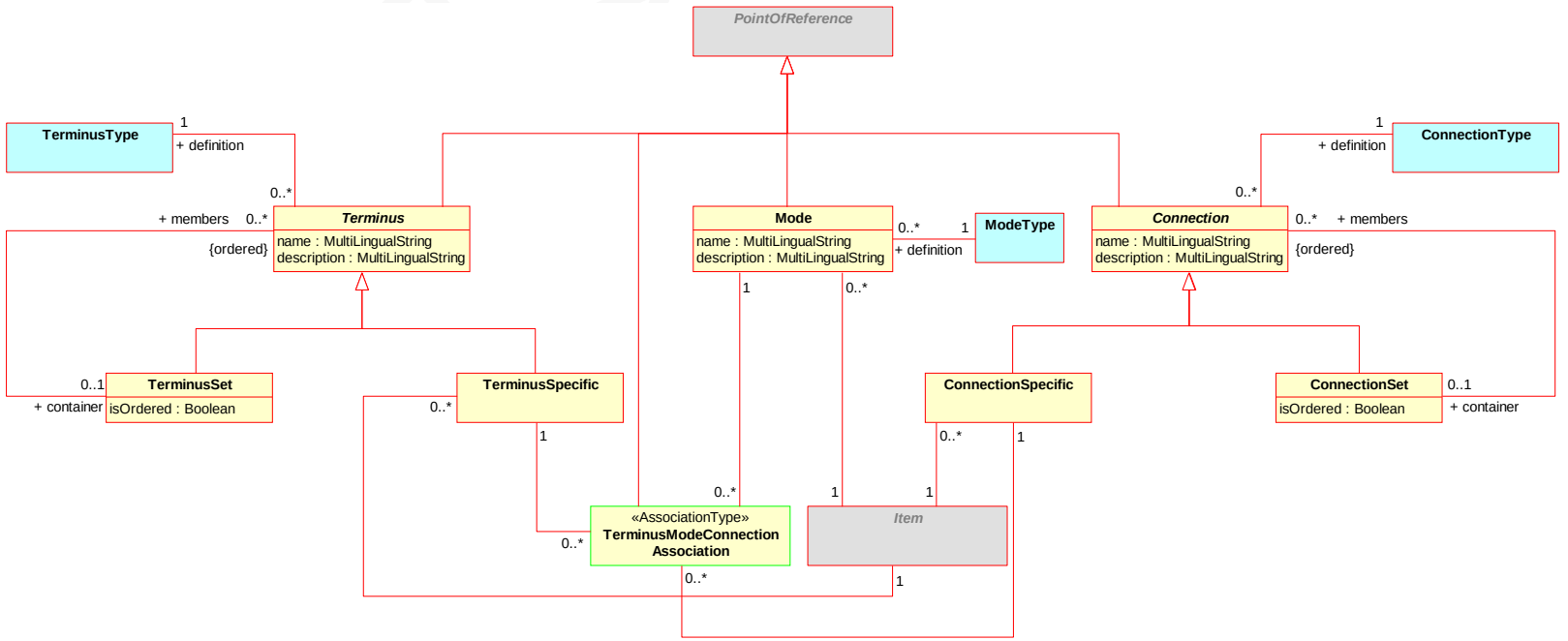



Fig 56. Type model for “EC Domain library extensions”

# 10. Data Dictionary

## 10.1 Introduction to the Data Dictionary

This section describes all classes and attributes in detail. The content is organized per model package.

## 10.2 Descriptors of a model type attribute

The following table defines the descriptors used in the Data Dictionary section of this Specification.

**Table 126: Model type attribute descriptors**

| Descriptor | Definition |
|---|---|
| **Descriptions** | Name for the following group of textual descriptors:: definition, note, remark, example and rule. |
| **Definition** | Statement that describes the meaning of a Class or Attribute in an unambiguous and unique manner to permit its differentiation from all other Classes or Attributes |
| **Note** | Statement which provides further information on the definition, which is essential to the understanding of that definition |
| **Remark** | Explanatory text to further clarify the meaning of the definition |
| **Example** | A representative instance of an Attribute |
| **Rule** | Statement which provides further information on the Attribute which is essential to the application of the Attribute |
| **UML** | Name for a group of descriptors defining the data type of an Attribute following UML conventions |
| **Data type** | A classification identifying one of various types of data |
| **Cardinality** | The number of elements in a given mathematical set |
| **XML Schema** | Name for a group of descriptors defining the data type of an Attribute following XML Schema conventions; for additional information see [XSD] |
| **Built-in data type** | Built-in data types are those which are defined in [XSD] |
| **User-derived data type** | User-derived datatypes are those that are defined in terms of other datatypes |
| **Lexical space** | A lexical space is the set of valid literals for a data type |
| **Constraining facet** | A constraining facet is an optional property that can be applied to a datatype to constrain its ·value space |
| **Implementation notes** | Statement which provides further information on the Attribute which may be helpful to the application of the Attribute in an implementation |
| **References** | This section contains references to external standards, specifications or other sources |
| **Source for definition** | A reference to source from which the values in a value list have been derived or copied |

## 10.3 RosettaNet Core

### 10.3.1 Dictionary

#### 10.3.1.1 Associations

**Table 127: Definition of ASSOCIATIONS in the "Dictionary" View**

| Name | Description |
|---|---|
| DictionaryVersionTrace | Defines a linked list of history for a **DictionaryItem**. The linked list is used to trace the versions of a **DictionaryItem**. |
| Constraint | Defines that when a particular **DataElementType** is used to specify a **Parameter** in the library, the reported **Characteristics** must also define a set of conditions specifying the values of the "requiredCondition" **DataElementTypes** |
| DictionaryItem <-> SpecificationReference | Allows a **DictionaryItem** to have a set of **SpecificationReferences**. A **DictionaryItem** uses the **SpecificationReferences** to define the origin of its definition. |
| SpecificationReference <-> Specification | Defines which **Specification** the **SpecificationReference** refers to. |
| NonSemanticDictionary ItemSet <-> DictionaryItem | Defines which **DictionaryItems** belong to the **NonSemanticDictionaryItemSet** |
| ExtensibleDictionaryItem super <-> sub | Defines an inheritance between **ExtensibleDictionaryItems**. When this construct is used, then all definitions of the **ExtensibleDictionaryItem** playing the role of "super" applies to the **ExtensibleDictionaryItem** playing the role of "sub". |
| DataElementType <-> SemanticDictionaryItemSet | Defines **Conditions** common to a set of **DataElementTypes** |
| DataElementType <-> Unit | A **DataElementType** may define a **Unit**. In that case, all reported values on **Characteristics** for a **Parameter** that takes its definition from this **DataElementType** are implicitly reported in this **Unit**. |
| DataElementType <-> primary: Symbol | Defines the preferred **Symbol** used for a **DataElementType** |
| DataElementType <-> alternates: Symbol | Defines alternate **Symbols** for a **DataElementType** (other than the primary **Symbol**) |
| DataElementType <-> ValueList | Defines a set of possible values to be used when reporting values for a **DataElementType**. |
| ValueList <-> ValueListItem | Defines all the **ValueListItems** that can be picked in the library when the **Value** is to be picked from the **ValueList** |
| ValueListItem <-> Value | Defines the actual **Value** for the **ValueListItem** |
| ValueSpaceType <-> ValueSpace | A **ValueSpace** always has a corresponding **ValueSpaceType**. The **ValueSpaceType** defines a reference to the language used to specify the **ValueSpace**. |
| DataElementType <-> ValueSpace | A **DataElementType** may restrict **Values** reported to a **ValueSpace**, hence, a **DataElementType** may link to a **ValueSpace**. |
| DataElementType <-> ValueQualifierSet | A **DataElementType** may specify that all **Characteristics** which are based on its specification must provide a value for each **ValueQualifier** in a **ValueQualifierSet** |
| ValueQualifierSet <-> ValueQualifier | The **ValueQualifierSet** is defined by the **ValueQualifier** held by this association. |

**Table 127: Definition of ASSOCIATIONS in the "Dictionary" View**

| Name | Description |
|---|---|
| Association <-> DictionaryItem | This association represents the target dictionary item for the **Association**. In other words, the **SemanticDictionaryItemSet** requires N number (specified by the cardinality defined on the **Association** type) of the **DictionaryItem** linked to the association. A **DictionaryItem** may be picked up by many Associations (that is, it may be a building block for many **SemanticDictionaryItemSets**), however each **Association** specifies exactly one **DictionaryItem** |
| Association <-> SematicDictionaryItemSet | This association links the **SemanticDictionaryItemSet** to its **Associations**. The **Association** in return picks up the **DictionaryItem** as specified above. |
| Restriction: origin <-> ExtensibleDictionaryItem | The origin is represents the source for a restriction. |
| Restriction: derivedBy <-> ExtensibleDictionaryItem | Defines **DictionaryItems** to be restricted. |
| NonSemanticDictionary ItemSet <-> DictionaryItem | Defines the members of the **NonSemanticDictionaryItemSet**. |
| DictionaryItem <-> ChangeRequest | Defines the link to the change requests associated with a **DictionaryItem**. Notice that the multiplicity allows a change request to be linked to a single item or not to any items. |
| DictionaryItem <-> DictionaryItemReference | Defines the link between **DictionaryItems** and their references to other **DictionaryItems**. |
| ChangeRequest <-> WorkflowEvent | Defines the activities relevant for a **ChangeRequest**. |

### 10.3.1.2 DictionaryItem

**Table 128: CLASS: DictionaryItem**

| Field | Value |
|---|---|
| Name | DictionaryItem |
| Definition | An abstract SuperType defining the attributes common to all entries in a Dictionary |
| Note | - |
| Remark | - |

**Table 129: ATTRIBUTE: registrationAuthorityIdentifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | registrationAuthorityIdentifier |
| Definition | Any organization authorized to register DictionaryItems |
| Note | - |
| Remark | See ISO/IEC 6523-1:1998(E) for structure of the Registration Authority Identifier string format; The combined attributes ‘Registration Authority’, ‘Identifier’ and ‘Major version’ serve to uniquely identify a data element. A fully qualified example is: “60:023936581:::DE06AC:009” |
| Example | 60:023936581 |
| Rule | - |
| **UML** | |
| Data type | OrganizationIdentifier |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | OrganizationIdentifier |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | ISO/IEC 11179-3:1994(E) 6.1.4 |

**Table 130: ATTRIBUTE: identifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | identifier |
| Definition | A six character string unique within a RegistrationAuthority |
| Note | - |
| Remark | Whether the identifier should be changed so that a new DictionaryItem is created or wether the majorVersion of a DictionaryItem should be changed is to be determined for each case separately |
| Example | DE06AC |
| Rule | Upper-case latin letters A through Z (to avoid misunderstanding, the upper-case Latin letters O and I shall not be used) digits 0 through 9.<br>The codes are issued sequential and should not have any relationship with the meaning of the DictionaryItems.<br>In case of at least one attribute of the DictionaryItem, which affects the meaning and or communication of the DictionaryItem is changed, a new (other) DictionaryItem, having a new identifier, shall be defined. Such attributes are:<br>• definition<br>• unit<br>• requiredConditions<br>• valueSpace |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | For interoperability reasons the constraining facet shall be: <xs:pattern value="\w[\w\|\s\|\i]{0,13}"/>. However for RosettaNet DictionaryItem Identifiers the constraining facet shall be: <xsd:pattern value="[A-HJ-NP-Z]{2}[0-9]{2}[A-HJ-NP-Z]{2}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 131: ATTRIBUTE: majorVersion**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | majorVersion |
| Definition | Identification of an issue of a DictionaryItem in a series of evolving DictionaryItems within a RegistrationAuthority |
| Note | - |
| Remark | Whether the identifier should be changed so that a new DictionaryItem is created or wether the majorVersion of a DictionaryItem should be changed is to be determined for each case separately |
| Example | 009 |
| Rule | Consecutive majorVersion numbers shall be issued in ascending order. A new version of the DictionaryItem shall be generated if at least one attribute of the DictionaryItem is changed |
| **UML** | |
| Data type | Integer |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | positiveInteger |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:minInclusive value=”001”/><br><xs:maxInclusive value=”999”/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 132: ATTRIBUTE: minorVersion**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | minorVersion |
| Definition | Number used for administrative control of a DictionaryItem |
| Note | - |
| Remark | - |
| Example | 01 |
| Rule | Consecutive minorVersion numbers shall be issued in ascending order. Per DictionaryItem, unique by its identifier, only one minorVersion number is current at any moment. A new minorVersion number of a DictionaryItem shall be generated if an attribute of the DictionaryItem is changed which neither affects the use (communication, data base definitions etc.) nor the meaning of the DictionaryItem, or when editorial changes of typing and spelling errors have been implemented. Those attributes which, when changed, will result in a minorVersion change are:<br>• alternateName<br>• alternate Symbol<br>• remark<br>• formula<br>• specificationReference<br>• spelling error in the text of the definition<br>• figure<br>The minorVersion number shall be reset to the starting number 01 when the minorVersion number is changed. |
| **UML** | |
| Data type | Integer |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | positiveInteger |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:minInclusive value=”01”/><br><xs:maxInclusive value=”99”/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 133: ATTRIBUTE: name**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | name |
| Definition | Single-word or multi-word designation assigned to a DictionaryItem |
| Note | - |
| Remark | - |
| Example | output short-circuit current |
| Rule | Those characters from the character set of ISO/IEC 10646-1, as defined in annex A of IEC 61360-1:2002-02 |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | <xsd:pattern value="\w[\w\|\s]{0,69}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | IEC 61360-1:2002-02 3.2.4 |

**Table 134: ATTRIBUTE: shortName**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | shortName |
| Definition | Shortened representation of the name of the data DictionaryItem |
| Note | - |
| Remark | For additional constraints see: IEC 61360-1:2002-02 3.2.6 |
| Example | I_OS |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | <xs:pattern value="\w[\w|\s|\i]{0,16}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | IEC 61360-1:2002-02 3.2.6 |

**Table 135: ATTRIBUTE: alternateNames**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | alternateNames |
| Definition | Single-word or multi-word designation that differs from the given name but represents the same DictionaryItem concept |
| Note | - |
| Remark | - |
| Example | dynamic output current |
| Rule | Those characters from the character set of ISO/IEC 10646-1, as defined in annex A of IEC 61360-1:2002-03 |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 0..10 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | <xsd:pattern value="\w[\w\|\s]{0,69}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | ISO/IEC 11179-3:1994(E) 6.1.5 |

**Table 136: ATTRIBUTE: definition**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | definition |
| Definition | Statement that describes the meaning of a DictionaryItem in an unambiguous and unique manner to permit its differentiation from all other DictionaryItems |
| Note | - |
| Remark | The definition shall be a single sentence. The unit of measure (if present) in which the value of the data element type is expressed shall always be included in the definition |
| Example | The value as specified by a ValueQualifierSet (MinMax) of the output current (in A) of a digital TTL IC, when an output is short-circuited, at maximum supply voltage, and in a temperature range between specified temperatures (T1 and T2). |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | IEC 61360-1:2002-02 3.3.1 |

**Table 137: ATTRIBUTE: note**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | note |
| Definition | Statement which provides further information on the definition, which is essential to the understanding of that definition |
| Note | - |
| Remark | - |
| Example | 1 Not more than one output should be shorted at a time.<br>2 The duration of the short-circuit should not exceed one second. |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | IEC 61360-1:2002-02 3.3.2 |

**Table 138: ATTRIBUTE: remark**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | remark |
| Definition | Explanatory text to further clarify the meaning of the definition |
| Note | - |
| Remark | Remarks shall not influence the meaning of the definition |
| Example | The output short-circuit current was intended originally to reassure the TTL user that the device would withstand accidental grounding e.g. during in-circuit testing. |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | IEC 61360-1:2002-02 3.3.3 |

**Table 139: ATTRIBUTE: sourceLanguage**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | sourceLanguage |
| Definition | The language in which the DictionaryItem was originally defined |
| Note | - |
| Remark | KNOWN LIMITATION: this solution is XSD compliant but does not provide the capability to “officially” designate “simplified Chinese”. [SIM] uses “zh-CN” for this purpose. |
| Example | en-US |
| Rule | - |
| **UML** | |
| Data type | Language |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | Language |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | The facet is defined in [XSD] |
| **References** | |
| Source for definition | - |

**Table 140: ATTRIBUTE: status**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | status |
| Definition | A designation of the position in the registration life-cycle of a DictionaryItem |
| Note | - |
| Remark | - |
| Example | RELEASED |
| Rule | - |
| **UML** | |
| Data type | RegistrationStatus |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | RegistrationStatus |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 141: ATTRIBUTE: whenProposed**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | whenProposed |
| Definition | The date and time at which a DictionaryItem was proposed for inclusion into the Dictionary |
| Note | - |
| Remark | - |
| Example | 2004-11-25T14:12:24Z |
| Rule | - |
| **UML** | |
| Data type | DateTime |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | dateTime |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 142: ATTRIBUTE: whenReleased**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | whenReleased |
| Definition | The date and time at which a DictionaryItem was released for inclusion into the Dictionary |
| Note | - |
| Remark | - |
| Example | 2005-01-17T17:55:00Z |
| Rule | - |
| **UML** | |
| Data type | DateTime |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | dateTime |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 143: ATTRIBUTE: whenEffective**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | whenEffective |
| Definition | The date and time at which a DictionaryItem becomes applicable as a standard |
| Note | - |
| Remark | - |
| Example | 2006-01-01T00:00:00Z |
| Rule | - |
| **UML** | |
| Data type | DateTime |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | dateTime |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 144: ATTRIBUTE: whenObsolete**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | whenObsolete |
| Definition | The date and time at which a DictionaryItem became obsolete |
| Note | - |
| Remark | - |
| Example | 2030-12-08T20:15:00Z |
| Rule | - |
| **UML** | |
| Data type | DateTime |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | dateTime |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.3 ChangeRequest

**Table 145: CLASS: ChangeRequest**

| Field | Value |
|---|---|
| Name | ChangeRequest |
| Definition | Specification of semantic and administrative aspects belonging to a change |
| Note | - |
| Remark | - |

**Table 146: ATTRIBUTE: identifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | identifier |
| Definition | A six character numeric string unique within a RegistrationAuthority |
| Note | The sequence of identifiers starts with 000001 and is incremented by 1 for every new ChangeRequest |
| Remark | - |
| Example | “000001” |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:pattern value=”[0-9]{5}[1-9]|[0-9]{4}[1-9][0-9]|[0-9]{3}[1-9][0-9]{2}|[0-9]{2}[1-9][0-9]{3}|[0-9][1-9][0-9]{4}|[1-9][0-9]{5}”/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 147: ATTRIBUTE: changeDescription**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | changeDescription |
| Definition | The description and/or reason for change |
| Note | - |
| Remark | - |
| Example | Please add the following ValueListItem to the color pick list: “P” for “Purple” |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 148: ATTRIBUTE: type**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | type |
| Definition | The category of ChangeRequest indicating an action to be performed on one or more DictionaryItems |
| Note | - |
| Remark | - |
| Example | CREATE |
| Rule | - |
| **UML** | |
| Data type | ChangeRequestType |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | ChangeRequestType |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 149: ATTRIBUTE: status**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | status |
| Definition | A string designating the phase in the life cycle of a ChangeRequest |
| Note | - |
| Remark | - |
| Example | EVALUATION |
| Rule | - |
| **UML** | |
| Data type | ChangeRequestStatus |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | ChangeRequestStatus |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 150: ATTRIBUTE: responsibleOrganizationIdentifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | responsibleOrganizationIdentifier |
| Definition | The organization or unit within an organization that is responsible for the validation and maintenance of DictionaryItems |
| Note | - |
| Remark | The organization shall be considered the “owner” of the DictionaryItem |
| Example | “60:023936581” = ICD:OI = DUNS:RosettaNet |
| Rule | - |
| **UML** | |
| Data type | OrganizationIdentifier |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | OrganizationIdentifier |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 151: ATTRIBUTE: submittingOrganizationIdentifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | submittingOrganizationIdentifier |
| Definition | The organization or unit within an organization that submitted the DictionaryItem for addition, change or cancellation/withdrawal in the DictionaryItem dictionary |
| Note | - |
| Remark | - |
| Example | “60:023936581” = ICD:OI = DUNS:RosettaNet |
| Rule | - |
| **UML** | |
| Data type | OrganizationIdentifier |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | OrganizationIdentifier |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

#### 10.3.1.4 WorkflowEvent

**Table 152: CLASS: WorkflowEvent**

| Field | Value |
|---|---|
| Name | WorkflowEvent |
| Definition | Records the history of a ChangeRequest |
| Note | - |
| Remark | An Event can occur multiple times to report additional information in a comment |

**Table 153: ATTRIBUTE: timeStamp**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | timeStamp |
| Definition | A record linking an Event to a time and date |
| Note | In this context the time and date a new WorkflowEvent is created |
| Remark | Lexical space as defined in [XSD] 3.2.7.1 |
| Example | 2004-11-25T14:12:24Z |
| Rule | The timeStamp shall be relative to UTC [Universal Coordinated Time] |
| **UML** | |
| Data type | DateTime |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | dateTime |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 154: ATTRIBUTE: event**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | event |
| Definition | A string indicating a change in status of a ChangeRequest or a comment related to a ChangeRequest |
| Note | - |
| Remark | - |
| Example | REVISE |
| Rule | - |
| **UML** | |
| Data type | EventType |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | EventType |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 155: ATTRIBUTE: comment**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | comment |
| Definition | Text providing supplementary information relevant to a WorkflowEvent |
| Note | - |
| Remark | - |
| Example | This revision is requested to incorporate new ValueListItems proposed by Philips Semiconductors |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

#### 10.3.1.5 DictionaryItemReference

**Table 156: CLASS: DictionaryItemReference**

| Field | Value |
|---|---|
| Name | DictionaryItemReference |
| Definition | A reference to a DictionaryItem in a Dictionary |
| Note | - |
| Remark | The Dictionary may be identical to the one that contains the DictionaryItemRefererence or may be different |

**Table 157: ATTRIBUTE: registrationAuthorityIdentifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | registrationAuthorityIdentifier |
| Definition | Any organization authorized to register DictionaryItems |
| Note | - |
| Remark | See ISO/IEC 6523-1:1998(E) for structure of the Registration Authority Identifier string format; The combined attributes ‘Registration Authority’, ‘Identifier’ and ‘Major version’ serve to uniquely identify a data element. A fully qualified example is: “60:023936581:::DE06AC:009” |
| Example | 60:023936581 |
| Rule | - |
| **UML** | |
| Data type | OrganizationIdentifier |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | OrganizationIdentifier |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | ISO/IEC 11179-3:1994(E) 6.1.4 |

**Table 158: ATTRIBUTE: identifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | identifier |
| Definition | A six character string unique within a RegistrationAuthority |
| Note | - |
| Remark | Whether the identifier should be changed so that a new DictionaryItem is created or wether the majorVersion of a DictionaryItem should be changed is to be determined for each case separately |
| Example | DE06AC |
| Rule | Upper-case latin letters A through Z (to avoid misunderstanding, the upper-case Latin letters O and I shall not be used) digits 0 through 9<br><br>The codes are issued sequential and should not have any relationship with the meaning of the DictionaryItems.<br>In case of at least one attribute of the DictionaryItem, which affects the meaning and or communication of the DictionaryItem is changed, a new (other) DictionaryItem, having a new identifier, shall be defined. Such attributes are:<br>definition<br>Unit<br>requiredConditions<br>ValueSpace |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | For interoperability reasons the constraining facet shall be: <xs:pattern value="\w[\w\|\s\|\i]{0,13}"/>. However for RosettaNet DictionaryItem Identifiers the constraining facet shall be: <xsd:pattern value="[A-HJ-NP-Z]{2}[0-9]{2}[A-HJ-NP-Z]{2}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 159: ATTRIBUTE: majorVersion**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | majorVersion |
| Definition | Identification of an issue of a DictionaryItem in a series of evolving DictionaryItems within a RegistrationAuthority |
| Note | - |
| Remark | Whether the identifier should be changed so that a new DictionaryItem is created or wether the majorVersion of a DictionaryItem should be changed is to be determined for each case separately |
| Example | 009 |
| Rule | Consecutive majorVersion numbers shall be issued in ascending order. A new version of the DictionaryItem shall be generated if at least one attribute of the DictionaryItem is changed |
| **UML** | |
| Data type | Integer |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | positiveInteger |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:minInclusive value=”001”/><br><xs:maxInclusive value=”999”/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 160: ATTRIBUTE: type**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | type |
| Definition | Descriptor identifying the type of a DictionaryItemReference |
| Note | - |
| Remark | - |
| Example | DERIVED |
| Rule | - |
| **UML** | |
| Data type | DictionaryItemReferenceType |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | DictionaryItemReferenceType |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.6 DataElementType

**Table 161: CLASS: DataElementType**

| Field | Value |
|---|---|
| Name | DataElementType |
| Definition | Unit of data for which the identification, description, representation and permissible values are specified by means of a set of attributes |
| Note | The purpose of DataElementTypes are to provide a shared definition of reported Parameters and their associated Characteristics. |
| Remark | - |

**Table 162: ATTRIBUTE: formula**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | formula |
| Definition | Formal rule or statement expressing semantics of a DataElementType providing an alternative definition |
| Note | - |
| Remark | - |
| Example | This example encodes the expression “x squared * y squared”: <math mode=”display” xmlns=”http://www.w3.org/1998/Math/MathML”><mrow><msup><mi>x</mi><mn>2</mn></msup><msup><mi>y</mi><mn>2</mn></msup></mrow></math> |
| Rule | - |
| **UML** | |
| Data type | FormulaLanguage |
| Cardinality | 0..* |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | FormulaLanguage |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | Encoding of Formulas must be limited to MathML [presentational] |
| **References** | |
| Source for definition | IEC 61360-1:2002-02 3.3.4 |

**Table 163: ATTRIBUTE: figure**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | figure |
| Definition | Illustration to clarify the meaning of the definition of a DataElementType |
| Note | A Figure shall not change any essential information of the meaning of the definition |
| Remark | - |
| Example | http://www.semiconductors.philips.com/crsc/images/mainlogo.gif |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 0..* |
| **XML Schema [XSD]** | |
| Built-in data type | anyURI |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | IEC 61360-1:2002-02 3.3.5 |

**Table 164: ATTRIBUTE: allowsValueListExtension**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | allowsValueListExtension |
| Definition | Indicator expressing whether extending values in a value list associated with a DataElementType is allowed or not |
| Note | - |
| Remark | - |
| Example | TRUE |
| Rule | - |
| **UML** | |
| Data type | Boolean |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | boolean |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.7 Association

**Table 165: CLASS: Association**

| Field | Value |
|---|---|
| Name | Association |
| Definition | A semantic relationship between a SemanticDictionaryItemSet and member DictionaryItems. |
| Note | The Association allows the dictionary specifier to define constraints on the library. |
| Remark | - |

**Table 166: ATTRIBUTE: cardinality**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | cardinality |
| Definition | The number of elements in a given mathematical set |
| Note | - |
| Remark | - |
| Example | 0..1 |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xsd:restriction base=”xsd:string”><xsd:enumeration value=”0..1”/><xsd:enumeration value=”0..n”/><xsd:enumeration value=”1..n”/>…</xsd:restriction> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 167: ATTRIBUTE: roleName**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | roleName |
| Definition | The type of association between the SemanticDictionaryItemSet and member DictionaryItems |
| Note | - |
| Remark | - |
| Example | Buyer |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 168: ATTRIBUTE: isComposition**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | isComposition |
| Definition | Signifies that the DictionaryItem can only exist as part of the associated SemanticDictionaryItemSet |
| Note | - |
| Remark | - |
| Example | TRUE |
| Rule | - |
| **UML** | |
| Data type | Boolean |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | boolean |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.8 ValueListItem

**Table 169: CLASS: ValueListItem**

| Field | Value |
|---|---|
| Name | ValueListItem |
| Definition | Representation of a permissible instance of a DataElementType as an element of a ValueList |
| Note | - |
| Remark | - |

**Table 170: ATTRIBUTE: valueCode**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | valueCode |
| Definition | Coded representation of a permissible Value of a non-quantitative DataElementType |
| Note | - |
| Remark | - |
| Example | R |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 171: ATTRIBUTE: valueMeaning**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | valueMeaning |
| Definition | Descriptive part of a permissible Value of a non-quantitative DataElementType |
| Note | - |
| Remark | - |
| Example | Red |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.9 SpecificationReference

**Table 172: CLASS: SpecificationReference**

| Field | Value |
|---|---|
| Name | SpecificationReference |
| Definition | Reference to a Specification that is a source for or provides context to the semantic parts of a DictionaryItem |
| Note | Allows a DictionaryItem to refer to a Specification and provide context (both the type of reference and a description) for the reference. |
| Remark | - |

**Table 173: ATTRIBUTE: sourceReference**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | sourceReference |
| Definition | Descriptor identifying the part of a Specification being referenced |
| Note | - |
| Remark | - |
| Example | 3.3.5 |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 174: ATTRIBUTE: type**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | type |
| Definition | Descriptor identifying the type of sourceReference |
| Note | - |
| Remark | - |
| Example | Clause |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.10 Specification

**Table 175: CLASS: Specification**

| Field | Value |
|---|---|
| Name | Specification |
| Definition | A Specification external to the Dictionary, providing additional context to a DictionaryItem |
| Note | Defines context for a DictionaryItem's reference to a Specification. This type allows a DictionaryItem to refer to an external specification (Specification) and provide context (both the kind of reference and some description) for the reference. |
| Remark | - |

**Table 176: ATTRIBUTE: description**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | description |
| Definition | Informal text describing a Specification |
| Note | - |
| Remark | - |
| Example | Standard data element types with associated classification scheme for electric component |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 177: ATTRIBUTE: source**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | source |
| Definition | Descriptor identifying a Specification being referenced |
| Note | - |
| Remark | - |
| Example | IEC 61360-1:2002-02 |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.11 RelationType

**Table 178: CLASS: RelationType**

| Field | Value |
|---|---|
| Name | RelationType |
| Definition | An association between two Items |
| Note | - |
| Remark | - |

**Table 179: ATTRIBUTE: roleADefinition**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | roleADefinition |
| Definition | The definition of the role Item A has within the association between two Items A and B |
| Note | - |
| Remark | - |
| Example | replaced by |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 180: ATTRIBUTE: roleBDefinition**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | roleBDefinition |
| Definition | The definition of the role Item B has within the association between two Items A and B |
| Note | - |
| Remark | - |
| Example | replaces |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.12 Symbol

**Table 181: CLASS: Symbol**

| Field | Value |
|---|---|
| Name | Symbol |
| Definition | Mark or characters used as a sign for representing a DataElementType |
| Note | - |
| Remark | - |

**Table 182: ATTRIBUTE: rendition**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | rendition |
| Definition | The formal representation of a Symbol |
| Note | The encoding of the rendition shall be limited to presentational MathML |
| Remark | For encoding guidelines see: Units in MathML: http://www.w3.org/TR/mathml-units/ |
| Example | VCC |
| Rule | - |
| **UML** | |
| Data type | MathML |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MathML |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | Mathematical Markup Language (MathML) Version 2.0 (Second Edition) |

**Table 183: ATTRIBUTE: letterSymbol**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | letterSymbol |
| Definition | A simplified, alpha-numeric representation of a Symbol |
| Note | - |
| Remark | - |
| Example | V_CC |
| Rule | See IEC 61360-1:2002-02 3.2.6 |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:pattern value="\w[\w|\s|\i]{0,16}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.13 ValueSpace

**Table 184: CLASS: ValueSpace**

| Field | Value |
|---|---|
| Name | ValueSpace |
| Definition | A restriction for reported Values on Characteristics for a DataElementType. |
| Note | - |
| Remark | The ValueSpace is defined by a language identified by the associated ValueSpaceType. A typical scenario would be to restrict the Values by XML Schema types. |

**Table 185: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | The specification of a restriction for reported Values on Characteristics for a DataElementType. |
| Note | - |
| Remark | - |
| Example | xsd: string |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 186: ATTRIBUTE: allowsExpression**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | allowsExpression |
| Definition | Boolean value indicating if expressions in ValueSpaces are allowed |
| Note | - |
| Remark | - |
| Example | TRUE |
| Rule | - |
| **UML** | |
| Data type | Boolean |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | boolean |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.14 Unit

**Table 187: CLASS: Unit**

| Field | Value |
|---|---|
| Name | Unit |
| Definition | Prescription of the Unit in which the Value of a quantitative DataElementType shall be expressed |
| Note | - |
| Remark | 1) Preference shall be given to SI Units as defined in ISO 31 and to Units as listed in Annex A of IEC 61360-1:2002-02<br>2) For encoding guidelines see: Units in MathML: http://www.w3.org/TR/mathml-units/<br>3) Only base Units, i.e. Units without prefixes shall be used<br>4) Units shall be specified for quantitative DataElementTypes |

**Table 188: ATTRIBUTE: rendition**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | rendition |
| Definition | The formal representation of a Unit |
| Note | The encoding of the rendition shall be limited to presentational MathML |
| Remark | - |
| Example | |
| Rule | - |
| **UML** | |
| Data type | MathML |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MathML |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | Mathematical Markup Language (MathML) Version 2.0 (Second Edition) |

**Table 189: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | An alpha-numeric representation of a Unit |
| Note | - |
| Remark | - |
| Example | Ohm |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:pattern value="\w[\w|\s|\i]{0,16}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.1.15 ValueList

**Table 190: CLASS: ValueList**

| Field | Value |
|---|---|
| Name | ValueList |
| Definition | A set specifying permissible Values of a DataElementType |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.16 ExtensibleDictionaryItem

**Table 191: CLASS: ExtensibleDictionaryItem**

| Field | Value |
|---|---|
| Name | ExtensibleDictionaryItem |
| Definition | An abstract type allowing extension or restriction of DictionaryItems as well as the definition of an inheritance structure |
| Note | Only subtypes of ExtensibleDictionaryItem can be extended or restricted. |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.17 ExtensibleDictionaryItem

**Table 192: CLASS: ExtensibleDictionaryItem**

| Field | Value |
|---|---|
| Name | ExtensibleDictionaryItem |
| Definition | An abstract type allowing extension or restriction of DictionaryItems as well as the definition of an inheritance structure |
| Note | Only subtypes of ExtensibleDictionaryItem can be extended or restricted. |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.18 NonSemanticDictionaryItemSet

**Table 193: CLASS: NonSemanticDictionaryItemSet**

| Field | Value |
|---|---|
| Name | NonSemanticDictionaryItemSet |
| Definition | Represents an arbitrary grouping of DictionaryItems. |
| Note | - |
| Remark | Allows DictionaryItems to be grouped for navigation or maintenance purpose |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.19 SemanticDictionaryItemSet

**Table 194: CLASS: SemanticDictionaryItemSet**

| Field | Value |
|---|---|
| Name | SemanticDictionaryItemSet |
| Definition | Represents a semantic grouping of DictionaryItems |
| Note | - |
| Remark | Allows the specifier to provide definition of types and restriction of information by means of Generalization/Specialization or Restriction or Associations; provides context for the definition of DataElementTypes |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.20 Restriction

**Table 195: CLASS: Restriction**

| Field | Value |
|---|---|
| Name | Restriction |
| Definition | Defines restrictions on two associated ExtensibleDictionaryItems |
| Note | - |
| Remark | Association |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.21 Term

**Table 196: CLASS: Term**

| Field | Value |
|---|---|
| Name | Term |
| Definition | A DictionaryItem defining a word or expression that has a precise meaning in a particular context |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.22 ValueSpaceType

**Table 197: CLASS: ValueSpaceType**

| Field | Value |
|---|---|
| Name | ValueSpaceType |
| Definition | Defines a reference to a language used to define ValueSpaces |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.23 ValueQualifier

**Table 198: CLASS: ValueQualifier**

| Field | Value |
|---|---|
| Name | ValueQualifier |
| Definition | A DictionaryItem defining a qualification for a Value |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.24 ValueQualifierSet

**Table 199: CLASS: ValueQualifierSet**

| Field | Value |
|---|---|
| Name | ValueQualifierSet |
| Definition | A DictionaryItem defining a set with a fixed sequence of ValueQualifiers applicable to the Value of a DataElementType |
| Note | ValueQualifiers in a ValueQualifierSet qualify a single semantic concept as defined in a DataElementType |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.1.25 Dictionary

**Table 200: CLASS: Dictionary**

| Field | Value |
|---|---|
| Name | Dictionary |
| Definition | A computer-sensible set of DictionaryItems defining shared knowledge about one or more domains |
| Note | - |
| Remark | A Dictionary would typically be the root element for all DictionaryItems. |

**Remark:** NO ATTRIBUTES DEFINED

## 10.3.2 Library

### 10.3.2.1 Associations

**Table 201: Definition of ASSOCIATIONS in the "Library" view**

| Name | Description |
|---|---|
| VersionTrace | Defines a linked list of history for an **ItemSpecific**. The linked list is used to trace the versions of the ItemSpecific to enable history of specifications. |
| ItemAggregation | An **Item** can assemble different specifications from **ItemSpecifics** specified else were in the item hierarchy |
| ItemGrouping | Defines a semantic tree of **Items**. If an **Item** is a descendent of another Item, it "inherits" all **Characteristics** of that **Item** |
| ItemRelation::Item roleA | The **RelationshipType** defines two roles (roleA and roleB). This association defines which **Item** plays roleA |
| ItemRelation::Item roleB | The **RelationshipType** defines two roles (roleA and roleB). This association defines which **Item** plays roleB |
| PointOfReference<-> Parameter | A **Parameter** must define a **PointOfReference**. |
| Parameter<-> CharacteristicSpecific | A **Parameter** may have a set of **CharacteristicSpecifics** reported under different conditions. |
| CharacteristicSet<-> Characteristic | A **CharacteristicSet** is made up of a set of **Characteristics** (which in turn may be **CharacteristicSets**) |
| CharacteristicSpecific<-> Value | A **CharacteristicSpecific** may define a set of values (as long as these values have different **ValueQualifiers**) |
| Item<-> CharacteristicSet | This association is in for navigation purpose. **CharacterisiticSets** must be rooted at some **Item**. |
| Value <-> ValueQualifier | Defines which **ValueQualifiers** are relevant for a specified **Value** on a **CharacteristicSpecific**. |
| Parameter <-> DataElementType | Specifies the link between the **Parameter** and its semantic definition i a **Dictionary**. |
| Characteristic <-> BooleanExpression | A **Characteristic** may define a condition under which it applies. The condition is a BooleanExpression. |
| CharacteristicSet <-> DictionaryItem | A **CharacteristicSet** must have link to its semantic definition in a **Dictionary**. The semantic definition is either a **Term** or a **SemanticDictionaryItemSet**. |

### 10.3.2.2 Characteristic

**Table 202: CLASS: Characteristic**

| Field | Value |
|---|---|
| Name | Characteristic |
| Definition | Property of an Item as defined by one or more DataElementTypes |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.2.3 CharacteristicSet

**Table 203: CLASS: CharacteristicSet**

| Field | Value |
|---|---|
| Name | CharacteristicSet |
| Definition | A set of Characteristics |
| Note | - |
| Remark | If a condition is reported on a CharacteristicSet, the semantic is that the conditions implicitly applies to all its members. If a CharacteristicSet's definition is a SemanticDictionaryItemSet, its members are constrained by the rules in this SemanticDictionaryItemSet |

**Remark:** NO ATTRIBUTES DEFINED

#### 10.3.2.4 CharacteristicSpecific

**Table 204: CLASS: CharacteristicSpecific**

| Field | Value |
|---|---|
| Name | CharacteristicSpecific |
| Definition | Represents a set of Values reported on a Parameter under a Condition as specified by a DataElementType |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.2.5 Item

**Table 205: CLASS: Item**

| Field | Value |
|---|---|
| Name | Item |
| Definition | An object type for which Characteristics can be specified |
| Note | An Item is the main “thing” that information [Characteristics, including Conditions and Values] is reported on |
| Remark | The Item type is an abstract model type used to make visible the commonality between ItemSpecific and ItemSet. Here used to show that instances of both ItemSpecific and ItemSet can have ItemRelation, Parameters and CharacteristicSets. |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.2.6 ItemRelation

**Table 206: CLASS: ItemRelation**

| Field | Value |
|---|---|
| Name | ItemRelation |
| Definition | A relationship between two Items for which the semantics are specified by a RelationType |
| Note | - |
| Remark | Every instance of an ItemRelation is associated with an instance of a RelationType (information stored in dictionaries). |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.2.7 ItemSet

**Table 207: CLASS: ItemSet**

| Field | Value |
|---|---|
| Name | ItemSet |
| Definition | A set of Items sharing the same Characteristics |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.2.8 ItemSpecific

**Table 208: CLASS: ItemSpecific**

| Field | Value |
|---|---|
| Name | ItemSpecific |
| Definition | The most atomic object type for which Characteristics can be specified |
| Note | - |
| Remark | - |

**Table 209: ATTRIBUTE: frozen**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | frozen |
| Definition | Signifies that Characteristics, Conditions and Values reported on an ItemSpecific are decoupled from any information specified by inheritance or aggregation |
| Note | - |
| Remark | - |
| Example | FALSE |
| Rule | - |
| **UML** | |
| Data type | Boolean |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | boolean |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.2.9 Parameter

**Table 210: CLASS: Parameter**

| Field | Value |
|---|---|
| Name | Parameter |
| Definition | A concept that can be reported-on for a PointOfReference associated with an Item |
| Note | A Parameter is defined in the Dictionary by a DataElementType |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.2.10 PointOfReference

**Table 211: CLASS: PointOfReference**

| Field | Value |
|---|---|
| Name | PointOfReference |
| Definition | A point of reference associated with an Item for which a Parameter can be specified |
| Note | Can be Item, Terminus, Mode, Connection or TerminusModeConnectionAssociation [TMCA] |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.3 Enumerations

**Remark:** NO ASSOCIATIONS DEFINED

**Table 212: RegistrationStatus**

| Enumeration | Definition |
|---|---|
| **RegistrationStatus** | A designation of the position in the registration life-cycle of a DictionaryItem |
| PROPOSED | Status of a DictionaryItem from the time a DictionaryItem has been assigned an identifier until a draft DictionaryItem is available for review and validation |
| DRAFT | Status of a DictionaryItem from the time a draft DictionaryItem is available for review and validation until the status of the DictionaryItem becomes either RELEASED or REJECTED |
| IN VALIDATION | Status of a DictionaryItem while it is being validated |
| RELEASED | Status of a DictionaryItem that will be released for use on the effectiveDate |
| REJECTED | Status of a DictionaryItem that will be not be released for use |
| OBSOLETE | Status of a DictionaryItem that is no longer recommended for use but is still available for reference only |

**Table 213: ChangeRequestStatus**

| Enumeration | Definition |
|---|---|
| **ChangeRequest Status** | A string designating the phase in the life cycle of a ChangeRequest |
| SUBMITTED | Status of a ChangeRequest from the time it was submitted to the time it has been assigned for evaluation |
| IN EVALUATION | Status of a ChangeRequest during the time it is being evaluated for disposition |
| IN PROGRESS | Status of a ChangeRequest while it is being resolved |
| ON HOLD | Status of a ChangeRequest that is no longer in PROGRESS but not yet RESOLVED |
| RESOLVED | Status of a ChangeRequest for which the final disposition is known |

**Table 214: TranslationStatus**

| Enumeration | Definition |
|---|---|
| **TranslationStatus** | The status specific to the state of a translation activity for a localized string |
| PROPOSED | Status of a Translation from the time a Translation has been submitted until the time a draft Translation is available for review and validation |
| DRAFT | Status of a Translation from the time a draft Translation is available for review and validation until the status of the Translation becomes RELEASED |
| APPROVED | Status of a Translation that has been approved |

**Table 215: ChangeRequestType**

| Enumeration | Definition |
|---|---|
| **ChangeRequestType** | A designator classifying a ChangeRequest |
| CREATE | Designator for a ChangeRequest which is submitted with the intent to CREATE one or more DictionaryItems |
| REVISE | Designator for a ChangeRequest which is submitted with the intent to REVISE one or more DictionaryItems |
| TRANSLATE | Designator for a ChangeRequest which is submitted with the intent to TRANSLATE one or more DictionaryItems |
| OBSOLETE | Designator for a ChangeRequest which is submitted with the intent to OBSOLETE one or more DictionaryItems |

**Table 216: EventType**

| Enumeration | Definition |
|---|---|
| **EventType** | A designator describing a specific step in the evolution of a ChangeRequest |
| SUBMITTED | Status of a ChangeRequest from the time it was submitted to the time it has been assigned for evaluation |
| IN EVALUATION | Status of a ChangeRequest during the time it is being evaluated for disposition |
| IN PROGRESS | Status of a ChangeRequest while it is being resolved |
| IN VALIDATION | Status of a ChangeRequest while it is being validated |
| RESOLVED | Status of a ChangeRequest for which the final disposition is known |
| ON HOLD | Status of a ChangeRequest that is no longer in PROGRESS but not yet RESOLVED |
| REJECTED | Status of a ChangeRequest that will not be implemented |
| COMMENT | Event that records a comment at a date and time specified by a DateTimeStamp |

**Table 217: SpecificationReferenceType**

| Enumeration | Definition |
|---|---|
| **SpecificationReferenceType** | Defines context for a DictionaryItem's reference to a Specification. This type allows a DictionaryItem to refer to an external specification (Specification) and provide context (both the kind of reference and some description) for the reference. |
| DERIVED | A relation between a SpecificationReference and a DictionaryItem meaning that the DictionaryItem is based upon a DictionaryItem defined in the SpecificationReference |
| UNSPECIFIED | A relation between a SpecificationReference and a DictionaryItem meaning that is of unknown nature |

### 10.3.4 Text and language primitives

#### 10.3.4.1 Associations

**Table 218: Definition of ASSOCIATIONS in the "Text & language primitives" view**

| Name | Description |
|---|---|
| MultiLingualString <-> Translation | Defines the links to various translations of the source definition to other languages. |
| MultiLingualString <-> LocalizedString | Defines the content in its source language for the MultiLingualString |
| Translation <-> TranslationAuthority | Defines the link to the party that performed a specific translation. |
| Translation <-> LocalizedString | Defines the link to the content holding a translation of the source content of the MultiLingualString. |
| LocalizedString <-> Language | Defines the language of the content for a LocalizedString. |
| LocalizedString <-> Script | Defines the Script used for the content of a LocalizedString. |
| LocalizedString <-> StructuredText | Defines the link to the content of a LocalizedString. |

#### 10.3.4.2 MultiLingualString

**Table 219: CLASS: MultiLingualString**

| Field | Value |
|---|---|
| Name | MultiLingualString |
| Definition | A primitive type enabling and managing language variants for attributes |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.4.3 LocalizedString

**Table 220: CLASS: LocalizedString**

| Field | Value |
|---|---|
| Name | LocalizedString |
| Definition | A human-readable text expressed in a locale-specific language and script |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

#### 10.3.4.4 RichText

**Table 221: CLASS: RichText**

| Field | Value |
|---|---|
| Name | RichText |
| Definition | Text that may have an structure as commonly found in XHTML documents |
| Note | - |
| Remark | RichText is limited to XHTML modules 5.2, 5.4.3, 5.6.1, 5.7 |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.4.5 Translation

**Table 222: CLASS: Translation**

| Field | Value |
|---|---|
| Name | Translation |
| Definition | A rendering from one language into another |
| Note | - |
| Remark | - |

**Table 223: ATTRIBUTE: status**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | status |
| Definition | The status specific to the state of a translation activity for a localized string. |
| Note | - |
| Remark | - |
| Example | APPROVED |
| Rule | - |
| **UML** | |
| Data type | TranslationStatus |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | TranslationStatus |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.4.6 TranslationAuthority

**Table 224: CLASS: TranslationAuthority**

| Field | Value |
|---|---|
| Name | TranslationAuthority |
| Definition | The organizational entity responsible for translating a LocalizedString |
| Note | - |
| Remark | - |

**Table 225: ATTRIBUTE: identifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | identifier |
| Definition | A code to uniquely identify a TranslationAuthority conforming to the format specified for OrganizationIdentifier |
| Note | - |
| Remark | - |
| Example | “60:023936581” = ICD:OI = DUNS:RosettaNet |
| Rule | - |
| **UML** | |
| Data type | OrganizationIdentifier |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | OrganizationIdentifier |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.4.7 Language

**Table 226: CLASS: Language**

| Field | Value |
|---|---|
| Name | Language |
| Definition | The words, their pronunciation, and the methods of combining them used and understood by a community |
| Note | - |
| Remark | - |

**Table 227: ATTRIBUTE: languageCode**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | languageCode |
| Definition | Language designator as described in [RFC3066] |
| Note | The LanguageCode is specified as the 2 letter language letter code according to ISO 639-1 followed by a hyphen ["-"], followed by the 2 letter country code according to ISO 3166 |
| Remark | "zh-CN" shall be the languageCode designating "simplified Chinese" |
| Example | en |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 228: ATTRIBUTE: countryCode**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | countryCode |
| Definition | A two letter code identifying a country as defined in ISO 3166 |
| Note | - |
| Remark | - |
| Example | US |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.4.8 Script

**Table 229: CLASS: Script**

| Field | Value |
|---|---|
| Name | Script |
| Definition | A collection of characters for displaying written text, all of which have a common characteristic that justifies their consideration as a distinct set. |
| Note | - |
| Remark | One script can be used for several different languages (for example, Latin script, which covers all of Western Europe). Some written languages require multiple scripts (for example, Japanese, which requires at least three scripts: the hiragana and katakana syllabaries and the kanji ideographs imported from China). |

**Table 230: ATTRIBUTE: scriptName**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | scriptName |
| Definition | A code representing the name of a Script |
| Note | The scriptName shall be the 3 digit code from ISO 15924:2004 |
| Remark | - |
| Example | 215 for Latin, 220 for Cyrillic, 501 for Han [simplified variant], 500 for Han [Kanji], 286 for Hangul |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1..* |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.5 Specialized primitives

**Remark:** NO ASSOCIATIONS DEFINED

#### 10.3.5.1 OrganizationIdentifier

**Table 231: CLASS: OrganizationIdentifier**

| Field | Value |
|---|---|
| Name | OrganizationIdentifier |
| Definition | A Primitive type for uniquely identifying an organization |
| Note | The OrganizationIdentifier is globally unique |
| Remark | NOTE: delimiter between the next 5 attributes is COLON ":" |

**Table 232: ATTRIBUTE: internationalCodeDesignator**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | internationalCodeDesignator |
| Definition | A code that uniquely identifies an organization identification scheme according to ISO/IEC 6523-1 (1998) |
| Note | - |
| Remark | - |
| Example | 60: identifies DUNS which will be used by RosettaNet so "60:023936581" = ICD:OI = DUNS:RosettaNet |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:pattern value="[1-9][0-9]{0,3}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 233: ATTRIBUTE: organizationIdentifierCode**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | organizationIdentifierCode |
| Definition | A code that uniquely identifies an organization within an organization identification scheme |
| Note | - |
| Remark | - |
| Example | 023936581: identifies RosettaNet in the DUNS namespace; so “60:023936581” = ICD:OI = DUNS:RosettaNet |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:pattern value=”[\w]{1,35}”/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 234: ATTRIBUTE: OrganizationPartIdentifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | OrganizationPartIdentifier |
| Definition | A code that uniquely identifies a particular organization part within an identified organization. |
| Note | - |
| Remark | - |
| Example | 0239365810002: is an imaginary identifier for the Asian part of RosettaNet in the RosettaNet namespace |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:pattern value="[\w]{1,35}"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 235: ATTRIBUTE: OrganizationPartIdentifierSourceIndicator**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | OrganizationPartIdentifierSourceIndicator |
| Definition | The particular value (digit or capital letter) designating the source of an organization part identifier. |
| Note | - |
| Remark | - |
| Example | 1 |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | <xs:pattern value="[019]"/> |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 236: ATTRIBUTE: DictionaryIdentifier**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | DictionaryIdentifier |
| Definition | The identifier assigned to a particular dictionary within the identified organization or organization part. |
| Note | - |
| Remark | - |
| Example | 61360_4_1 |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 0..1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.5.2 FormulaLanguage

**Table 237: CLASS: FormulaLanguage**

| Field | Value |
|---|---|
| Name | FormulaLanguage |
| Definition | A formally defined method to encode Formulas |
| Note | - |
| Remark | - |

### 10.3.5.3 MathML

**Table 238: CLASS: MathML**

| Field | Value |
|---|---|
| Name | MathML |
| Definition | Mathematical Markup Language |
| Note | The markup must follow Presentation Markup rules as defined in http://www.w3.org/TR/2003/REC-MathML2-20031021/ |
| Remark | - |

### 10.3.6 Values

#### 10.3.6.1 Associations

**Table 239: Definition of ASSOCIATIONS in the "Value" view**

| Name | Description |
|---|---|
| Value <-> ValueQualifier | Defines a set of value qualifiers for a particular value. The value qualifier provides some semantic for the value. E.g., "Max" specifying that the value represents the maximum allowed value for a characteristic |
| BooleanExpression::right <-> BinaryBooleanExpression | Defines the right hand side value of a binary boolean expression. |
| BooleanExpression::left <-> BinaryBooleanExpression | Defines the left hand side value of a binary boolean expression. |
| BinaryComparator::right <-> ComparableValue | Defines the right hand side value of a comparison.. |
| BinaryComparator::left <-> ComparableValue | Defines the left hand side value of a comparison |
| ComparableValue::from <-> Range | Defines the (inclusive) lower bound in a range. |
| ComparableValue::to <-> Range | Defines the (inclusive) upper bounds in a range |
| NumericValue::right <-> BinaryNumericExpression | Defines the right hand side in a binary numeric expression. E.g., in an expression A + B, this association defines the link between the "+" and B. |
| NumericValue::left <-> BinaryNumericExpression | Defines the left hand side in a binary numeric expression. E.g., in an expression A + B, this association defines the link between the "+" and A. |
| NumericValue <-> UnaryNumericExpression | Defines the link between the unary numeric expression and what it modifies. E.g., in an expression A!, defines the relationship between the factorial and A. |
| Parameter <-> ParameterReference | Defines which parameter a parameter references refers to. |

#### 10.3.6.2 ParameterReference

**Table 240: CLASS: ParameterReference**

| Field | Value |
|---|---|
| Name | ParameterReference |
| Definition | Allows a Parameter to be used as a variable in an expression |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.3 FigureReference

**Table 241: CLASS: FigureReference**

| Field | Value |
|---|---|
| Name | FigureReference |
| Definition | Allows a Figure to be used as a Value |
| Note | - |
| Remark | - |

**Table 242: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | Reference to a Figure |
| Note | - |
| Remark | - |
| Example | http://www.semiconductors.philips.com/crsc/images/mainlogo.gif |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | anyURI |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.4 Value

**Table 243: CLASS: Value**

| Field | Value |
|---|---|
| Name | Value |
| Definition | A quantity represented by a symbol or set of symbols |
| Note | - |
| Remark | The Value type is an modeling artifact capturing commonality between all concrete values |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.5 NoValue

**Table 244: CLASS: NoValue**

| Field | Value |
|---|---|
| Name | NoValue |
| Definition | A NULL Value for which the reason is explicitly stated |
| Note | - |
| Remark | The Value type is an modeling artifact capturing commonality between all concrete values |

**Table 245: ATTRIBUTE: noValueReason**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | noValueReason |
| Definition | The reason as defined in a ValueList for not supplying a Value |
| Note | - |
| Remark | - |
| Example | NOT MEASURED |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 246: ATTRIBUTE: valueCode**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | valueCode |
| Definition | Coded representation of a permissible Value in a ValueList |
| Note | - |
| Remark | - |
| Example | NM |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.6 NumericValue

**Table 247: CLASS: NumericValue**

| Field | Value |
|---|---|
| Name | NumericValue |
| Definition | A Value of the type Numeric |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.7 StringValue

**Table 248: CLASS: StringValue**

| Field | Value |
|---|---|
| Name | StringValue |
| Definition | A Value of the type String |
| Note | - |
| Remark | - |

**Table 249: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | The content of a StringValue |
| Note | - |
| Remark | - |
| Example | Strings are here, strings are there, strings are everywhere |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.8 DurationValue

**Table 250: CLASS: DurationValue**

| Field | Value |
|---|---|
| Name | DurationValue |
| Definition | A Value of the type DateTime |
| Note | - |
| Remark | - |

**Table 251: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | The content of a DurationValue |
| Note | - |
| Remark | - |
| Example | 2005-01-14T12:45:00Z |
| Rule | - |
| **UML** | |
| Data type | Duration |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | duration |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.9 DateTimeValue

**Table 252: CLASS: DateTimeValue**

| Field | Value |
|---|---|
| Name | DateTimeValue |
| Definition | A Value of the type Duration |
| Note | - |
| Remark | - |

**Table 253: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | The content of a DateTimeValue |
| Note | - |
| Remark | - |
| Example | 2005-01-14T12:45:00Z |
| Rule | - |
| **UML** | |
| Data type | DateTime |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | dateTime |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.10 RealValue

**Table 254: CLASS: RealValue**

| Field | Value |
|---|---|
| Name | RealValue |
| Definition | A Value of the type Real |
| Note | - |
| Remark | - |

**Table 255: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | The content of a RealValue |
| Note | - |
| Remark | - |
| Example | 20.00 |
| Rule | - |
| **UML** | |
| Data type | Real |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | float |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.11 IntegerValue

**Table 256: CLASS: IntegerValue**

| Field | Value |
|---|---|
| Name | IntegerValue |
| Definition | A Value of the type Integer |
| Note | - |
| Remark | - |

**Table 257: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | The content of a IntegerValue |
| Note | - |
| Remark | - |
| Example | 20 |
| Rule | - |
| **UML** | |
| Data type | Integer |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | integer |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.12 Range

**Table 258: CLASS: Range**

| Field | Value |
|---|---|
| Name | Range |
| Definition | A Value of the type Range |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.13 BooleanConstant

**Table 259: CLASS: BooleanConstant**

| Field | Value |
|---|---|
| Name | BooleanConstant |
| Definition | A Constant of the type Boolean |
| Note | - |
| Remark | - |

**Table 260: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | The content of a BooleanConstant |
| Note | - |
| Remark | - |
| Example | TRUE |
| Rule | - |
| **UML** | |
| Data type | Boolean |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | boolean |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.14 BooleanExpression

**Table 261: CLASS: BooleanExpression**

| Field | Value |
|---|---|
| Name | BooleanExpression |
| Definition | A mathematical expression in which all variables involved are either 0 or 1; it evaluates to either 0 or 1 |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.15 InformalBooleanExpression

**Table 262: CLASS: InformalBooleanExpression**

| Field | Value |
|---|---|
| Name | InformalBooleanExpression |
| Definition | Represents a value that requires a human to interpret but that conceptually yields a Boolean |
| Note | - |
| Remark | - |

**Table 263: ATTRIBUTE: content**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | content |
| Definition | The content of an InformalBooleanExpression |
| Note | - |
| Remark | - |
| Example | In accordance with the Absolute Maximum Rating System (IEC 60134) |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.16 BinaryBooleanExpression

**Table 264: CLASS: BinaryBooleanExpression**

| Field | Value |
|---|---|
| Name | BinaryBooleanExpression |
| Definition | An abstract type for a Boolean expression with two operands |
| Note | - |
| Remark | Supported functions are AND, OR, XOR |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.17 BinaryNumericExpression

**Table 265: CLASS: BinaryNumericExpression**

| Field | Value |
|---|---|
| Name | BinaryNumericExpression |
| Definition | An abstract type for a Numeric expression with two operands |
| Note | - |
| Remark | Supported functions are PLUS, MINUS, MULTIPLY, DIVIDE |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.18 UnaryNumericExpression

**Table 266: CLASS: UnaryNumericExpression**

| Field | Value |
|---|---|
| Name | UnaryNumericExpression |
| Definition | An abstract type for a Numeric expression with one operand |
| Note | - |
| Remark | Supported functions are ABS, SQRT |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.19 ComparableValue

**Table 267: CLASS: ComparableValue**

| Field | Value |
|---|---|
| Name | ComparableValue |
| Definition | An abstract SuperType representing a value that can be compared to another value of the same type. |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.20 BinaryComparator

**Table 268: CLASS: BinaryComparator**

| Field | Value |
|---|---|
| Name | BinaryComparator |
| Definition | An abstract type for a Boolean expression that compares two operands |
| Note | - |
| Remark | Supported functions are LT, GT, LTEQ, GTEQ, EQ |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.21 Xor

**Table 269: CLASS: Xor**

| Field | Value |
|---|---|
| Name | Xor |
| Definition | The binary operator Exclusive OR |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.22 Or

**Table 270: CLASS: Or**

| Field | Value |
|---|---|
| Name | Or |
| Definition | The binary operator OR |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.23 And

**Table 271: CLASS: And**

| Field | Value |
|---|---|
| Name | And |
| Definition | The binary operator AND |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.24 Gt

**Table 272: CLASS: Gt**

| Field | Value |
|---|---|
| Name | Gt |
| Definition | The binary operator GREATER THAN |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.25 Lt

**Table 273: CLASS: Lt**

| Field | Value |
|---|---|
| Name | Lt |
| Definition | The binary operator LESS THAN |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.26 GtEq

**Table 274: CLASS: GtEq**

| Field | Value |
|---|---|
| Name | GtEq |
| Definition | The binary operator GREATER OR EQUAL THAN |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.27 LtEq

**Table 275: CLASS: LtEq**

| Field | Value |
|---|---|
| Name | LtEq |
| Definition | The binary operator LESS OR EQUAL THAN |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.28 Eq

**Table 276: CLASS: Eq**

| Field | Value |
|---|---|
| Name | Eq |
| Definition | The binary operator EQUAL |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.29 Plus

**Table 277: CLASS: Plus**

| Field | Value |
|---|---|
| Name | Plus |
| Definition | The arithmetic operator PLUS |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.30 Minus

**Table 278: CLASS: Minus**

| Field | Value |
|---|---|
| Name | Minus |
| Definition | The arithmetic operator MINUS |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.31 Multiply

**Table 279: CLASS: Multiply**

| Field | Value |
|---|---|
| Name | Multiply |
| Definition | The arithmetic operator MULTIPLY |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.32 Divide

**Table 280: CLASS: Divide**

| Field | Value |
|---|---|
| Name | Divide |
| Definition | The arithmetic operator DIVIDE |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.33 Sqrt

**Table 281: CLASS: Sqrt**

| Field | Value |
|---|---|
| Name | Sqrt |
| Definition | The arithmetic operator SQUARE ROOT |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.34 Abs

**Table 282: CLASS: Abs**

| Field | Value |
|---|---|
| Name | Abs |
| Definition | The arithmetic operator ABOLUTE VALUE |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.3.6.35 BooleanValueListItemReference

**Table 283: CLASS: BooleanValueListItemReference**

| Field | Value |
|---|---|
| Name | BooleanValueListItemReference |
| Definition | Reference to a ValueListItem of type Boolean |
| Note | - |
| Remark | - |

**Table 284: ATTRIBUTE: valueCode**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | valueCode |
| Definition | Coded representation of a permissible Value of a non-quantitative DataElementType |
| Note | - |
| Remark | - |
| Example | TRUE, 0 |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.36 InformalValueListItemReference

**Table 285: CLASS: InformalValueListItemReference**

| Field | Value |
|---|---|
| Name | InformalValueListItemReference |
| Definition | Reference to a ValueListItem that represents a Value unsuitable for use in an expression |
| Note | - |
| Remark | - |

**Table 286: ATTRIBUTE: valueCode**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | valueCode |
| Definition | Coded representation of a permissible Value of a non-quantitative DataElementType |
| Note | - |
| Remark | - |
| Example | |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.3.6.37 NumericValueListItemReference

**Table 287: CLASS: NumericValueListItemReference**

| Field | Value |
|---|---|
| Name | NumericValueListItemReference |
| Definition | Reference to a ValueListItem of type Numeric |
| Note | - |
| Remark | - |

**Table 288: ATTRIBUTE: valueCode**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | valueCode |
| Definition | Coded representation of a permissible Value of a non-quantitative DataElementType |
| Note | - |
| Remark | - |
| Example | 230 |
| Rule | - |
| **UML** | |
| Data type | String |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | string |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

## 10.4 RosettaNet EC Domain Extensions

### 10.4.1 EC Domain Dictionary Extensions

**Remark:** NO ASSOCIATIONS DEFINED

#### 10.4.1.1 TerminusType

**Table 289: CLASS: TerminusType**

| Field | Value |
|---|---|
| Name | TerminusType |
| Definition | A DictionaryItem defining metadata for a Terminus |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

#### 10.4.1.2 ModeType

**Table 290: CLASS: ModeType**

| Field | Value |
|---|---|
| Name | ModeType |
| Definition | A DictionaryItem defining metadata for a Mode |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

#### 10.4.1.3 ConnectionType

**Table 291: CLASS: ConnectionType**

| Field | Value |
|---|---|
| Name | ConnectionType |
| Definition | A DictionaryItem defining metadata for a Connection |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

## 10.4.2 EC Domain Library Extensions

### 10.4.2.1 Associations

**Table 292: Definition of ASSOCIATIONS in the "EC Library extensions" view**

| Name | Description |
|---|---|
| TerminusSet<->Terminus | Specification of the grouping of **Termini** |
| Terminus<->TerminusType | Represents the link from a **Terminus** to its metadata definition |
| ConnectionSet<-> Connection | Specification of the grouping of **Connections**. |
| Connection<-> ConnectionType | Represents the link from **Connection** to its metadata definition. |
| Item<->ConnectionSpecific | Specifies that a **ConnectionSpecific** must be associated with one and only one **Item** |
| Item<->TerminusSpecific | Specifies that a **TerminusSpecific** must be linked to one and only one **Item** |
| Item<->Mode | Specifies that a **Mode** must be linked to one and only one **Item**. |
| TerminusModeConnectionAssociation<-> TerminusSpecific | Linking the **Terminus** part of a **TerminusModeConnectionAssociation** |
| TerminusModeConnectionAssociation<-> ConnectionSpecific | Linking the **Connection** part of a **TerminusModeConnectionAssociation** |
| TerminusModeConnectionAssociation<-> Mode | Linking the **Mode** part of a **TerminusModeConnectionAssociation** |
| Mode<->ModeType | Defines the meta information for a **Mode**. |

### 10.4.2.2 TerminusSpecific

**Table 293: CLASS: TerminusSpecific**

| Field | Value |
|---|---|
| Name | TerminusSpecific |
| Definition | A physical connection to an Item |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.4.2.3 TerminusSet

**Table 294: CLASS: TerminusSet**

| Field | Value |
|---|---|
| Name | TerminusSet |
| Definition | A group of Termini |
| Note | - |
| Remark | - |

**Table 295: ATTRIBUTE: isOrdered**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | isOrdered |
| Definition | A Boolean designating whether the order of Termini in a TerminusSet is significant or not |
| Note | - |
| Remark | Boolean TRUE means “is significant” |
| Example | FALSE |
| Rule | - |
| **UML** | |
| Data type | Boolean |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | boolean |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.4.2.4 Terminus

**Table 296: CLASS: Terminus**

| Field | Value |
|---|---|
| Name | Terminus |
| Definition | An abstract SuperType defining the commonality between TerminusSpecific and TerminusSet |
| Note | - |
| Remark | In the Electronic Component domain common alternate names are pin, ball and lead |

**Table 297: ATTRIBUTE: name**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | name |
| Definition | A string identifying a Terminus |
| Note | - |
| Remark | - |
| Example | Pin 1 |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 298: ATTRIBUTE: description**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | description |
| Definition | Text describing a Terminus |
| Note | - |
| Remark | - |
| Example | supply voltage |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.4.2.5 ConnectionSpecific

**Table 299: CLASS: ConnectionSpecific**

| Field | Value |
|---|---|
| Name | ConnectionSpecific |
| Definition | A logical connection to an Item |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

### 10.4.2.6 ConnectionSet

**Table 300: CLASS: ConnectionSet**

| Field | Value |
|---|---|
| Name | ConnectionSet |
| Definition | A group of Connections |
| Note | - |
| Remark | - |

**Table 301: ATTRIBUTE: isOrdered**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | isOrdered |
| Definition | A Boolean designating whether the order of Connections in a ConnectionSet is significant or not |
| Note | - |
| Remark | Boolean TRUE means “is significant” |
| Example | TRUE [when ConnectionSet is an Address Bus - A0, A1, A2, .. An] |
| Rule | - |
| **UML** | |
| Data type | Boolean |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | boolean |
| User-derived data type | - |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.4.2.7 Connection

**Table 302: CLASS: Connection**

| Field | Value |
|---|---|
| Name | Connection |
| Definition | An abstract SuperType defining the commonality between ConnectionSpecific and ConnectionSet |
| Note | - |
| Remark | - |

**Table 303: ATTRIBUTE: name**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | name |
| Definition | A string identifying a Connection |
| Note | - |
| Remark | In the Electronic Component domain this is commonly known as a signal |
| Example | Address Bus |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 304: ATTRIBUTE: description**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | description |
| Definition | Text describing a Connection |
| Note | - |
| Remark | - |
| Example | The signals used to address external memory listed from least significant to most significant |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.4.2.8 Mode

**Table 305: CLASS: Mode**

| Field | Value |
|---|---|
| Name | Mode |
| Definition | The state of an Item |
| Note | - |
| Remark | - |

**Table 306: ATTRIBUTE: name**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | name |
| Definition | A string identifying a Mode |
| Note | - |
| Remark | A change in Mode may cause a different Connection to appear on a particular Terminus. Characteristics and associated Values may be different for the same Connection in different Modes. |
| Example | Sleep |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

**Table 307: ATTRIBUTE: description**

| Field | Value |
|---|---|
| **Descriptions** | |
| Name | description |
| Definition | Text describing a Mode |
| Note | - |
| Remark | - |
| Example | A very low power state for the device where all outputs are high impedance and a HIGH signal on the Wake-Up pin is required to return to normal operation. |
| Rule | - |
| **UML** | |
| Data type | MultiLingualString |
| Cardinality | 1 |
| **XML Schema [XSD]** | |
| Built-in data type | - |
| User-derived data type | MultiLingualString |
| Lexical space | - |
| Constraining facet | - |
| Implementation notes | - |
| **References** | |
| Source for definition | - |

### 10.4.2.9 TerminusModeConnectionAssociation

**Table 308: CLASS: TerminusModeConnectionAssociation**

| Field | Value |
|---|---|
| Name | TerminusModeConnectionAssociation |
| Definition | An association class linking a TerminusSpecific, a ConnectionSpecific and a Mode |
| Note | - |
| Remark | - |

**Remark:** NO ATTRIBUTES DEFINED

# 11. Definition of key terms used in this specification

Table 309 “Definition of key terms” on page 246 defines terms used in this Specification that are not explicitly defined in Section 10 “Data Dictionary” on page 106. The terms are defined as applied in the context of this Specification.

**Table 309: Definition of key terms**

| Term | Definition |
|---|---|
| abstract super type | Stated without reference to a specific instance of a grouping of objects within a hierarchical structure; a super type that exists only to show the relationships between other items in the model that is not intended to be realized in any implementation of the model |
| abstract type | Stated without reference to a specific instance of a group; a type that has irrelevant details removed |
| abstract type model | A model of interrelated entity types that has abstracted some details in order to focus on key properties and interrelations |
| acyclic | Non-repeating |
| analysis model | An abstract formal model that is a partial concretization of the domain type model (e.g., the RDA) in terms of the system requirements for a given company or organization; this model is still more abstract than the design model, which further concretizes the analysis model in terms of a given processing platform and specific computer languages |
| assembly | The fitting together of partial specifications into a higher-level specification. |
| attribute | A property inherent in or ascribed to an object |
| built-in data type | Built-in datatypes are those which are defined in [XSD] |
| cardinality | A number or set of numbers indicating the number of times a given item can or must exist |
| class | A set, collection, or group of objects containing members that have certain attributes or characteristics in common |
| concrete dictionary entries | The actual list of items or words in a given version of the dictionary |
| constraining facet | Constrains the permitted values of a datatype |
| constraints | Restrictions that define the limits of an item |
| descendant | An item within a hierarchical relationship; or an item derived from a prototype or earlier form in a hierarchy |
| dictionary | A collection of metadata elements representing a repository of shared knowledge |
| domain expert | A person with a high degree of skill or knowledge within a certain field or area |
| EC domain | The domain of Electronic Components |
| ECIX | Electronic Component Information Exchange |
| enumeration type | A value type whose members are defined by a fixed list of literals |
| formal model | A representation of the system conforming to rigorous rules. A model is said to be formal when it is based on a language that has a well-defined form or syntax (such as UML), meaning “semantics”, and possibly rules of analysis, inference, or proof for its constructs. The syntax may be graphical or textual. |

**Table 309: Definition of key terms**

| Term | Definition |
|---|---|
| formal semantic | The official meaning, especially meaning in a language |
| IEC | International Electrotechnical Committee: A global organization that prepares and publishes international standards for all electrical, electronic and related technologies. These serve as a basis for national standardization and as references when drafting international tenders and contracts. |
| information structure | The organization of elements within a dictionary |
| inheritance | In a predecessor to successor relationship this the ability of the child to take on the characteristics, conditions, and values of the parent |
| instance diagram | A specific example used to explain how (a part of) a model works |
| instantiate | To create a specific example versus an abstract rendering |
| invariant business rule | A rule that is constant across organizations |
| item inheritance hierarchy | An organizational structure depicting the relationship between items that allow one item to inherit attributes from another |
| item inheritance tree | An organizational structure depicting the relationship between items that allow one item to inherit attributes from another |
| item specifier | The name for the mechanism by which characteristic can are attached to an item set |
| lexical space | The set of valid literals for a datatype |
| library | A collection of data or objects arranged for ease of use |
| lifecycle | A progression through a series of differing stages in the development, maturity, and discontinuance of a concept, activity, or product |
| MathML | Mathematical Markup Language - an XML application for describing mathematical notation and capturing both its structure and content that is provided by the World Wide Web Consortium (W3C). |
| mutable business rule | A rule that can vary, e.g. across organizations, with time or with locale |
| non-quantitative | Cannot be specified as a finite or definite number or amount |
| optionality | Description of the requirement to use zero, one or more instances of an object or association. |
| parseable string | A sequence of alphanumeric characters that can be processed one character at a time by a computer program according to a predefined set of syntax rules. |
| predecessor association | An association to a DictionaryItem that links to the DictionaryItem that is replaced by the current DictionaryItem |
| prescriptive | Authoritative |
| primitive | Not derived from something else, primary or basic |
| product classification tree | A hierarchical structure that depicts the relationship between products |
| RDA | RosettaNet Dictionary Architecture |
| recursively | In terms of this document this is a rule that is applied to the entire successor/predecessor relationship in a hierarchy. |

**Table 309: Definition of key terms**

| Term | Definition |
|---|---|
| reference model | • A schematic description of a system that can be used to provide direction<br>• A schematic description of a system that can be used to provide direction; a formal structure and a set of definitions for describing the implicit and explicit concepts and relationships used in a specific area, e.g., OSI reference model<br>• An idealized application that implements much or most of the functionality of a model to provide proof of concept and suggestions to actual applications |
| RNIF | RosettaNet Implementation Framework |
| semantic relationship | A connection between two items based on meaning. |
| SI Units | A unit is a particular physical quantity, defined and adopted by convention, with which other particular quantities of the same kind are compared to express their value. This is based on the International System (SI) of Units which defines the modern metric system of measurement. |
| structural rules | Guidelines used for determining or defining how objects are put together |
| successor association | An association to a DictionaryItem that links to the DictionaryItem that is replaces the current DictionaryItem |
| super type | A grouping of objects within a hierarchical structure; a type that is more abstract than those types lower than it in the hierarchy |
| UML | Unified Modeling Language - a method for specifying, visualizing, and documenting the artifacts of an object-oriented system under development. |
| user-derived data type | User-derived datatypes are those that are defined in terms of other datatypes |
| XML Schema | An expression of shared vocabularies that allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents. |
| XSD | See XML Schema |

# 12. Appendix

## 12.1 Alternative designs

During the creation of the RDA models, two extreme alternatives have been considered:

- Using an explicit model
- Using a completely flexible model

### 12.1.1 Why not use an explicit model?

We could have created an explicit model of the information to be interchanged. That would result in a UML model with significantly more type information. For example, instead of Item we may have modeled Resistor, instead of Characteristics we would use explicitly modeled attributes on the resistors (e.g., resistance, size, etc.). This would suggest the use of explicit schemas for interchanging information between trading partners.

There are two main reasons for rejecting this approach. The first is its maintenance burden and the second is its lack of flexibility which would cause significant delays in implementing new business processes.

The maintenance issue comes from the number of models (probably in the thousands) to be standardized. The primary use of RDA (at least the first use) is to provide means for interchange of product information. It is hard, if not impossible, to predict what information structure is required to interchange products and the number of variations seems infinite.

The definition of the information structure would have to be agreed upon between trading partners. This requires a significant standardization effort leading to unacceptable delays in electronic communication between trading partners

### 12.1.2 Why not use a more flexible model?

We may have chosen to use a more flexible model, allowing the trading partners to define the information interchange structure they require. A trading partner could for instance select to use the XML Metadata Interchange (XMI) standard that allows for interchange of any object graph.

The argument against this approach has been primarily that we want to reduce the expense in building software for the trading partners.

The design philosophy used has been:

- We want to only allow flexibility where it is required
- As much as possible of the information structure should be static

## 12.2 Known limitations

### 12.2.1 No support for multiple inheritance for Dictionary content

During the development of the dictionary model, a discussion on support for multiple inheritance emerged. The model currently does not support multiple inheritance. This is a deliberate choice to reduce the risk of semantic confusion.

A single inheritance model has been selected where ExtensibleDictionaryItems can inherit its definition only from a single parent.

### 12.2.2 Restrictions not defined

The dictionary model has an incomplete model for restrictions. A requirement to support restricted inheritance has been raised. In particular, there has been an expressed requirement to limit ValueLists (i.e., define a ValueList as a subset of another ValueList).

The model does not discuss or provide a full model for restricted inheritance.

### 12.2.3 No explicit model for ValueSpace

The model does not explicitly model the language for definition of value spaces. We recommend the use of a subset of XSD in the first release. However, an additional document describing the guidelines for how to use the XSD value space is required.

### 12.2.4 No meta-metadata

The dictionary has a fixed model for definition of metadata semantics. A discussion of creating a reflective model for the dictionary content has been raised but not fully explored.

### 12.2.5 No explicit support for collection types

One may envision values on characteristics expressed as collections. The model does not support this.

### 12.2.6 Limited support for complex expressions

Only a small subset of mathematical formulas are supported in the value model. These represent the set of know expressions required. It is possible that as the model includes more business domains that more complex expressions will be required.

### 12.2.7 Characteristics reported in tree structures

The characteristics can currently only be reported in tree structures. This seems to cover all business scenarios raised during the requirements gathering. However, it is possible that some business domains would prefer to report characteristics using more sophisticated graph structures.

### 12.2.8 Limited support for assemblies

The current model has only limited support for assembly structures. The model allows for provision of item structures where an item may compose characteristics from other components. It is also possible to create interesting composite structures using the ItemRelation feature. However, there is no explicit model for assemblies or selection guides.

At one point a discussion of including feature models have been discussed but the effort has been postponed to future releases.

### 12.2.9 No support for hierarchical Modes

The EC extension defining Terminus, Modes and Connection allows for hierarchical structures for Termini and Connections, however Modes can not be hierarchical.

## 12.3 OIDDI

The Open Interoperable Domain Dictionary Initiative (OIDDI) is an initiative carried by a network of dictionary providers and has been set-up to facilitate the use of domain dictionaries in e-business and e-engineering. RosettaNet is one of the dictionary providers supporting the initiative and has joined OIDDI by means of a Memorandum of Understanding (MOU).

OIDDI is aiming to solve the problems caused by the existence and partially uncoordinated evolvement of domain dictionaries in various industrial sectors: As a result, industry has to support several dictionaries which increase their costs and efforts to participate in global e-business relationships.

The OIDDI has been launched to overcome this obstacle. Its goals are:

- to support the interoperability of product dictionaries
- to avoid uncontrolled overlap among different dictionaries
- to support the harmonization of existing dictionaries

The basic idea underlying OIDDI is to enable dictionary providers to reuse existing dictionary elements that have been defined by other dictionary providers. This can be done by reference mechanisms and requires that OIDDI members get the allowance to use dictionary elements of other OIDDI member's dictionaries.

According to this idea and the goals of OIDDI, the following requirements have been defined, and any OIDDI member will commit to ensure these rules.

1. Each dictionary provider in OIDDI shall specify its own identification using IS0 6523-1 (in ISO 13584-26:2000 syntax).
2. The identifier of each entry of a dictionary (product or service class, or property data element type) shall be a GUI defined according to ISO 13584-42/IEC 61360-2, i.e. it shall contain the identification of the dictionary provider plus a version number.
3. It shall be possible to describe the content of a dictionary according to the common ISO/IEC information model, also known as the PLIB model, specified in ISO 13584-25 and IEC 61360-5 in conformance class 1, 2, 3 or 4.
4. The content of each dictionary shall be published in a common electronic publishing format. In a first stage, this format will consist of ISO 10303-21 file compliant with the common ISO/IEC dictionary model published in ISO 13584-25:2004 / IEC 61360-5. If and when an XML schema will be agreed upon between all participants of the OIDDI initiative, and when tools for mapping this format back and forth onto ISO 10303-21 files will be available, publication will be done in this format.
5. OIDDI members (in the following called providers) grant other OIDDI members (users) the right to reference elements of their dictionaries and to include them into their user's dictionaries

## 12.4 References

The following external standards or specifications are referenced in this document.

**Table 310: External references**

| Reference | Title |
|---|---|
| [RFC2119] | Key words for use in RFCs to Indicate Requirement Levels |
| [XML] | Extensible Markup Language (XML) 1.0 (Third Edition) |
| [XSD] | XML Schema Part 2: Datatypes |
| [XHTML] | XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition) |
| [MATHML] | Mathematical Markup Language (MathML) Version 2.0 (Second Edition) |
| [RFC3066] | Tags for the Identification of Language |
| [ISO639] | Codes for the Representation of Names of Languages |
| [ISO3166] | ISO 3166: Codes for Country Names |
| [UNICODE] | UNICODE 4.0.0 |
| [IEC61360] | Standard data element types with associated classification scheme for electric components - IEC 61360-1 - Consol. Ed. 2.1 (incl. am1) - English [2004] |
| [OIDDI] | Please contact Guy Pierra [plib@ensma.fr] |
| [UML] | Unified Modeling Language |

# 13. Tables



continued >>

continued >>

continued >>

continued >>

## 14. Figures





continued >>

## 15. Contents



**continued >>**

continued >>

continued >>

**Document date: 21 April 2005 Document number: RDA CS - April, 2005**