**Date**: March 2016

WE SET THE STANDARD

# Date-Time Vocabulary (DTV)

*Version 1.3 RTF convenience document (clean) – March, 2016*

---


OMG Document Number: dtc/2016-02-20

Standard document URL: http://www.omg.org/spec/DTV/1.3/PDF

Associated Schema Files: http://www.omg.org/spec/DTV/20160301

dtc/2015-02-11 (Annex C Business Usage Guidelines)


---

## OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (http://www.omg.org/report_issue.htm).

# Preface

## About the Object Management Group

### OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

## OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Formal Specifications are available from this URL:

http://www.omg.org/spec

Specifications are organized by the following categories:

### Business Modeling Specifications

### Middleware Specifications

- **CORBA/IIOP**
- **Data Distribution Services**
- **Specialized CORBA**

**IDL/Language Mapping Specifications**

**Modeling and Metadata Specifications**

- **UML, MOF, CWM, XMI**
- **UML Profile**

**Modernization Specifications**

OMG Domain Specifications

**Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications**

- **CORBAServices**
- **CORBAFacilities**

**OMG Domain Specifications**

**CORBA Embedded Intelligence Specifications**

**CORBA Security Specifications**

**Signal and Image Processing Specifications**

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
109 Highland Avenue
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: pubs@omg.org

Certain OMG specifications are also available as ISO standards. Please consult http://www.iso.org

## Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to http://www.omg.org/report_issue.htm.

# Table of Contents

# List of Figures

# 1 Scope

Many SBVR rules involve common, generic, cross-domain concepts such as date and time. Characteristics of these concepts are frequent usage in everyday and business activities and wide usage across all business domains such as finance and manufacturing. These concepts exclude specialized needs such as sidereal time and real-time processing requirements. This document uses the term "foundation vocabularies" due to the foundational nature of these vocabularies for all these potential SBVR users.

SBVR tool vendors and users need standard vocabularies for such concepts to improve interoperability among vendors and to ensure that they share the same concepts in the same way. Vendors also need an agreed format for exchange of date and time literals when used in rules. The SBVR community in general needs such vocabularies as a foundation to avoid the startup cost of defining vocabularies for basic concepts, and as an example for interoperability testing among tools. The OMG wants SBVR to be successful, and sees value in lowering the "cost of entry" for potential SBVR users.

This document addresses two different, but complementary, aspects of time:

- **Type 1:** *Temporal noun concepts* (such as time coordinate, duration, calendar, etc.) that model attributes of SBVR noun concepts, and *temporal verb concepts* (such as time coordinate *is in the past*, time interval$_1$ *is before* time interval$_2$, time interval$_1$ *includes* time interval$_2$, etc.) that model relationships between temporal noun concepts. See Clauses 8 through 8.2.
- **Type 2:** Fact types that relate situation kinds and occurrences (such as a person being married to another person) to temporal concepts (e.g., to a time interval). See normative clause 16, as well as informative clauses 7.9 and 7.11, and informative Annex E.

These two aspects reflect the use/mention distinction well known from analytical philosophy: the first mentions temporal concepts, whereas the second uses temporal concepts in order to anchor situation kinds and occurrences in time.

The OMG's Model Driven Architecture (MDA) anticipates mappings between business-layer or Computation Independent Models (CIM) and implementation-layer Platform Independent (PIM) and Platform Specific (PSM) Models. To encourage such mappings, this document provides date and time models in UML (Unified Markup Language) plus OCL (Object Constraint Language), partially in CLIF (Common Logic Interchange Format), and partially in OWL (Web Ontology Language) modeled in ODM (Ontology Definition Metamodel). The UML, CLIF, and OWL/ODM date and time models are "equivalent" to the SBVR date and time vocabulary while being "true" to the spirit of their respective technologies.

# 2 Conformance

Conformance to this specification is defined with respect to three types of software:

1. Software that manages ontologies complies with this specification if and only if it can import the entire set of concepts defined by the Date-Time Vocabulary in at least one of the normative forms specified here.
2. Software that implements machine reasoning about time complies with this specification if and only if it interprets the entire set of concepts defined by the Date-Time Vocabulary according to the semantics defined here.
3. The compliance of software that interchanges documents containing date and time concepts is specified in Clause 18.

# 3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- Bureau International des Poids et Mesures (BIPM), *The International System of Units*, 8th edition, 2006.
- International Electrotechnical Commission (IEC) 60050-111, *Physics and Chemistry, Edition* 2.0, 1996-07
- International Standards Organization (ISO) 8601, Data elements and interchange formats - Information interchange - Representation of Dates and Times, Third edition. December 1, 2004.
- International Standards Organization/International Electrotechnical Commission (ISO/IEC), JCGM 200: 2008, *International Vocabulary for Metrology - Basic and General Concepts and Associated Terms* (VIM), 3rd edition
- International Standards Organization (ISO), ISO/IEC 24707, Information Technology - Common Logic (CL): a framework for a family of logic-based languages, first edition, 2007-10-01
- International Standards Organization (ISO), ISO/IEC 80000-3, *Quantities and units -- Part 3: Space and time*, 2006
- International Standards Organization (ISO) 18026. *Information technology - Spatial Reference Model (SRM)*, 2009
- Object Management Group (OMG), *Object Constraint Language*, version 2.0, May 2006
- Object Management Group (OMG), *Ontology Definition Metamodel*, version 1.0, May 2009
- Object Management Group (OMG), *Semantics of Business Vocabulary and Business Rules* (SBVR), v1.0, January 2008, OMG document formal/2008-01-02.
- Object Management Group (OMG), *Unified Modeling Language* (UML), v2.3, May 2010
- World Wide Web Consortium (W3C), *OWL 2 Web Ontology Language Document Overview*, 27 October 2009
- World Wide Web Consortium (W3C) Recommendation, *XML Schema Part 2: Datatypes Second Edition*, 28 October 2004

# 4 Terms and Definitions

Because the Date Time Vocabulary is intended to be a formal vocabulary, the content of this specification is primarily terms, definitions, and examples. Where terms are drawn from other sources, this is noted in the vocabulary entry by a Source caption.

The following terms are taken directly from SBVR and used only with the SBVR meaning, regardless of markup:

- designation
- individual concept
- noun concept
- *ranges over*, as 'role *ranges over* concept'
- verb concept
- verb concept role

Note: The unmarked term 'role' used in this specification means 'verb concept role'. The marked up term role refers to a property of something, which SBVR calls a 'situational role'.

The following additional terms are taken from SBVR and have the definitions and other descriptions given therein, when they are marked as SBVR terms. Note: The list below is ordered by the symbol being defined, while SBVR practice is to define verb symbols in the context of the subject term.

- cardinality *of* set and set *has* cardinality
- categorization type
- characteristic

- concept
- concept type
- meaning *corresponds to* thing
- definite description
- definition
- element *of* set and set *has* element statement *expresses* proposition
- expression
- extensional definition
- general concept
- set *includes* thing (= set *has* element)
- instance *of* concept and thing *is* instance *of* concept
- intensional definition
- $thing_1$ *is* $thing_2$
- thing *is in* set (= set *includes* thing)
- name *of* thing and thing *has* name
- proposition
- representation
- representation *has* expression
- representation *of* meaning and meaning *has* representation
- representation *represents* meaning (= meaning *has* representation)
- res
- roleset
- $concept_1$ *specializes* $concept_2$
- statement
- terminological dictionary
- thing
- unitary concept
- vocabulary

The following concepts have their usual mathematical meaning but are formally marked as the SBVR terms:

- integer
- nonnegative integer
- number

# 5 Symbols

This clause specifies the intended meaning of the symbols and other special text of this specification.

## 5.1 SBVR Vocabulary

Clauses 8 through 17 of this specification introduce the Date-Time Vocabulary as a 'vocabulary,' as defined by the OMG Semantics of Business Vocabulary and Rules specification.

This specification presents the Date-Time Vocabulary in the forms specified in Annex C of SBVR. The intent is that the Date-Time Vocabulary is to be interpreted as specified in SBVR Annex C.2 and C.3, and is to be rendered as an XML document that conforms to the SBVR Metamodel XML Schema that is described in SBVR sub clause 15.2, according to the patterns given in SBVR sub clause 13.6.

The following captions are used as specified by SBVR in formulating vocabularies and terminological entries. In some cases, the corresponding SBVR term is used (with markup, see clause 4) directly in DTV definitions and rules,

- Concept type
- General concept
- Definition
- Dictionary basis
- Example
- Included Vocabulary
- Language
- Namespace URI
- Necessity
- Note
- Possibility
- Source
- Synonym
- Synonymous Form
- Vocabulary

Annex A of this specification identifies the normative attachment that contains the formal representation of the Date-Time Vocabulary as an SBVR Vocabulary in the normative XML document form prescribed by SBVR sub clauses 13.6 and 15.2. The XML document includes all the meanings, definitions, rules, and other representations that are given in this specification in text form.

It is possible to represent most, but not all, of the definitions and rules given in this specification in the formal logical form specified by SBVR Clause 9. That representation may be a normative part of a future version of this specification.

## 5.2 SBVR Structured English

For definitions of vocabulary terms, and for 'structural rules' (necessities, axioms) that relate to those terms, this document adopts the "SBVR Structured English" syntax and font styles described in Annex C of the SBVR specification [SBVR]:

- Underlined teal indicates noun concepts.
- *Italic blue* identifies the fact symbols of verb concepts.
- Orange font indicates keywords.
- Double underlined teal marks individual concepts.
- Black normal font is regular text.

This specification uses the following symbols for the meanings indicated:

| | |
|---|---|
| ≤ | less than or equal |
| ≥ | greater than or equal |
| < | less |
| > | greater |
| = | equal |
| + | addition |
| - | subtraction |
| * | multiplication |
| / | division |

Ordinary arithmetic is meant when these symbols are used, unstyled, with numbers (e.g., "$number_1$ = $number_2$"). The meaning is explicitly defined in this specification when these symbols are applied (and styled as verb concepts) to other operand types.

Sets are formed using the BNF syntax '{' <element>+ (, <element>)* '}', where <element> gives the members of the set, separated by commas. An empty set is specified by "{}".

This specification uses the SBVR definition of 'thing1 *is* thing2,' meaning "The $thing_1$ and the $thing_2$ are the same thing." Verb concepts using the fact symbol '*equals*,' '*=*,' or '*is equivalent to*' are explicitly defined for usages where the intended meaning is that two values can be distinct things, but are equivalent in terms of their relationship to some other thing. In particular, two quantity values are different things if they involve different units but are *equal* or *equivalent* if they *quantify* the same quantity.

The SBVR specification does not discuss dates and times, and thus does not specify the styling of literal time coordinates (e.g., "January 21 2009"), literal times of day (e.g., "3:00 pm"), and literal duration values (e.g., 3 months 13 days). These values identify themselves, meaning that each such expression identifies exactly one time coordinate, time of day, or duration value – they are what SBVR calls 'individual concepts.' For this reason, literal time coordinates and times of day are styled as individual concepts in this document. For example, January 21 2009 3:00 pm.

In this specification, duration values provide the reference scheme for durations, and time coordinates provide the reference scheme for time points. Verb concept roles that apply to durations or time points can be filled by duration values or time coordinates, respectively. For example, "17:00 *is* 1 hour *before* the start *of* the meeting" applies the verb concept "$time\ interval_2$ *is* duration *before* $time\ interval_1$" using time coordinate "17:00" to fill the "$time\ interval_2$" role, and duration value "1 hour" to fill the "duration" role. The example assumes that "start *of* meeting" is a time interval that fills the "$time\ interval_1$" role.

This specification distinguishes between comparing durations or time periods, and quantifying time periods. Comparisons uses verb concepts defined in this document and styled as verb concepts. For example, "if the length of the meeting *is greater than* 3 hours …" or "if the date of the meeting *is before* the contract due date …" Quantifications use keyword style, as in "The party is on each July 4."

Definitions that are drawn from another specification are preceded by "Source" or "Dictionary Basis" captions. "Source" indicates that the definition is adopted exactly from the indicated specification. "Dictionary Basis" identifies definitions that are paraphrased from the specified source.

## 5.3 UML and OCL

This specification includes a normative UML (Unified Modeling Language) model of the concepts represented in the Date-Time Vocabulary, using the same terms as the SBVR vocabulary to the extent possible. The intent of the UML model is two-fold: (a) to provide a normative PIM (Platform Independent Model) UML representation of the concepts, for use in software models of date and time concepts, and (b) to illustrate the Date-Time Vocabulary with UML diagrams. Annex A of this specification identifies the normative attachment that is the UML model.

The UML model is derived manually from the Date-Time Vocabulary presented in the SBVR form. The UML model is constructed generally following the principles in [SBVR] Clause 13. The names in the UML model are identical to the primary vocabulary terms for the same concepts.

Some SBVR vocabulary items are modeled in the UML model using stereotypes. The stereotypes are formally specified in Annex I.

- Each SBVR general concept maps to a UML class.
- Each SBVR concept type maps to a UML class with the stereotype «concept type». Where specific concepts that are instances of a concept type are also modeled, the fact that each such a concept is an instance of the concept type is modeled by a UML dependency with the stereotype «instance of».
- Each SBVR categorization type maps to a UML class with the stereotype «categorization type». The relationship between the categorization type and the general concept it categorizes is modeled by a UML dependency with the

stereotype «for general concept».

- Each binary verb concept maps to a UML association. The association is named for the primary verb concept form for the verb concept, discarding all markup. The placeholders (role names) in the verb concept are mapped to the association end names, with subscripts being elevated to plain text.

- Each binary verb concept that uses the SBVR verb symbol *has* in any of its synonymous forms maps to a UML Property of the class that is the subject of the verb; that is, the association end is owned by the class. In some cases, this means that the association end name (the property name) is taken from the *has* form, rather than the primary form.

- Regardless of the verb symbol, where the intent of the binary verb concept is that the association represents a property of the class that plays the subject role, the corresponding association end is owned by the class. Similarly, where there is a Synonymous Form that represents a property of the other role (as the subject of that form), the corresponding association end of the same association is owned by the class that plays that role.

- Binary verb concepts that do not clearly imply a property of either participating class, such as 'time interval$_1$ *is before* time interval$_2$', are mapped to associations in which both association ends are owned by the association.

- Verb concepts with more than two roles map to UML classes stereotyped as «verb concept». The roles in these verb concepts are modeled by UML associations from the «verb concept» class to the UML classes that model the ranges of the roles. These associations are stereotyped «verb concept role» and are properties of the «verb concept» class. These properties always have multiplicity '1', because each instance of the class represents a single instance of the relationship, having exactly one participant in each role. The multiplicity of the association-owned end of a «verb concept role» association represents the number of situations in which a given object in the range class can play that role.

- In general, the operation is named for the primary verb concept wording, and is attached to the class that is the range of the subject role in that wording. The operation takes one argument for each other role in the verb concept wording and returns a Boolean result. The Boolean result indicates whether the subject instance ("self"), together with a given set of argument values as participants in the corresponding association roles, represents an actual instance of the association. In addition, in those cases where it is convenient for stating rules, a synonymous form of the verb concept is used to create an operation on the class that is the subject of that form. That operation is named for the synonymous form, and its arguments correspond to the remaining roles in the synonymous form. It returns Boolean with the same interpretation..

- Some verb concepts that have more than two roles also map to a UML operation that returns the unique object that plays one of the roles, as a function of the objects that play the other roles. The operation is on the class that is the range of the subject role in one of the verb concept wordings, and that is one of the inputs to the function. The operation has one argument for each of the other roles that serves as an input to the function, and it returns the unique object that plays the remaining ("result") role in the corresponding state of affairs. For example, the verb concept 'duration$_3$ *=* duration$_1$ *plus* duration$_2$' has the synonymous form ' 'duration$_1$ *plus* duration$_2$' *gives* duration$_3$'. This latter form is mapped to an operation on class 'duration' – plus(duration2: duration): duration – which returns the value of 'duration3'.

- All formal SBVR definitions and rules (Necessities) in Clauses 8 and 16 are also formally specified as OCL definitions and constraints. The "noun forms", if any, of the verb concepts in those sections are mapped to UML Properties or Operations, and those Properties and Operations have formal definitions in OCL.

- Definitions, notes, and examples that are attached to entries in the Date-Time Vocabulary are intentionally omitted from the UML model to avoid the requirement to maintain consistency between the specification text and ownedComments in the model.

- Because UML does not support the concept of Synonym (for a noun concept) or Synonymous Form (for a verb concept), the UML model does not include any formal model elements for those elements of the vocabulary.

For the definitions and rules in the Date-Time Vocabulary, this specification adds Object Constraint Language (OCL) rules to the UML model, to the extent possible. (The definitions of primitive concepts, and some rules, cannot be formally stated in terms of classes and associations in the model.)

OCL constraints are incorporated into the document text and the UML model as follows:

- Each fully-formal SBVR definition has an equivalent OCL definition or constraint, captioned as "OCL Definition:". The constraint captures the distinguishing characteristics of the formal definition. For example, if the formal definition of an SBVR object type '*luxury car*' is '*car* that *is gold*,' the corresponding OCL constraint is given as:

OCL Constraint:
```
context 'luxury car'
   inv:self._'is gold'
```

- Each SBVR Necessity (that is not a cardinality constraint) has an equivalent OCL constraint, captioned as "OCL Constraint"
- Necessities and Possibilities that specify cardinalities are modeled as UML cardinalities, rather than OCL constraints.
- OCL name-quoting syntax is applied as necessary to quote UML names with embedded spaces. For example the term 'consecutive sequence' is quoted in OCL as "_'consecutive sequence'".

OCL is provided for sub clauses 8.1, 8.2, and Annex D. These parts of the specification require the most rigorous definition.

## 5.4 CLIF Axioms

This specification includes a file of matching and normative Common Logic Interchange Format (CLIF) axioms that is inventoried in Annex A. The axioms are provided to precisely specify the formal Definitions and Necessities of this specification in a form that is meaningful to logicians and that can be input (in the future) to software that automatically checks for consistency among the axioms. The CLIF axioms in this document have been syntactically checked using the Kojeware CLIF validation service that is available at http://www.kojeware.com/clif-file-validator. No automated quality analysis has yet been performed.

The CLIF axioms are derived manually from the SBVR-based text in this document. In case of any discrepancies between the SBVR-based text in this document and these axioms, the text prevails because it is the original model.

Names in the CLIF axioms are based directly on the corresponding SBVR names, using CLIF name-quoting as necessary to address embedded spaces. For example the SBVR term 'consecutive sequence' is quoted in CLIF as "consecutive sequence."

The file of CLIF axioms is derived automatically from CLIF statements that are incorporated directly in the text of this specification as follows:

- Each fully-formal SBVR definition has an equivalent CLIF axiom, captioned as "CLIF Definition:". The axiom defines how the corresponding concept is derived from some other concept. For example, if the formal definition of an SBVR object type '*luxury car*' is '*car* that *is gold*,' the corresponding CLIF axiom is given as shown below. Read this as "each car is a luxury car if and only if the car is gold."

CLIF Definition:
```
(forall ((car car))
 (iff ("luxury car" car)
     ("is gold" car))
```

- Each SBVR Necessity has an equivalent CLIF axiom, captioned as "CLIF Axiom:". The axiom expresses the same constraint as the SBVR Necessity.

Many SBVR Necessities specify cardinality constraints. Basic CLIF cannot express these constraints in the absence of functions that generate collections, give the cardinality of collections, and compare the values of integers. Therefore this specification assumes the following in order to express cardinality constraints in CLIF:

- For each SBVR verb concept, there is a corresponding CLIF predicate, and also *n-1* CLIF functions, where *n* is the number of roles of the verb concept. The predicate and all the functions have the name of the verb concept, quoted if necessary. The distinction among them is the number of terms they take and which terms they take. The predicate takes one term for each role of the verb concept, and returns true or false according to whether the verb concept is satisfied for the specific terms. Each function omits one role and produces a collection of instances that fulfill that role in relationship to the other terms of the function.

For example, given an SBVR verb concept '<u>driver</u> *drives* <u>car</u> *to* <u>city</u>,' the predicate ("`driver drives car to city`" `John` "`car 123`" `Paris`) is true or false according to whether John drives car 123 to Paris. The function ("`driver drives car to city`" `John Paris`) returns the collection of cars that John drives to Paris.

- A primitive `count` function that returns the cardinality of a collection. For example, (`count` ("`driver drives car to city`" `John Paris`)) produces the number of cars that John drives to Paris.
- CLIF defines the = predicate as testing whether two terms are equal. This specification uses primitive functions <, <=, >, >=, and + to mean the standard numeric relationships. For example `(< (count ("driver drives car to city" John Paris)) 2)` tests whether John drives fewer than two cars to Paris.
- This document also uses the allDifferent function as defined in [IKL Guide].

CLIF is provided for sub clauses 8.1, 8.2, and Annex D. These parts of the specification require the most rigorous definition.

## 5.5 OWL Formulation

In addition to the normative SBVR, UML/OCL and CLIF specifications of the Date Time concepts, an informative model of the same concepts expressed in the Web Ontology Language (OWL) is provided. The OWL model - a set of OWL "ontologies" - was developed by a rote transformation from the Date Time vocabulary entries. The transformation converts the primary SBVR terms to OWL classes, properties, and individuals, and it converts each other element of an SBVR terminological entry to a specialized OWL annotation.

Each SBVR vocabulary presented in Clauses 8 through 17, and each supporting vocabulary presented in Annex D, was transformed to a separate OWL ontology in this way. The OWL ontologies are not presented in the specification per se. They are provided as an informative attachment to this specification in the standard OWL/RDF exchange form.

# 6 Additional Information

## 6.1 How to Read this Specification

This document serves different purposes for first-time readers versus implementers. First-time readers should start with informative Clause 7, "Rationale" that offers introductory text, and describes the motivations behind the design of this vocabulary. These readers may wish to refer to the normative clauses (Clause 8 through Clause 13), as well as informative Annex D, for definitions, notes, examples, and diagrams that describe the Date-Time Vocabulary concepts. The other Annexes provide additional examples and supporting information that should also be useful to these readers.

Implementers of this vocabulary will focus on the normative clauses and Annex D and on the supporting machine-readable files. The specific aspects of interest will depend upon the intended conformance goal, as described in Clause 2. Implementers should study the material in the normative clauses in detail. The supporting informative material will also provide some guidance.

## 6.2 About this Specification

The first 6 clauses include information that is applicable to most OMG specifications. The rest of the document includes the following key topics:

Clause 7 - Rationale (informative) - introduces this document and discusses some of the key technical choices made by this specification.

Clause 8 - Time Infrastructure (normative) - describes fundamental concepts about time intervals, durations, and their relationships.

Clause 9 - Duration Values (normative) - Duration values are amounts of time stated as multiples of time units, for example "5 hours 30 minutes". The model of duration values presented here accommodates the complexities introduced by the varying number of calendar days in each calendar month and calendar year.

Clause 10 - Calendars (normative) - defines the basic concepts used to organize time as time scales and calendars, and to identify locations in time via time coordinates, such as "July 31".

Clause 11 - Gregorian Calendar (normative) - defines the standard Gregorian calendar, and the time points, time scales, and time coordinates of this calendar.

Clause 12 - ISO Week Calendar (normative) - defines the standard calendar based on weeks, and the time points, time scales, and time coordinates of this calendar.

Clause 13 - Time of Day (normative) - specifies the time points, time scales, and time coordinates that jointly identify the time periods within a day.

Clause 14 - Internet Time (normative) - specifies the calendar used by the Network Time Protocol.

Clause 15 - Indexical Time (normative) - Indexical time concepts use terms such as "in the past" and "now" to refer to time. These terms are defined in this specification, despite their inherent ambiguity, because they are frequently used in everyday communication.

Clause 16 - Situations (normative) - provides concepts that relate situations to time.

Clause 17 - Schedules (normative) - defines time tables, and schedules of events that may repeat over time.

Clause 18 - Interchange of Duration Values and Time Coordinates (normative) - defines how duration values and time coordinates should be exchanged between tools that implement this specification. The interchange format is based on the existing [XML Schema] and [ISO 8601] specifications.

Annexes

Annex A: Attachments (normative) - Lists the machine-readable files that accompany this specification.

Annex B: References (informative) - this annex lists the standards documents and academic papers that were consulted in the preparation of this specification.

Annex C: Business Usage Guidelines (informative) – is published as a separate document for the convenience of business users who need not read the normative specification. This annex offers counsel on the use of DTV by a discussion of and examples of "calendar expressions", and an inventory of the Date-Time noun and verb concepts recommended for business use.

Annex D: Fundamental Concepts (normative) - International standards, for example [VIM], [ISO 80000:3], and [ISO 18026] define duration as just one of many quantity kinds, and time scales as one of many kinds of coordinate systems. This permits the formation of derived quantities based on durations (e.g., velocity, which is length / duration), and multi-dimensional coordinate systems that include time as one dimension. Coordinate systems themselves depend upon mathematical concepts, such as sequences and scales. Unfortunately, there is no existing SBVR vocabulary or ODM ontology that addresses these concepts. The authors recognize that they are out-of-scope for this specification, but felt it necessary to imagine how this Date-Time Vocabulary would fit into a complete schema that addresses them. Annex D summarizes that schema in the form of several SBVR vocabularies.

Annex D.2: Sequences (normative) - presents a complete model of sequences that provides the formal foundation for time scales.

Annex D.3 Quantities Vocabulary (informative) - defines a minimal vocabulary for quantities and units of measure. This vocabulary is informative because it does not address requirements beyond those of this Date-Time Vocabulary.

Annex D.4: Mereology (normative) specifies a basic model of mereology that provides the formal basis for the part-of relationship among time intervals.

Annex E: Formalizing English Tense and Aspect (informative) - The normative clauses of this specification deal with the semantics of time as used in natural languages. This Annex describes how propositions that are given in English language syntax may be formulated using the Date-Time Vocabulary.

Annex F: This annex formally lists the vocabularies provided by the Date-Time Vocabulary specification.

Annex G: UML Profile for the SBVR Elements used in the Date-Time Vocabulary (normative) - documents the stereotypes used in the UML model of this vocabulary.

Index of Date Time Designations (informative) - contains an index to the business designations defined in this document.

## 6.3 Structure of this Specification

Figure 6.1 summarizes the structure of the SBVR vocabularies and UML packages that are defined in this specification.

The SBVR-DTV package contains the concepts from the SBVR specification that are used in this specification. The corresponding excerpts from the SBVR vocabularies are specified in Clause 4. The SBVR Profile defines UML stereotypes for some of these SBVR concepts. These stereotypes are used to mark up UML representations of some DTV concepts as described in Annex I. The «apply» relationship provides the Profile as the interpretation of those markups in the SBVR-DTV package, and in every UML package that directly or indirectly imports the SBVR-DTV package.

The content of each remaining element of the figure is a vocabulary and a UML package that corresponds to a top-level clause of this specification, or to a sub-clause of Annex D. The dependency relationships shown in the figure match the dependency relationships among the corresponding specification clauses.

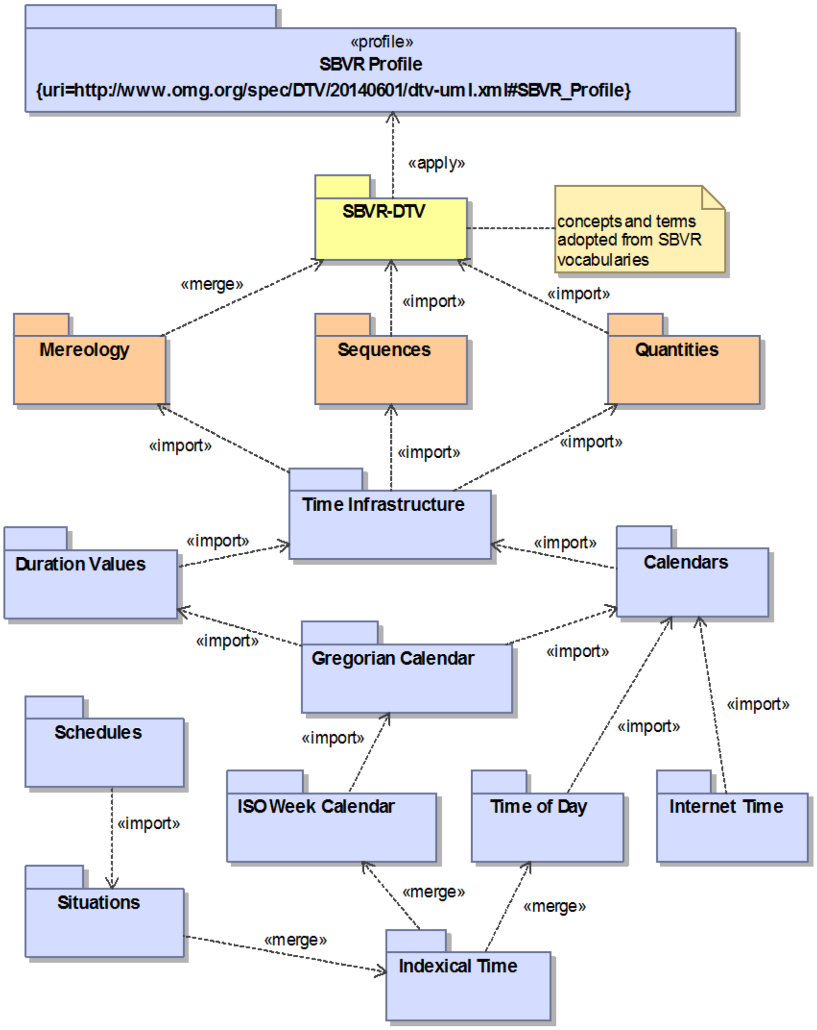


**Figure 6.1 - SBVR Vocabulary and UML Package Structure**

The «import» relationship shown in Figure 6.1 indicates both SBVR '$\text{vocabulary}_1$ *incorporates* $\text{vocabulary}_2$' (as indicated by the 'Included Vocabulary' caption) and UML package import. For both SBVR and UML, the entire contents of the imported package are incorporated into the importing package. For example, the Duration Values vocabulary incorporates the Time Infrastructure vocabulary, and the corresponding Duration Values UML package imports the Time Infrastructure UML package.

Vocabulary incorporation and UML package import are both transitive. For example, the Calendars vocabulary and package indirectly import the Sequences vocabulary and package.

The «merge» relationship used in Figure 6.1 is UML "package merge." This means that the entire contents of the merged package are incorporated into the merging package and some elements of the merged package are modified by the merging package. For example, the Indexical Time package merges the Calendars package, and thus the Time Infrastructure package, because Indexical Time adds UML attributes (such as the 'is current' attribute that represents the concept 'time interval *is current*') to the 'time interval' class defined in the Time Infrastructure package.

SBVR does not distinguish vocabulary incorporation from 'vocabulary merge,' because what is added is just additional vocabulary or additional constraints. So the 'merge' relationships shown in Figure 6.1 are accomplished by vocabulary incorporation in the SBVR text in this specification.

## 6.4 Acknowledgments

The following companies submitted and/or supported parts of this specification:

- Automata, Inc. - Paul Haley
- Business Rule Solutions, LLC - Ron Ross
- Business Semantics, Ltd - Donald Chapin
- Deere & Co - Roger Burkhart
- Hendryx & Associations - Stan Hendryx
- International Business Machines - Mark H. Linehan (team lead)
- KnowGravity Inc - Markus Schacher
- LogicBlox - Terry Halpin
- Microsoft - Don Baisley
- Model Driven Solutions - Cory Casanave
- Model Systems - John Hall
- National Institute of Standards and Technology - Ed Barkmeyer
- PNA Group - Sjir Nijssen
- Ravi Sharma
- Thematics Partners - Elisa Kendall

# 7 Rationale

## 7.1 General

This Informative clause introduces this document, and discusses various design considerations that impacted it.

## 7.2 Multiple Goals

This vocabulary attempts to satisfy several goals that tend to conflict.

- Provide a Standard Business Vocabulary for Date and Time Concepts - Provide a vocabulary of date and time concepts that business users can share and exploit in their business domain vocabularies and rules. Quoting Donald Chapin, this requires an "… SBVR Foundation Business Terminology that is conceptualized optimally for the way people think and communicate about things in their organizations using natural language." To satisfy this goal, the date and time vocabulary needs to include terms that make intuitive sense to business users.
- Support Machine Reasoning about Time - Provide a formal ontology that enables machine interpretation and reasoning. This means that processing by automated reasoners is possible, based on a well-grounded formal

representation. For example, it should be possible for a reasoning system to determine whether a payment is more than 30 days late compared to some due date. Satisfying this goal requires carefully-defined vocabulary concepts, to the point of making distinctions that would not occur to business users. The business vocabulary is grounded on the formal ontology, so these distinctions show through in the business vocabulary.

- Enable implementation - Enable tool vendors and other software developers to implement the date and time vocabulary with a "reasonable" amount of development effort – meaning that the value obtained is commensurate with the development cost. That cost is driven by the size of the vocabulary – the more there is to implement, the greater the cost. Implementation cost is also driven by the effort required to resolve ambiguities, omissions, and inconsistencies in the specification. Including a formal grounding and concise vocabulary is expected to facilitate both development of tools and use of the specification by vendors, business users, and those who want to apply formal reasoning systems.

This specification employs several techniques to reconcile these different modeling goals. The vocabulary is presented as an SBVR business vocabulary, with extensive examples and notes. Many formally-defined concepts are also presented in CLIF and OCL. Wherever possible, terms and examples are chosen to make sense to business users. Parallel construction of terms ensures that related terms are used consistently. Every concept is precisely defined. Multiple distinct concepts are defined where needed to distinguish between concepts that are intuitively similar but have different reasoning implications.

Annex D, "Foundational Concepts" documents general concepts that, though out-of-scope for a date and time vocabulary, nevertheless must be implemented consistently by reasoning systems. Annex D includes formal mathematical definitions of sequences, on which all scales, not just time scales, are based, and a general treatment of quantities and units, and of basic mereology. Although Annex D is not normative, it will provide guidance that should ease formal integration of future possible normative specifications, perhaps published by the OMG or other standards bodies, of the Annex D concepts with the normative vocabulary of this specification. Implementers of this specification are encouraged to support or assure compatibility with Annex D. Normative concepts of this specification that specialize Annex D concepts formally includes Annex D concepts in their definitions, as if Annex D were normative.

Implementors and reasoning systems are also addressed by providing this date and time vocabulary in SBVR, UML, and CLIF forms.

## 7.3 Reckoning of Time

The scientific community, and some time standards such as OWL-Time, typically conceive of time as continuous, meaning that any moment of the Time Axis can be subdivided into an infinite number of smaller moments. This Date and Time Vocabulary follows that pattern by modeling time as a segment of the Time Axis called a time interval, and describing amounts of time as durations.

Mathematically, both time intervals and durations correspond to contiguous sets of real numbers, making modeling of time-varying phenomena amenable to continuous mathematics. This specification gives a rigorous account of the operations that may be performed on time intervals and durations, providing the basis for formal reasoning about time.

Since antiquity, the passage of time has been reckoned by counting discrete time intervals demarcated by the diurnal and annual cycles of the Earth and the Moon's cycle – giving rise to 'time point' concepts such as 'calendar day', 'calendar month', and 'calendar year'. To identify a particular element of a cycle, each cycle is mapped onto a 'calendar'.

Calendars define time scales used refer to time points by name or by scale index. The combination of a time scale and an index or a name (e.g., 'February') is called a 'time coordinate'. An individual time coordinate is called an 'atomic time coordinate', whereas combinations of time coordinates (e.g., "February 3") are called 'compound time coordinates' (sub clauses 7.5 and 10.6.3). Time coordinates provide a reference scheme for time points via the verb concept 'time coordinate *indicates* time point'. Thus time points can be referred to either by definition descriptions (e.g, "the day after the meeting") or by time coordinates (e.g., "3:00 p.m.").

Each time point is a concept whose instances are time intervals. Thus, every 'time interval' fact type role in this specification can be filled by a time coordinate that *indicates* a time point. For example, the statement "the meeting time *is before* 3:00 p.m." uses the "time interval$_1$ *is before* time interval$_2$" verb concept (sub clause 8.2.2) to compare one time interval given as a definite description with another time interval given as a time coordinate.

Many calendars have been devised, ancient and modern. Time coordinates of most calendars can be correlated to jointly reference the same time interval. Calendars are anchored to the Time Axis by associating a noteworthy event with a particular time point on the calendar, e.g., the signing of the Convention du Mètre in Paris on May 20, 1875, which established the International Bureau of Weights and Measures (BIPM), and is the anchoring event for the modern Gregorian Calendar.

Timekeeping is significantly complicated by the incommensurable and irregular periods of rotation and revolution of the Earth and Moon. These variations are accounted for at the granularity of 'day' by incorporating intercalary leap days in the Gregorian Calendar, and at the granularity of 'second' by incorporating intercalary leap seconds in UTC. Businesses sensitive to elapsed 'seconds' should use TAI, while those that are concerned with calendar alignment may prefer UTC.

Time is measured by clocks, or tracked by calendars, in discrete time intervals called 'time periods', which *instantiate* time point sequences, as discussed in the next subclause. A particular member of a time scale – and a time period that *instantiates* a time point sequence of just one member – is called a 'time point'. Every time scale divides the Time Axis into time points with a specified duration, called the 'granularity' of the time scale. One consequence of this model is that every time period is aligned to the time points of a time scale: the time period *starts on* the first time point of some time point sequence of the time scale, and the time period *ends on* the last time point of some time point sequence of the time scale. Another consequence is that the duration of every time period is a multiple of the granularity of the time scale.

Of course, any time point can be subdivided by another time scale with a finer granularity. For example, a time point with duration "1 second" can be divided into milliseconds. But subdivision in this sense is still a discrete process. The finer time scale has a finite number of time points for each time point on the original time scale.

In everyday activity, people and businesses talk about durations such as years and hours, and about time periods such as calendar years, hours of day, and so forth. These discrete time concepts are used in ordinary conversation, in business contracts, in legislation and regulations, and in corporate policies. They also form the basis for identifying time intervals for scientific purposes (International Atomic Time) and for navigation (Global Positioning System). Representation of time in computers is inherently discrete and finite. Consequently, this specification also defines discrete time modeled by time scales.

## 7.4 Time Scales

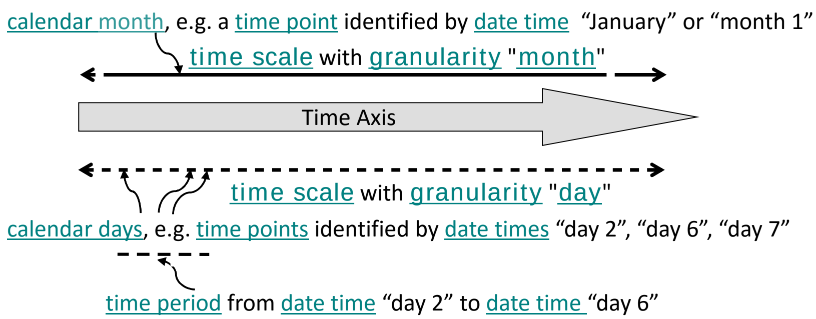


**Figure 7.1 - The Time Axis and Time Scales**

Following [ISO 8601], this specification considers that there is a single Time Axis that is measured by multiple time scales. The Time Axis represents "the succession in time of instantaneous events". Figure 7.1 shows the Time Axis with one time scale for calendar months, and another for calendar days.

Each time scale comprises a consecutive sequence *of* time points at regular or irregular time intervals. The time points of each time scale have a duration that is called the granularity of the time scale. Month scales have irregular time intervals because different calendar months have different durations. Thus, the Time Axis is continuous time, while time scales partition the Time Axis into discrete segments. Time scales define concepts that are meaningful in business and everyday life.

Time coordinates label individual time points on a time scale. For example, the top time scale in Figure 7.1 has a calendar month labeled "January", while "day 2", "day 6", and "day 7" are indicated on the time scale for calendar days. A time coordinate can have multiple labels. For example, "January" is also labeled "month 1".

A time period *instantiates* a time point sequence, a sequence of consecutive time points on a time scale. "Instantiation" means that the time point sequence *corresponds to* the time period, analogous to SBVR's "meaning *corresponds to* thing". Each time point sequence has a first time point, a last time point (the final time point of the time point sequence), and a duration (the length of the time period). For example, the time point sequence from "day 2" to "day 6" has a first time point of "day 2", a last time point of "day 6", and a duration of "5 days".

Conventionally, and by international agreement, on some time scales (hours, minutes) the first time point is designated "hour 0" or "minute 0", while on others (months, weeks, days) the first time point is designated "month 1", "week 1", or "day 1". Historically and in [XML Schema], calendar years are numbered from 1 but scientific practice and [ISO 8601] counts a year 0.

Conversion between time scales is possible via formulae that specify how a time point on a coarser time scale indicates the same time interval as a time period on a finer time scale.

## 7.5 Distinctions

The distinction among time coordinate and duration values is significant. A time coordinate gives a location on a time scale. A duration value specifies an amount of time. For example, a meeting might occur at "3:00 p.m." (a time coordinate) for "3 hours" (a duration value). This distinction leads to separate terms for concepts such as "day" (a time unit used with duration values) and "calendar day" (a time point indicated by a time coordinate).

There is a many-to-one relationship between time coordinates and time points. For example, "January 2009" and "month 1 of 2009" are two time coordinates for the same time point. In SBVR terms, time coordinates provide the reference scheme for time points. In human language, a thing and a reference to the thing are often not distinguished, but the difference is important in ontological reasoning.

Similarly, there is a many-to-one relationship between duration values and durations. "1 hour" and "60 minutes" are two duration values for the same duration. Again, the distinction is significant ontologically but often blurred in human discourse.

## 7.6 Compound Time Coordinates

Compound time coordinates are time coordinates composed from multiple time scales. Compound time coordinates are used to designate a time interval whose duration is much less than the span of a time scale. For example, to identify a particular calendar day on a time scale that spans millennia, the compound designation "3 January, 2010" is used, rather than something like "day 733 795". Compound time coordinates originated historically as counts of the apparent cycles of the Sun, the Moon, and the stars.

Around the globe, different cultures express compound time coordinates in different ways. For example, "January 3, 2010", "3 January 2010", "2010-01-03", "1/3/10", "3/1/10" represent the same date in different parts of the world. Similarly, the same time may be expressed as "6:00 p.m." or "18:00". For example purposes only, this document gives dates and times in various formats. However, this specification does NOT standardize any particular way of expressing dates and times. (See [ISO 8601] for such a standard.) Instead, this specification focuses on formally capturing the meaning of compound time coordinates that may be expressed in various date and time formats and in different languages.

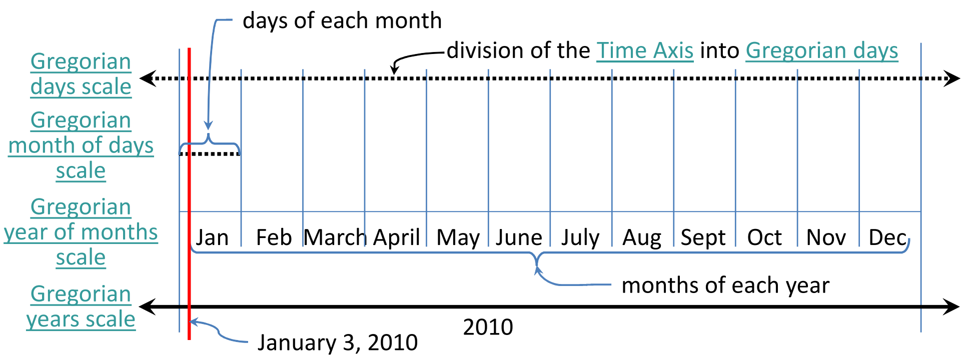


**Figure 7.2 - Example of Gregorian calendar**

This specification models "2010" as a time coordinate (a date time, or timestamp) on the Gregorian years scale. It models "3 January 2010" as a compound time coordinate that references multiple time scales of the Gregorian calendar. The compound time coordinate specifies time points on the Gregorian years scale, the Gregorian year of months scale, and the Gregorian month-of-days scale. Put together, these time points on these time scales *indicate* (by definition of 'compound time point') a particular time point on the Gregorian days scale.

Notionally, the Gregorian days scale is a time scale of granularity 'day' that extends indefinitely into the past and the future. "3 January 2010" can be understood as the time interval *indicated by* a particular time point on the Gregorian days scale. Sub clause 10.6.2 gives details about this. Sub clause 10.8 gives details about conversions between time scales.

Not all time scales can be combined in compound time coordinates. For example, "day 33 second 45" makes no sense. Clauses 11 through 13 details both the time scale combinations that form legitimate compound time coordinates, and their meaning in terms of atomic time coordinates. For example, "01:35" is a compound time coordinate (using the day of hours scale and the hour of minutes scale) that means minute of day 95 on the day of minutes scale.

The meaning of some compound time coordinates as time intervals depends upon the presence or absence of leap days. For example, the relationship of March, April, etc. on the Gregorian year of months scale to the Gregorian days scale depends upon the number of days in February. In leap years, there is an additional day in February that "bumps" March over by one day on the Gregorian days scale. Hence, a compound time coordinate such as "3 March" does not mean a single Gregorian day on the Gregorian days scale if the calendar year is not given. Instead, such a date is understood as a choice among two possible Gregorian days. The choice is called a 'time set' and denoted (in this example) as "{Gregorian day 62, Gregorian day 63}".

## 7.7 Compound Duration Values

Compound duration values are duration values composed from multiple time units. Examples are "3 weeks 4 days", and "1 hour 30 minutes". The meaning of these is durations using the smallest time unit of the compound duration values. For example, "3 weeks 4 days" means "25 days", and "1 hour 30 minutes" means "90 minutes".

Some compound duration values that use nominal time units are ambiguous. For example, "5 months 3 days" is ambiguous because the number of Gregorian days in a Gregorian month of year varies. Similarly, the number of Gregorian days in a Gregorian year varies according to whether the Gregorian year is a leap year. The concept 'duration value set' models the ambiguity. For example, "2 years 1 day" means the duration value set {730 days, 731 days}.

## 7.8 Granularity of Time Coordinates and Time Points

The granularity of a time coordinate is understood as the finest granularity of the components of the time coordinate. For example, the granularity of "3 January 2010" is 'day'. This is important when understanding the meaning of a phrase such as "the meeting happens on 3 January 2010". The phrase means that the meeting happens sometime during that calendar day, but does not say whether it happened at noon or 18:00 or throughout the entire calendar day because the granularity means the whole day. A phrase such as "the meeting happens at 18:00 3 January 2010" is more specific because it uses a compound time coordinate with granularity 'hour'. It means that the meeting happens sometime within the hour indicated by "18:00". To specify the time more precisely, add minutes or seconds or even fractional seconds to the compound time coordinate to achieve the desired temporal resolution. The granularity chosen in giving a time coordinate should be as specific as required for any particular use case.

Similarly, the time unit of a compound duration value is the least time unit of the individual atomic duration values that makeup the whole duration value. For example, "6 hours 00 minutes" has a time unit of "minute", while "6 hours" has a time unit of "hour".

## 7.9 Time Point Relationships

This specification provides relationships among time points and durations that permits comparing, adding, and subtracting them in various combinations. These are described in Clause 8 in terms of fundamental relationships (e.g., the mereological aspects of time intervals, the *is before* relationship between time intervals, the Allen relations), and various derived relationships.

Some duration value relationships, when applied to operands that have nominal time units, may have no meaning. For example, it makes sense to compare two duration values that are in months with each other (e.g., "5 months *is greater than* 3 months"), but comparing some duration values in months to some duration values in days (e.g., "2 months is less than 59 days") may be meaningless since months have varying numbers of days. Whether a relationship has meaning may depend upon both the values and time units of the relationship operands. For example, "10 days *is less than* 1 month" is always true, even though individual Gregorian months may be 28, 29, 30, or 31 Gregorian days. Clause 15 addresses these issues.

Similarly, time relationships may be ambiguous when applied to time coordinates or time points. For example, the time interval from 8 January through 13 March (given without the Gregorian year) has one of two durations, the duration value set {65 days, 66 days}. Clause 16 discusses these complexities.

## 7.10 Temporal Reasoning

A major goal of the Date-Time vocabulary is to enable reasoning about time in fact models. Such reasoning presupposes that the temporal aspects of each sentence are described in the logical formulation of the sentence. This sub clause provides a summary of issues involved and describes how this specification supports temporal reasoning. A more thorough treatment is provided in sub clause 16.5.

Fundamentally, time is associated with events and with the lifecycle of things. This specification uses the term "situation" to refer to events, activities, states, etc. Linguists often categorize situations in various ways, for example as "events," "situations," "actions," and so forth. This specification chooses not to categorize situations, but instead to focus on various relationships between situations and time.

Situations are said to *occur*, which is a primitive notion. Some situations that are conceptualized never occur. This specification uses the term 'occurrence' for a situation that occurs at some time in the world that is taken to be actual. When one is making a decision in the real world, what is taken to be actual is what the decision maker knows or believes about the real world. When one is analyzing a what-if situation (as in a business plan), the hypothetical elements of that situation are taken to be 'actual.'

When something occurs, there is always a time associated with the occurrence. The time may be present, past, or future, relative to the decisions being made. This permits distinctions among different instances of some situations that recur. For example, "Oceanic Air flight 815 flies from NY to Los Angeles" may be a situation that occurs many times and for which the individual occurrences may be distinguished by time. However, many types of occurrences are not distinguishable by time. For example, multiple child births often happen at the same time, so are not distinguishable purely by time.

The basic element of time introduced in Date-Time is a time interval, a portion of time having a non-zero duration. One basic fact type relates occurrence to time interval: 'occurrence *occurs throughout* time interval'. It represents the idea that the occurrence is ongoing at every point in the time interval. From it, we derive the characterizing relationship 'occurrence *occurs for* time interval' (sub clause 16.2). This fact type represents the idea that the occurrence starts at the beginning of the time interval and ends at the end of that time interval. For any occurrence, there is exactly one such time interval, called the occurrence interval.

A situation kind is a potential situation that could occur in some possible world. In a given world of interest (the world taken to be actual), each situation kind has zero, one, or more occurrences. We say that an occurrence *exemplifies* a situation kind. The situation kind itself is said to *occur for* each time interval that is the occurrence interval of an occurrence of the situation kind. Other verbs that relate occurrences to time intervals are used to relate situation kinds to time intervals by extension. The critical difference is that an occurrence is a single actual situation and *occurs for* exactly one time interval; a situation kind is an abstraction of zero or more occurrences and may *occur for* zero or more time intervals, one for each distinguished occurrence.

Occurrences are partially ordered by the times of their occurrence – their occurrence intervals. This specification provides the basic vocabulary to describe the ordering of occurrences in sub clause 16.3. Ordering of occurrences allows some statements to be made about the ordering of situation kinds, and those verbs are defined in sub clause 16.5.

This document uses 'proposition' to mean the logical interpretation of a sentence. Each proposition (that is not paradoxical) *corresponds to* exactly one situation kind. This viewpoint was famously championed by Donald Davidson, that a proposition is a definite description of a situation ([Davidson], p. 504). This specification adopts this viewpoint. A proposition is either true or false in a given world. A situation kind either *has* or does not *have* occurrences in the universe of discourse. There is a duality in that a proposition may simultaneously have a truth value and correspond to a situation kind. A proposition is true when it *corresponds to* a situation kind that *has* at least one current occurrence.

Since a proposition describes exactly one situation kind, it is said to *describe* every occurrence of that situation kind as well. In many cases, this is the critical fact type: proposition *describes* occurrence. For example, "the books of corporation XYZ are reviewed annually at corporate headquarters" can be formally represented as:

> In every fiscal year (a business-defined time period), there is an occurrence that is described by the proposition "the books of corporation XYZ are reviewed", and that occurrence occurs at the corporate headquarters.

A statement contains explicit and implicit references to time that restrict the time interval of the situation it describes. Time is inescapable in a temporal model, it is pervasive. There is a time interval(s) associated with every fact statement, explicitly or implicitly. Explicit references are time coordinates, indexicals, and definite descriptions. References to time are implicit in the tense and aspect of verbs. This specification includes definitions of time coordinates, indexicals, and calendar terms used in statements, and formulations for the most common tenses and aspects.

Each example given above assumes that the relevant concepts are defined in domain-specific vocabularies. Such vocabularies include verb concepts, such as "flight *takes off*". Human languages use many different prepositions ("at," "on," "in," "during," etc.) for relationships with time. This specification supports verb concepts with a few of these prepositions, with the expectation that business vocabularies will define verb concepts using other prepositions as appropriate for particular business domains.

## 7.11 Temporal Granularity

The granularity of a time point is important to the semantic meaning of a statement such as "Apollo 13 *launched on* 11 April 1970".

Since we know from background knowledge that the launch took much less than a day, we understand this as "the occurrence 'Apollo 13 *launched*' happened *within* the specified calendar day". Public records show that Apollo 13 actually launched at "14:13 EST" on that day. But the statement "Apollo 13 *launched on* 11 April 1970" does not give any hours or minutes; it just gives the day. It tells us that the occurrence happened sometime during the day or perhaps throughout the day. It tells us no more. If given as "Apollo 13 *launched on* 11 April 1970 at 14:13 EST", and assuming the launch took less than a minute, then we would know the time with minute granularity, that is that the launch happened within the specified minute of hour.

## 7.12 Language Tense and Aspect

Most human languages incorporate *tenses*, to indicate whether propositions occur in the past, the present, or the future with respect to the time of utterance of the proposition. For example, "company x *traded with* company y" is past tense. This specification captures the semantic meaning of tenses by associating situation kinds and occurrences with time and then indicating whether that time is past, present, or future with respect to current time. For example "company x *traded with* company y" is understood as "the occurrence 'company x *trades with* company y' *is in the past*". This approach to formalizing human sentences about tense follows [Parsons].

Many human languages also incorporate *simple*, *progressive*, and *perfect* aspects. *Simple aspect* applies to activities independent of whether they are ongoing or completed. For example "company x *traded with* company y", meaning that the two companies did trade, but does not say whether the trading is ongoing or completed. *Progressive aspect* means that an activity was ongoing or is ongoing or will be ongoing. For example "company x *was trading with* company y", meaning that the trading was continuing.

*Perfect* aspect indicates that an activity is accomplished. For example, "company x *will have traded with* company y" says that at some time in the future, the trading activity will be achieved. The difference between the simple and perfect aspects is shown by comparing the phrases "John *writes* a book" and "John *has written* a book". The second example, using "*has written*" applies the perfect aspect to indicate that the writing is complete. The first example, using "*writes*" uses the simple aspect. It does not say whether the writing is finished.

The progressive and perfect aspects may be combined to indicate that an activity both was ongoing, and is achieved. For example, "John *has been writing* a book" indicates that the writing occurred over time and the writing is completed or achieved.

In this specification, the progressive and perfect aspects are formally captured by characteristics of situation kinds and occurrences: "situation kind *is continuing*" and "situation kind *is accomplished*". Thus, any situation kind may be progressive or not, and may be perfected or not. Both are independent of whether the situation kind is in the past, the present, or in the future.

Human languages enable combinations of tense and aspect. The following table gives a grammatical term and shows an example for each combination. The table assumes a domain vocabulary has a verb concept "$\text{company}_1$ *trades with* $\text{company}_2$". The table shows semantic concepts of tense and aspect using English syntax for illustration purposes only. Different natural languages use different syntaxes to express these semantics. Some natural languages do not distinguish each combination shown in the table. Annex E contains an informative formal analysis of English language syntax for tense and aspect.

**Table 7.1 - Language Tense and Aspect**

| | | **Aspect** | | | |
|---|---|---|---|---|---|
| | | Simple | Progressive | Perfect | Progressive & Perfect |
| **Tense** | Past | past simple<br>company x *traded with* company y | past progressive<br>company x *was trading with* company y | past perfect, pluperfect<br>company x *had traded with* company y | pluperfect progressive<br>company x *had been trading with* company y |

| | | | | | |
|---|---|---|---|---|---|
| | Present | present simple<br>companyx *trades with* companyy | present progressive<br>companyx *is trading with* companyy | present perfect<br>companyx *has traded with* companyy | present perfect progressive<br>companyx *has been trading with* companyy |
| | Future | future simple<br>companyx *will trade with* companyy | future progressive<br>companyx *will be trading with* companyy | future perfect<br>companyx *will have traded with* companyy | future perfect progressive<br>companyx *will have been trading with* companyy |

These combinations can be employed in business rules, as shown in these examples. They presume a domain vocabulary verb concept "company$_1$ *merges with* company$_2$".

1. "If some company$_1$ *merged with* the companyx …" – asking whether a merger happened in the past, independent of whether the trading is ongoing, completed, or both.
2. "If some company$_1$ *was merging with* the companyx …" – asking whether a merger was continuing over some time in the past.
3. "If some company$_1$ *will have merged with* the companyx …" – asking whether a merger will be accomplished in the future.
4. "If some company$_1$ *will have been merging with* the companyx …" – asking whether a completed merger will be ongoing in the future.

One intended use case for these many combinations is annotation of existing text, as in [TimeML].

Sub clause 16.9 provides vocabulary for formulating tenses and aspects, and describes how these may be combined in rules.

## 7.13 Domain Vocabularies and Time

This specification provides foundational date and time concepts that are intended for use in domain-specific business vocabularies and rules. Annex C gives a complete example. This sub clause shows an abbreviated example in order to introduce how a domain vocabulary can build on this Date-Time Vocabulary.

Consider the example of a contract that has a "start date," a "contract length," a "contract term," and a "payment schedule." A business vocabulary might specify these as follows:

---

### Example Vocabulary

General Concept: terminological dictionary
Language: English

---

### contract

Definition: Agreement between two companies for one to provide goods or services, and for the other to pay for those goods or services

### start date

General Concept: calendar day

Note: The granularity of a domain vocabulary time concept is defined via the time point kind. Defining 'start date' as a calendar day means that the granularity of 'start date' is 'day' rather than 'week' or 'month', etc.

Note: Domain vocabulary time concepts should be defined as kinds of 'time point' or 'duration', rather than 'time coordinate' or 'duration value'. Actual 'time points' and 'durations' can be specified as definite descriptions as well as 'time coordinates' and 'duration values'.

## contract *has* start date

## contract length

General Concept: duration

Necessity: The granularity *of* 'contract length' *is* 'day'.

## contract *has* contract length

## contract term

Definition: Time interval during which the goods should be delivered or the services provided.

Necessity: The time interval *of* a contract *is from* the start date *of* the contract *for* the contract length.

## contract *has* contract term

## payment schedule

Definition: schedule *for* contract payments in which the time span *is* the contract term, and the repeat duration *is* 1 month

## contract *has* payment schedule

## contract payment

Definition: amount to be paid according to the payment schedule

## contract *has* contract payment

A business rule example might be:

> It is obligatory that a contract payment *be paid on* each time table entry *of* the payment schedule.

The example is simplified since it does not specify all the details that would exist in a real contract. For example, it does not indicate who makes the payment or who receives the payment, nor does it allow for payments other than monthly. But it does illustrate some basic ideas:

1. Defining domain vocabulary concepts that make use of time points (start date), durations (contract length), time intervals (contract term), and schedules (payment schedule).
2. Using Definitions (start date, contract term, payment schedule) and Necessities (contract term) to precisely capture the semantic meaning of domain concepts.
3. Specifying business rules that build upon this Date-Time Vocabulary and domain vocabularies to model business requirements.

Consider a business rule such as "It is obligatory that the contract length *of* each contract *is less than* 1 year." Notice that it compares 'contract length' to '1 year'. It does not quantify over 'year' because time is a mass noun concept. In contrast, a rule such as "It is obligatory that each rental *has* at most 3 additional drivers" uses quantification because 'additional driver' is a countable noun concept. Mass noun concepts are measured (possibly in fractional units of measure) while countable noun concepts are counted in whole units.

## 7.14 Enabling Other Calendars

The world has many different time-keeping and calendar systems. Specialized business calendars include fiscal calendars, tax calendars, and manufacturing calendars. Examples of historical, religious, and cultural calendars include the Julian calendar, various lunar calendars, and the 14-year calendar cycle of some Asian nations. Examples of time-keeping systems are those based on mariners' "bells", and religious "vespers".

This specification defines vocabularies for the standard, globally recognized "Universal Date Coordinated" (UTC) time system, and the Gregorian Calendar. In addition, this specification provides a Time Infrastructure Vocabulary that enables others to define business domain-specific, cultural, religious, or historical calendars and time schemas. The Time of Day Vocabulary and Gregorian Calendar Vocabulary show how time and calendar systems can be defined using the foundational concepts of the Time Infrastructure Vocabulary. Specifying time systems and calendars in terms of the foundational concepts of the Time Infrastructure Vocabulary enables conversions between different calendars and different time keeping schemas.

## 7.15 Precise and Nominal Time Units

This specification distinguishes precise time units from nominal time units, as defined in sub clause 8.4. Precise time units are measurement units (Annex D.3.2) in the sense of VIM: quantities of quantity kind 'duration' that are defined by convention. All precise time units are defined (sub clause 8.4) in terms of the SI 'second': picosecond, nanosecond, millisecond, microsecond, minute, hour, day, week.

Two other time units – 'month' and 'year' – are called 'nominal time units'. The duration of 'year' varies, depending upon whether a given calendar year includes a leap day. The duration of 'month' varies by definition. These time units are mentioned but not formally defined in [SI]. This specification formally defines these nominal time units (sub clause 8.4) in terms of sets of durations. For example, 'year' is defined as the set {365 days, 366 days}. Sub clauses 11.5 and 11.6 develop algorithms that specify the meaning of multiples of these nominal time units. For example, 2 years is {730 days, 731 days}, not {730 days, 732 days} because 2 calendar years contains just one leap day. This method enables well-defined results for comparisons such as "2 years ≥ 730 days" and arithmetic expressions such as "4 years – 3 months", which is {1369 days, 1370 days, 1371 days, 1372 days}. This permits logical reasoning systems to infer results that otherwise would be unreachable.

Domain-specific vocabularies may define their own precise time units and nominal time units as required by particular business conventions.

## 7.16 Temporal Aspects of Rules

Broadly speaking, all business rules define, constrain, or guide situations in some way. Some rules require a temporal relationship among situations, for example forbidding two situations from occurring concurrently:

> A person who *is driving* must not *be texting*.

SBVR Clause 10 states that rules apply to possible worlds, and that each possible world captures a 'fact population'. As time progresses, the fact population evolves. Rules, such as the example given above, are evaluated with respect to an individual fact population at a specific time, the reference or current time.

In the example given above, the verbs '*is driving*' and '*be texting*' use the present progressive tense as described in sub clause 16.7: the activities are unfinished at some reference time interval. The "reference time interval" is understood to be any time that the rule is considered. This can be made explicit with the following wording, which is shown here to make the meaning clear. The previous phrasing is shorter, clearer, and recommended.

> A person who *is driving* for some time interval must not *be texting during* the time interval.

Unless otherwise stated, rules apply at all times. To limit a rule to some time interval, a behavioral rule can state when it applies. For example:

*After* January 1, 2012, each expense that *costs more than* $1,000 *must be approved by* a director.

The examples given above are all behavioral (deontic) rules: prohibitions and obligations. By their nature, structural (alethic) rules (necessities, impossibilities) apply to all times in all possible worlds, but they can still specify relationships among the times of situations. For example:

It is necessary that the birth date *of* each person *is after* the birth dates *of* the parents of the person.

The first two example rules, above, apply to occurrences of two different situation kinds. When behavioral and structural rules pertain to multiple occurrences of a *single* situation kind, the rules may be abbreviated. For example:

It is prohibited that a renter *has possession of* more than one rental car.

What is prohibited is a possible world in which a renter possesses multiple rental cars. This is equivalent to the following rule, which is not recommended because it is much more complex, and significantly harder to understand:

It is prohibited that a renter *has possession of* a rental $car_1$ *at* a time $interval_1$ and the renter *has possession of* a rental $car_2$ *at* a time $interval_2$ and time $interval_1$ *overlaps* time $interval_2$.

SBVR Clause 10 distinguishes between *static* constraints and *dynamic* constraints. Static constraints "impose[s] a restriction on what fact populations are possible or permitted, for each fact population taken individually." [SBVR sub clause 10.1.1.2] Dynamic constraints "impose[s] a restriction on transitions between fact populations." [ibid] The examples given above are static constraints. The previous example may also be stated as a dynamic constraint:

It is prohibited that a renter *takes possession of* a rental $car_1$ *while* the renter *has possession of* a rental $car_2$.

… where the verb concept 'renter *takes possession of* rental car' uses the simple present tense to identify an event and 'renter *has possession of* rental car' uses the present progressive tense to indicate an ongoing situation. See sub clause 16.9 for a discussion of the tense and aspect of verbs.

Domain modelers have the choice of writing static or dynamic constraints, but static constraints are recommended in SBVR because static constraints capture the complete business requirement, whereas dynamic constraints tend to address specific aspects of the business practice – possibly ignoring other aspects. In the last example, there might be other ways that a renter could end up possessing two rental cars, but the example rule only addresses one such way.

# 8 Time Infrastructure (normative)

## 8.1 General

Many time schemes and calendars are in use to support a variety of business needs, and due to historical, cultural, and religious traditions. The Time Infrastructure vocabulary provides a foundation for defining any time keeping or calendar system. Relating different time and calendar schemes to each other is made possible by using the foundational concepts provided in this clause.

---

Time Infrastructure Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Language: | English |
| Included Vocabulary: | Mereology Vocabulary |
| Included Vocabulary: | Quantities Vocabulary |

| | |
|---|---|
| Included Vocabulary: | Sequences Vocabulary |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#TimeInfrastructureVocabulary |

______________________________________________

## 8.2 The Time Axis and Time Intervals

The principal concept in this sub clause is time interval. This concept is used to define many of the business terms that are specified in other clauses of this specification. Formally, time interval is a primitive concept – an intuitive notion that does not have a mathematical definition. Its properties are defined by a set of axioms that are presented here as SBVR definitions and Necessities with matching CLIF and OCL statements. Much of this clause is the presentation of those axioms.

### Time Axis

| | |
|---|---|
| Dictionary Basis: | IEC 60050-111 (‘time axis’) |
| Dictionary Basis: | IEC 8601 (2.1.1, ‘time axis’) |
| Definition: | mathematical model of the succession in time of events along a unique axis |
| Source: | NODE (‘time’) |
| Definition: | the indefinite continued progress of existence and events in the past, present, and future, regarded as a continuum |
| Necessity: | There exists exactly one Time Axis. |
| Note: | The above necessity is questionable in light of the theory of relativity, but relativistic effects are not considered in this model. Some applications need to take these effects into account, e.g., GPS, in which the clocks in satellites are adjusted on the ground to compensate for relativistic shifts in their rates in orbit, due to the lower gravitational field in orbit (+) and orbital motion (-). |
| Note: | Time Axis is the conceptual time dimension. |
| Note: | “Time” could be a synonym of Time Axis, but “time” is often confused with other concepts, such as duration and time of day. |

### time interval

| | |
|---|---|
| Definition: | segment of the time axis, a location in time |
| Note: | Every time interval has a beginning, an end, and a duration, even if not known. Every time interval is “finite”, a bounded segment of the Time Axis. The beginning or end of a time interval may be defined by reference to events that *occur for* a time interval that is not known. |
| Note: | Time intervals may be ‘indefinite’, meaning that their beginning is ‘primordiality’ or their end is ‘perpetuity’, or both (‘eternity’). This vocabulary assumes that indefinite time intervals exist and have some duration, but their duration is unknown. |
| Reference Scheme: | an absolute time coordinate that *refers to* the time interval |
| Note: | Absolute time coordinates are related to calendars, and are introduced in clause 10.6. |
| Example: | The lifetime of Henry V. |
| Example: | The day whose Gregorian calendar date is September 11, 2001. |

### 8.2.1 The Whole-Part Relationship Among Time Intervals

The mereological principles described in Annex D.4 apply to time intervals.

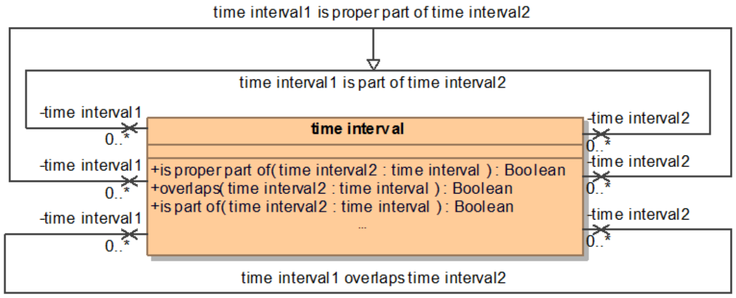


**Figure 8.1 - Mereology as Applied to Time Intervals**

## time interval$_1$ *is part of* time interval$_2$

Synonymous Form: time interval$_2$ *includes* time interval$_1$
Synonymous Form: time interval$_1$ *is in* time interval$_2$
Synonymous Form: time interval$_1$ *in* time interval$_2$
Definition: Time interval$_2$ is a component of time interval$_1$. Every instant in time interval$_1$ is also in time interval$_2$. Everything that happens in time interval$_1$ happens in time interval$_2$
Note: Like the concept time interval itself, this relationship is also primitive – intuitive. It is a mathematical ordering of time intervals by containment.
CLIF Axiom:

```
(forall (t1 t2)
  (if ("time interval1 is part of time interval2" t1 t2)
    (and ("time interval" t1) ("time interval" t2)
      ("thing1 is part of thing2" t1 t2))))
```

Note: The OCL operation signature implies this constraint.
Note: This relationship is based on the mereological verb concept 'part *is part of* whole' (Annex D.4). All the axioms cited there for 'part *is part of* whole' apply to 'time interval$_1$ *is part of* time interval$_2$'.
Note: The axioms of reflexivity, anti symmetry, and transitivity (Annex D.4) make 'time interval$_1$ *is part of* time interval$_2$' a partial ordering relationship on time intervals. The relationship is *partial* because two arbitrary time intervals might be disjoint or might overlap, so that there is no part-whole relationship between them.

## time interval$_1$ *overlaps* time interval$_2$

Note: This relationship is the mereological verb concept 'thing1 overlaps thing2' in Annex D.4.
CLIF Axiom:

```
(forall (t1 t2)
  (if ("time interval1 overlaps time interval2" t1 t2)
    (and ("time interval" t1) ("time interval" t2)
      ("thing1 overlaps thing2" t1 t2))))
```

## time interval$_1$ *is a proper part of* time interval$_2$

Note: This relationship is based on the mereological verb concept 'part *is a proper part of* whole' (Annex D.4). See the definition of that concept for details. For time intervals, stronger supplementation axioms are given in 8.2.6.
CLIF Axiom:

```
(forall (t1 t2)
  (if ("time interval1 is proper part of time interval2" t1 t2)
    (and ("time interval" t1) ("time interval" t2)
      ("thing1 is proper part of thing2" t1 t2))))
```

Note: The OCL operation signature implies this constraint.

Note: A proper part is a part that is not the whole.

**Axiom**: There is no smallest time interval.

Necessity: For each time interval$_1$, there is at least one time interval$_2$ that *is a proper part of* time interval$_1$.

CLIF Axiom:
```
(forall (ti1 "time interval")
  (exists (ti2 "time interval")
    ("proper part of" ti2 ti1) ))
```

OCL Constraint:
```
context _'time interval'
    inv: self._'time interval1 is proper part of time interval2'::_'time interval1'->notEmpty()
```

Note: This axiom requires the Open World Assumption: Things can exist without being explicitly included in a population.

### 8.2.2 The Temporal Ordering Relationship

A fundamental property of time intervals is the totally ordered '*is before*' relationship, which defines temporal ordering.

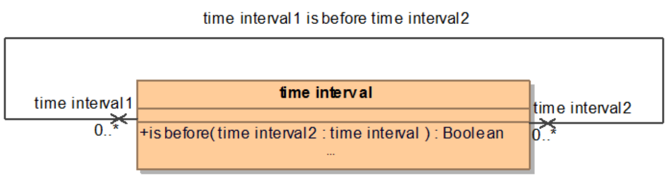


**Figure 8.2 - Temporal Ordering**

## time interval$_1$ *is before* time interval$_2$

Synonymous Form: time interval$_2$ *is after* time interval$_1$
Synonymous Form: time interval$_1$ < time interval$_2$
Synonymous Form: time interval$_2$ > time interval$_1$
Synonymous Form: time interval$_1$ *precedes* time interval$_2$
Synonymous Form: time interval$_2$ *is preceded by* time interval$_1$
Synonymous Form: time interval$_2$ *follows* time interval$_1$
Synonymous Form: time interval$_1$ *is followed by* time interval$_2$
Definition: time interval$_1$ ends before/when time interval$_2$ starts
Example: In any given calendar, the time interval identified by 2010 is before the time interval identified by 2011.
Note: This relationship is also primitive – intuitive. It is a mathematical ordering of time intervals by position on the Time Axis. *Is before* captures the intuition of the direction of time, of past and future: if x is before y, then y is in the future relative to x and x is in the past relative to y.

CLIF Axiom:
```
(forall (t1 t2)
  (if ("time interval1 is before time interval2" t1 t2)
    (and ("time interval" t1) ("time interval" t2))))
```

Note: The OCL operation signature implies this constraint.
Note: The actual determination of the ordering of time intervals may be based on direct observation, on calendar knowledge, on historical knowledge, or on practical knowledge. One can see the order in which two vehicles enter an intersection and infer the corresponding facts about the time intervals involved (observation). One can know from calendar rules that November 11, 1918 was before September 1, 1939. One can know from the reports of others (historical knowledge) that railroads were in use for many years before automobiles first appeared. Knowing that every airplane takes off before it lands (practical knowledge), and that a particular airplane has taken off and landed, one can infer that the

| | |
|---|---|
| | time interval of the takeoff was before the time interval of the landing. And, of course, these knowledge elements can be mixed in determining time interval ordering. When such knowledge elements are formalized as facts and rules in an ontology, the inferences about the ordering of time intervals can be automated. |
| Note: | The following axioms define the properties of this primitive concept. |

**Axiom**: time interval$_1$ *is before* time interval$_2$ can only be true of time intervals that do not overlap.

Necessity: If a time interval$_1$ *overlaps* a time interval$_2$, then the time interval$_1$ *is* not *before* the time interval$_2$.

CLIF Axiom:
```
(forall (t1 t2)
  (if ("time interval1 overlaps time interval2" t1 t2)
    (and
      (not ("time interval1 is before time interval2" t1 t2))
      (not ("time interval1 is before time interval2" t2 t1)) )))
```

OCL Constraint:
```
context _'time interval'
    inv: _'time interval'.allInstances->
        forAll(t2 | self.overlaps(t2) implies not self._'is before'(t2))
```

**Corollary**:

Necessity: If a time interval$_1$ *overlaps* a time interval$_2$, then the time interval$_1$ *is* not *after* the time interval$_2$.

Note: This follows from the fact that 'time interval$_1$ *overlaps* time interval$_2$' is symmetric.

**Axiom:** For any two time intervals that do not overlap, one *is before* the other.

Necessity: If a time interval$_1$ does not *overlap* a time interval$_2$, then the time interval$_1$ *is before* the time interval$_2$ or the time interval$_2$ *is before* the time interval$_1$.

CLIF Axiom:
```
(forall ((t1 "time interval") (t2 "time interval"))
  (if (not ("time interval1 overlaps time interval2" t1 t2))
     (or ("time interval1 is before time interval2"
         t1 t2)
        ("time interval1 is before time interval2"
         t2 t1))))
```

OCL Constraint:
```
context _'time interval'
    inv: _'time interval'.allInstances->
        forAll(t2 |
          not self.overlaps(t2)
          implies (self._'is before'(t2) or t2._'is before'(self)) )
```

**Corollary (*irreflexivity*)**: No time interval *is before* itself.

Necessity: A given time interval *is* not *before* the time interval.

CLIF Axiom:
```
(forall ((t1 "time interval"))
  (not ("time interval1 is before time interval2" t1 t1)))
```

OCL Constraint:
```
context _'time interval'
    inv: not self._'is before'(self)
```

**Axiom of *asymmetry***: No time interval is both *before* and *after* the same time interval.

Necessity: If a time interval$_1$ *is before* a time interval$_2$, then the time interval$_2$ *is* not *before* the time interval$_1$.

CLIF Axiom:
```
(forall (t1 t2)
  (if ("time interval1 is before time interval2" t1 t2)
      (not ("time interval1 is before time interval2" t2 t1))))
```

OCL Constraint:
```
context _'time interval'
    inv: _'time interval'.allInstances->
        forAll(t2 |
```

```
self._'is before'(t2)
  implies not t2._'is before'(self))
```

**Corollary** (totality): For any two time intervals *t1* and *t2*, exactly one of the following is true:

- t1 *overlaps* t2
- t1 *is before* t2
- t2 *is before* t1

Necessity: Each time interval$_1$ *overlaps* each time interval$_2$ and time interval$_1$ *is* not *before* time interval$_2$ and time interval$_2$ *is* not *before* time interval$_1$, or time interval$_1$ *is before* time interval$_2$ and time interval$_1$ *does* not *overlap* time interval$_2$ and time interval$_2$ *is* not *before* time interval$_1$, or time interval$_2$ *is before* time interval$_1$ and time interval$_1$ *does* not *overlap* time interval$_2$ and time interval$_1$ *is* not *before* time interval$_2$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval"))
  (or
    ("time interval1 overlaps time interval2" t1 t2)
    (and
      ("time interval1 is before time interval2" t1 t2)
      (not ("time interval1 overlaps time interval2" t1 t2)))
    (and
      ("time interval1 is before time interval2" t2 t1)
      (not ("time interval1 overlaps time interval2" t1 t2)))
 ))
```

OCL Constraint:

```
context _'time interval'
    inv: _'time interval'.allInstances->
        forAll(t2 |
         (self.overlaps(t2)
           and not self._'is before'(t2)
           and not t2._'is before'(self))
        or (self._'is before'(t2)
            and not self.overlaps(t2)
            and not t2._'is before'(self))
        or (t2._'is before'(self)
            and not self.overlaps(t2)
            and not self._'is before'(t2)))
```

**Axiom** of *transitivity*: Every time interval that *is before* a given time interval is also *before* every time interval that is after the given time interval.

Necessity: If a time interval$_1$ *is before* a time interval$_2$ and the time interval$_2$ is before a time interval$_3$ then the time interval$_1$ *is before* the time interval$_3$.

CLIF Axiom:

```
(forall (t1 t2 t3)
  (if
    (and
      ("time interval1 is before time interval2" t1 t2)
      ("time interval1 is before time interval2" t2 t3))
    ("time interval1 is before time interval2" t1 t3)))
```

OCL Constraint:

```
context _'time interval'
    inv: _'time interval'.allInstances->
         forAll(t2, t3 |
           self._'is before'(t2)
           and t2._'is before'(t3)
           implies self._'is before'(t3))
```

The preceding 3 axioms specify that 'time interval$_1$ ***is before*** time interval$_2$' is anti-reflexive, weakly antisymmetric, and transitive. The relationship does *not* apply to all pairs of time intervals. This characterizes a kind of partial ordering on time intervals.

### 8.2.3 The Allen Relations

In a 1983 paper [Allen], James F. Allen asserted that there are exactly thirteen ways in which an ordered pair of time intervals can be related. His Figure 2, showing these relationships, is reproduced below.

| Relation | Symbol | Symbol for Inverse | Pictoral Example |
|---|---|---|---|
| X *before* Y | < | > | XXX YYY |
| X *equal* Y | = | = | XXX<br>YYY |
| X *meets* Y | m | mi | XXXYYY |
| X *overlaps* Y | o | oi | XXX<br>  YYY |
| X *during* Y | d | di |   XXX<br>YYYYYY |
| *X starts Y* | s | si | XXX<br>YYYYY |
| *X finishes Y* | f | fi |    XXX<br>YYYYY |

**FIGURE 2. The Thirteen Possible Relationships.**

**Figure 8.3 - Allen's Original Diagram of the 13 Time Relationships**

According to Thomas Alspaugh [Alspaugh], these relations are *distinct* ("because no pair of definite intervals can be related by more than one of these relationships"), *exhaustive* ("because any pair of definite intervals are described by one of the relations"), and *qualitative*, rather than *quantitative*, ("because no numeric time spans are considered").

The word '***properly***' is used in the terms for some of the Allen relations below, in order to distinguish those relations from the more general relations defined in 8.2.1 and 8.2.2. In each case of terminology clash, the Allen's term is narrower. The business use of the general term – before, after, part of, includes, during, overlaps – almost always means the more general relationship.

The Allen relations are *independent*: none is entailed by another and none is defined in terms of the others. They are, however, all defined here in terms of the two fundamental relationships: '***part of***' and '***before***'.

The '***properly before***' and '***meets***' relations are mutually exclusive. The primitive relationship '***before***' subsumes both. Allen's '***before***' concept is designated here as '***properly before***' to indicate there is necessarily an intervening time interval.

The '***properly overlaps***' relation distinguishes the case in which there is a part of each time interval that is not a part of the other from all the cases in which one time interval is entirely a part of the other. The general '***overlaps***' relation subsumes all of them. '***Properly overlaps***' describes the first time interval as starting earlier than the second starts and ending earlier than the second ends, whereas '***is properly overlapped by***' describes the first time interval as starting later than the second starts, and ending later than the second ends.

The '***properly during***,' '***starts***', and '***finishes***' relationships are mutually exclusive. The general '***part of***' relationship subsumes all of them. They are distinguished by the temporal relationship of the included time interval to the supplementary parts of the whole.

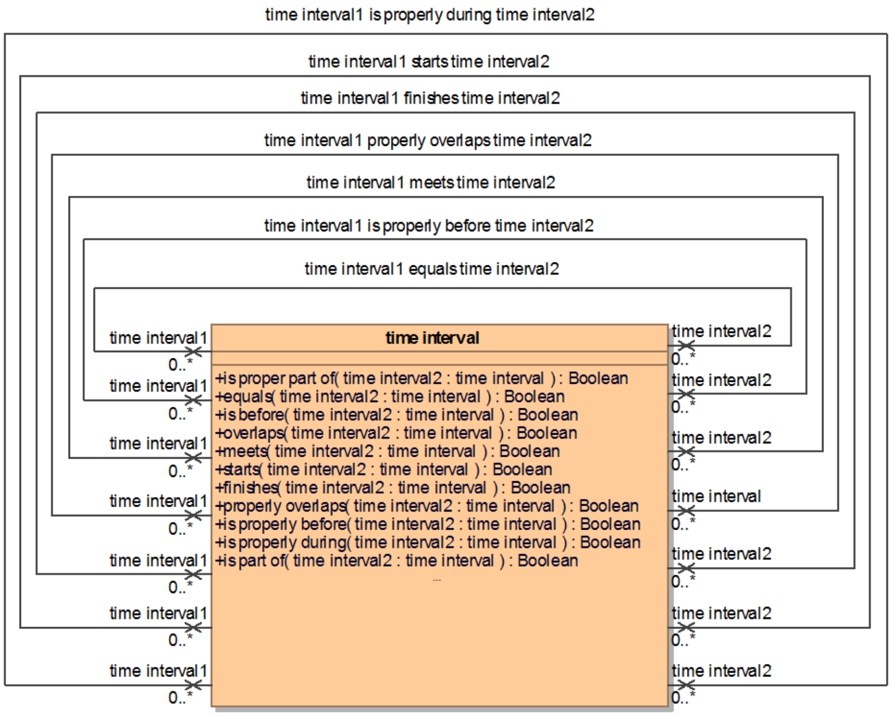


**Figure 8.4 - UML Diagram of Allen Relations**

## time interval$_1$ *is properly before* time interval$_2$

Synonymous Form: time interval$_2$ *is properly after* time interval$_1$

Definition: time interval$_1$ *is before* time interval$_2$ and some time interval$_3$ *is after* time interval$_1$ and *is before* time interval$_2$

Description: time interval$_1$ is before time interval$_2$ and there is some time interval between them.

CLIF Definition:
```
(forall (t1 t2)
  (iff ("time interval1 is properly before time interval2" t1 t2)
    (and
      ("time interval" t1) ("time interval" t2)
      ("time interval1 is before time interval2" t1 t2)
      (exists (t3)
        (and ("time interval1 is before time interval2" t1 t3)
             ("time interval1 is before time interval2" t3 t2))
) )))
```

OCL Definition:
```
context _'time interval'
def: _'time interval1 is properly before time interval2' (t2: _'time interval'): Boolean =
   self._'is before'(t2) and
   'time interval'.allInstances->exists(t3 | self._'is before'(t3) and t3._'is before'(t2))
```

Example: In any given calendar, 2009 *is properly before* 2011

## time interval$_1$ *equals* time interval$_2$

Synonymous Form: time interval$_1$ *is the same as* time interval$_2$
Synonymous Form: time interval$_1$ = time interval$_2$
General Concept: thing$_1$ *is* thing$_2$
Definition: the time interval$_1$ *is part of* the time interval$_2$ and the time interval$_2$ *is part of* the time interval$_1$
CLIF Definition:
```
(forall (t1 t2)
  (iff ("time interval1 equals time interval2" t1 t2)
    (and ("time interval1 is part of time interval2" t1 t2)
         ("time interval1 is part of time interval2" t2 t1)) ))
```
OCL Definition:
```
context _'time interval'
def: _'time interval1 equals time interval2' (t2: _'time interval'): Boolean =
    self._'is part of'(t2) and t2._'is part of'(self)
```
Note: That is, the mereology axiom of antisymmetry in Annex D.4 is really the formal definition of '*equals*.' Two time intervals are equal if and only if each is part of the other.
Note: SBVR uses the verb *is* for this relationship, but the equals relationship here is a specialization of 'thing *is* thing' for time intervals.
Necessity: A time interval$_1$ *equals* a time interval$_2$ if and only if time interval$_1$ *is* time interval$_2$
CLIF Axiom:
```
(forall (ti1 ti2)
  (if (and ("time interval" ti1) ("time interval" ti2))
     (iff ("time interval equals time interval" ti1 ti2)
          ("thing1 is thing2" ti1 ti2) )))
```
OCL Constraint:
```
context _'time interval'
     inv: _'time interval'.allInstances->
     forAll(t2 |
       self.equals(t2) implies self.is(t2)
       and (self.is(t2) implies self.equals(t2) ))
```
Example: January 2011 *through* December 2011 *equals* 2011

## time interval$_1$ *meets* time interval$_2$

Synonymous Form: time interval$_2$ *is met by* time interval$_1$
Synonymous Form: time interval$_1$ *immediately precedes* time interval$_2$
Synonymous Form: time interval$_2$ *immediately follows* time interval$_1$
Definition: time interval$_1$ *is before* time interval$_2$ and no time interval$_3$ *is after* time interval$_1$ and *is before* time interval$_2$
Description: time interval$_1$ is before time interval$_2$ and there is no time interval between them: time interval$_2$ starts at the instant time interval$_1$ ends.
CLIF Definition:
```
(forall (t1 t2)
  (iff ("time interval1 meets time interval2" t1 t2)
    (and
      ("time interval1 is before time interval2" t1 t2)
      (not (exists (t3)
        (and ("time interval1 is before time interval2" t1 t3)
             ("time interval1 is before time interval2" t3 t2))
)) )))
```
OCL Definition:
```
context _'time interval'
def: _'time interval1 meets time interval2'(t2: _'time interval'): Boolean =
    self._'is before'(t2) and
    not _'time interval'.allInstances->
      exists(t3 | self._'is before'(t3) and t3._'is before'(t2))
```
Example: 2009 *meets* 2010

## time interval₁ *properly overlaps* time interval₂

Synonymous Form: time interval₂ *is properly overlapped by* time interval₁

Definition: time interval₁ *overlaps* time interval₂ and some part *of* time interval₁ *is before* time interval₂

Description: Part of time interval₁ is before time interval₂ and the rest of time interval₁ is also part of time interval₂.

CLIF Definition:
```
(forall (t1 t2)
 (iff ("time interval1 properly overlaps time interval2" t1 t2)
  (and
   ("time interval1 overlaps time interval2" t1 t2)
   (exists (t3)
    (and ("time interval1 is proper part of time interval2" t3 t1)
         ("time interval1 is before time interval2" t3 t2))
) )))
```

OCL Definition:
```
context _'time interval'
def: _'time interval1 properly overlaps time interval2' (t2: _'time interval'): Boolean =
   self.overlaps(t2) and
   'time interval'.allInstances->
    exists(t3 | t3._'is a proper part of'(self) and t3._'is before'(t2))
```

Example: July 2010 *through* February 2011 *properly overlaps* January 2011 *through* March 2011

## time interval₁ *is properly during* time interval₂

Synonymous Form: time interval₂ *properly includes* time interval₁

Definition: time interval₁ *is part of* time interval₂ and some part *of* time interval₂ *is before* time interval₁ and some part *of* time interval₂ *is after* time interval₁

CLIF Definition:
```
(forall (t1 t2)
 (iff ("time interval1 is properly during time interval2" t1 t2)
  (and
   ("time interval1 is proper part of time interval2" t1 t2)
   (not ("time interval1 starts time interval2" t1 t2))
   (not ("time interval1 finishes time interval2" t1 t2))
)))
```

OCL Definition:
```
context _'time interval'
def: _'time interval1 is properly during time interval2' (t2: _'time interval'): Boolean =
   self._'is a proper part of'(t2) and
   'time interval'.allInstances->
     exists(t3, t4 |
      t3._'is a proper part of'(t2) and t4._'is a proper part of'(t2) and
       t3._'is before'(self) and self._'is before'(t4))
```

Example: July 2010 *is properly during* 2010

## time interval₁ *starts* time interval₂

Synonymous Form: time interval₂ *is started by* time interval₁

Definition: time interval₁ *is a proper part of* time interval₂ and no part *of* time interval₂ *is before* time interval₁

Description: time interval₁ is a proper part of time interval₂ and they both start at the same instant.

CLIF Definition:
```
(forall (t1 t2)
 (iff ("time interval1 starts time interval2" t1 t2)
  (and
   ("time interval1 is proper part of time interval2" t1 t2)
   (not (exists (t3)
      (and ("time interval1 is proper part of time interval2" t3 t2)
           ("time interval1 is before time interval2" t3 t1)) ))
)))
```

OCL Definition:
```
context _'time interval'
def: _'time interval1 starts time interval2' (t2: _'time interval'): Boolean =
    self._'is a proper part of'(t2) and
    not 'time interval'.allInstances->
      exists(t3 | t3._'is a proper part of'(t2) and t3._'is before'(self))
```

Example: January 2010 *starts* 2010

## time interval₁ *finishes* time interval₂

Synonymous Form: time interval₂ *is finished by* time interval₁

Definition: time interval₁ *is a proper part of* time interval₂ and no part *of* time interval₂ *is after* time interval₁

Description: time interval₁ is a proper part of time interval₂ and they both end at the same instant.

CLIF Definition:
```
(forall (t1 t2)
 (iff ("time interval1 finishes time interval2" t1 t2)
  (and
   ("time interval1 is proper part of time interval2" t1 t2)
   (not (exists (t3)
      (and ("time interval1 is proper part of time interval2" t3 t2)
           ("time interval1 is before time interval2" t1 t3)) ))
)))
```

OCL Definition:
```
context _'time interval'
def: _'time interval1 finishes time interval2'(t2: _'time interval'): Boolean =
    self._'is a proper part of'(t2) and
    not _'time interval'.allInstances->
      exists(t3 | t3._'is a proper part of'(t2) and self._'is before'(t3))
```

Example: December 2010 *finishes* 2010

### 8.2.4 Additional Time Interval Relationships

As described in [Alspaugh], the basic Allen relationships can be combined in $2^{13}$ (8192) ways. This sub clause defines a few of these "combination" relationships that have particular value to everyday and business uses.

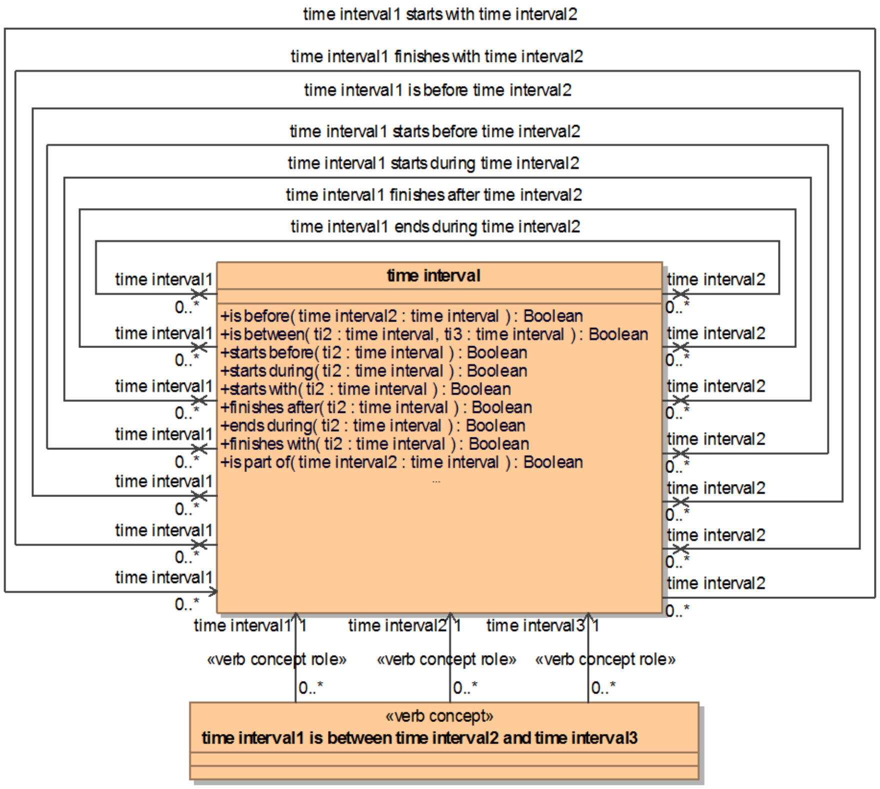


**Figure 8.5 - Additional Time Interval Relationships**

## time interval$_1$ *starts before* time interval$_2$

| | |
|---|---|
| Synonymous Form: | time interval$_2$ *starts after* time interval$_1$ |
| Definition: | some time interval$_3$ *is part of* time interval$_1$ and *is before* time interval$_2$ |
| Description: | Time interval$_1$ starts earlier than time interval$_2$ starts. |

CLIF Definition:

```
(forall (t1 t2)
  (iff ("time interval1 starts before time interval2" t1 t2)
    (exists (t3)
      (and
        ("time interval1 is before time interval2" t3 t2)
        ("time interval1 is part of time interval2" t3 t1) ))))
```

OCL Definition:

```
context _'time interval'
def: _'starts before'(t2: _'time interval'): Boolean =
'time interval'.allInstances->
  exists(t3 |
    t3._'is part of'(self)
    and t3._'is before'(t2))
```

Example: 2009 *starts before* 2010

Example: 2010 *starts before* February 2010

## time interval$_1$ *starts with* time interval$_2$

Synonymous Form: time interval$_1$ *starts when* time interval$_2$ *starts*

Definition: time interval$_1$ *starts* time interval$_2$ or time interval$_2$ *starts* time interval$_1$ or time interval$_1$ *equals* time interval$_2$

Description: The two time intervals start together, but either may end first. All of the following relationships are possible:

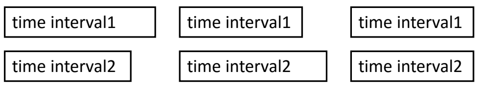


**Figure 8.6 - time interval1 starts with time interval2**

CLIF Definition:
```
(forall (t1 t2)
  (iff ("time interval1 starts with time interval2" t1 t2)
    (or
      ("time interval1 starts time interval2" t1 t2)
      ("time interval1 starts time interval2" t2 t1)
       ("time interval1 equals time interval2" t1 t2)) ))
```

OCL Definition:
```
context _'time interval'
def: _'time interval1 starts with time interval2'(t2: _'time interval'): Boolean =
  self.starts(t2) or t2.starts(self) or self.equals(t2)
```

Necessity: If time interval$_1$ *starts with* time interval$_2$ then time interval$_2$ *starts with* time interval$_1$

CLIF Axiom:
```
(forall ((t1 "time interval") (t2 "time interval"))
  (if ("time interval1 starts with time interval2" t1 t2)
    ("time interval2 starts with time interval1" t2 t1) ))
```

OCL Constraint:
```
context _'time interval'
    inv:_'time interval'.allInstances->forAll(t2 |
      self._'time interval1 starts with time interval2'(t2)
        implies t2. _'time interval1 starts with time interval2'(self)
```

## time interval$_1$ *starts during* time interval$_2$

Synonymous Form: time interval$_1$ *starts within* time interval$_2$

Definition: some time interval$_3$ *starts* time interval$_1$ and *is part of* time interval$_2$

Description: The start of time interval$_1$ is within time interval$_2$.

CLIF Definition:
```
(forall (t1 t2)
  (iff ("time interval1 starts during time interval2" t1 t2)
    (exists (t3)
      (and
        ("time interval1 starts time interval2" t3 t1)
        ("time interval1 is part of time interval2" t3 t2) ))))
```

OCL Definition:
```
context _'time interval'
def: _'starts during'(t2: _'time interval'): Boolean =
'time interval'.allInstances->
  exists(t3 |
    t3._'is part of'(t2)
    and t3.starts(self))
```

Example: Fiscal Year 2015 starts within Calendar Year 2014

Note: In most uses of this verb concept, one of the time intervals involved is described by an occurrence.

## time interval$_1$ *finishes with* time interval$_2$

Synonymous Form: time interval$_1$ *finishes when* time interval$_2$ *finishes*

Definition: time interval$_1$ *finishes* time interval$_2$ or time interval$_2$ *finishes* time interval$_1$ or time interval$_1$ *equals* time interval$_2$

Description: Either time interval may start first, but they finish together. All of the following relationships are possible:

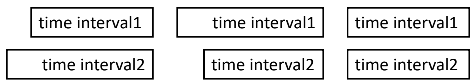


**Figure 8.7 - time interval1 finishes with time interval2**

CLIF Definition:
```
(forall (t1 t2)
  (iff ("time interval1 finishes with time interval2" t1 t2)
     (or
        ("time interval1 finishes time interval2" t1 t2)
        ("time interval1 finishes time interval2" t2 t1)
         ("time interval1 equals time interval2" t1 t2)) ))
```

OCL Definition:
```
context _'time interval'
def: _'time interval1 finishes with time interval2'(t2: _'time interval'): Boolean =
  t1.finishes(t2) or t2.finishes(t1) or t1.equals(t2)
```

Necessity: If time interval$_1$ *finishes with* time interval$_2$ then time interval$_2$ *finishes with* time interval$_1$

CLIF Axiom:
```
(forall ((t1 "time interval") (t2 "time interval"))
  (if ("time interval1 finishes with time interval2" t1 t2)
     ("time interval2 finishes with time interval1" t2 t1) ))
```

OCL Constraint:
```
context _'time interval'
       inv:_'time interval'.allInstances->forAll(t2 |
         self._'time interval1 finishes with time interval2'(t2)
           implies t2. _'time interval1 finishes with time interval2'(self) )
```

## time interval$_1$ *finishes after* time interval$_2$

Definition: some time interval$_3$ *is part of* time interval$_1$ and *is after* time interval$_2$

CLIF Definition:
```
(forall (t1 t2)
  (iff ("time interval1 finishes after time interval2" t1 t2)
    (exists (t3)
      (and
        ("time interval1 is before time interval2" t2 t3)
         ("time interval1 is part of time interval2" t3 t1) ))))
```

OCL Definition:
```
context _'time interval'
def: _'finishes after'(t2: _'time interval'): Boolean =
'time interval'.allInstances->
  exists(t3 |
    t3._'is part of'(self)
    and t2._'is before'(t3))
```

Example: 2010 *finishes after* February 2010

## time interval$_1$ *ends during* time interval$_2$

Synonymous Form: time interval$_1$ *ends within* time interval$_2$

Definition: some time interval$_3$ *finishes* time interval$_1$ and *is part of* time interval$_2$

Description: The end of time interval$_1$ is within time interval$_2$.

CLIF Definition: (forall (t1 t2)
(iff ("time interval1 ends during time interval2" t1 t2)
(exists (t3)
(and
("time interval1 finishes time interval2" t3 t1)
("time interval1 is part of time interval2" t3 t2) ))))

OCL Definition: context _'time interval'
def: _'ends during'(t2: _'time interval'): Boolean =
'time interval'.allInstances->
exists(t3 |
t3._'is part of'(t2)
and t3.finishes(self))

Example: The grace period will end in December.

Note: In most uses of this verb concept, one of the time intervals involved is described by an occurrence.

### time interval₁ *is between* time interval₂ *and* time interval₃

Synonymous Form: time interval₁ *is between* time interval₂ *to* time interval₃

Definition: time interval₁ *is after* time interval₂ and time interval₁ *is before* time interval₃

CLIF Definition: (forall (t1 t2 t3)
(iff ("time interval1 is between time interval2 and time interval3" t1 t2 t3)
(and
("time interval" t1) ("time interval" t2) ("time interval" t3)
("time interval1 precedes time interval2" t2 t1)
("time interval1 precedes time interval2" t1 t3) ) ))

OCL Definition: context _'time interval'
def: _'time interval1 is between time interval2 and time interval3'
(t2: _'time interval', t3: _'time interval'): Boolean =
t2.precedes(self) and self.precedes(t3)

Example: July 2012 *is between* June 2012 *to* August 2012

### 8.2.5 Time Interval Sum

This sub clause describes the " sum" of two time intervals – the smallest time interval that contains both of them.

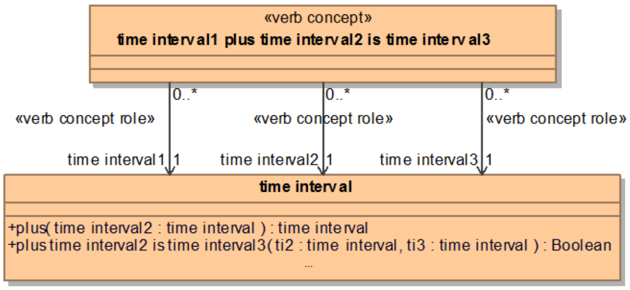


**Figure 8.8 - Time Interval Sum**

This concept of 'sum' is generalized. It may be said to represent the 'convex hull' of the two intervals, and it may contain intervals that lie between them. It is particularly useful, however, when t1 meets t2 or t2 meets t1, i.e., in those cases where t1 and t2 are disjoint and there is no time interval between them.

## time interval[1] *plus* time interval[2] *is* time interval[3]

Synonymous Form: time interval[1] + time interval[2] = time interval[3]
Synonymous Form: time interval[3] *is* time interval[1] *plus* time interval[2]
Synonymous Form: time interval[3] = time interval[1] + time interval[2]
Synonymous Form: time interval[1] *plus* time interval[2]
Synonymous Form: time interval[1] + time interval[2]
Synonymous Form: *sum of* time interval[1] + time interval[2]
Definition: time interval[3] *includes* time interval[1] and time interval[3] *includes* time interval[2] and time interval[3] *is part of* each time interval that *includes* time interval[1] and time interval[2]

CLIF Definition:

```
(forall  (t1 t2 t3)
   (iff ("time interval1 plus time interval2 is time interval3" t1 t2 t3)
        (and
           ("thing1 is part of thing2"  t1 t3)
           ("thing1 is part of thing2"  t2 t3)
           (forall (t4)
              (if (and
                      ("thing1 is part of thing2"  t1 t4)
                      ("thing1 is part of thing2"  t2 t4))
                  ("thing1 is part of thing2"  t3 t4)))
)))
```

OCL Definition:

```
context _'time interval'
def:  _'plus time interval2 is time interval3'
   (t2: _'time interval', t3: _'time interval'): Boolean =
      self._'is part of'(t3) and t2._'is part of'(t3) and
      _'time interval'.allInstances->
          forAll(t4 | self._'is part of'(t4) and t2._'is part of'(t4)
              implies t3._'is part of'(t4))
```

Necessity: if a time interval[1] *is before* a time interval[2] or time interval[1] *properly overlaps* time interval[2], then time interval[1] *plus* time interval[2] *is started by* time interval[1] and *is finished by* time interval[2]

CLIF Axiom:

```
(forall  (t1 t2 t3)
    (if
      (or
         ("time interval1 is before  time interval2"  t1 t2)
          ("time interval1 properly overlaps   time interval2" t1 t2))
      (iff
          ("time interval1 plus time interval2  is time interval3" t1 t2 t3)
          (and
            ("time interval1 starts  time interval2"  t1 t3)
            ("time interval1 finishes  time interval2"  t2 t3))
)))
```

OCL Constraint:

```
context _'time interval'
inv: _'time interval'.allInstances->(forAll  t2 |
   (self._'is before'(t2)  or self._'properly overlaps'(t2)) implies
   (self.starts(self.plus(t2))  and t2.finishes(self.plus(t2))  )
```

Necessity: if a time interval[1] *is after* a time interval[2] or time interval[1] *is properly overlapped by* time interval[2], then time interval[1] *plus* time interval[2] *is started by* time interval[2] and *is finished by* time interval[1].

CLIF Axiom:
```
(forall (t1 t2 t3)
  (if
    (or
      ("time interval1 is before time interval2" t2 t1)
       ("time interval1 properly overlaps time interval2" t2 t1))
    (iff
       ("time interval1 plus time interval2 is time interval3" t1 t2 t3)
       (and
         ("time interval1 starts time interval2" t2 t3)
         ("time interval1 finishes time interval2" t1 t3))
)))
```

OCL Constraint:
```
context _'time interval'
inv: _'time interval'.allInstances->(forAll t2 |
  (t2._'is before'(self) or t2._'properly overlaps'(self)) implies
  (t2.starts(self.plus(t2)) and self.finishes(self.plus(t2)) )
```

Necessity: if a time interval$_1$ *is part of* a time interval$_2$, then time interval$_1$ *plus* time interval$_2$ *is* time interval$_2$.

CLIF Axiom:
```
(forall (t1 t2 t3)
  (if
    ("time interval1 is part of time interval2" t1 t2)
    (iff
       ("time interval1 plus time interval2 is time interval3" t1 t2 t3)
       (= t3 t2)
)))
```

OCL Constraint:
```
context _'time interval'
inv: _'time interval'.allInstances->(forAll t2 |
  (self._'is part of'(t2) implies self.plus(t2) = t2 )
```

Necessity: if a time interval$_2$ *is part of* a time interval$_1$, then time interval$_1$ *plus* time interval$_2$ *is* time interval$_1$.

CLIF Axiom:
```
(forall (t1 t2 t3)
  (if
    ("time interval1 is part of time interval2" t2 t1)
    (iff
       ("time interval1 plus time interval2 is time interval3" t1 t2 t3)
       (= t3 t1)
)))
```

OCL Constraint:
```
context _'time interval'
inv: _'time interval'.allInstances->(forall t2 |
  (t2._'is part of'(self) implies self.plus(t2) = self )
```

Example: January 2010 *through* December 2010 *is* 2010

**Axiom Sum:** For any time intervals t1 and t2, there is a time interval t3 that *is equal to* t1 *plus* t2.

Necessity: For each time interval$_1$ and each time interval$_2$, there is a time interval$_3$ that *is* time interval$_1$ *plus* time interval$_2$.

CLIF Axiom:
```
(forall ((t1 "time interval") (t2 "time interval"))
  (exists ((t3 "time interval"))
    ("time interval1 plus time interval2 is time interval3" t1 t2 t3) ))
```

OCL Constraint:
```
context _'time interval'
     inv: _'time interval'.allInstances->forAll(t2 |
     'time interval'.allInstances->exists(t3 |
     self._'time interval1 plus time interval2 is time interval3'(t2, t3)))
```

**Corollary**: For any two time intervals t1 and t2, t1+t2 is unique.

Necessity: A time interval$_1$ *plus* a time interval$_2$ *is* exactly one time interval$_3$.

CLIF Axiom:
```
(forall (t1 t2 t3)
  (if ("time interval1 plus time interval2 is time interval3" t1 t2 t3)
    (forall (t4)
      (if
        ("time interval1 plus time interval2 is time interval3" t1 t2 t4)
        (= t4 t3)) )) ))
```

OCL Constraint:
```
context _'time interval'
    inv: _'time interval'.allInstances->
        forAll(t2 |
          'time interval'.allInstances->
            one(t4 | t4 = self.plus(t2)))
```

### 8.2.6 Time Interval Complement

The following start-complement and end-complement verb concepts construct the complementary time interval given a time interval that starts or ends a larger time interval. Note that a complementary time interval does not exist in the case where one time interval *is properly during* another time interval.

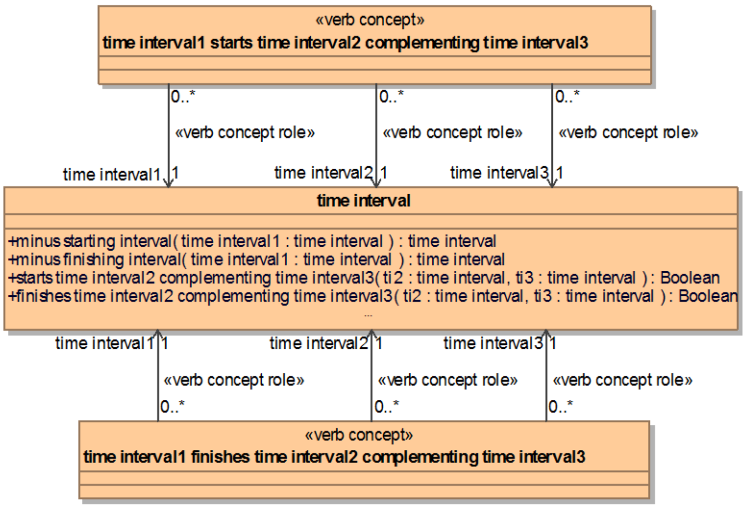


**Figure 8.9 - Time Interval Complement**

## time interval$_1$ *starts* time interval$_2$ *complementing* time interval$_3$

Definition: time interval$_1$ *starts* time interval$_2$ and time interval$_3$ *finishes* time interval$_2$ and time interval$_1$ *meets* time interval$_3$

CLIF Definition:
```
(forall (t1 t2 t3)
  (iff ("time interval1 starts time interval2 complementing time interval3" t1 t2 t3)
      (and
        ("time interval1 starts time interval2" t1 t2)
        ("time interval1 finishes time interval2" t3 t2)
        ("time interval1 meets time interval2" t1 t3)) ))
```

| | |
|---|---|
| OCL Definition: | context _'time interval'<br>def: _'starts time interval2 complementing time interval3'<br>(t2: _'time interval', t3: _'time interval'): Boolean =<br>self.starts(t2) and t3.finishes(t2) and self.meets(t3) |
| Example: | January 2010 *starts* 2010 *complementing* February 2010 *through* December 2010 |

**Axiom Start-complement:** If t1 and t2 are time intervals and t1 *starts* t2, then there is a time interval t3 such that t3 *finishes* t2 *complementing* t1.

| | |
|---|---|
| Necessity: | If a time interval$_1$ *starts* a time interval$_2$, then some time interval$_3$ *finishes* time interval$_2$ *complementing* time interval$_1$. |
| CLIF Axiom: | (forall (t1 t2)<br>(if ("time interval1 starts time interval2" t1 t2)<br>(exists (t3)<br>("time interval1 finishes time interval2 complementing time interval3" t3 t2 t1) ))) |
| OCL Constraint: | context 'time interval'<br>inv: _'time interval'.allInstances->forAll(t2 \|<br>self.starts(t2) implies<br>'time interval'.allInstances->exists(t3 \|<br>t3._'finishes time interval2 complementing time interval3'<br>(t2, self))) |
| Note: | This formalizes the axiom above: If a time interval1 starts a time interval2, there is a time interval3 that is the start complement. |

**Corollary:** For all time intervals t1, t2 and t3, such that t1 *starts* t2 *complementing* t3, and for all time intervals t4, such that t4 *is part of* t2 and t4 does not *overlap* t1, t4 *is part of* t3. That is, t3 is the largest time interval that *is part of* t2 but does not *overlap* t1.

| | |
|---|---|
| Necessity: | If a time interval$_1$ *starts* a time interval$_2$ *complementing* a time interval$_3$, then each time interval$_4$ that *is part of* the time interval$_2$ and that *does* not *overlap* the time interval$_1$ *is part of* the time interval$_3$. |
| CLIF Axiom: | (forall (t1 t2 t3)<br>(if ("time interval1 starts time interval2 complementing time interval3" t1 t2 t3)<br>(forall (t4)<br>(if<br>(and<br>("time interval1 is part of time interval2" t4 t2)<br>(not ("time interval1 overlaps time interval2" t4 t1)))<br>("time interval1 is part of time interval2" t4 t3))))) |
| OCL Constraint: | context _'time interval'<br>inv: _'time interval'.allInstances-><br>forAll(t2, t3, t4 \|<br>(t3 = t2._'minus starting interval'(self)<br>and (t4._'is part of'(t2)<br>and not t4.overlaps(self))<br>implies t4._'is part of'(t3))) |

**Corollary:** For any two time intervals t1 and t2 such that t1 *starts* t2 *complementing* some time interval t3, t3 is unique.

| | |
|---|---|
| Necessity: | If a time interval$_1$ *starts* a time interval$_2$ then the time interval$_1$ *starts* the time interval$_2$ *complementing* exactly one time interval$_3$. |
| CLIF Axiom: | (forall (t1 t2 t3)<br>(if ("time interval1 starts time interval2 complementing time interval3" t1 t2 t3)<br>(forall (t4)<br>(if ("time interval1 starts time interval2 complementing time interval3" t1 t2 t4)<br>(= t4 t3))) )) |

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances ->
          forAll(t2 |
           'time interval'.allInstances ->
            isUnique(t2._'minus starting interval'(self))
```

## time interval$_1$ *finishes* time interval$_2$ *complementing* time interval$_3$

Definition: time interval$_1$ *finishes* time interval$_2$ and time interval$_3$ *starts* time interval$_2$ and time interval$_1$ *is met by* time interval$_3$

CLIF Definition:
```
(forall (t1 t2 t3)
  (iff ("time interval1 finishes time interval2 complementing time interval3" t1 t2 t3)
       (and
          ("time interval1 finishes time interval2" t1 t2)
          ("time interval1 starts time interval2" t3 t2)
          ("time interval1 meets time interval2" t3 t1)) ))
```

OCL Definition:
```
context _'time interval'
def: _'finishes time interval2 complementing time interval3'
      (t2: _'time interval', t3: _'time interval'): Boolean =
  self.finishes(t2) and t3.starts(t2) and t3.meets(self)
```

Example: December 2010 *finishes* 2010 *complementing* January 2010 *through* February 2010

**Axiom End-complement**: If t1 and t2 are time intervals and t1 *finishes* t2, then there is a time interval t3 such that t3 *starts* t2 *complementing* t1.

Necessity: If a time interval$_1$ *finishes* a time interval$_2$, then some time interval$_3$ *starts* time interval$_2$ *complementing* time interval$_1$.

CLIF Axiom:
```
(forall (t1 t2)
  (if ("time interval1 finishes time interval2" t1 t2)
      (exists (t3)
        ("time interval1 starts time interval2 complementing time interval3" t3 t2 t1) )))
```

OCL Constraint:
```
context 'time interval'
      inv: _'time interval'.allInstances->forAll(t2 |
      self.finishes(t2) implies
      'time interval'.allInstances->exists(t3 |
      t3._'starts time interval2 complementing time interval3'
          (t2, self)))
```

Note: This formalizes the axiom End-complement above: If a time interval1 finishes a time interval2, there is a time interval3 that is the end complement.

**Corollary:** For all time intervals t1, t2 and t3, such that t1 *finishes* t2 *complementing* t3, and for all time intervals t4, such that t4 *is part of* t2 and t4 does not *overlap* t1, t4 *is part of* t3. That is, t3 is the largest time interval that *is part of* t2 but does not *overlap* t1.

Necessity: If a time interval$_1$ *finishes* a time interval$_2$ *complementing* a time interval$_3$, then each time interval$_4$ that *is part of* the time interval$_2$ and that *does* not *overlap* the time interval$_1$ *is part of* the time interval$_3$.

CLIF Axiom:
```
(forall (t1 t2 t3)
  (if ("time interval1 finishes time interval2 complementing time interval3" t1 t2 t3)
     (forall (t4)
       (if
         (and
            ("time interval1 is part of time interval2" t4 t2)
            (not ("time interval1 overlaps time interval2" t4 t1)))
          ("time interval1 is part of time interval2" t4 t3)))))
```

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances->
```

```
forAll(t2, t3, t4 |
(t3 = t2._'minus finishing interval'(self)
and (t4._'is part of'(t2)
and not t4.overlaps(self))
implies t4_.'is part of'(t3)))
```

**Corollary:** For any two time intervals t1 and t2 such that t1 *finishes* t2 *complementing* some time interval t3, t3 is unique.

Necessity: If a time interval$_1$ *finishes* a time interval$_2$ then the time interval$_1$ *finishes* the time interval$_2$ *complementing* exactly one time interval$_3$.

CLIF Axiom:
```
(forall (t1 t2 t3)
  (if ("time interval1 finishes time interval2 complementing time interval3" t1 t2 t3)
    (forall (t4)
      (if ("time interval1 finishes time interval2 complementing time interval3" t1 t2 t4)
         (= t4 t3))) ))
```

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances ->
          forAll(t2 |
           'time interval'.allInstances ->
            isUnique(t2._'minus finishing interval'(self))
```

**Axiom:** For any time intervals t1 and t2 such that t2 *is properly during* t1, t2 has both a start complement in t1 and an end complement in t1.

Necessity: For each time interval$_1$ and each time interval$_2$ that *is properly during* time interval$_1$, there is a time interval$_3$ that ***starts*** time interval$_1$ and ***meets*** time interval$_2$.

Necessity: For each time interval$_1$ and each time interval$_2$ that *is properly during* time interval$_1$, there is a time interval$_4$ that *finishes* time interval$_1$ and *is met by* time interval$_2$.

CLIF Axiom:
```
(forall ((ti1 "time interval")
        (ti2 "time interval"))
  (if ("time interval1 is properly during  time interval2" t2 t1)
   (exists (ti3 "time interval")
    (and
     ("time interval1 starts time interval2" ti3 ti1)
     ("time interval1 meets time interval2" ti3 ti2)) )) ))
```

CLIF Axiom:
```
(forall ((ti1 "time interval")
        (ti2 "time interval"))
  (if ("time interval1 is properly during time interval2" t2 t1)
   (exists (ti4 "time interval")
    (and
     ("time interval1 finishes time interval2" ti4 ti1)
     ("time interval1 meets time interval2" ti2 ti4)) )) ))
```

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances->
           forAll(t2 | t2._'is properly during'(self)
           implies _'time interval'.allInstances
            -> exists(t3 | t3.starts(self) and t3.meets(t2)))
```

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances->
            forAll(t2 | t2._'is properly during'(self)
             implies _'time interval'.
                 allInstances-> exists(t3 | t3.ends(self) and t2.meets(t3)))
```

Necessity: For each time interval$_1$ at least one time interval$_2$ *starts* time interval$_1$.

Necessity: For each time interval$_1$ at least one time interval$_2$ *finishes* time interval$_1$.

### 8.2.7 Time Interval Intersection

This verb concept generates the intersection of two overlapping **time intervals**.

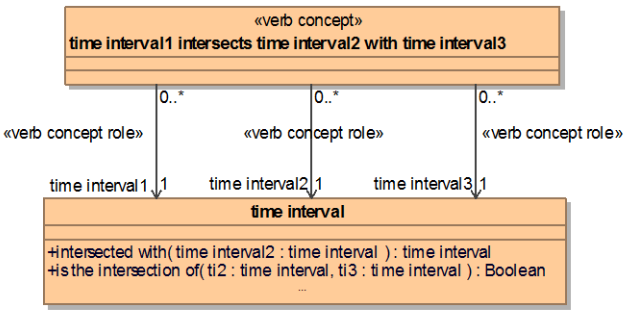


**Figure 8.10 - Time Interval Intersection**

## time interval$_1$ *intersects* time interval$_2$ *with* time interval$_3$

Synonymous Form: time interval$_1$ *is the intersection of* time interval$_2$ *with* time interval$_3$

Synonymous Form: *intersection of* time interval$_2$ *with* time interval$_3$

Definition: time interval$_1$ *is part of* time interval$_2$ and time interval$_1$ *is part of* time interval$_3$ and time interval$_1$ *includes* each time interval that *is part of* time interval$_2$ and time interval$_3$

CLIF Definition:

```
(forall (t1 t2 t3)
  (iff
    ("time interval1 intersects time interval2 with time interval3"
           t1 t2 t3)
    (and
        ("thing1 is part of thing2" t1 t2)
        ("thing1 is part of thing2" t1 t3)
        (forall (t4)
            (if (and
                    ("thing1 is part of thing2" t4 t2)
                    ("thing1 is part of thing2" t4 t3))
                ("thing1 is part of thing2" t4 t1) ))
)))
```

OCL Definition:

```
context _'time interval'
def: _'is intersection of '(t2: _'time interval', t3: _'time interval'):
          Boolean =
    self._'is part of'(t2) and self._'is part of'(t3) and
    _'time interval'.allInstances->
        forAll(t4 | (t4._'is part of'(t2) and t4._'is part of'(t3))
            implies t4._'is part of'(self))
```

Note: The alternative definitions describe construction of the intersection. Technically, these are corollaries to the Definition

Definition: if time interval$_2$ *is part of* time interval$_3$, then time interval$_1$ *equals* time interval$_3$, and
if time interval$_3$ *is part of* time interval$_2$, then time interval$_1$ *equals* time interval$_2$, and
if time interval$_2$ *properly overlaps* time interval$_3$, then time interval$_1$ *finishes* time interval$_2$ and time interval$_1$ *starts* time interval$_3$, and
if time interval$_3$ *properly overlaps* time interval$_2$, then time interval$_1$ *finishes* time interval$_3$ and time interval$_1$ *starts* time interval$_2$

CLIF Definition:

```
(forall (t1 t2 t3)
  (iff
    ("time interval1 intersects time interval2 with time interval3"
         t1 t2 t3)
    (and
       (if ("thing1 is part of thing2" t2 t3) (=t1 t2))
       (if ("thing1 is part of thing2" t3 t2) (=t1 t3))
       (if ("time interval1 properly overlaps time interval2" t2 t3)
          (and
            ("time interval1 finishes time interval2" t1 t2)
            ("time interval1 starts time interval2" t1 t3)) )
       (if ("time interval1 properly overlaps time interval2" t3 t2)
          (and
            ("time interval1 finishes time interval2" t1 t3)
            ("time interval1 starts time interval2" t1 t2)) )
)))
```

OCL Definition:

```
context _'time interval'
def: _'is intersection of '(t2: _'time interval', t3: _'time interval'):
          Boolean =
    (t2.includes(t3) implies self.equals(t3)) and
    (t3.includes(t2) implies self.equals(t2)) and
    (t2._'properly overlaps'(t3) implies
       self.finishes(t2) and self.starts(t3) ) and
    (t3._'properly overlaps'(t2) implies
       self.finishes(t3) and self.starts(t2) )
```

Example: January 2010 *through* June 2010 *intersects* March 2010 *through* August 2010 *with* March 2010 *through* June 2010

**Axiom Intersection**: For any time intervals t1 and t2 such that t1 *overlaps* t2, there is a time interval t1*t2 that *intersects* t1 *with* t2.

Necessity: If a time interval$_1$ *overlaps* a time interval$_2$, then some time interval$_3$ *intersects* time interval$_1$ *with* time interval$_2$.

CLIF Axiom:

```
(forall (t1 t2)
  (if
    ("time interval1 overlaps time interval2" t1 t2)
    (exists (t3)
       ("time interval1 intersects time interval2 with time interval3" t3 t1 t2) )))
```

OCL Constraint:

```
context 'time interval'
    inv: _'time interval'.allInstances->forAll(t2 |
    self.overlaps(t2) implies
    'time interval'.allInstances->exists(t3 |
      t3._'is intersection of'(self, t2)))
```

**Corollary**: For all time intervals t1, t2, and t4, such that t1 *overlaps* t2 and t4 *is part of* t1 and t4 *is part of* t2, t4 *is a part of* t1*t2. That is, t1*t2 is the largest time interval that *is part of* t1 and *part of* t2.

Necessity: If a time interval$_1$ *intersects* a time interval$_2$ *with* a time interval$_3$ and a time interval$_4$ *is part of* the time interval$_1$ and the time interval$_4$ *is part of* the time interval$_2$, then the time interval$_4$ *is part of* the time interval$_3$.

CLIF Axiom:

```
(forall (t1 t2 t3 t4)
  (if
    (and
      ("time interval1 intersects time interval2 with time interval3" t1 t2 t3)
      ("time interval1 is part of time interval2" t4 t2)
      ("time interval1 is part of time interval2" t4 t1))
    ("time interval1 is part of time interval2" t4 t3) ))
```

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances->
           forAll(t2, t3, t4 |
              (self.overlaps(t2) and t4._'is part of'(self) and t4_.'is part of'(t2))
             implies
              t4._'is part of'(self._'intersected with'(t2)))
```

**Corollary**: For any two time intervals t1 and t2 such that t1 *overlaps* t2, t1*t2 is unique.

Necessity: If a time interval$_1$ *overlaps* a time interval$_2$, then the time interval$_1$ *intersects* a time interval$_2$ *with* exactly one time interval$_3$.

CLIF Axiom:
```
(forall (t1 t2 t3)
  (if
    ("time interval1 intersects time interval2 with time interval3" t3 t1 t2)
    (forall (t4)
      (if
        ("time interval1 intersects time interval2 with time interval3" t4 t1 t2)
         (= t4 t3))) ))
```

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances->forAll(t2 |
           self.overlaps(t2) implies
           'time interval'.allInstances->
           isUnique(self._'intersected with'(t2)))
```

**Corollary** (Intervening): For all time intervals t1 and t2 such that t1 *is properly before* t2, there is a unique time interval t3 such that t1 *meets* t3 and t3 *meets* t2. The intervening time interval t3 is the *intersection* of the start-complement (t5) of t1+t2 (t4), and the end-complement of t1+t2 (t4).

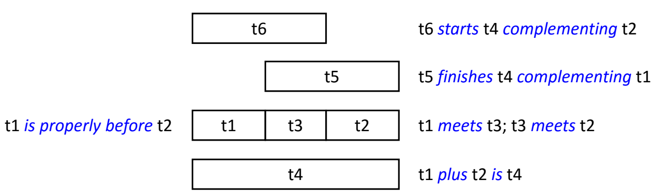


**Figure 8.11 - Illustration of 'Intervening' Corollary**

Necessity: If a time interval$_1$ *is properly before* a time interval$_2$ then the time interval$_1$ *meets* a time interval$_3$ and the time interval$_3$ *meets* the time interval$_2$ and the time interval$_1$ *plus* the time interval$_2$ *is* a time interval$_4$ and the time interval$_1$ *starts* the time interval$_4$ *complementing* a time interval$_5$ and the time interval$_2$ *finishes* the time interval$_4$ *complementing* a time interval$_6$ and the time interval$_5$ *intersects* the time interval$_6$ *with* the time interval$_3$.

CLIF Axiom:
```
(forall (t1 t2)
  (if ("time interval1 is properly before time interval2" t1 t2)
      (exists (t3 t4 t5 t6)
        (and
          ("time interval1 meets time interval2" t1 t3)
          ("time interval1 meets time interval2" t3 t2)
          ("time interval1 plus time interval2 is time interval3" t1 t2 t4)
          ("time interval1 finishes time interval2 complementing time interval3" t5 t4 t1)
          ("time interval1 starts time interval2 complementing time interval3" t6 t4 t2)
          ("time interval1 intersects time interval2 with time interval3" t3 t6 t5))) ))
```

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances->
          forAll(t2|
```

```
self._'is properly before'(t2)
implies
'time interval'.allInstances->
  exists(t3, t4, t5, t6|
   self.meets(t3)
   and t3.meets(t2)
   and t4 = self.plus(t2)
   and t5 = t4._'minus starting interval'(self)
   and t6 = t4._'minus finishing interval(t2)
    and t3 = t5._'intersected with''(t6)))
```

### 8.2.8 Time intervals defined by start and end

The above sections specify mathematical means of defining a time interval as the sum, complement, or intersection of two other time intervals. In practice, a time interval is more commonly defined by specifying when it starts and when it ends. This section introduces two verb concepts that support such a mechanism.

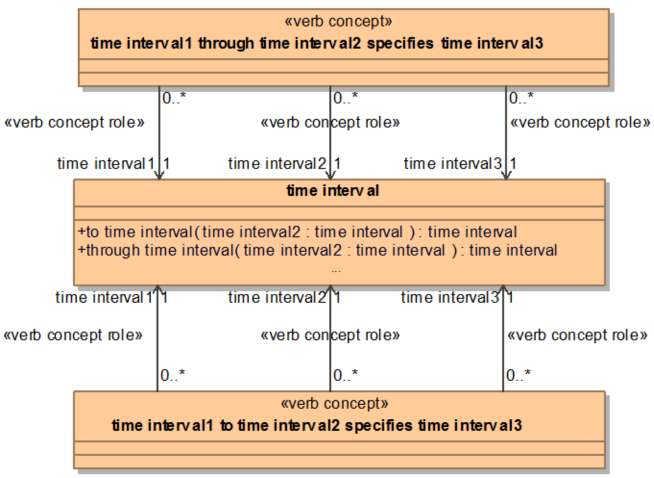


**Figure 8.12 - Time intervals defined by start and end**

**time interval$_1$ *through* time interval$_2$ *specifies* time interval$_3$**

Synonymous Form: time interval$_1$ *through* time interval$_2$ *is* time interval$_3$

Synonymous Form: time interval$_3$ *is from* time interval$_1$ *through* time interval$_2$

Definition: time interval$_1$ *starts before* time interval$_2$, and
time interval$_1$ *starts* time interval$_3$, and
time interval$_2$ *finishes* time interval$_3$

CLIF Definition:

```
(forall (t1 t2 t3)
  (iff
    ("time interval1 through time interval2 specifies  time interval3" t1 t2 t3)
    (and
      ("time interval1 starts before  time interval2" t1 t2)
      ("time interval1 starts time interval2" t1 t3)
      ("time interval1 finishes time interval2" t2 t3)
)))
```

OCL Definition:
```
context _'time interval'
    def: _'through time interval2 is time interval3'
         (t2: _'time interval', t3: _'time interval'): Boolean =
       self._'starts before'(t2) and
       self.starts(t3) and t2.finishes(t3)
```

Synonymous Form: time interval$_1$ *through* time interval$_2$

Note: This is a noun form of the verb concept. It refers to the specified time interval.

CLIF Definition:
```
(forall (t1 t2 t3)
    (iff (= t3 ("time interval1 through time interval2" t1 t2))
    ("time interval1 through time interval2 specifies time interval3" t1 t2 t3) ))
```

OCL Definition:
```
context _'time interval'
    def: _'through time interval' (t2: _'time interval'): _'time interval' =
      _'time interval'.allInstances->(t3 |
        self.starts(t3) and t2.finishes(t3))
```

Example: The time interval that *is from* 2006 *through* 2007 has duration 2 years.

Necessity: For each time interval$_1$ that *starts before* a given time interval$_2$, exactly one time interval$_3$ *is* time interval$_1$ *through* time interval$_2$.

Note: This follows from the definition.

CLIF Axiom:
```
(forall (t1 t2)
  (if ("time interval1 starts before time interval2" t1 t2)
     (exists (t3)
       ("time interval1 is time interval2 through time interval3"
          t3 t1 t2) )))
```

CLIF Axiom:
```
(forall (t1 t2 t3 t4)
  (if (and
       ("time interval1 is time interval2 through time interval3"
          t3 t1 t2)
       ("time interval1 is time interval2 through time interval3"
          t4 t1 t2) )
     (= t3 t4) ))
```

OCL Constraint:
```
context _'time interval'
    inv: _'time interval'.allInstances->
       forAll(t2 |
         self._'starts before'(t2) implies
         _'time interval'.allInstances->
              one(t3 | t3 = self._'through time interval'(t2)) )
```

## time interval$_1$ *to* time interval$_2$ *specifies* time interval$_3$

Synonymous Form: time interval$_1$ *to* time interval$_2$ *is* time interval$_3$

Synonymous Form: time interval$_3$ *is from* time interval$_1$ *to* time interval$_2$

Synonymous Form: time interval$_3$ *is from* time interval$_1$ *until* time interval$_2$

Definition: time interval$_1$ *is before* time interval$_2$, and time interval$_3$ *is* time interval$_1$ if time interval$_1$ *meets* time interval$_2$, and time interval$_3$ *is* the time interval that *meets* time interval$_2$ and *is started by* time interval$_1$ if time interval$_1$ *is properly before* time interval$_2$

CLIF Definition:
```
(forall (t1 t2 t3)
  (iff
   ("time interval1 to time interval2 specifies time interval3" t1 t2 t3)
   (and
    ("time interval1 is before time interval2" t1 t2)
    (if ("time interval1 meets time interval2" t1 t2)
        (= t1 t3))
    (if
     ("time interval1 is properly before time interval2" t1 t2)
     (and
        ("time interval1 starts time interval2" t1 t3)
        ("time interval1 meets time interval2" t3 t2)
     ))
)))
```

OCL Definition:
```
context _'time interval'
      def: _'to time interval2 is time interval3'
           (t2: _'time interval', t3: _'time interval'): Boolean =
          self._'is before'(t2) and
          (if self.meets(t2) then t3 = self
              else self.starts(t3) and t3.meets(t2))
```

Synonymous Form: time interval$_1$ *to* time interval$_2$

Note: This is a noun form of the verb concept. It refers to the specified time interval.

CLIF Definition:
```
(forall (t1 t2 t3)
      (iff (= t3 ("time interval1 to time interval2" t1 t2))
      ("time interval1 to time interval2 specifies time interval3"
          t1 t2 t3) ))
```

OCL Definition:
```
context _'time interval'
      def: _'to time interval' (t2: _'time interval'): _'time interval' =
         if (not (self._'is before(t2)) then null
         else if (self.meets(t2)) then self
         else _'time interval'.allInstances->
               forall(t3 | t3.meets(t2)and self.starts(t3))
```

Note: Contrast '*through*' with '*to*.' '*through*' is inclusive of time interval$_2$, while '*to*' is exclusive of time interval$_2$.

Example: The time interval "2006" *to* "2007" has duration 1 year.

Necessity: For each time interval$_1$ that *is before* a given time interval$_2$, exactly one time interval$_3$ *is* time interval$_1$ *to* time interval$_2$.

Note: This follows from the definition.

CLIF Axiom:
```
(forall (t1 t2)
  (if ("time interval1 is before time interval2" t1 t2)
     (exists (t3)
      ("time interval1 is time interval2 to time interval3"
          t3 t1 t2) )))
```

CLIF Axiom:
```
(forall (t1 t2 t3 t4)
  (if (and
      ("time interval1 is time interval2 to time interval3"
          t3 t1 t2)
      ("time interval1 is time interval2 to time interval3"
          t4 t1 t2) )
    (= t3 t4) ))
```

OCL Constraint:
```
context _'time interval'
      inv: _'time interval'.allInstances->
         forAll(t2 |
           self.before(t2) implies
           _'time interval'.allInstances->
               one(t3 | t3 = self._'to time interval'(t2)) )
```

### 8.2.9 Indefinite time intervals

Indefinite time intervals provide the basis for describing time intervals that extend indefinitely into the past or the future. One example is a British bond of the 1910s that pays interest "in perpetuity."

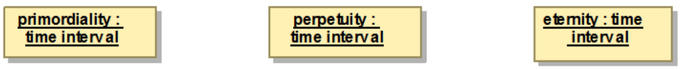


**Figure 8.13 - primordiality, perpetuity, and eternity**

## eternity

Synonym: forever

Definition: the time interval that *includes* each time interval

CLIF Definition:
```
(forall (t) (iff (= t eternity)
    (and
      ("time interval" t)
      (forall (ti2) (iff ("time interval" ti2) ("is part of" ti2 t) ))
)))
```

OCL Constraint:
```
context _'time interval'
    inv: self.'is part of'(eternity)
```

Description: The time interval that extends across the entire Time Axis.

Note: eternity is an individual concept because there can be only one such time interval.

Note: eternity is not the same thing as the Time Axis, even though it 'covers' the Time Axis.

## primordiality

Definition: the time interval that *is before* each time interval that *is* not primordiality or eternity.

Description: The time interval that is at the beginning of time, or at least so far back in time that it is before all interesting time intervals.

CLIF Definition:
```
(forall (t)
  (iff (= t primordiality)
       (and
         ("time interval" t)
         (forall (ti2) (or
           (= ti2 primordiality)
           (= ti2 eternity)
           ("time interval1 is before time interval2" t ti2) )))))
```

OCL Constraint:
```
context 'time interval'
    inv: self = primordiality or self = eternity
        or primordiality._'is before'(self)
```

Note: primordiality is an individual concept. There can be only one time interval that *is before* every other time interval.

Note: This concept can be used in formulations such as "primordiality *through* current day" to define time intervals that began at some indefinite time in the past. Tools may choose to support a convenient syntax such as "until today".

Example: "primordiality *to* 2005" meaning "until 2005".

Note: primordiality has a duration but it is not known.

Necessity: primordiality *starts* eternity.

Note: This follows from the definitions. No part of eternity can be before primordiality.

## perpetuity

Definition: the time interval that *is after* each time interval that *is* not perpetuity or eternity.

| | |
|---|---|
| Description: | The time interval that is at the end of time, or at least so far forward in time that it is after all interesting time intervals. |
| CLIF Definition: | (forall (t)<br>(iff (= t perpetuity)<br>(and<br>("time interval" t)<br>(forall (ti2) (or<br>(= ti2 perpetuity)<br>(= ti2 eternity)<br>("time interval1 is before time interval2" ti2 t) ))))) |
| OCL Constraint: | context 'time interval'<br>inv: self = perpetuity or self = eternity<br>or self._'is before'(perpetuity) |
| Note: | perpetuity is an individual concept. There can be only one time interval. that *is after* every other time interval. |
| Note: | This concept can be used in formulations such as "2012 *through* perpetuity" to define time intervals that extend indefinitely into the future. Tools may choose to support a convenient syntax such as "after 2012". |
| Example: | "Contract signing *through* perpetuity" meaning "after the contract signing". |
| Note: | perpetuity has a duration but it is not known. |
| Necessity: | perpetuity *finishes* eternity. |
| Note: | This follows from the definitions. No part of eternity can be after perpetuity. |

## 8.3 Durations

A second foundational temporal concept is 'duration,' the amount of time in a time interval. This clause presents various properties of 'duration' and of the relationship between 'duration' and 'time interval'.

### duration

| | |
|---|---|
| Synonym: | time |
| Definition: | base quantity of the International System of Quantities, used for measuring time intervals |
| Note: | Duration is a quantity kind, whose instances are quantities of time. Each duration is an equivalence class of particular durations: a duration equals all the measurements for the same amount of time. |
| Note: | 'Duration' is a different concept from 'duration value'. 'Duration' is the amount of time in a time interval. 'Duration value' is a quantification of 'duration' in terms of a time unit. There is a one-to-many relationship between durations and duration values. For example, the same duration may be quantified as any of the duration values "1 hour", or "60 minutes", or "3600 seconds". |
| Reference Scheme: | a precise atomic duration value that *quantifies* the duration |
| Source: | [ISO/IEC 80000-3] |

### 8.3.1 Duration Ordering

'Duration' has relationships, '=', '≤', and '<' with the following properties. These relationships neither follow from nor entail the duration properties defined in the next clause.The four axioms defined in this section, taken together, define a total ordering on 'duration'.

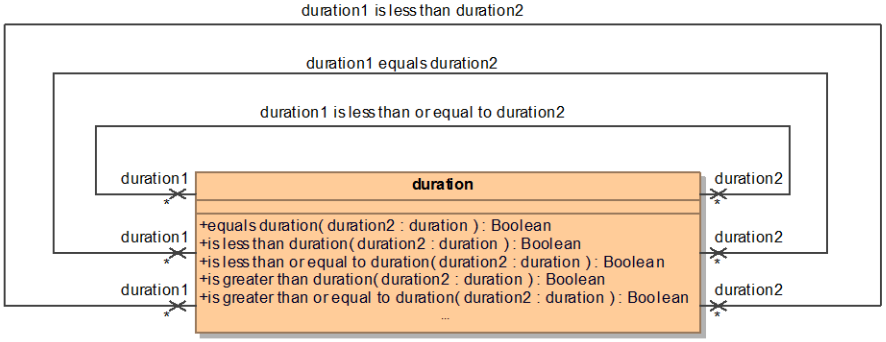


**Figure 8.14 - Duration Ordering**

## duration$_1$ *is less than or equal to* duration$_2$

| | |
|---|---|
| Synonymous Form: | duration$_1$ ≤ duration$_2$ |
| Synonymous Form: | duration$_2$ ≥ duration$_1$ |
| Synonymous Form: | duration$_2$ *is greater than or equal to* duration$_1$ |
| Definition: | A total ordering on quantities of time. |
| Note: | This is a primitive concept. |
| Example: | Two runners start a race at the same time. The duration of the run of one runner is less than or equal to the duration of the run of the other runner. |

## duration$_1$ *equals* duration$_2$

| | |
|---|---|
| Synonymous Form: | duration$_1$ = duration$_2$ |
| Definition: | duration$_1$ ≤ duration$_2$ and duration$_2$ ≤ duration$_1$ |
| Example: | Two runners start and complete a race at the same time. The duration of the run of one runner is equal to the duration of the run of the other runner. |

Axiom O.1 (Reflexive): If d1 is a duration, then d1 ≤ d1.

Necessity: Each duration ≤ the duration.

CLIF Axiom:
```
(forall ((d1 duration))
  ("duration ≤ duration" d1 d1))
```

OCL Constraint:
```
context duration
    inv: self._'is less or equal'(self))
```

Axiom O.2 (Total): If d1 and d2 are durations, then either d1 ≤ d2 or d2 ≤ d1.

Necessity: Each duration$_1$ ≤ each duration$_2$ or duration$_2$ ≤ duration$_1$.

CLIF Axiom:
```
(forall ((d1 duration) (d2 duration))
  (or
    ("duration ≤ duration" d1 d2)
    ("duration ≤  duration" d2 d1)))
```

OCL Constraint:
```
context duration
    inv: duration.allInstances->forAll(d2 |
        self._'is less or equal(d2)
        or d2._'is less or equal'(self)
```

Axiom O.3 (Antisymmetric): If d1 and d2 are durations, and d1 ≤ d2 and d2 ≤ d1, then d1 = d2.

Necessity: If some duration$_1$ ≤ some duration$_2$ and the duration$_2$ ≤ the duration$_1$, then the duration$_1$ *equals* the duration$_2$.

CLIF Axiom:
```
(forall ((d1 duration) (d2 duration))
 (if (and
      ("duration ≤ duration" d1 d2)
      ("duration ≤ duration" d2 d1))
      (= d1 d2)))
```
OCL Constraint:
```
context duration
     inv: duration.allInstances->forAll(d2 |
         self._'is less or equal'(d2)
         and d2._'is less or equal'(self)
         implies self = d2)
```

Axiom O.4 (Transitive): If d1, d2, d3 are durations, and d1 ≤ d2 and d2 ≤ d3, then d1 ≤ d3.

Necessity: If some $duration_1$ ≤ some $duration_2$ and the $duration_2$ ≤ the $duration_3$ then the $duration_1$ ≤ the $duration_3$.

CLIF Axiom:
```
(forall ((d1 duration) (d2 duration) (d3 duration))
  (if (and
        ("duration ≤ duration" d1 d2)
        ("duration ≤ duration" d2 d3))
     ("duration ≤ duration" d1 d3)))
```
OCL Constraint:
```
context duration
     inv: duration.allInstances->forAll(d2, d3 |
         self._'is less or equal'(d2)
         and d2._'is less or equal(d3)
         implies self._'is less or equal'(d3))
```

**Corollary** (Equals is transitive): If d1, d2, d3 are durations, and d1 = d2 and d2 = d3, then d1 = d3.

Necessity: If some $duration_1$ = some $duration_2$ and the $duration_2$ = some $duration_3$ then the $duration_1$ = the $duration_3$.

CLIF Axiom:
```
(forall (d1 d2 d3)
  (if (and
        ("duration = duration" d1 d2)
        ("duration = duration" d2 d3))
     ("duration = duration" d1 d3)))
```
OCL Constraint:
```
context duration
     inv: duration.allInstances->forAll(d2, d3 |
         self._equals(d2) and d2.equals(d3)
         implies self.equals(d3))
```

## $duration_1$ *is less than* $duration_2$

Synonymous Form: $duration_1$ < $duration_2$

Synonymous Form: $duration_2$ > $duration_1$

Synonymous Form: $duration_2$ *is greater than* $duration_1$

Definition: $duration_1$ ≤ $duration_2$ and $duration_1$ *does* not *equal* $duration_2$

Example: Two runners start a race at the same time. The duration of the run of the first runner to cross the finish line is less than the duration of the run of the other runner.

CLIF Definition:
```
(forall ((d1 duration) (d2 duration))
  (iff ("duration < duration" d1 d2)
      (and
       ("duration ≤ duration" d1 d2)
       (not (= d2 d1)))))
```
OCL Definition:
```
context duration
def: _'is less than'(d2: duration): Boolean =
    self._'is less or equal'(d2)
    and not self.equals(d2)
```

### 8.3.2 Duration Operations

From a mathematical point of view, the extension of 'duration' is a vector space over the real numbers. That is, two operations – addition and scalar multiplication – are defined on durations. They operations obey the following axioms:

**Axiom V.1** (Addition is Closed): If d1 and d2 are durations, then d1 + d2 is a duration.

**Axiom V.2** (Addition is Associative): If d1, d2, d3 are durations, then (d1 + d2) + d3 = d1 + (d2 + d3).

**Axiom V.3** (Addition is Commutative): If d1 and d2 are durations, then d1 + d2 = d2 + d1.

**Axiom V.4** (Additive Identity): There is a duration D0 such that, for every duration d1, d1 + D0 = d1.

**Axiom V.5** (Additive Inverse): For each duration d1, there is a duration d2, such that d1 + d2 = D0.

Note: The existence of the inverse (-d1) is a mathematical necessity for the vector space. Whether it has physical meaning is quite another thing entirely.

**Axiom V.6** (Scalar multiplication is closed): if d1 is a duration and n1 is a number, n1 * d1 is a duration.

**Axiom V.7** (Scalar multiplication is distributive over durations): if d1 and d2 are durations and n1 is a real number, n1 * (d1 + d2) = (n1 * d) + (n1 * d2)

**Axiom V.8** (Scalar multiplication is distributive over reals): if d1 is a duration, and n1 and n2 are numbers, (n1 + n2) * d1 = n1 * d1 + n2 * d1.

**Corollary**: For all durations d1, 0 * d1 = D0

**Corollary**: If n1 is a number and d1 is a duration, then n1 * d1 = D0 iff n1 = 0 or d1 = D0

**Corollary (Ratio)**: If d1 and d2 are durations and not d2 = D0, then there exists a number n1 such that d2 = n1 * d1. We call n1 the "ratio of d2 to d1."

Note that the above does not depend on the concept 'time unit.' In fact, the usefulness of 'time unit' depends on this property.

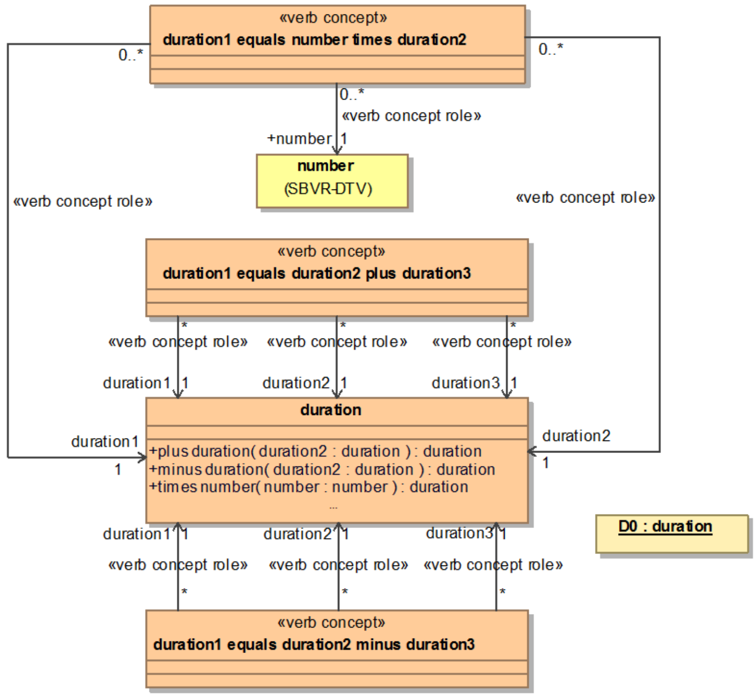


**Figure 8.15 - Duration Operations**

## duration$_3$ *equals* duration$_1$ *plus* duration$_2$

| | |
|---|---|
| Synonymous Form: | duration$_3$ = duration$_1$ *plus* duration$_2$ |
| Synonymous Form: | duration$_1$ *plus* duration$_2$ *gives* duration$_3$ |
| Synonymous Form: | duration$_1$ + duration$_2$ *gives* duration$_3$ |
| Synonymous Form: | duration$_1$ *plus* duration$_2$ |
| Synonymous Form: | duration$_1$ + duration$_2$ |
| Note: | This is a “ground concept” that cannot be defined in terms of other concepts. |
| Example: | Some race consists of a run and a swim. For each racer, the duration of the race is the duration of the run *plus* the duration of the swim. |
| Note: | The following definition defines the CLIF duration addition function in terms of the verb concept. The verb concept is primitive and has no formal definition. |

CLIF Definition:

```
(forall ((d1 duration) (d2 duration) d3)
  (iff
    (= d3 (+ d1 d2))
    (and
      (duration d3)
      ("duration3 = duration1 + duration2" d3 d1 d2) )))
```

Axiom V.1 (Addition is closed): For all durations d1 and d2, there is a duration d3 such that d3 = d1 + d2.

Necessity: For each duration$_1$ and each duration$_2$ some duration$_3$ *equals* duration$_1$ *plus* duration$_2$.

| | |
|---|---|
| CLIF Axiom: | (forall ((d1 duration) (d2 duration))<br>(exists (d3 duration)<br>(= d3 (+ d1 d2)))) |
| OCL Constraint: | OCL Constraint: context duration<br>inv: duration->allInstances(forAll d2 \|<br>duration->allInstances(exists d3 \| d3 = self.plus(d2)) ) |

Corollary: The sum of two durations is unique.

| | |
|---|---|
| Necessity: | For each duration1 and each duration2 exactly one duration3 *equals* duration1 *plus* duration2. |

This follows from the transitivity of equality of durations in 8.3.1.

Axiom V.2 (Addition is Associative): If d1, d2, d3 are durations, then (d1 + d2) + d3 = d1 + (d2 + d3).

| | |
|---|---|
| Necessity: | If a duration4 *equals* a duration1 *plus* a duration2, and a duration5 *equals* duration4 *plus* duration3, and a duration6 *equals* duration2 *plus* duration3, then duration5 *equals* duration1 *plus* duration6. |
| CLIF Axiom: | (forall ((d1 duration) (d2 duration) (d3 duration))<br>(= (+ (+ d1 d2) d3) (+ d1 (+ d2 d3))) ) |
| OCL Constraint: | context duration<br>inv: duration.allInstances-><br>forAll(d2, d3 \|<br>(self._'plus duration'(d2)<br>._'plus duration'(d3))<br>.equals(self._'plus duration'<br>(d2._'plus duration'(d3)))) |

Axiom V.3 (Addition is Commutative): If d1 and d2 are durations, then d1 + d2 = d2 + d1.

| | |
|---|---|
| Necessity: | Each duration1 *plus* duration2 *equals* duration2 *plus* duration1. |
| CLIF Axiom: | (forall ((d1 duration))<br>(exists ((d2 duration))<br>(= (+ d2 d1) (+ d1 d2)))) |
| OCL Constraint: | context duration<br>inv: duration.allInstances->forAll(d2 \|<br>self._'plus duration'(d2).equals<br>(d2._'plus duration'(self))) |

Axiom V.4 (Additive Identity): There is a duration D0 such that, for every duration d1, d1 + D0 = d1.

D0

| | |
|---|---|
| Synonym: | zero duration |
| Definition: | duration that is the additive identity whose existence is required by Axiom V.4. |
| Necessity: | Each duration *plus* D0 = the duration. |
| CLIF Axiom: | (and<br>(duration D0)<br>(forall (d duration) (= (+ d D0) d) )) |
| OCL Constraint: | context duration<br>inv: self = self._'plus duration'(D0) |
| Note: | Declaring the individual concept (a logical "constant") asserts its existence. |
| Note: | D0 is unique. The uniqueness of D0 follows from the definition and the commutative axiom (V.3): If there is some other Dx such that d + Dx = d for all durations d, then D0 + Dx = D0, but D0 + Dx = Dx + D0 and Dx + D0 = Dx, by definition of D0. |

## duration$_3$ *equals* duration$_1$ *minus* duration$_2$

| | |
|---|---|
| Synonymous Form: | duration$_3$ = duration$_1$ - duration$_2$ |
| Synonymous Form: | duration$_1$ *minus* duration$_2$ *gives* duration$_3$ |
| Synonymous Form: | duration$_1$ - duration$_2$ *gives* duration$_3$ |
| Synonymous Form: | duration$_1$ *minus* duration$_2$ |
| Synonymous Form: | duration$_1$ - duration$_2$ |
| Definition: | duration$_1$ *equals* duration$_3$ *plus* duration$_2$ |
| Note: | There are no time intervals with negative durations, but negative durations can arise when subtracting one duration from another duration. In common usage, a negative duration is a combination of a direction and a magnitude. |
| Example: | A business process consists of task A immediately followed by task B. In any instance of the business process, the duration of task B is the duration of the entire business process minus the duration of task A. |

CLIF Definition:

```
(forall ((d1 duration) (d2 duration) d3)
  (iff
    (= d3 (– d1 d2))
    (and
      (duration d3)
      ("duration3 = duration1 – duration2" d3 d1 d2) )))
```

Axiom V.5 (Additive Inverse): For each duration d1, there is a duration d2, such that d1 + d2 = D0.

| | |
|---|---|
| Necessity: | D0 *equals* each duration$_1$ *plus* some duration$_2$. |

CLIF Axiom:

```
(forall ((d1 duration))
  (exists ((d2 duration))
    (= D0 (+ d1 d2))))
```

OCL Constraint:

```
context duration
    inv: duration.allInstances->exists(d2 |
        D0 = self + d2)
```

## duration$_2$ *equals* number *times* duration$_1$

| | |
|---|---|
| Synonymous Form: | duration$_2$ *equals* duration$_1$ *times* number |
| Synonymous Form: | duration$_2$ = number * duration$_1$ |
| Synonymous Form: | duration$_2$ = duration$_1$ * number |
| Definition: | duration$_2$ is the result of duration$_1$ *plus* duration$_1$, repeated number times |
| Example: | 50 seconds *equals* 50 *times* 1 second |
| Note: | The following are noun forms of the above verb concept. |
| Synonymous Form: | number *times* duration$_1$ |
| Synonymous Form: | duration$_1$ *times* number |
| Synonymous Form: | number * duration$_1$ |
| Synonymous Form: | duration$_1$ * number |

CLIF Definition:

```
(forall ((d1 duration) (n number) d2)
  (iff
    (= d2 (times n d1))
    (and
      (duration d2)
      ("duration2 = number times duration1" d2 n d1) )))
```

Axiom V.6 (Scalar multiplication is closed): if d1 is a duration and n1 is a number, n1 * d1 is a duration.

| | |
|---|---|
| Necessity: | For each number and each duration$_1$, some duration$_2$ *is* number *times* duration$_1$. |

CLIF Axiom:

```
(forall ((n1 number) (d1 duration))
  (exists ((d2 duration))
    (= d2 (times n1 d1))))
```

OCL Constraint:

```
context duration
    inv: Integer.allInstances->forAll(n |
```

```
self._'times number'(n)
 .oclIsKindOf(duration))
```

Corollary: The product of a number and a duration is unique.

Necessity: For each $duration_1$ and each number exactly one $duration_2$ *equals* number *times* $duration_1$.

This follows from the transitivity of equality of durations in 8.3.1.

Axiom V.7 (Scalar multiplication is distributive over durations): if d1 and d2 are durations and n1 is a number, n1 * (d1 + d2) = (n1 * d1) + (n1 * d2)

Necessity: If a $duration_3$ *equals* a $number_1$ *times* (a $duration_1$ *plus* a $duration_2$) then $duration_3$ *equals* ($number_1$ *times* $duration_1$) *plus* ($number_1$ *times* $duration_2$).

CLIF Axiom:
```
(forall ((d1 duration) (d2 duration)
        (d3 duration) (n1 number))
(if (= d3 (times n1 (+ d1 d2)))
   (= d3 (+ (* n1 d1) (times n1 d2)))))
```

OCL Constraint:
```
context duration
     inv: duration.allInstances->forAll(d2 |
         Integer.allInstances->forAll(n |
         self._'plus duration'(d2)
          ._'times number'(n).equals(
           self._'times number'(n)
            .self._'plus duration'(
              d2._'times number'(n)))))
```

Axiom V.8 (Scalar multiplication is distributive over reals): if d1 is a duration, and n1 and n2 are numbers, (n1 + n2) * d1 = n1 * d1 + n2 * d1.

Necessity: If a ($number_1$ plus a $number_2$) *times* a $duration_1$ *equals* a $duration_2$, then $duration_2$ *equals* ($number_1$ *times* $duration_1$) *plus* ($number_2$ *times* $duration_1$).

CLIF Axiom:
```
(forall ((d1 duration) (n1 number) (n2 number))
   (= (times (+ n1 n2) d1) (+ (times n1 d1) (times n2 d1))))
```

OCL Constraint:
```
context duration
     inv: Integer.allInstances->
      forAll(n1, n2 |
       self._'times number'(n1 + n2).equals(
        self._'times number'(n1)._'plus duration'
         (self_.'times number'(n2))))
```

Corollary: For all durations d1, 0 * d1 = D0.

Necessity: D0 *equals* 0 *times* each $duration_1$.

CLIF Axiom:
```
(forall ((d1 duration))
 (times 0 d1 D0))
```

OCL Constraint:
```
context duration
inv: self._'times duration' = D0
```

Corollary: If n1 is a number and d1 is a duration, then n1 * d1 = D0 iff n1 = 0 or d1 = D0.

Necessity: D0 *equals* a $number_1$ *times* a $duration_1$ if and only if $number_1$ *equals* 0 or $duration_1$ *equals* D0.

CLIF Axiom:
```
(forall ((n1 number) (d1 duration))
  (iff (= D0 (times n1 d1))
      (or
        (= n1 0)
        (= d1 D0)) ))
```

OCL Constraint:
```
context duration
inv: Integer.allInstances->forAll( n |
   (self._'times number'(n) = D0) eqv (self = D0 or n = 0) ))
```

Corollary (Ratio): If d1 and d2 are durations and not d2 = D0, then there exists a number n1 such that d2 = n1 * d1.

Necessity: If a $duration_1$ *does* not *equal* D0, then a $duration_2$ *equals* a $number_1$ *times* $duration_1$.

CLIF Axiom:
```
(forall ((d1 duration))
  (if (not (= d1 D0))
     (exists ((d2 duration) (n1 number))
        (* d1 n1 d2))))
```

OCL Constraint:
```
context duration
inv: if (not (self = D0)) then
   self.duration.allInstances->forAll(d |
     Integer.allInstances->exists(n |
       self._'times number'(n) = d) )
```

### 8.3.3 Relationships between 'Duration' and 'Time Interval'

The intent of the 'duration' concept is to measure time intervals, but the model presented above is a mathematical abstraction that does not depend on time intervals for its properties. What makes it useful is the following set of relationships between durations and time intervals.

Each time interval has a unique duration attribute that is a measure of its size, i.e., the amount of time the time interval occupies. This attribute is mathematically a function that maps time intervals into durations. This mapping function is sometimes called the "range" of a time interval, and some times called the "measure" of a time interval. Following SBVR practice, this specification calls it the duration *of* a time interval.

This sub clause describes the only special cases in which the durations of constructed time intervals are well-defined.

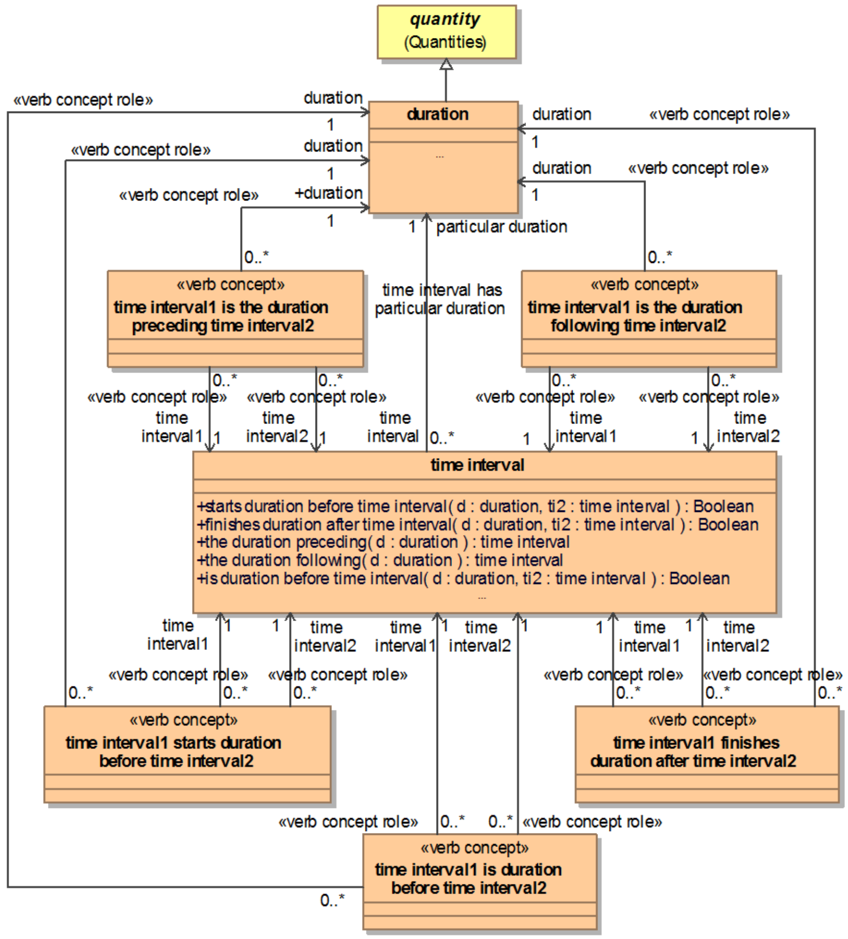


Figure 8.16 - Relationships between 'Duration' and 'Time Interval'

## particular duration

Concept Type: role
General Concept: duration
Definition: the amount of time in a given time interval
Note: particular duration is an instance of particular quantity whose values are of the quantity kind 'duration'.
Example: Particular duration of a particular meeting.

## time interval *has* particular duration

Synonymous Form: particular duration *of* time interval
Synonymous Form: time interval *has* duration

Synonymous Form: duration *of* time interval
Definition: the particular duration is the duration that is the amount of time in the time interval
Note: This is a primitive concept. It is the fundamental relationship between time intervals and durations. It has no formal definition. But there is a corresponding CLIF function, and a corresponding UML operation, and they can be formally defined in terms of the primitive verb concept.
CLIF Definition:
```
(forall (d ti) (iff
 (= d ("duration of time interval" ti))
 (and
  ("time interval" ti) (duration d)
  ("time interval has duration" ti d) )))
```
Example: The duration of Henry V's life is given by the duration value "35 years."
CLIF Axiom:
```
(forall (t d)
 (if ("time interval has duration" t d)
     (and ("time interval" t) (duration d)) ))
```
CLIF Definition:
```
(forall (t d)
 (iff (= ("duration of" t) d)
      ("time interval has duration" t d)))
```

Axiom D.1: Each time interval has exactly one duration.

Necessity: Each time interval *has* exactly one duration.
CLIF Axiom:
```
(forall ((t "time interval") (d1 duration) (d2 duration))
 (if (and ("time interval has duration" t d1)
          ("time interval has duration" t d2))
     (= d1 d2)))
```
OCL Constraint:
```
context _'time interval'
    inv: self._'particular duration'->size() = 1
```

Axiom D.2: Every time interval has a positive duration.

Necessity: The duration *of* each time interval *is greater than* D0.
CLIF Axiom:
```
(forall ((t "time interval"))
 (> ("duration of" t) D0))
```
OCL Constraint:
```
context _'time interval'
    inv: self.duration > D0
```

Corollary: No time interval has duration D0.

Necessity: The duration *of* no time interval *equals* D0.
CLIF Axiom:
```
(forall ((t "time interval"))
 (not (= ("duration of" t) D0)))
```
OCL Constraint:
```
context _'time interval'
    inv: not self.duration = D0
```

Corollary: No time interval has a duration that is the additive inverse of the duration of any time interval. Thus, the vector space 'duration' is larger than the image of the time intervals.

Necessity: For each time interval$_1$ there is no time interval$_2$ such that the duration *of* time interval$_1$ *plus* the duration *of* time interval$_2$ *equals* D0.
CLIF Axiom:
```
(not (exists ((t1 t2))
 ("duration3 = duration1 plus duration2"
  D0 ("duration of time interval" t1) ("duration of time interval" t2)) ))
```
OCL Constraint:
```
context _'time interval'
    inv: _'time interval'.allInstances->forAll(t2 |
      not ((self.duration() + t2.duration()) = D0))
```

Axiom D.3: If t1 and t2 are time intervals such that t1 is a *part of* t2, then $D(t1) \leq D(t2)$.

| | |
|---|---|
| Necessity: | For each time interval1 and each time interval2 that *is a part of* time interval1, the duration *of* time interval2 *is less than or equal to* the duration *of* time interval1. |

CLIF Axiom:
```
(forall (t1 t2)
 (if ("time interval1 is part of time interval2" t1 t2)
   ("duration duration" ("duration of" t1) ("duration of" t2)) ))
```
OCL Constraint:
```
context _'time interval'
   inv: 'time interval'.allInstances->forAll(t2 |
    self._'is part of'(t2)
    implies self_.'particular duration'._'is less than'(d2._'particular duration'))
```

Axiom D.4. If t1 and t2 are time intervals such that t1 meets t2, D(t1+t2) = D(t1) + D(t2).

Necessity: For each time interval1 and each time interval2 that *meets* a time interval1, the duration *of* time interval1 *plus* time interval2 *is equal to* the duration *of* time interval1 *plus* the duration *of* time interval2.

CLIF Axiom:
```
(forall (t1 t2 t3)
 (if (and
     ("time interval1 meets time interval2" t1 t2)
     ("time interval3 equals time interval1 plus time interval2" t3 t1 t2))
   (= (+ ("duration of" t1) ("duration of" t2)) ("duration of" t3)) ))
```
OCL Constraint:
```
context _'time interval'
   inv: 'time interval'_.allInstances->forAll(t2 |
    self.meets(t2)
    implies self.plus(t2)._'particular duration'
     .equals(self._'particular duration'._'plus duration'(t2._'particular duration')))
```

Corollary: If t1, t2, and t3 are time intervals such that t1 *starts* t2 *complementing* t3, then D(t1) = D(t2) – D(t3).

Necessity: For each time interval2 and each time interval3 that *finishes* time interval2, the duration *of* the time interval1 that *starts* time interval2 *complementing* time interval3 *is equal* to the duration *of* time interval2 *minus* the duration *of* time interval3.

CLIF Axiom:
```
(forall (t1 t2 t3)
 (if ("time interval1 starts time interval2 complementing time interval3" t1 t2 t3)
   (= ("duration of" t1)
    (- ("duration of" t2)
      ("duration of" t3)) ) ))
```
OCL Constraint:
```
context _'time interval'
   inv: 'time interval'.allInstances->forAll(t2 |
    self.starts(t2)
    implies t2._'minus starting interval'(self)._'particular duration'.equals(
      t2._'particular duration'._'minus duration'(self._'particular duration')))
```

Corollary: If t1 and t2 are time intervals such that t1 *finishes* t2 *complementing* t3, then D(t1 = D(t2) – D(t3).

Necessity: For each time interval2 and each time interval3 that *starts* time interval2, the duration *of* the time interval1 that *finishes* time interval2 *complementing* time interval3 *is equal* to the duration *of* time interval2 *minus* the duration *of* time interval3.

CLIF Axiom:
```
(forall (t1 t2 t3)
 (if ("time interval1 finishes time interval2 complementing time interval3" t1 t2 t3)
   (= ("duration of" t1) (- ("duration of" t2) ("duration of" t3)) ) ))
```
OCL Constraint:
```
context _'time interval'
   inv:'time interval'.allInstances->forAll(t2 |
    self.finishes(t2)
    implies t2._'minus finishing interval'(self)._'particular duration'.equals(
      t2._'particular duration'._'minus duration'(self._'particular duration')))
```

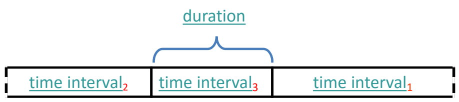


**Figure 8.17 - time interval$_2$ *is* duration *before* time interval$_1$**

## time interval$_2$ *is* duration *before* time interval$_1$

Synonymous Form: time interval$_2$ *is* duration *after* time interval$_1$
Synonymous Form: time interval$_1$ *ends* duration *before* time interval$_2$
Synonymous Form: time interval$_2$ *starts* duration *after* time interval$_1$
Synonymous Form: duration *is between* time interval$_1$ *and* time interval$_2$
Definition: time interval$_1$ *meets* some time interval$_3$ that *has* the duration and *meets* time interval$_2$
Description: The end of one time interval is duration before the start of the other time interval.
Necessity: Each duration that *is between* a time interval$_1$ *and* a time interval$_2$ *is greater than or equal to* D0.
Example: A time interval that "10:55" *refers to* is the duration that is *quantified by* "7 minutes" *before* a time interval that "11:02" *refers to*.
CLIF Definition:
```
(forall (t1 t2 d)
 (iff ("time interval2 is duration before time interval1" t1 d t2)
    (and
      ("time interval" t1)
      ("time interval" t2)
      (duration d)
      ("time interval1 is before time interval2" t2 t1)
      (exists ("time interval" t3)
       (and
        ("time interval1 meets time interval2" t2 t3)
        ("time interval1 meets time interval2" t3 t1)
        ("duration1 equals duration2" d
          ("time interval has particular duration" t3)) )))))
```
OCL Definition:
```
context _'time interval'
def: _'is duration before'(d: duration): _'time interval' =
    'time interval'.allInstances->
       exists(t2, t3 |
        t2._'is before'(self)
        and t2.meets(t3)
        and t3.meets(self)
        and t3._'particular duration'.equals(d))
```

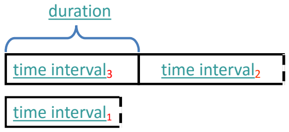


**Figure 8.18 - time interval$_1$ *starts* duration *before* time interval$_2$**

## time interval$_1$ *starts* duration *before* time interval$_2$

Definition: time interval$_1$ *starts with* the time interval$_3$ that *has* the duration and *meets* time interval$_2$
Description: The start of one time interval is duration before the start of the other time interval.

Note: This says nothing about the relationship between time interval₂ and the end of time interval₁

CLIF Definition:
```
(forall (t1 t2 d)
  (iff ("time interval1 starts duration before time interval2" t1 d t2)
    (and
      ("time interval" t1) ("time interval" t2) (duration d)
      (exists (t3 "time interval")
        (and
          ("time interval1 meets time interval2" t3 t2)
          ("time interval1 starts with time interval2" t1 t3)
          ("time interval has duration" t3 d) )) )))
```

OCL Definition:
```
context _'time interval'
def: _'starts duration before'(d: duration, t2: _'time interval'):Boolean =
        'time interval'.allInstances->
        exists(t3 |
         self._'starts with'(t3)
         and t3.meets(t2)
         and t3._'particular duration'.equals(d))
```

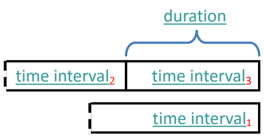


**Figure 8.19 - time interval$_1$ *finishes* duration *after* time interval$_2$**

## time interval$_1$ *finishes* duration *after* time interval$_2$

Definition: time interval$_1$ *finishes with* the time interval$_3$ that *has* the duration and *is met by* time interval$_2$

Description: The end of one time interval is duration after the end of the other time interval.

Note: This says nothing about the relationship between time interval$_2$ and the beginning of time interval$_1$

CLIF Definition:
```
(forall (t1 t2 d)
  (iff ("time interval1 finishes duration after time interval2" t1 d t2)
    (and
      ("time interval" t1) ("time interval" t2) (duration d)
      (exists (t3 "time interval")
        (and
          ("time interval1 meets time interval2" t3 t2)
          ("time interval1 finishes with time interval2" t1 t3)
          ("time interval has duration" t3 d) )) )))
```

OCL Definition:
```
context _'time interval'
def: _'finishes duration after'(d: duration, t2: _'time interval')
         :Boolean =
        'time interval'.allInstances->
        exists(t3 |
         self._'finishes with'(t3)
         and t2.meets(t3)
         and t3._'particular duration'.equals(d))
```

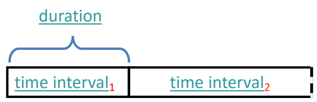


**Figure 8.20 - time interval$_1$ *is the* duration *preceding* time interval$_2$**

## time interval$_1$ *is the* duration *preceding* time interval$_2$

Synonymous Form: *the* duration *preceding* time interval$_2$

Definition: time interval$_1$ *is* the time interval that *has* the duration and *meets* time interval$_2$

Description: The time interval of interest (time interval$_1$) is the time period that has the given duration and is immediately before the other time interval(time interval$_2$).

Note: The word 'the' before the 'duration' phrase is a required part of the verb phrase.

Example: *the* two weeks *preceding* the meeting date

CLIF Definition:

```
(forall (t1 t2 d)
   (iff ("time interval1 is the duration preceding time interval2" t1 d t2)
      (and
         ("time interval" t1)
         ("time interval" t2)
         (duration d)
         ("time interval1 meets time interval2" t1 t2)
         ("time interval has duration" t1 d) )))
```

OCL Definition:

```
context _'time interval'
def: _'is the duration preceding'(d: duration, t2:'time interval'): Boolean =
          self.meets(t2)
          and self._'particular duration'.equals(d))
```

Necessity: For each time interval$_2$ and for each duration, exactly one time interval$_1$ *is the* duration *preceding* time interval$_2$.

Note: This follows from the definition.

CLIF Axiom:

```
(forall (t1 d) (exists (t2)
  (and
   ("time interval1 is the duration preceding time interval2" t2 d t1)
   (forall (t3)
    (if
     ("time interval1 is the duration preceding time interval2" t3 d t1)
     (= t3 t2) ))
)))
```

OCL Constraint:

```
context _'time interval'
     inv: _'time interval'.allInstances->
          forAll(t2 |
            duration.allInstances->forAll(d |
              'time interval'.allInstances->
               one(t3 | t3 = self._'is the duration preceding'(d, t2))))
```

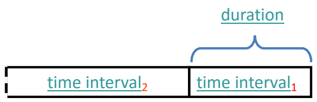


**Figure 8.21 - time interval$_1$ *is the* duration *following* time interval$_2$**

## time interval$_1$ *is the* duration *following* time interval$_2$

Synonymous Form: *the* duration *following* time interval$_2$

Definition: time interval$_1$ *is* the time interval that *has* the duration and *is met by* time interval$_2$

| | |
|---|---|
| Description: | The time interval of interest (time interval₁) is the time period that has the given duration and is immediately after the other time interval (time interval₂). |
| Note: | The word 'the' before the 'duration' phrase is a required part of the verb phrase. |
| Example: | *the* week *following* next week |
| Example: | The item is on sale during *the* two weeks *following* the holiday. |

CLIF Definition:

```
(forall (t1 t2 d)
   (iff ("time interval1 is the duration following time interval2" t1 d t2)
      (and
         ("time interval" t1)
         ("time interval" t2)
         (duration d)
         ("time interval1 meets time interval2" t2 t1)
         ("time interval has duration" t1 d) )))
```

OCL Definition:

```
context _'time interval'
def: _'is the duration following'
          (d: duration, t2:'time interval'): Boolean' =
           t2.meets(self)
           and self._'particular duration'.equals(d))
```

Necessity: For each time interval$_2$ and for each duration, exactly one time interval$_1$ *is the* duration *following* time interval$_2$.

Note: This follows from the definition.

CLIF Axiom:

```
(forall (t1 d) (exists (t2)
  (and
   ("time interval1 is the duration following time interval2" t2 d t1)
   (forall (t3)
     (if
      ("time interval1 is the duration following time interval2" t3 d t1)
      (= t3 t2) ))
)))
```

OCL Constraint:

```
context _'time interval'
      inv: _'time interval'.allInstances->
          forAll(t2 |
            duration.allInstances ->forAll(d |
              'time interval'.allInstances->
               one(t3 | t3 = self._'is the duration following'(d, t2))))
```

## 8.4 Time Units

As with other quantity kinds, durations are measured in terms of units. Unlike other quantity kinds, common time units are not simple ratios of each other. This makes for considerable complexity in specifying these time units. The details of this complexity are deferred to Clause 10.

The fundamental source of the complexity is that one of the main time units, 'year,' is incommensurable with other time units, such as 'month' and 'day.' This fact is due to the derivation of "year" and "day" from physical characteristics of our world.

### 8.4.1 Time Unit Concepts

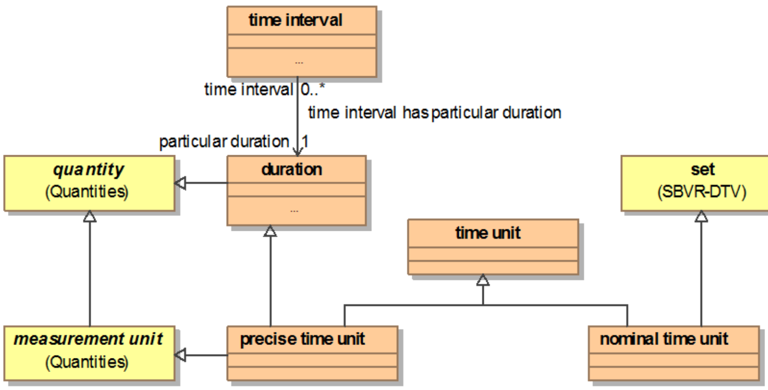


**Figure 8.22 - Time Units**

## time unit

Definition: precise time unit or nominal time unit
Example: year, week, hour

## precise time unit

Definition: measurement unit that *is* a duration
Note: [SI] defines 'hour', 'minute', and 'day' precisely. Although not addressed by [SI], 'week' also meets the definition of 'precise time unit'. Leap seconds are considered to introduce discontinuities in UTC, rather than variation in the definition of 'day'.
Example: second, minute, hour

## nominal time unit

Definition: set *of* durations that is defined and adopted by convention, meaning some duration *of* the set
Note: Sets *of* durations are *quantified* as 'duration value sets' in sub clause 8.7.
Note: Each nominal time unit can be traced to counting cycles of some natural phenomenon. Historically the phenomena have been astronomical: the orbital cycles of the Earth and the Moon and the diurnal cycle of the Earth. Unfortunately for time keeping, these cycles are incommensurable, requiring intercalary time periods to maintain synchronization. Leap days have been used since 46 BC with the introduction of the Julian calendar to keep the calendar aligned with seasons of the year.
Note: 'Year' and 'month' are said to be 'nominal time units' because of the effects of leap days.
Example: Year defined as {365 days, 366 days}.
Example: Month defined as {28 days, 29 days, 30 days, 31 days}. Each month on the Gregorian calendar is a choice of 28, 29, 30, or 31 days.

### 8.4.2 Standard Time Units

This sub clause provides standard concepts about times of day, as found in [ISO 8601] and [SI], and generally accepted around the world.

## second

| | |
|---|---|
| Synonym: | s |
| Synonym: | sec |
| Definition: | the precise time unit that is equal to the amount of time required for 9 192 631 770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the caesium 133 atom |
| Definition: | the base unit that *is defined for* the base quantity 'time' *by* the International System of Units (SI) |
| Dictionary Basis: | The International System of Units (SI) 2.1.1.3 'Unit of time (second)' |
| Note: | The duration of a second is a constant. In 1972, the second broke with astronomy and went to an atomic clock standard. |

## millisecond

| | |
|---|---|
| Synonym: | ms |
| Source: | SI |
| General Concept: | derived unit |
| General Concept: | precise time unit |
| Definition: | .001 seconds |

## microsecond

| | |
|---|---|
| Synonym: | µs |
| General Concept: | derived unit |
| General Concept: | precise time unit |
| Source: | SI |
| Definition: | $10^{-6}$ second |

## nanosecond

| | |
|---|---|
| Synonym: | ns |
| General Concept: | derived unit |
| General Concept: | precise time unit |
| Source: | SI |
| Definition: | $10^{-9}$ second |

## picosecond

| | |
|---|---|
| Synonym: | ps |
| General Concept: | derived unit |
| General Concept: | precise time unit |
| Source: | SI |
| Definition: | $10^{-12}$ second |

## minute

| | |
|---|---|
| Synonym: | min |
| General Concept: | derived unit |
| General Concept: | precise time unit |
| Source: | ISO 31-1 |
| Definition: | the precise time unit that *is quantified by* '60 seconds' |

## hour

| | |
|---|---|
| Synonym: | h |
| General Concept: | derived unit |

General Concept: precise time unit
Source: ISO 31-1
Definition: the precise time unit that *is quantified by* '3600 seconds'

## day

Synonym: d
Definition: the precise time unit that *is quantified by* 86 400 seconds
Note: 'Day' is defined in [SI] as 86 400 seconds. Leap seconds are intercalary seconds of day that are inserted as needed into UTC. Leap seconds do not affect the definition of 'day'.
Note: The duration of a calendar day is not necessarily 1 day, due to leap seconds and discontinuities arising when a locality switches between standard time and daylight time.
Note: Different calendars may define "day" differently. Particularly, in calendars based on solar time rather than ephemeris time, the calendar day may be defined by sunrise to sunrise, sunset to sunset or noon to noon. In such cases, the duration of a calendar day varies cyclically through the calendar year by as much as half an hour, a phenomenon known as the Equation of Time. Solar time is measured by observations and instruments such as sun dials, ephemeris time is measured by clocks.

## year

Definition: the nominal time unit that is the duration of a time interval required for one revolution of the Earth around the Sun, approximated to an integral number of days
Source: ISO 8601 (2.2.13, 'calendar year')
Definition: the nominal time unit that *is quantified by* {365 days, 366 days}
Note: There are several methods for reckoning a year. The main method is the return of the Vernal Equinox. This is called a tropical year, whose length is 365.2424 days of 86 400 seconds. There are several other year schemes, whose length in days of 86 400 seconds varies from about 347 days to about 384 days, depending how a year is measured. Such schemes use the term 'year' for different nominal units.
Note: The definition of a year is dependent on the use of a specific calendar. See "Gregorian year."
Note: The business term '*n* years' commonly refers to the duration of a specific consecutive sequence of 'year periods' (see 10.3).

## month

Definition: the nominal time unit that is the duration of a time interval required for one rotation of the Moon in its orbit around the Earth, approximated to a number of days.
Source: ISO 8601 (2.2.12, 'month')
Definition: the nominal time unit that *is quantified by* {28 days, 29 days, 30 days, 31 days}
Note: The business term '*n* months' commonly refers to the duration of a specific consecutive sequence of 'month periods' (see 10.3).
Note: A lunar month is about 28 days, and is incommensurable with a year. Different calendars define the number of days in a month differently. And the same calendar may define different calendar months to have different numbers of days. The Gregorian calendar has 12 calendar months that were rather arbitrarily set to a certain number of days by Roman politicians, without synchronizing with the lunar cycle.

## week

Source: ISO 8601 (2.2.9, 'week')
Definition: the precise time unit that *is quantified by* 7 days
Definition: the precise time unit that *is quantified by* 604 800 seconds

## 8.5 Time Scales

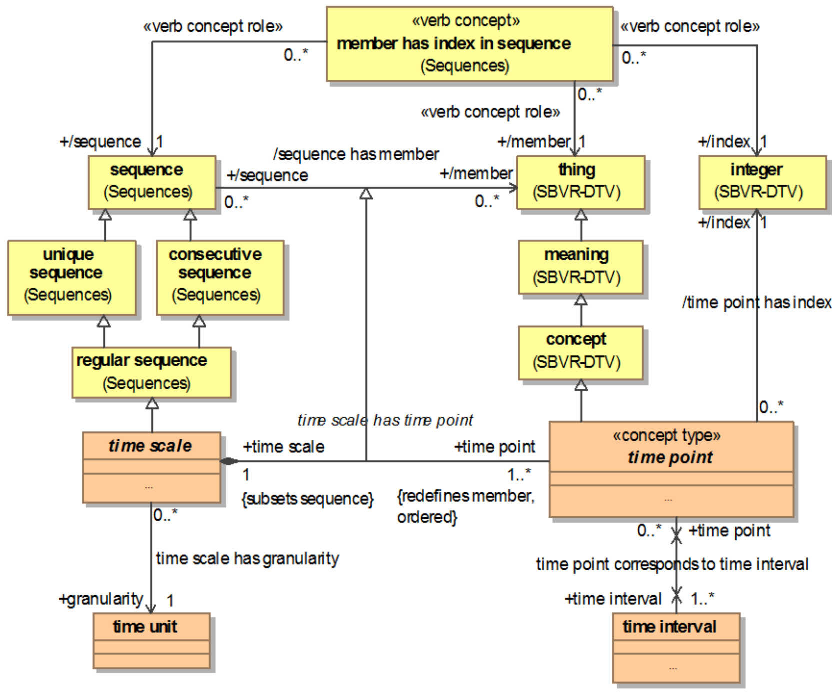


**Figure 8.23 - Time Scales**

## time scale

| | |
|---|---|
| Definition: | regular sequence that each member *of* the regular sequence *is* a time point |
| Necessity: | Each time scale *has* exactly one granularity |
| Necessity: | If a member *of* a time scale *has* a previous member then each time interval that *is* an instance *of* the member *is met by* some time interval that *is* an instance *of* the previous member. |
| Necessity: | If a member *of* a time scale *has* a next member then each time interval that *is* an instance *of* the member *meets* some time interval that *is* an instance *of* the next member. |
| Note: | These Necessities are really part of the definition of 'time scale'. |
| Dictionary Basis: | IEC 60050-111, ("time scale") |
| Dictionary Basis: | IEC 8601, (2.1.4, "time scale") |
| Definition: | system of ordered marks that can be associated with time intervals on the Time Axis, with one time interval being chosen as the reference point |
| Note: | [from ISO 8601] A time scale may among others be chosen as: |

- continuous, e.g., international atomic time (TAI) (see IEC 60050-713, item 713-05-18);
- continuous with discontinuities, e.g., Coordinated Universal Time (UTC) due to leap seconds, standard time due to summer time and winter time;

- successive steps, e.g., usual calendars, where the Time Axis is split up into a succession of consecutive time intervals and the same mark is attributed to all instants of each time interval;
- discrete, e.g., in digital techniques.

Note: [from ISO 8601] For physical and technical applications, a time scale with quantitative marks is preferred, based on a chosen initial instant together with a unit of measurement.
Note: [from ISO 8601] Customary time scales use various units of measurement in combination, such as second, minute, hour, or various time intervals of the calendar such as calendar day, calendar month and calendar year.
Note: [from ISO 8601] A time scale has a reference point which attributes one of the marks of the time scale to one of the instants, thus determining the attribution of marks to instants for the Time Scale.
Note: Each semantic community should agree on a closed set of time scales.
Example: The clock face of a traditional clock is a time scale.

## granularity

Synonym: resolution
Concept Type: role
General Concept: time unit
Dictionary Basis: VIM (4.15, 'resolution (2)')
Definition: the smallest duration that can be distinguished with a given time scale
Necessity: Each time scale *has* exactly one granularity
Example: "Second" as the granularity for a time scale in which each time point has the duration "1 second."

## time scale *has* granularity

Definition: The granularity *of* the time scale *is* the duration *of* the time points *of* the time scale.

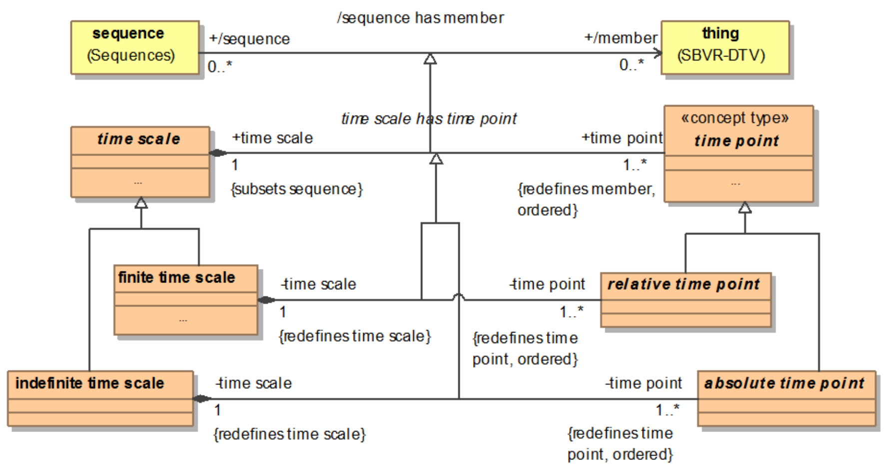


**Figure 8.24 - Time Scale Kinds**

## finite time scale

Definition: time scale that *has* a first member and that *has* a last member
Note: A finite time scale has a cardinality.
Necessity: Each time point *of* a finite time scale *is* a relative time point
Example: the Gregorian year of months scale
Example: the hour of minutes scale

## indefinite time scale

Definition: time scale that *is* not a finite time scale
Necessity: Each time point *of* an indefinite time scale *is* an absolute time point.
Note: An indefinite time scale has no cardinality because it has no first member, no last member, or both.
Example: the Gregorian years scale

## absolute time point

Definition: time point that *is of* an indefinite time scale
Necessity: Each absolute time point *corresponds to* exactly one time interval.
Example: The absolute time coordinate 'September 11, 2011' *indicates* an absolute time point.

## relative time point

Definition: time point that *is of* a finite time scale
Necessity: Each relative time point *corresponds to* more than one time interval.
Example: The relative time coordinate 'September 11' *refers to* multiple time intervals, one in each Gregorian year.

## 8.6 Time Points

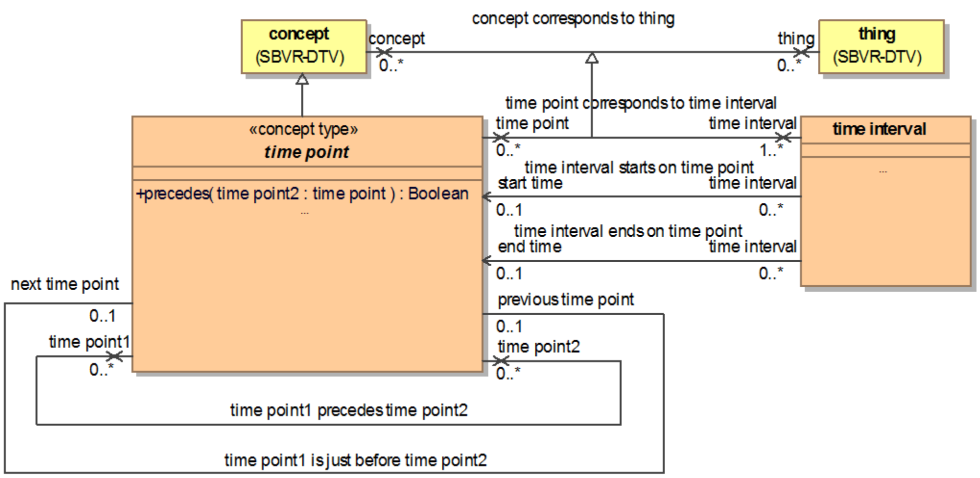


Figure 8.25 - Time Points

## time point

| | |
|---|---|
| Concept Type: | concept type |
| General Concept: | time period |
| Definition: | concept that *specializes* the concept 'time interval' and that *is* a member *of* a time scale |
| Necessity: | The duration *of* each time interval that *is* an instance *of* the time point *is* the granularity *of* the time scale *of* the time point. |
| Note: | Each time point is a concept whose instances are time intervals. |
| Reference Scheme: | an occurrence *at* the time point |
| Reference Scheme: | a time coordinate that *indicates* the time point |
| Reference Scheme: | the time scale *of* the time point and the index *of* the time point |
| Reference Scheme: | the time point kind *of* the time point and the index *of* the time point |
| Note: | This is a total reference scheme: every time point *is indicated by* at least one time coordinate, and some time points may *be indicated by* multiple time coordinates. |
| Example: | The Battle of Hastings *was on* "14 October 1066". (This gives the Julian date of the battle at a granularity of "day". If desired, the battle could be given more precisely as a time period within that calendar day.) |

## time scale *has* time point

| | |
|---|---|
| Synonymous Form: | time point *is on* time scale |
| General Concept: | sequence *has* member |
| Necessity: | Each time scale *has* at least one time point. |
| Necessity: | Each time point *is of* exactly one time scale. |

## time point *has* index

| | |
|---|---|
| Definition: | the index *is* the index *of* the sequence position that *is in* the time scale *of* the time point and that *has* a member that *is* the time point |
| Necessity: | Each time point *has* exactly one index. |

## time point$_1$ *precedes* time point$_2$

| | |
|---|---|
| Synonymous Form: | time point$_2$ *follows* time point$_1$ |
| Definition: | the time scale *of* time point$_1$ *is* the time scale *of* time point$_2$ and the index *of* time point$_1$ *is less than* the index *of* time point$_2$ |
| Note: | This is a special case of member *precedes* member in the unique sequence that is the time scale of the two time points |

## time point$_1$ *is just before* time point$_2$

| | |
|---|---|
| Synonymous Form: | time point$_2$ *is next after* time point$_1$ |
| Definition: | the time scale *of* time point$_1$ *is* the time scale *of* time point$_2$ and the sequence position *of* time point$_1$ *is just before* the sequence position *of* time point$_2$ *in* the time scale *of* time point$_1$ |

## time interval *starts on* time point

| | |
|---|---|
| Synonymous Form: | time point *starts* time interval |
| Definition: | some time interval that *is* an instance *of* the time point *starts* the time interval |

## time interval *ends on* time point

| | |
|---|---|
| Synonymous Form: | time point *ends* time interval |
| Definition: | some time interval that *is* an instance *of* the time point *finishes* the time interval |

## 8.7 Time Periods and Time Point Sequences

This sub clause introduces a general mechanism for references to time intervals.

Many references to time intervals involve expressions using time points to denote the ends of the time interval, such as "2 p.m. to 4 p.m." Formally, such time intervals may be said to instantiate consecutive sequences of time points on some time scale, what is here called a time point sequence. A single time point used to refer to a time interval may be regarded as a special case of a time point sequence. And like a time point, a time point sequence can refer to more than one time interval, e.g., "2 p.m. to 4 p.m. on Mondays."

The time intervals that are specified in this way are common in business, and are considered a special class of time interval, called time period. The business user understands the names for the time points and the time period concept; the user need not be aware of the formal model.

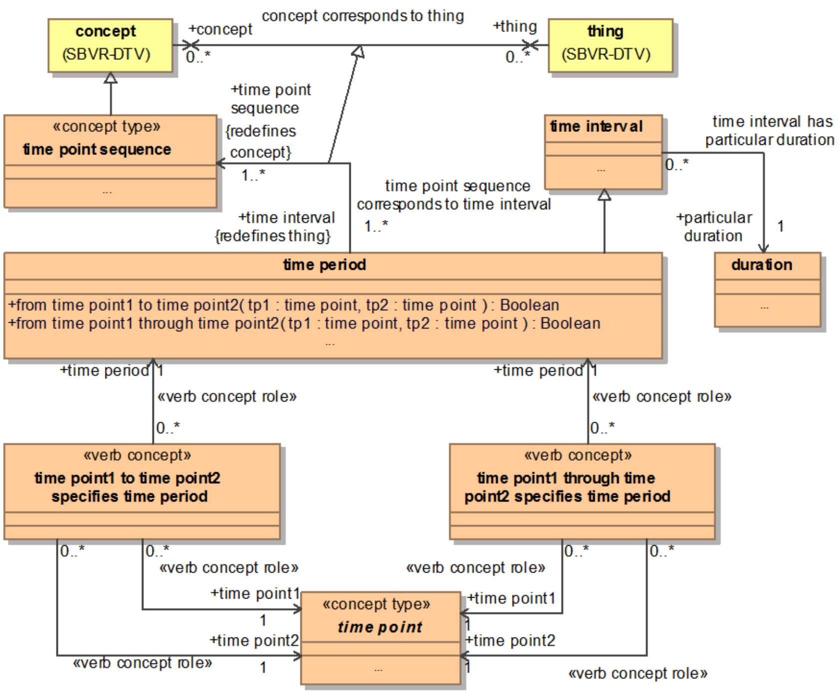


Figure 8.26 - Time periods and time point sequences

### time point sequence

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | consecutive sequence *of* time points |
| Necessity: | All the time points of a given time point sequence are on the same time scale. |
| Note: | This is formalized by the Definition and Necessity under 'time point sequence is on time scale'. |
| Note: | A time point sequence is not necessarily a subsequence of a time scale because a time point sequence may "wrap around" a finite time scale by including time points from the end of the time scale, followed by time points from the start of the time scale. |

| | |
|---|---|
| Reference Scheme: | The first time point *of* the time point sequence and the last time point *of* the time point sequence. |
| Reference Scheme: | The first time point *of* the time point sequence and the duration *of* the time point sequence. |
| Reference Scheme: | The last time point *of* the time point sequence and the duration *of* the time point sequence. |
| Necessity: | Each time point sequence *has* at least one member. |
| Necessity: | Each time point sequence *has* exactly one first time point. |
| Necessity: | Each time point sequence *has* exactly one last time point. |
| Note: | It is not possible to specify an indefinite time point sequence; i.e. one that has no first time point or no last time point. A time point sequence is a specific section of a calendar. It is possible to specify a time point sequence by specifying the first time point or last time point to be the date or time of an event, including primordiality and perpetuity, if appropriate. It is also possible to specify a time interval by means other than a time point sequence (see clause 16.7). |
| Necessity: | The first time point *of* each time point sequence that *is on* an indefinite time scale and that *has* more than one member *precedes* the last time point *of* the time point sequence. |
| Note: | In a time point sequence on an indefinite time scale, the time points are consecutive. But a time point sequence can "wrap around" the end of a finite time scale. For example, "December 25 through January 4". The definition of 'time point sequence corresponds to time interval' just requires the start and finish of the time interval to instantiate the first and last time point. The relationship of the time point sequence to the time scale follows from that requirement. |
| Example: | 22:00 to 06:00 |
| Example: | The time point sequence from July 1, 2009 to August 3, 2010. |

## time point sequence *corresponds to* time interval

| | |
|---|---|
| Synonymous Form: | time interval *instantiates* time point sequence |
| Definition: | the time interval *starts on* the first time point *of* the time point sequence and the duration *of* the time interval *is* the duration *of* the time point sequence |
| Necessity: | Each time point sequence that *is on* an indefinite time scale *corresponds to* exactly one time interval. |
| Note: | The corresponding time intervals are determined by the first time point and the cardinality of the time point sequence. This is correct even when the time point sequence "wraps around" the end of a finite time scale. |

## time point sequence *has* duration

| | |
|---|---|
| Definition: | the duration *equals* the cardinality *of* the time point sequence *times* the granularity *of* the time point sequence |
| Necessity: | Each time point sequence that *has* a first time point and a last time point *has* exactly one duration. |
| Necessity: | Each time point sequence that *has* no first time point or no last time point *has* no duration. |
| Note: | The duration of such a time sequence is infinite. |
| Example: | The duration of the time point sequence consisting of Monday, Tuesday, and Wednesday is 3 days. |

## time period

| | |
|---|---|
| Definition: | time interval that *instantiates* some time point sequence |

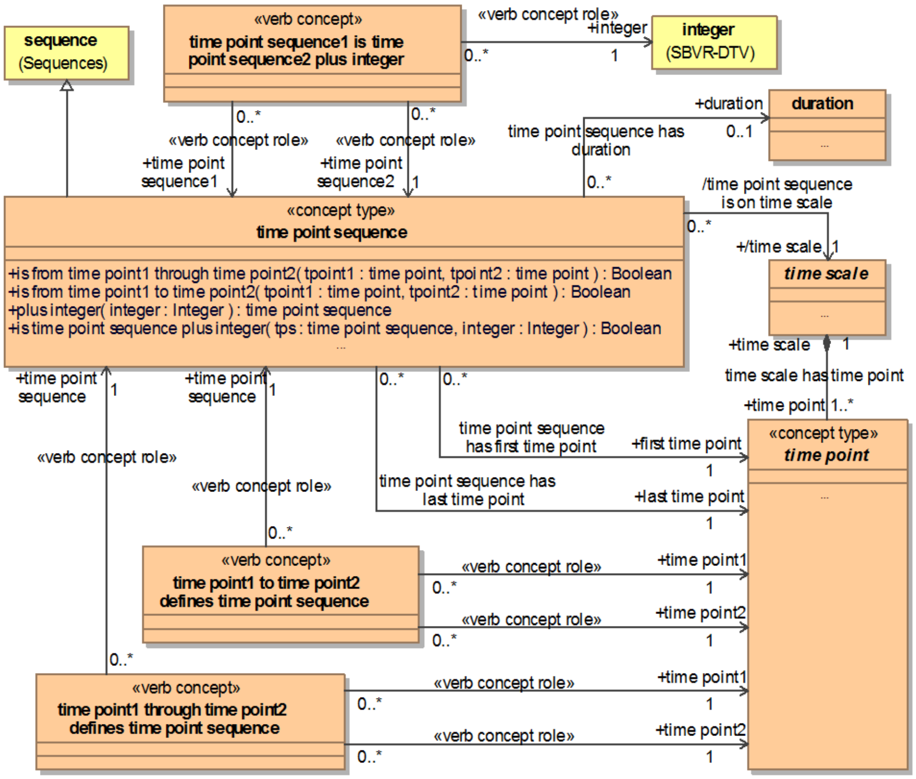


**Figure 8.27 - Time point sequence structure**

## time point sequence *is on* time scale

| | |
|---|---|
| Synonymous Form: | time scale *of* time point sequence |
| Definition: | each time point *of* the time point sequence *is* a member *of* the time scale |
| Necessity: | Each time point sequence *is on* exactly one time scale. |
| Example: | A time point sequence *consisting of* seconds of day *is on* the day of seconds scale. |

## time point sequence$_2$ *is* time point sequence$_1$ *plus* integer

| | |
|---|---|
| Synonymous Form: | time point sequence$_2$ = time point sequence$_1$ + integer |
| Synonymous Form: | time point sequence$_1$ *plus* integer |
| Synonymous Form: | time point sequence$_1$ + integer |
| Definition: | time point sequence$_2$ *is on* the time scale *of* time point sequence$_1$ and the index origin position *of* time point sequence$_2$ *is* the index origin position *of* time point sequence$_1$ + the integer |
| Description: | The time point sequence$_1$ *is shifted* by the integer. |
| Necessity: | If a time point sequence$_1$ *is* a time point sequence$_2$ *plus* an integer, then time point sequence$_1$ *is on* an indefinite time scale and time point sequence$_2$ *is on* the indefinite time scale. |

Example: The time point sequence 2 July 2012 *through* 4 July 2012 *is* the time point sequence 1 July 2012 *through* 3 July 2012 *plus* 1.

## first time point

Synonym: start time point
Concept Type: role
General Concept: time point

## time point sequence *has* first time point

Synonymous Form: first time point *of* time point sequence
Definition: the first time point is the first member *of* the time point sequence
Example: The time coordinate of the first time point of the time point sequence from July 1, 2009 to August 3, 2010 is July 1, 2009.

## last time point

Synonym: end time point
Concept Type: role
General Concept: time point

## time point sequence *has* last time point

Synonymous Form: last time point *of* time point sequence
Definition: the last time point is the last member *of* the time point sequence
Example: The time coordinate of the last time point of the time point sequence from July 1, 2009 to August 3, 2010 is August 3, 2010.

## time point$_1$ *through* time point$_2$ *defines* time point sequence

Synonymous Form: time point sequence *is from* time point$_1$ *through* time point$_2$
Definition: time point$_1$ *is* the first time point *of* the time point sequence and time point$_2$ *is* the last time point *of* the time point sequence

## time point$_1$ *to* time point$_2$ *defines* time point sequence

Synonymous Form: time point sequence *is from* time point$_1$ *to* time point$_2$
Definition: time point$_1$ *is* the first time point *of* the time point sequence, and if time point$_2$ *is* the first member *of* the time scale *of* the time point sequence, the last time point *of* the time point sequence *is* the last member *of* the time scale,
and if time point$_2$ *is* not the first member *of* the time scale, the last time point *of* the time point sequence *is* the time point that *is just before* time point$_2$ (on the time scale)

## time point$_1$ *through* time point$_2$ *specifies* time period

Synonymous Form: time point$_1$ *through* time point$_2$
Definition: the time point sequence that *is from* time point$_1$ *through* time point$_2$ *corresponds to* the time period
Possibility: If the time scale *of* time point$_1$ is a finite time scale then time point$_1$ *through* time point$_2$ *specifies* more than one time period.
Note: Contrast '*through*' with '*to*.' '*Through*' is inclusive of time point$_2$, while '*to*' is exclusive of time point$_2$.
Example: "January *through* March", meaning the time interval of 3 months duration that starts with January and ends with March.

**time point$_1$ *to* time point$_2$ *specifies* time period**

Synonymous Form: time point$_1$ *to* time point$_2$
Definition: the time point sequence that *is from* time point$_1$ *to* time point$_2$ *corresponds to* the time period
Possibility: If the time scale *of* time point$_1$ is a finite time scale then time point$_1$ *through* time point$_2$ *specifies* more than one time period.
Note: Contrast '*through*' with '*to*.' '*Through*' is inclusive of time point$_2$, while '*to*' is exclusive of time point$_2$.
Example: "January *to* March", meaning the time interval of 2 months duration that starts with January and ends with February.

________________________________________________
________________________________________________

# 9 Duration Values (normative)

## 9.1 General

A duration value is a conceptual structure of meaning that serves to identify a duration. Duration values are amounts of time stated in terms of one or more time units. For example, "60 seconds" or "1 minute". The concept 'duration value', and related concepts, specialize 'quantity value' (Annex D.2.3) and its related concepts. These concepts are restated here for clarification and to bring them into this normative text.

In this specification, a precise duration value *quantifies* a duration. The key difference between 'duration value' and 'duration' is that a single duration may be quantified by multiple precise duration values. For example, "60 seconds" and "1 minute" quantify the same duration: the two duration values are *equivalent*.

Complexity arises with duration values that use the nominal time units 'month' and 'year' because the number of calendar days varies among calendar months, and because some calendar years incorporate leap days. For example, "1 year" is equivalent to "12 months" but it is unclear in everyday usage how "12 months" compares to "365 days". To help answer the question, this clause introduces the concept of 'duration value set'. A duration value set specifies a set of duration values that are jointly considered equivalent to a nominal duration value. For example, "1 month" is any of {28 days, 29 days, 30 days, 31 days}.

Furthermore, this clause specifies common arithmetic and comparison operations on nominal duration values defined as duration value sets. This helps to define what expressions such as "3 months" or "3 months *plus* 3 days" mean. The advantage of this approach is that it clarifies the results of comparisons such as "3 months < 90 days."

________________________________________________

**Duration Values Vocabulary**

General Concept: terminological dictionary
Language: English
Included Vocabulary: Time Infrastructure Vocabulary
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#DurationValuesVocabulary

________________________________________________

## 9.2 Duration Values

### duration value

Definition: precise duration value or nominal duration value
Definition: atomic duration value or compound duration value
Necessity: Each duration value *has* at least one atomic duration value.
Note: A duration value can be either atomic or compound and either nominal or precise (see sub clause 9.3).
Example: 45 seconds, 1 year 3 days

### 9.2.1 Atomic and Compound Duration Values

Duration values can be either atomic (have just one component, such as 10 minutes) or be compound (a combination of multiple atomic duration values, such as 1 year 5 months). Atomic duration values consist of a number and a time unit, such as "4 weeks." Compound duration values comprise multiple atomic duration values. For example, "3 years 5 months".

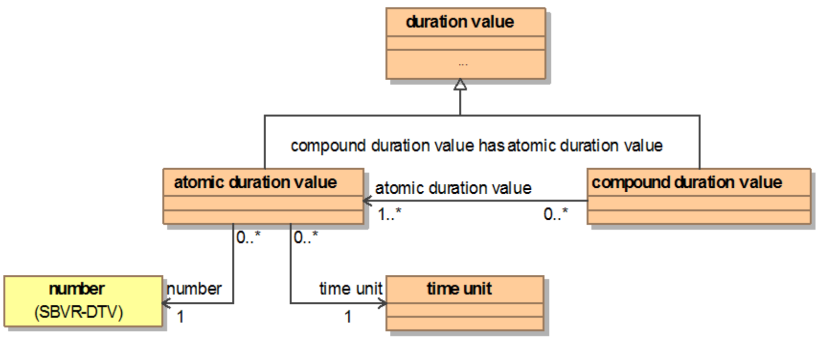


**Figure 9.1 - Duration Values**

### atomic duration value

Definition: number and time unit together giving magnitude of a duration
Dictionary Basis: VIM 1.19 'quantity value'
Example: 55 seconds is an atomic duration value

### atomic duration value *has* number

Definition: if the atomic duration value *is* a precise atomic duration value, then the number is the ratio of the duration *quantified by* the atomic duration value to the time unit of the atomic duration value
Definition: if the atomic duration value *is* a nominal atomic duration value, then the number is the ratio of exactly one of the elements of the duration value set that *is specified by* the atomic duration value to the time unit of the atomic duration value
Note: In the general case, the number is a mathematical real or complex number. Because the number is a ratio, rational fractions are commonly used in stating duration values. Thus, it is meaningful to say a task took 2.5 days to complete. Fractional numbers are not defined for nominal atomic duration values (except for ½ year, ¼ year, and ¾ year), because they have no clear meaning.
Example: 2.5 years, 5.6318 seconds

| | |
|---|---|
| Note: | When the number is a non-negative integer, it may be thought of as a count of the time units in the duration value. But that view only applies to certain measurement techniques, such as the count of ticks of a clock. |
| Example: | 8 months |
| Possibility: | The number *is less than* 0. |
| Note: | Although there are no negative durations, the number of an atomic duration value may be negative. A duration value may *quantify* a (positive) duration even though a component atomic duration value is negative. Typically, a negative atomic duration value arises as an intermediate result of a subtraction. |
| Note: | "1 hour 12 minutes - 14 minutes *equals* 1 hour –2 minutes", which *quantifies* the same duration that is *quantified by* "58 minutes". |

### atomic duration value *has* time unit

| | |
|---|---|
| Definition: | if the atomic duration value *is* a precise atomic duration value, then the time unit is the reference duration to which the ratio of the duration *quantified by* the atomic duration value is taken |
| Definition: | if the atomic duration value *is* a nominal atomic duration value, then the time unit is the reference duration to which the ratio of exactly one element of the duration value set *specified by* the atomic duration value is taken |
| Example: | "45 minutes" has the time unit 'minute' |

### compound duration value

| | |
|---|---|
| Definition: | combination of two or more atomic duration values that *have* different time units |
| Example: | "2 hours 20 minutes" *quantifies* the duration that may also be *quantified* as "140 minutes" |

### duration value *has* atomic duration value

| | |
|---|---|
| Definition: | the atomic duration value is one of the summands of the duration value |
| Example: | 1 hour 5 minutes 3 seconds is a compound duration value that is composed of three atomic duration values: 1 hour, 5 minutes, 3 seconds |

### 9.2.2 Precise Duration Values

Time units are either precise (such as seconds) or nominal (that is years, which can be either 365 days or 366 days; and months, which can be 28 days, 29 days, 30 days, or 31 days). Duration values are also nominal or precise according to whether they use nominal or precise time units.

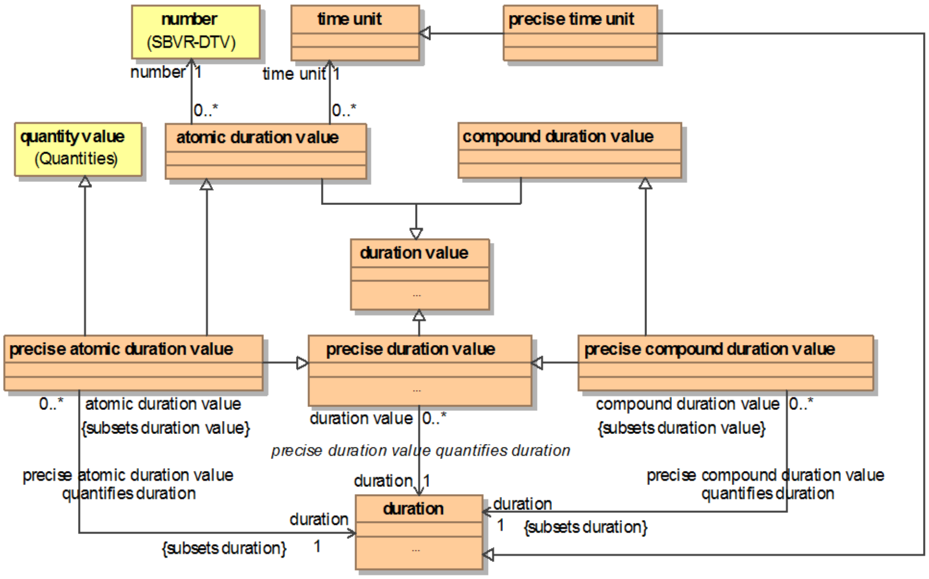


**Figure 9.2 - Precise Duration Values**

## precise duration value

| | |
|---|---|
| Definition: | precise atomic duration value or precise compound duration value |
| Example: | 5 hours |
| Example: | 3 days 5 hours |

## precise atomic duration value

| | |
|---|---|
| Definition: | quantity value that *is* an atomic duration value that *has* a precise time unit |
| Note: | The duration *quantified by* a precise atomic duration value is the duration whose ratio to the time unit is the number. |
| Example: | 30 seconds |

## precise compound duration value

| | |
|---|---|
| Definition: | compound duration value that is the combination of two or more precise atomic duration values that *have* different time units |
| Example: | 5 minutes 30 seconds |

Each precise time unit (i.e., the time units 'second,' 'minute,' 'hour,' 'day,' and 'week') is defined as *quantifying* a multiple of 'second' using the pattern 'the precise time unit that *quantifies* <some number of> seconds'. Thus, every precise atomic duration value (i.e., an atomic duration value that uses one of those time units) *quantifies* a duration that is some multiple of 'seconds'. For example, '3 hours' *quantifies* a duration of 10 800 seconds.

## precise atomic duration value *quantifies* duration

| | |
|---|---|
| Synonymous Form: | duration *is quantified by* precise atomic duration value |
| Definition: | the ratio of the duration to the time unit *of* the precise atomic duration value *is* the number *of* the precise atomic duration value |

Example: "2 seconds" *quantifies* a duration that is twice the duration of the time unit 'second'
Example: "1 minute 3 seconds" *quantifies* a duration that is 63 times the duration of the time unit 'second'

Precise compound duration values *quantify* durations via a computation that can be summarized as "*quantify* all the atomic duration values of the precise compound duration value as durations, and then sum them". For example, 2 hours 30 minutes 20 seconds *quantifies* a duration of '9 020 seconds'.

## precise compound duration value *quantifies* duration

Synonymous Form: duration *is quantified by* precise compound duration value
Definition: the duration *is* the sum of the durations that *are quantified by* each precise atomic duration value *of* the precise compound duration value
Example: 12 weeks 3 days *quantifies* the duration '8 380 800 seconds'

### 9.2.3 Nominal Duration Values

Nominal duration values are distinguished from precise nominal duration values because a nominal duration value is one of several durations as defined by a calendar. For example, the compound nominal duration value "1 year 1 day" is any of {366 days, 367 days} because 1 year plus 1 day could be either of those.

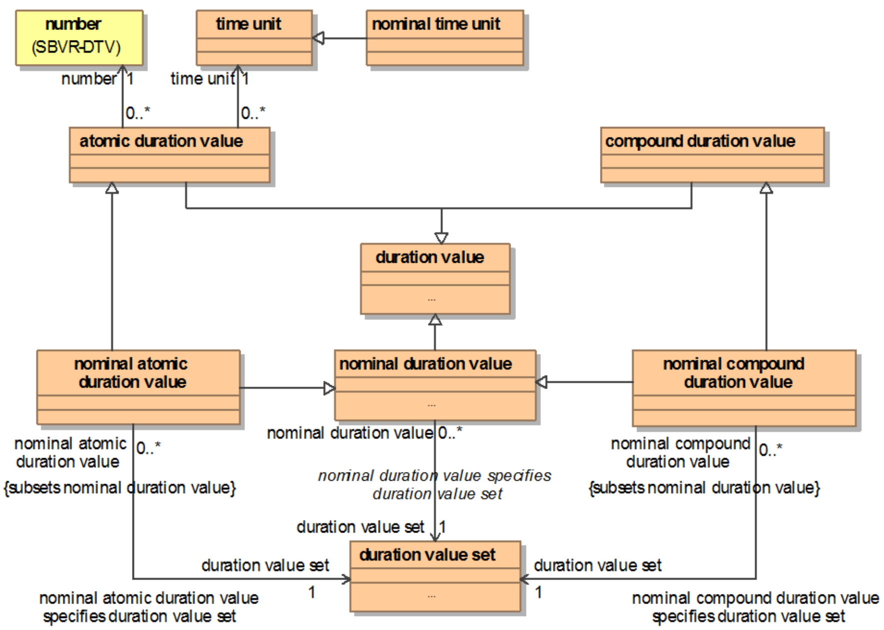


**Figure 9.3 - Nominal Duration Values**

## nominal duration value

Definition: nominal atomic duration value or nominal compound duration value
Necessity: The nominal duration value *is* the range *of* a time interval *identified by* a time period *of* a time calendar.

Example: 5 months, for example from February through June
Example: 2 years 6 months, for example from January 1990

## nominal atomic duration value

General Concept: atomic duration value
Definition: number and nominal time unit together that *specify* a duration value set
Note: See sub clauses 9.3 and 9.4 for the detailed definition of this concept.
Example: 30 months

## nominal compound duration value

Definition: compound duration value that *has* at least one atomic duration value that *is* a nominal atomic duration value
Possibility: An atomic duration value *of* the nominal compound duration value *is* a precise atomic duration value.
Example: 1 year 1 day

Each nominal time unit (i.e., the time units 'year' and 'month') is defined as *specifying* two or more choices among different numbers of 'days' using the pattern 'the nominal time unit that *specifies* {<number1> days, <number2> days, …, <numbern> days}'. This captures the idea that a year is either 365 days or 366 days, and a month is anywhere from 28 to 31 days.

## nominal atomic duration value *specifies* duration value set

Synonymous Form: duration value set *is specified by* nominal atomic duration value
Definition: the duration value set is a function of the nominal time unit *of* the nominal atomic duration value and the number *of* the nominal atomic duration value, and that function depends upon the nominal time unit
Note: The meaning of this verb concept is further defined in specializations, two which are defined in clauses 11.5 and 11.6: 'year value *specifies* duration value set' and 'month value *specifies* duration value set'. Other vocabularies can add their own for other nominal time units.
Example: 2 years *specifies* {730 days, 731 days} because the nominal time unit 'year' *specifies* the duration value set {365 days, 366 days} and there are no two consecutive leap years

Unlike precise atomic duration values, a nominal atomic duration value is not a simple multiple of the duration values of the duration value set *specified by* the nominal time unit of the nominal atomic duration value. For example, 2 years does not *quantify* "2 * 366 days" because, in the Gregorian calendar, two successive years cannot both be leap years. Thus,
2 years *specifies* one of {365 + 365 days, 365 + 366 days}. Sub clauses 11.5 and 11.6 formally define this for the 'year' and 'month' nominal time units.

A nominal compound duration value comprises two or more nominal atomic duration values. Each of these nominal atomic duration values *specifies* a duration value set, as described above. The entire nominal compound duration value *specifies* a duration value set that is the summation of the individual duration value sets. The summation is computed by pairwise addition of each of the duration values sets that *are quantified by* the nominal atomic duration values. Adding two duration value sets is defined by the verb concept 'duration $set_3$ = duration $set_1$ + duration $set_2$' in sub clause 9.5.

## nominal compound duration value *specifies* duration value set

Synonymous Form: duration value set *is specified by* nominal compound duration value
Definition: the duration value set *is* the sum of the duration value sets that *are specified by* each atomic duration value *of* the nominal compound duration value
Example: 14 months 3 days *specifies* the duration value set {427 days, 428 days, 429 days, 430 days, 431 days}

## 9.3 Duration Value Arithmetic

Addition and subtraction of duration values, and multiplication and division of duration values by scalar numbers, is defined in terms of the corresponding operations on the individual components of the duration values. For example, "1 year 5 months + 8 months 8 days" produces "1 year 13 months 8 days". This avoids the complexities of mixed-base arithmetic, which are not resolvable in the case of nominal duration values. (As an example of those complexities, consider that "14 days + 14 days" might be equivalent to either "28 days" or "1 month" depending upon the particular month.)

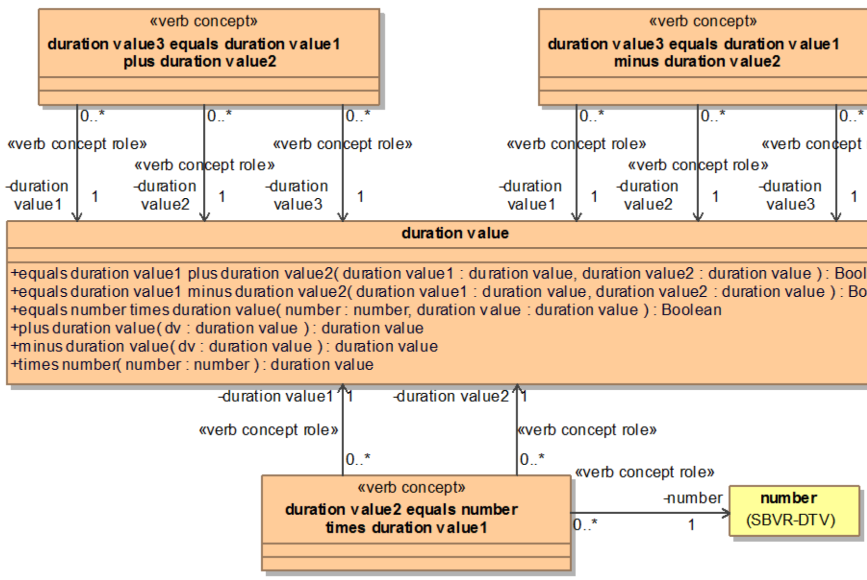


**Figure 9.4 - Duration Value Arithmetic**

### duration value$_3$ *equals* duration value$_1$ *plus* duration value$_2$

| | |
|---|---|
| Synonymous Form: | duration value$_1$ *plus* duration value$_2$ |
| Synonymous Form: | duration value$_3$ = duration value$_1$ + duration value$_2$ |
| Synonymous Form: | duration value$_1$ + duration value$_2$ |
| Definition: | each atomic duration value$_3$ *of* duration value$_3$ equals the sum of the number$_1$ *of* an atomic duration value$_1$ *of* duration value$_1$ and either the number$_2$ *of* some atomic duration value$_2$ *of* duration value$_2$ that *has* the same time unit, or 0 if there *does* not *exist* an atomic duration value$_2$ *of* duration value$_2$ that *has* the same time unit |
| Note: | This does not use "carries" among atomic duration values of different time units, because they don't work for nominal time units. The numbers of the atomic duration values that comprise duration value$_3$ may be greater than defined in the corresponding time unit. |
| Example: | 6 years 367 days 4 hours 61 minutes *equals* 5 years 3 days 4 hours 3 minutes *plus* 1 year 364 days 58 minutes |
| Note: | Tools may represent the results of duration value addition using mixed-base "carries" when practical. |
| Example: | 1 hour 80 minutes *equals* 1 hour 35 minutes *plus* 45 minutes. A tool may choose to display this result as 2 hours 20 minutes. |

### duration value$_3$ *equals* duration value$_1$ *minus* duration value$_2$

Synonymous Form: duration value$_1$ *minus* duration value$_2$
Synonymous Form: duration value$_3$ = duration value$_1$ - duration value$_2$
Synonymous Form: duration value$_1$ - duration value$_2$
Definition: each atomic duration value$_3$ *of* duration value$_3$ equals the number$_1$ *of* an atomic duration value$_1$ *of* duration value$_1$ minus either the number$_2$ *of* some atomic duration value$_2$ *of* duration value$_2$ that *has* the same time unit, or 0 if there *does* not *exist* an atomic duration value$_2$ *of* duration value$_2$ that *has* the same time unit
Possibility: The number *of* some atomic duration value *of* duration value$_3$ may be negative.
Note: This does not use "borrows" among atomic duration values of different time units, because they don't work for nominal time units. Negative atomic duration values may occur.
Example: 1 year -5 days *equals* 1 year 45 days *minus* 50 days

### duration value$_2$ *equals* number *times* duration value$_1$

Synonymous Form: duration value *equals* duration value *times* number
Synonymous Form: number *times* duration value
Synonymous Form: duration value *times* number
Synonymous Form: duration value = number * duration value
Synonymous Form: duration value = duration value * number
Synonymous Form: number * duration value
Synonymous Form: duration value * number
Definition: each atomic duration value$_1$ *of* duration value$_1$, multiplied by the given number equals some atomic duration value$_2$ *of* duration value$_2$
Example: 5 days *quantifies* the duration that *equals* 5 *times* 1 day
Possibility: The number is negative.
Example: -5 days
Note: Negative duration values arise from arithmetic formulae. However, a negative duration value does not *quantify* any duration.
Possibility: If duration value$_1$ *is* a precise duration value then the number is fractional.
Example: 5.5 days *quantifies* the duration that *equals* 5.5 *times* 1 day
Necessity: 3 months *equals* ¼ *times* 'year.'
Necessity: 6 months *equals* ½ *times* 'year.
Necessity: 6 months *equals* 2/4 *times* 'year.'
Necessity: 9 months *equals* ¾ *times* 'year.'
Note: This specification defines only the fractional nominal duration values ¼ year, ½ year, 2/4 year, and ¾ year because these are in common business use and they equal an integral number of months.
Example: 5.5 years *quantifies* the duration that *equals* 5.5 *times* 1 year

## 9.4 Duration Value Comparison

Comparison of duration values is defined in terms of the same operations on the *quantified* durations or *specified* duration value sets. The benefit of the unusual semantic for nominal duration values is that these comparisons have useful results for many nominal duration values. For example, the expression "1 year 1 day > 365 days" is true for both possible duration values that are *specified by* 1 year 1 day.

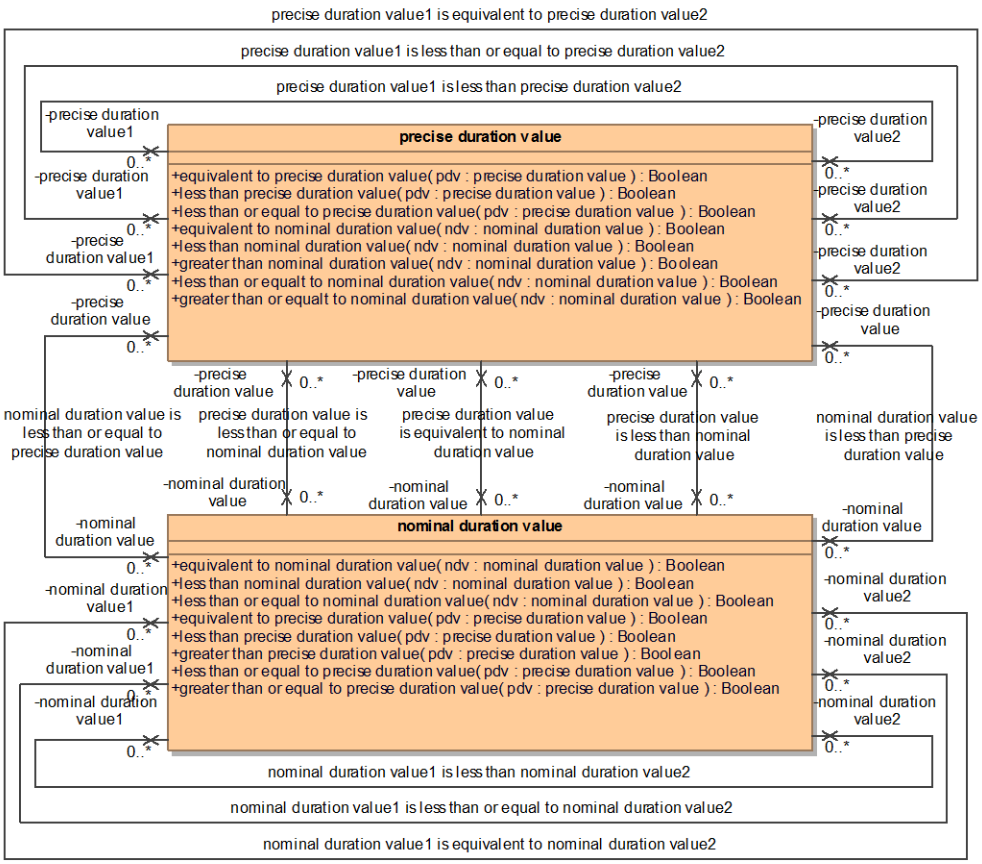


**Figure 9.5 - Duration Value Comparison**

## precise duration value$_1$ *is equivalent to* precise duration value$_2$

Synonymous Form: precise duration value$_1$ *equals* precise duration value$_2$
Synonymous Form: precise duration value$_1$ = precise duration value$_2$
Definition: precise duration value$_1$ *quantifies* duration$_1$ and precise duration value$_2$ *quantifies* duration$_2$ and duration$_1$ = duration$_2$
Example: "3 days 12 hours" *is equivalent to* "84 hours"

## nominal duration value$_1$ *is equivalent to* nominal duration value$_2$

Synonymous Form: nominal duration value$_1$ *equals* nominal duration value$_2$
Synonymous Form: nominal duration value$_1$ = nominal duration value$_2$
Definition: nominal duration value$_1$ = duration value set$_1$
and nominal duration value$_2$ = duration value set$_2$
and duration value set$_1$ = duration value set$_2$

Example: "1 month" *is equivalent to* "1 month"
Example: " 1 year 1 day" *is* not *equivalent to* "366 days"

## precise duration value *is equivalent to* nominal duration value

Synonymous Form: precise duration value *equals* nominal duration value
Synonymous Form: precise duration value = nominal duration value
Synonymous Form: nominal duration value *is equivalent to* precise duration value
Synonymous Form: nominal duration value *equals* precise duration value
Synonymous Form: nominal duration value = precise duration value
Definition: nominal duration value *quantifies* a duration value set and precise duration value *quantifies* a duration that = some duration *of* the duration value set
Example: "28 days" *is equivalent to* "1 month"

## precise duration value$_1$ *is less than or equal to* precise duration value$_2$

Synonymous Form: precise duration value$_2$ *is greater than or equal to* precise duration value$_1$
Synonymous Form: precise duration value$_1$ ≤ precise duration value$_2$
Synonymous Form: precise duration value$_2$ ≥ precise duration value$_1$
Definition: precise duration value$_1$ *quantifies* duration$_1$ and precise duration value$_2$ *quantifies* duration$_2$ and duration$_1$ ≤ duration$_2$
Example: "1 hour 30 minutes" *is less than or equal to* "2 days 30 minutes"

## nominal duration value$_1$ *is less than or equal to* nominal duration value$_2$

Synonymous Form: nominal duration value$_2$ *is greater than or equal to* nominal duration value$_1$
Synonymous Form: nominal duration value$_1$ ≤ nominal duration value$_2$
Synonymous Form: nominal duration value$_2$ ≥ nominal duration value$_1$
Definition: nominal duration value$_1$ *quantifies* duration value set$_1$ and nominal duration value$_2$ *quantifies* duration value set$_2$ and duration value set$_1$ ≤ duration value set$_2$
Example: "1 month 1 day" *is less than or equal to* "1 month 2 days"

## precise duration value *is less than or equal to* nominal duration value

Synonymous Form: precise duration value ≤ nominal duration value
Synonymous Form: nominal duration value *is greater than or equal to* precise duration value
Synonymous Form: nominal duration value ≥ precise duration value
Definition: precise duration value *quantifies* duration and nominal duration value *quantifies* duration value set and duration ≤ duration value set
Example: "366 days" *is less than or equal to* "1 year 1 day"

## nominal duration value *is less than or equal to* precise duration value

Synonymous Form: nominal duration value ≤ precise duration value
Synonymous Form: precise duration value *is greater than or equal to* nominal duration value
Synonymous Form: precise duration value ≥ nominal duration value
Definition: nominal duration value *quantifies* duration value set and precise duration value *quantifies* duration and duration value set ≤ duration
Example: "2 years 1 day" *is less than or equal to* "732 days"

## precise duration value$_1$ *is less than* precise duration value$_2$

Synonymous Form: precise duration value$_2$ *is greater than* precise duration value$_1$
Synonymous Form: precise duration value$_1$ < precise duration value$_2$
Synonymous Form: precise duration value$_2$ > precise duration value$_1$
Definition: precise duration value$_1$ *quantifies* duration$_1$ and precise duration value$_2$ *quantifies* duration$_2$ and duration$_1$ < duration$_2$

Example: "1 hour 30 minutes" *is less than* "91 minutes"

### nominal duration value$_1$ *is less than* nominal duration value$_2$

Synonymous Form: nominal duration value$_2$ *is greater than* nominal duration value$_1$
Synonymous Form: nominal duration value$_1$ < nominal duration value$_2$
Synonymous Form: nominal duration value$_2$ > nominal duration value$_1$
Definition: nominal duration value$_1$ *quantifies* duration value set$_1$ and nominal duration value$_2$ *quantifies* duration value set$_2$ and duration value set$_1$ < duration value set$_2$
Example: "1 month 1 day" *is less than* "1 month 2 days"

### precise duration value *is less than* nominal duration value

Synonymous Form: precise duration value < nominal duration value
Synonymous Form: nominal duration value *is greater than* precise duration value
Synonymous Form: nominal duration value > precise duration value
Definition: precise duration value *quantifies* duration and nominal duration value *quantifies* duration value set and duration < duration value set
Example: "366 days" *is less than* "1 year 2 days"

### nominal duration value *is less than* precise duration value

Synonymous Form: nominal duration value < precise duration value
Synonymous Form: precise duration value *is greater than* nominal duration value
Synonymous Form: precise duration value > nominal duration value
Definition: nominal duration value *quantifies* duration value set and precise duration value *quantifies* duration and duration value set < duration
Definition: "1 month 1 day" *is less than* "34 days"

## 9.5 Duration Value Sets

This sub clause defines the concept 'duration value set' and those relationships of that concept that are needed to semantically ground other features of this specification.

### duration value set

Definition: set *of* duration values
Possibility: the cardinality *of* a duration value set *is* 0
Example: the duration value set that *is quantified by* {60 seconds, 64 seconds}

The following concepts support comparison of two duration value sets.

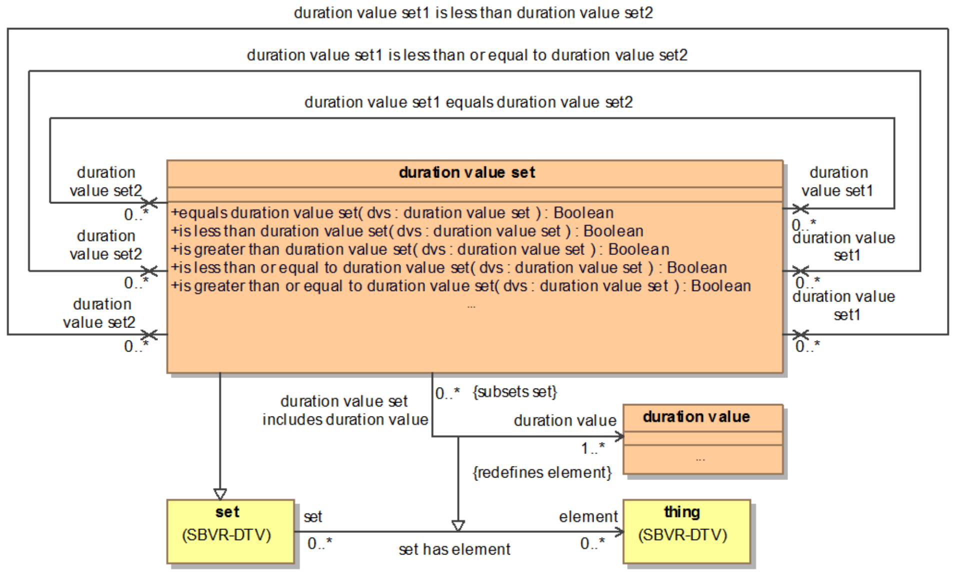


**Figure 9.6 - Duration Value Set Comparisons**

## duration value set$_1$ *equals* duration value set$_2$

Synonymous Form: duration value set$_1$ *is equal to* duration value set$_2$
Synonymous Form: duration value set$_1$ *is equivalent to* duration value set$_2$
Synonymous Form: duration value set$_1$ = duration value set$_2$
Definition: each duration$_1$ *of* duration value set$_1$ = some duration$_2$ *of* duration value set$_2$ and each duration$_2$ *of* duration value set$_2$ = some duration$_1$ *of* duration value set$_1$
Example: the duration value set {1 week, 2 weeks} *equals* the duration value set {7 days, 14 days}
Example: the duration value set {1 day, 2 days} *equals* the duration value set {2 days, 1 day}

## duration value set$_1$ *is less than or equal to* duration value set$_2$

Synonymous Form: duration value set$_2$ *is greater than or equal to* duration value set$_1$
Synonymous Form: duration value set$_1$ ≤ duration value set$_2$
Synonymous Form: duration value set$_2$ ≥ duration value set$_1$
Definition: each duration value$_1$ *of* duration value set$_1$ *is less than or equal to* each duration value$_2$ *of* duration value set$_2$
Example: the duration value set {1 day, 2 days} *is less than or equal to* the duration value set {2 days, 4 days}

## duration value set$_1$ *is less than* duration value set$_2$

Synonymous Form: duration value set$_2$ *is greater than* duration value set$_1$
Synonymous Form: duration value set$_1$ < duration value set$_2$
Synonymous Form: duration value set$_2$ > duration value set$_1$
Definition: each duration value$_1$ *of* duration value set$_1$ *is less than* each duration value$_2$ *of* duration value set$_2$
Example: the duration value set {1 day, 2 days} *is less than* the duration value set {3 days, 4 days}

Durations can be compared with duration value sets.

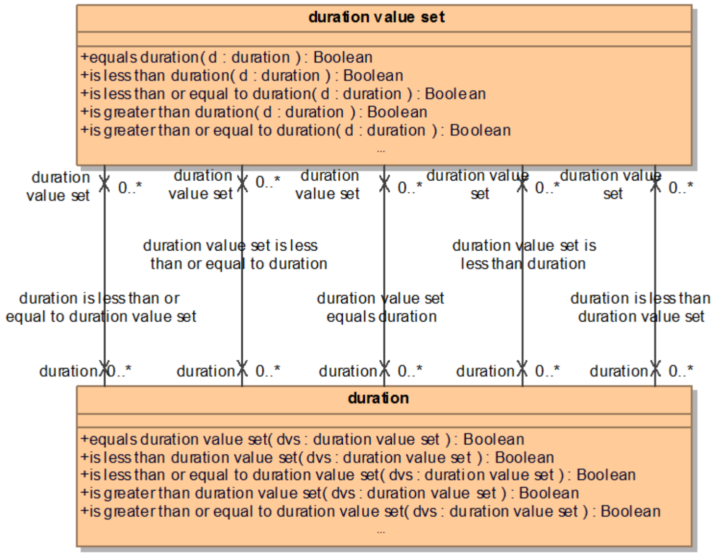


**Figure 9.7 - Comparisons among Duration Value Sets and Durations**

## duration value set *equals* duration

| | |
|---|---|
| Synonymous Form: | duration = duration value set |
| Synonymous Form: | duration value set = duration |
| Synonymous Form: | duration *equals* duration value set |
| Synonymous Form: | duration value set *is equivalent to* duration |
| Synonymous Form: | duration *is equivalent to* duration value set |
| Definition: | each duration value *of* the duration value set *equals* the given duration |
| Example: | the duration value set {1 day} *equals* the duration that *is quantified by* 1 day |

## duration value set *is less than or equal to* duration

| | |
|---|---|
| Synonymous Form: | duration *is greater than or equal to* duration value set |
| Synonymous Form: | duration value set ≤ duration |
| Synonymous Form: | duration ≥ duration value set |
| Definition: | each duration value *of* the duration value set *is less than or equal to* the given duration |
| Example: | the duration value set {1 day, 2 days} *is less than or equal to* the duration that *is quantified by* 2 days |

## duration *is less than or equal to* duration value set

| | |
|---|---|
| Synonymous Form: | duration value set *is greater than or equal to* duration |
| Synonymous Form: | duration ≤ duration value set |
| Synonymous Form: | duration value set ≥ duration |
| Definition: | duration *is less than or equal to* each duration value *of* the duration value set |
| Example: | the duration that *is quantified by* 28 days *is less than or equal to* the duration value set {28 days, 29 days} |

## duration value set *is less than* duration

| | |
|---|---|
| Synonymous Form: | duration *is greater than* duration value set |
| Synonymous Form: | duration value set < duration |
| Synonymous Form: | duration > duration value set |
| Definition: | each duration value *of* the duration value set *is less than* the given duration |
| Example: | the duration value set {1 day, 2 days} *is less than* the duration that *is quantified by* 3 days |

## duration *is less than* duration value set

| | |
|---|---|
| Synonymous Form: | duration value set *is greater than* duration |
| Synonymous Form: | duration < duration value set |
| Synonymous Form: | duration value set > duration |
| Definition: | duration *is less than* each duration value *of* the duration value set |
| Example: | the duration that *is quantified by* 364 days *is less than* the duration value set {365 days, 366 days} |

Specification of compound nominal duration values as duration value sets requires addition and subtraction among durations and duration value sets, and addition and subtraction among two duration value sets.

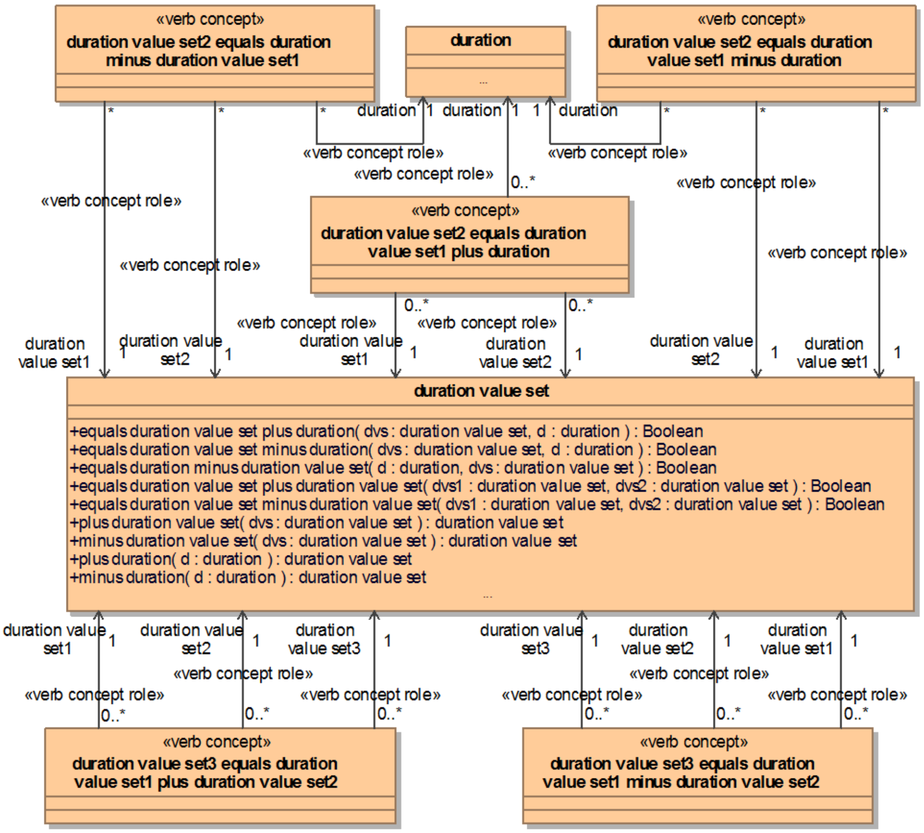


**Figure 9.8 - Duration Value Set Arithmetic**

## duration value set$_2$ *equals* duration value set$_1$ *plus* duration

| | |
|---|---|
| Synonymous Form: | duration value set$_2$ *equals* duration *plus* duration value set$_1$ |
| Synonymous Form: | duration value set$_2$ = duration value set$_1$ + duration |
| Synonymous Form: | duration value set$_2$ = duration + duration value set$_1$ |
| Synonymous Form: | duration value set$_1$ + duration |
| Synonymous Form: | duration + duration value set$_1$ |
| Synonymous Form: | duration *plus* duration value set$_1$ |
| Synonymous Form: | duration value set$_1$ *plus* duration |
| Definition: | each duration value$_2$ *of* the duration value set$_2$ *equals* some duration value$_1$ *of* duration value set$_1$ *plus* the duration |
| Necessity: | For each duration value set$_1$ and for each duration, exactly one duration value set$_2$ *is* the duration value set$_1$ *plus* the duration. |
| Example: | the duration value set {3 days, 4 days} *equals* the duration that *is quantified by* 2 days *plus* the duration value set {1 day, 2 days} |

## duration value set$_3$ *equals* duration value set$_1$ *plus* duration value set$_2$

Synonymous Form: duration value set$_3$ = duration value set$_1$ + duration value set$_2$
Definition: each duration value$_3$ *of* duration value set$_3$ =
some duration value$_1$ *of* duration value set$_1$
+ some duration value$_2$ *of* duration value set$_2$,
where the duration value$_1$ and duration value$_2$ are selected to form a Cartesian product of duration value set$_1$ and duration value set$_2$
Note: The result set disregards duplicates. Hence the cardinality of duration value set$_3$ may be less than the product of the cardinalities of duration value set$_1$ and duration value set$_2$.
Necessity: For each duration value set$_1$ and for each duration value set$_2$, exactly one duration value set$_3$ *is* the duration value set$_1$ *plus* the duration value set$_2$.
Example: the duration value set {4 days, 5 days, 6 days} *equals* the duration value set {1 day, 2 days} *plus* the duration value set
{3 days, 4 days}

## duration value set$_2$ *equals* duration value set$_1$ *minus* duration

Synonymous Form: duration value set$_2$ = duration value set$_1$ – duration
Synonymous Form: duration value set$_1$ *minus* duration
Synonymous Form: duration value set$_1$ – duration
Definition: each duration value$_1$ *of* duration value set$_1$ ≥ the duration and each duration value$_2$ *of* the duration value set$_2$ = some duration value$_1$ *of* duration value set$_1$ – the duration
Necessity: For each duration value set$_1$ and for each duration that *is less than or equal to* each duration value$_1$ *of* duration value set$_1$, exactly one duration value set$_2$ *is* the duration value set$_1$ *minus* the duration.
Example: the duration value set {2 days, 0 days} = the duration value set {3 days, 1 day} – the duration that *is quantified by* 1 day

## duration value set$_2$ *equals* duration *minus* duration value set$_1$

Synonymous Form: duration value set$_2$ = duration – duration value set$_1$
Synonymous Form: duration – duration value set$_1$
Synonymous Form: duration *minus* duration value set$_1$
Definition: each duration value$_1$ *of* duration value set$_1$ ≤ the duration and each duration value$_2$ *of* duration value set$_2$ = the duration – some duration value$_1$ *of* duration value set$_1$
Necessity: For each duration value set$_1$ and for each duration that *is greater than or equal to* each duration value$_1$ *of* duration value set$_1$, exactly one duration value set$_2$ *is* the duration *minus* the duration value set$_1$.
Example: the duration value set {1 day, 0 days} = the duration that *is quantified by* 2 days - the duration value set {1 day, 2 days}

## duration value set$_3$ *equals* duration value set$_1$ *minus* duration value set$_2$

Synonymous Form: duration value set$_3$ = duration value set$_1$ – duration value set$_2$
Synonymous Form: duration value set$_1$ *minus* duration value set$_2$
Synonymous Form: duration value set$_1$ – duration value set$_2$
Definition: duration value set$_2$ ≤ duration value set$_1$ and
each duration value$_3$ *of* duration value set$_3$ =
some duration value$_1$ *of* duration value set$_1$
- some duration value$_2$ *of* duration value set$_2$,
where the duration value$_1$ and duration value$_2$ are selected to form a Cartesian product of duration value set$_1$ and duration value set$_2$

Note: The result set disregards duplicates. Hence the cardinality of duration value set$_3$ may be less than the product of the cardinalities of duration value set$_1$ and duration value set$_2$.

Necessity: For each duration value set$_1$ and for each duration value set$_2$ that *is less than or equal to* duration value set$_1$, exactly one duration value set$_3$ *is* the duration value set$_1$ *minus* the duration value set$_2$.

Example: the duration value set {-1 days, 0 days, 2 days, 3 days } = the duration value set {3 days, 4 days} - the duration value set {1 days, 4 days}

____________________________________________________
____________________________________________________

# 10 Calendars (normative)

## 10.1 General

Calendars use time scales to impose structure on time.

____________________________________________________

### Calendars Vocabulary

General Concept: terminological dictionary
Language: English
Included Vocabulary: Time Infrastructure Vocabulary
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#CalendarsVocabulary

____________________________________________________

## 10.2 Calendar Fundamentals

This sub clause contains definitions true of calendars in general.

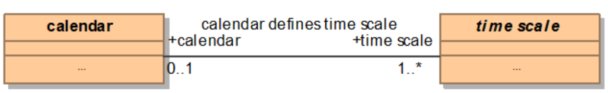


**Figure 10.1 - Calendars**

### calendar

Definition: system of time scales specified by a combination of concepts and rules

Note: This concept of calendar can include any date-time conceptual schema, of any granularity. This is more general than the usual calendar concept, which limits the finest granularity to "day". The two most prominent calendars are the Gregorian, whose finest granularity is "day", and the Universal Coordinated Time (UTC), whose finest granularity is "second". UTC uses the Gregorian calendar to get to a day and extends it to define the time of day down to a second calendar.

Note: There are many different calendars, some standard, some cultural, some defined for particular business needs.
Example: Gregorian calendar, lunar calendars, fiscal calendars, manufacturing calendars, tax calendars, religious calendars.
Reference Scheme: the time scales that *are defined by* a calendar

## calendar *defines* time scale

Synonymous Form: time scale *is defined by* calendar
Synonymous Form: time scale *of* calendar
Synonymous Form: time scale *on* calendar
Definition: the calendar specifies the details of the time scale
Example: The Gregorian calendar *defines* the Gregorian year time scale with other time scales.

# 10.3 Calendar Time Points and Time Periods

This sub clause defines categories of time points and time periods that *indicate* time intervals with duration 'day', 'month', or 'year', but are independent of any particular calendar design. These concepts are intended to apply to religious and cultural calendars as well as the Gregorian calendar.

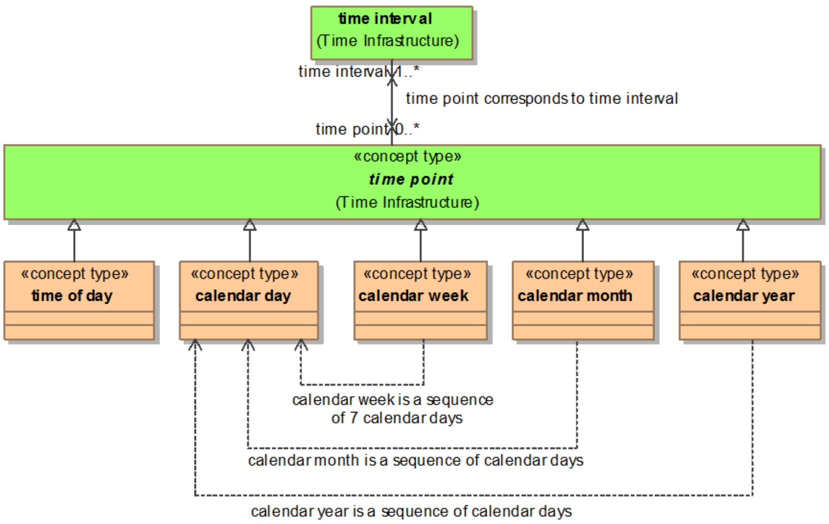


**Figure 10.2 - Calendar Time Points**

## calendar year

Dictionary Basis: ISO 8601 (2.2.13, 'calendar year')
Concept Type: concept type
Definition: time point that is defined by a given calendar as a consecutive sequence of calendar days, during which approximately one orbital rotation of the Earth around the Sun is completed
Note: See "Gregorian year".
Example: the year 2008 (as defined by the Gregorian calendar)
Example: the 15th year of the reign of the Pharaoh Akhenaton

## calendar month

Concept Type: concept type
Definition: time point that is defined by a given calendar as a consecutive sequence of calendar days in a calendar year, during which approximately one rotation of the Moon in its orbit around the Earth is completed
Example: August, 1945 (as defined by the Gregorian calendar)
Example: Ramadan in the 63rd year of the Prophet Mohammed

## calendar week

Concept Type: concept type
Definition: time point that is defined by a given calendar as 7 consecutive calendar days
Dictionary Basis: ISO 8601 (2.2.8, 'calendar week')
Note: ISO 8601 adds "starting on a Monday" to this definition. This vocabulary drops that phrase because it is culture-specific.
Note: This specification introduces two specific calendar week concepts: 'ISO week' and 'ISO week of year', both of which adopt the ISO 8601 convention that weeks start on Monday. See Clause 12.
Example: The third calendar week of 2009.

## calendar day

Concept Type: concept type
Definition: time point that is defined by a given calendar, and that *corresponds to* time intervals during which approximately one revolution of the Earth occurs on its axis
Necessity: For each calendar, each instance *of* each calendar day that *is defined by* the calendar *is met by* at most one instance *of* a calendar day that *is defined by* the calendar.
Example: July 4, 1776 (as defined by the Gregorian calendar)
Example: The time period from sunrise in Rome on the Ides of March in the year 753 after the founding of the City to the following sunrise.

## time of day

Definition: time point that *is on* a time scale that *has* a granularity that *is less than* 1 day
Note: time of day time points are defined and discussed in detail in sub clause 13.2. The intent here is that such time scales may be defined by a calendar.
Example: hour of day, second of minute

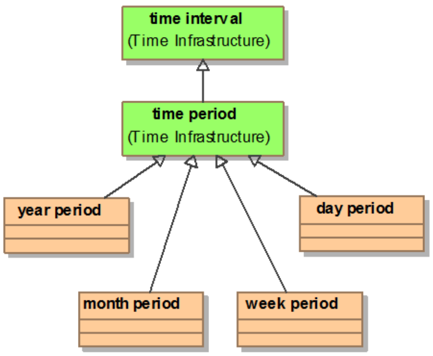

**Figure 10.3 - Time periods based on calendars**

## year period

Dictionary Basis: ISO 8601 (2.2.14, note 1)
Definition: time period which starts at a certain time of day at a certain calendar date of the calendar year and ends at the same time of day at the same calendar date of the next calendar year, if it exists. In other cases, the ending calendar date is defined by agreement.
Note: A calendar year corresponds to time periods that start and end as defined by a calendar.
Note: A year period starts at any time within an instance of a calendar year.
Example: The concept "fiscal year" defined as the year period from midnight of July 1 of one calendar year to midnight of July 1 of the following calendar year.

## month period

Source: ISO 8601 (2.2.12, note 1)
Definition: time period that starts at a certain time of day at a certain calendar date of the calendar month and ends at the same time of day at the same calendar date of the next calendar month, if it exists. In other cases, the ending calendar date is defined by agreement.
Note: A calendar month corresponds to time periods that start and end as defined by a calendar.
Note: A month period starts at any time within an instance of a calendar month.
Example: From July 15 at noon to August 15 at noon.

## week period

Definition: time period that starts at a certain time of day on a certain calendar day of the calendar week and ends at the same time of day at the same calendar day of the next calendar week.
Note: A calendar week is a period that starts and ends as defined by a calendar. A week period starts and ends at any time within a calendar week.
Example: Tuesday to Tuesday.

## day period

Definition: time period that begins and ends at the same local time of day on consecutive calendar days
Note: A calendar day corresponds to time periods that start and end as defined by a calendar.
Note: A day period starts at any time of day within an instance of a calendar day.
Note: A day period is defined by starting and ending at the same local time of day. When the local time of day is affected by a change of time offset between the starting and ending time intervals, the day period can have a duration that is not 24 hours. The duration of a month period or a year period may also be affected by changes in the time offset for the local time of day.
Example: Noon one calendar day to noon the following calendar day.

## 10.4 Time Point Subdivision

The purpose of finite time scales is to provide finer-grained resolution of time intervals within the time intervals that are instances of time points with coarser granularities. In this specification, the relationship between a finite time scale and a coarser time point is called "time point subdivision". Many finite time scales are defined by the category of time point they subdivide and the granularity of the time points they contain.

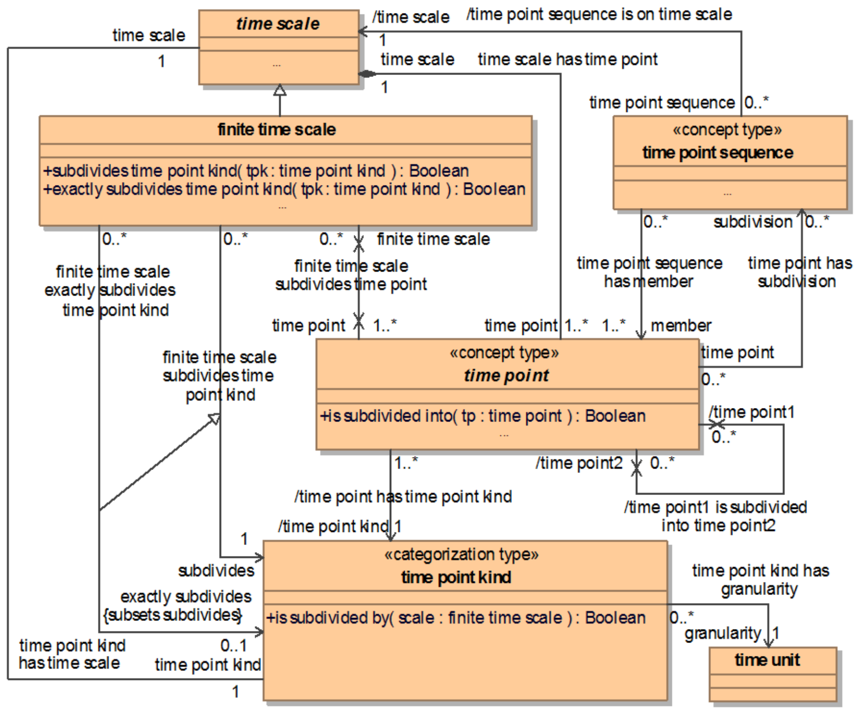

Figure 10.4 - Time Point Subdivision

## time point kind

Concept Type: categorization type
Definition: concept that *has* an extension that *is* the set *of* members *of* exactly one time scale
Necessity: Each time point kind *specializes* the concept 'time point'.
Necessity: The concept 'time point kind' *is* a categorization type that *is for* the concept 'time point'.
Note: 'Time point kind' is a *partial* categorization of 'time point'. A concept like 'time of day' specializes 'time point', but its extension is not just the members of one time scale.

## time point kind *has* time scale

Synonymous Form: time scale *defines* time point kind
Definition: each time point that *is* an instance *of* the time point kind *is* a member *of* the time scale
Necessity: Each time point kind *has* exactly one time scale.
Necessity: Each time scale *defines* exactly one time point kind.

## time point kind *has* granularity

Definition: the granularity *is* the granularity *of* the time scale *of* the time point kind
Necessity: Each time point kind *has* exactly one granularity.

## finite time scale *subdivides* time point

Definition: each instance *of* the time point *is* an instance *of* a time point sequence that *is on* the finite time scale and that has a first time point that *is* the index origin member *of* the finite time scale

Note: This verb concept is defined primarily to simplify other definitions.

## finite time scale *subdivides* time point kind

Definition: the finite time scale *subdivides* each time point that *is* an instance *of* the time point kind

Note: This verb concept describes the purpose of the finite time scale: each time point of the finite time scale corresponds to time intervals according to their position relative to the start of a time interval that is an instance of some time point of the time point kind. The first time point of the finite time scale corresponds to time intervals that start the larger time intervals and have a duration equal to the granularity of the finite time scale.

Necessity: The granularity *of* each finite time scale *is less than* the granularity *of* each time point kind that the finite time scale *subdivides*.

Note: The same time point sequence may be the subdivision of all time points of the time point kind, or different time points may be subdivided into time point sequences of different lengths.

Example: The day of hours scale *subdivides* Gregorian calendar day. Every time point that is a Gregorian calendar day is subdivided into 24 hour of day time points, and each corresponding time interval is divided into 24 time intervals, each of which is an instance of one hour of day.

Note: The time point sequence may correspond to more time intervals than the instances of the time point. For example, the day of hours time scale subdivides ISO day of week and day of month, but the time point sequence that is hour of day 0 to hour of day 23 corresponds to every one day time interval, not just every Tuesday and every first of the month.

Example: The Gregorian month of days scale *subdivides* month of year. Every time point that is a Gregorian month of year is subdivided into some number of day of month time points, and the time point sequences all begin with day of month 1, but the length of the time point sequence depends on which month time point is subdivided.

## subdivision

Concept Type: role
General Concept: time point sequence
Definition: time point sequence that *is coextensive with* a given time point

## time point *has* subdivision

Definition: the subdivision *is* a time point sequence that *corresponds to* each instance *of* the time point and that *is on* some finite time scale that *subdivides* the time point

Possibility: A time point *has* no subdivision.

Possibility: A time point *has* more than one subdivision.

## time point$_1$ *is subdivided into* time point$_2$

Definition: the subdivision *of* time point$_1$ *includes* time point$_2$

Note: This verb concept describes the relationship between a time point$_1$ and each individual time point$_2$ of a kind that *subdivides* it. In this specification it is used primarily to express the cardinality of subdivisions.

Example: Gregorian day 3 January 2010 *is subdivided into* exactly 24 'hour of day' time points. The time interval corresponding to Gregorian date 3 January 2010 is implicitly subdivided into 24 time intervals, each of which is an instance of one hour of day. But that same 24-hour time point sequence is the subdivision of every Gregorian day, and it corresponds to every time interval that is an instance of a Gregorian day.

## finite time scale *exactly subdivides* time point kind

Definition: for each time point that *is* an instance *of* time point kind, the time point sequence that *is* the finite time scale *corresponds to* each time interval that *is* an instance *of* the time point

Necessity: Each *finite time scale* that *exactly subdivides* a time point kind *subdivides* the time point kind.

Necessity: If a finite time scale *exactly subdivides* a time point kind$_1$, and each time point *of* the finite time scale *is* an instance *of* a time point kind$_2$, then the number *of* time point kind$_2$ that each time point that *is* an instance *of* time point kind$_1$ *has* *is* the cardinality *of* the finite time scale.

Example: The day of hours scale exactly subdivides Gregorian day of month. Every Gregorian day of month therefore has 24 of 'hour of day', because 24 is the cardinality of 'day of hours'.

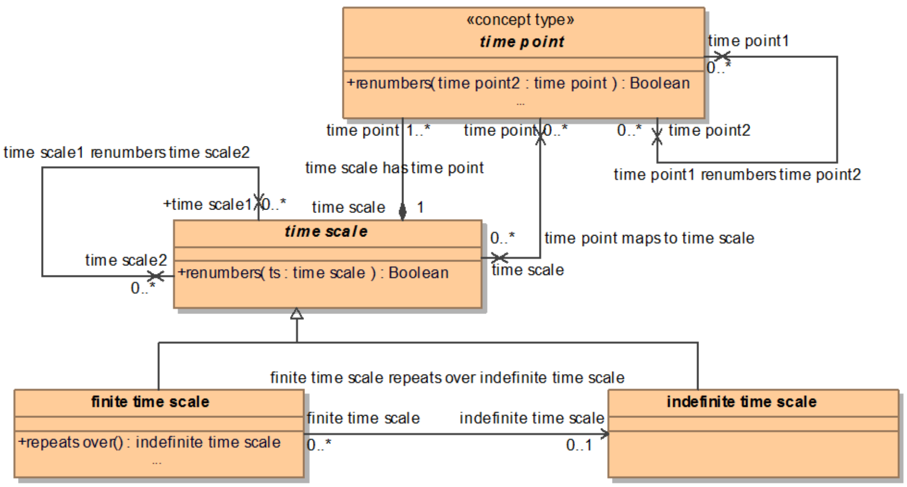


**Figure 10.5 - Time Scale Renumbering**

## time point *maps to* time scale

Definition: the time point *is* not *on* the time scale and each time interval that *is* an instance *of* the time point *is* an instance *of* some time point *of* the time scale

Note: This concept is introduced in order to simplify the definitions of time scale$_1$ *renumbers* time scale$_2$ and time point$_1$ *renumbers* time point$_2$.

Example: Every day-of-year on the year of days time scale (see xxx) maps to the indefinite time scale of calendar days. All of the time intervals involved are instances of calendar day.

## time point$_1$ *renumbers* time point$_2$

Synonymous Form: time point$_2$ *is renumbered by* time point$_1$

Definition: time point$_1$ *maps to* the time scale *of* time point$_2$ and time point$_2$ *specializes* time point$_1$

Description: Every time interval that is an instance of time point$_2$ is also an instance of time point$_1$

Possibility: A time point$_1$ *renumbers* more than one time point.

Note: In particular, a time point on a finite time scale can renumber an indefinite number of time points on an indefinite time scale

Example: Every day-of-year on the year of days time scale renumbers a set of time points on the indefinite time scale of calendar days

## time scale$_1$ *renumbers* time scale$_2$

Definition: each time point *of* time scale$_1$ *renumbers* some time point *of* time scale$_2$ and each time point *of* time scale$_2$ *is renumbered by* some time point *of* time scale$_1$

Necessity: The granularity *of* each time scale$_2$ that a time scale$_1$ *renumbers* *is* the granularity *of* time scale$_1$.

## finite time scale *repeats over* indefinite time scale

Definition: the finite time scale *renumbers* the indefinite time scale and each time point$_1$ *of* the indefinite time scale *is renumbered by* the time point$_3$ that *is on* the finite time scale and that *is just before* the time point$_4$ that *renumbers* the time point$_2$ that *is next after* time point$_1$, if time point$_2$ *is* not *renumbered by* the index origin member *of* the finite time scale

Description: Consecutive time points on the finite time scale renumber consecutive time points on the infinite time scale, and at some point the finite time scale starts over beginning with the origin time point.

Note: Figure 10.6 shows the relationship of a finite time scale to an indefinite time scale that it repeats over. The arrows show correspondence to time intervals.
The time points of the finite time scale, beginning at the origin, correspond to time intervals that are instances of time points on the indefinite time scale. So, in particular, time point O renumbers time point M and time point N, because it corresponds to the same time intervals. Further, time points M+1 and N+1 are renumbered by time point O+1, and similarly time points M+2 and N+2 are renumbered by time point O+2, and so on. This is the requirement stated in the definition above. Some " last" time point (T) on the finite time scale, however, renumbers the time point that is just before time point N, because the origin time point (O) renumbers time point N.

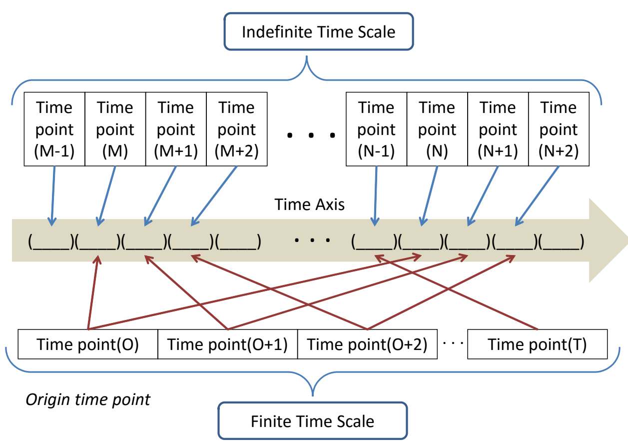


**Figure 10.6 - Time point renumbering**

Note: It is possible that time point (T) does not renumber the time point that is just before time point (M). Some other time point on the finite time scale (e.g., T-1) may renumber that time point. It is not a requirement that the entire finite time scale is repeated in every instance.

For example, the month of days (finite) time scale renumbers the Gregorian days (indefinite) time scale. The month of days has 31 day of month time points, but the repeating process can start over after index 30, or 29, or 28, as well.

## 10.5 Time Coordinates

A time coordinate is a conceptual structure of meaning that *refers to* time intervals using time scales. A time coordinate that refers to exactly one time interval is called an absolute time coordinate. When a time coordinate incorporates a year number, it is always an absolute time coordinate. For example, "January 3, 2011" refers to exactly one day over all time. A time coordinate that refers to more than one time interval is called a relative time coordinate. When a time coordinate omits the year, it is usually relative. For example, "January 3" refers to one day in every calendar year.

An atomic time coordinate is said to *indicate* a time point on some time scale, either by its name or by its number (called its index). For example, "January" indicates a Gregorian month of year time point, and "day of month 3" indicates a day of month time point. The atomic time coordinate *refers to* all the time intervals that are instances of that time point.

A compound time coordinate describes a category of the concept 'time interval', by *combining* multiple time coordinates to create a set of atomic time coordinates on different time scales. The compound time coordinate *refers to* the time intervals that are instances of the smallest granularity time point and that are contained in instances of the larger ones. For example, "July 1" is a compound time coordinate that refers to instances of 'day of month 1' that are part of an instance of July. Compound time coordinates don't always indicate time points. ("July 1" does not indicate a time point; because of leap years, it is not always the same day of year. "July 1, 2011", however, indicates a time point on the indefinite time scale of Gregorian days.)

Examples are "July 1, 2010 12:43:55", "ISO week of year 41 ISO day of week 6", and "1999 day 45". Clauses 11, 12, and 13 specify which combinations of atomic time coordinates form legitimate compound time coordinates. Invalid combinations typically omit intermediate time units. For example, "2011 12:43:55" makes no sense.

This specification does NOT specify how time coordinates are externally represented, for example on a monitor or in printed form. Many different external formats are employed among different languages and cultures. Representation formats are the choice of individual tools.

When more than one time coordinate *refers to* exactly the same time intervals, they are said to be *equivalent*. For example, "January 3, 2011" *is equivalent to* "2011 day 3" because the two time coordinates *refer to* the same calendar day time interval. Determining equivalence is not easy because of the incorporation of leap days in some calendar years. For example, whether the 182nd day of the calendar year is before or the same as July 1 of the same calendar year depends upon whether the calendar year is a leap year.

### 10.5.1 General

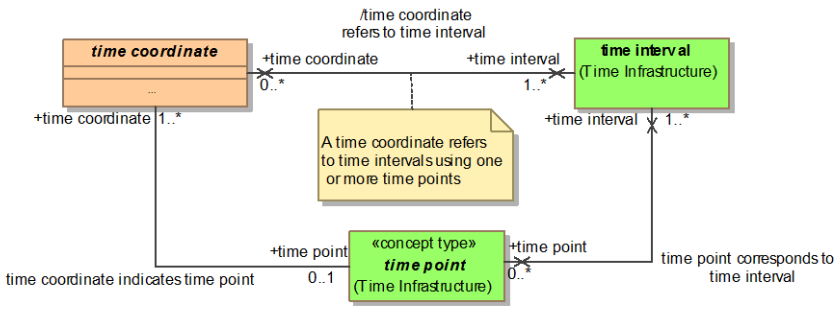


**Figure 10.7 - Time Coordinate**

### time coordinate

| | |
|---|---|
| Synonym: | time stamp |
| Definition: | conceptual structure of meaning that characterizes a category of the concept 'time interval' |
| Reference Scheme: | an expression that *represents* the time coordinate |
| Example: | January 2009, 2009 month 1, 2009 |
| Note: | Time coordinates may be either absolute or relative (see sub clause 10.6.2). |
| Note: | Time coordinates may be either atomic or compound (see sub clause 10.6.3). |
| Necessity: | Each time coordinate *is* either an absolute time coordinate or a relative time coordinate. |
| Necessity: | Each time coordinate *is* either an atomic time coordinate or a compound time coordinate. |
| Note: | Particular kinds of time coordinates are defined in Clauses 11, 12, and 13. |

### time coordinate *indicates* time point

| | |
|---|---|
| Definition: | the time coordinate characterizes the time point, either by instantiating a reference scheme for the concept 'time point', or by characterizing the time intervals that the time point *corresponds to* |
| Necessity: | Each time coordinate *indicates* at most one time point. |
| Possibility: | A time point *is indicated by* more than one time coordinate. |
| Note: | Atomic time coordinates and compound time coordinates indicate time points in different ways. Each is specified separately below. |
| Note: | See 'compound time coordinate *indicates* time point' for definitions of exactly how a compound time coordinate *indicates* a time point. |

### time coordinate *refers to* time interval

| | |
|---|---|
| Note: | The purpose of time coordinates is to identify time intervals, but atomic time coordinates and compound time coordinates do that in different ways. So this concept is separately defined for the two categories of time coordinate. |
| Necessity: | Each time coordinate *refers to* at least one time interval. |

### 10.5.2 Absolute and Relative Time Coordinates

It is convenient to distinguish between absolute time coordinates (time coordinates that include a calendar year and hence can be located on the Time Axis) and relative time coordinates (time coordinates that are relative to some larger time unit).

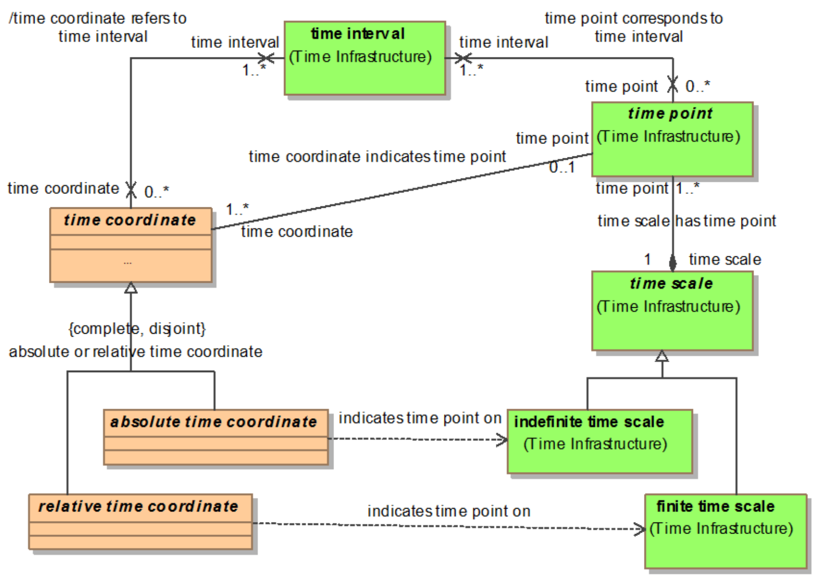


**Figure 10.8 - Absolute and Relative Time Coordinates**

## absolute time coordinate

| | |
|---|---|
| Definition: | time coordinate that *refers to* exactly one time interval |
| Necessity: | If an absolute time coordinate *indicates* a time point, the time point *is on* an indefinite time scale. |
| Necessity: | No absolute time coordinate *is* a relative time coordinate. |

## relative time coordinate

| | |
|---|---|
| Definition: | time coordinate that *refers to* more than one time interval |
| Necessity: | If a relative time coordinate *indicates* a time point, the time point *is on* a finite time scale. |
| Necessity: | No relative time coordinate *is* an absolute time coordinate. |
| Note: | A relative time coordinate *refers to* one time interval within each time period that is an instance of some time point with a greater granularity (e.g., an hour of day is part of a calendar day). Thus the relative time coordinate "recurs" in each instance of the larger time point. |
| Example: | 12 November (which recurs every calendar year) |

### 10.5.3 Atomic and Compound Time Coordinates

As with duration values, time coordinates can be atomic (reference just one time scale, as in "5 p.m.") or compound (referencing multiple time scales, as in "5:00 p.m.", which combines an hour-of-day and a minute-of-hour).

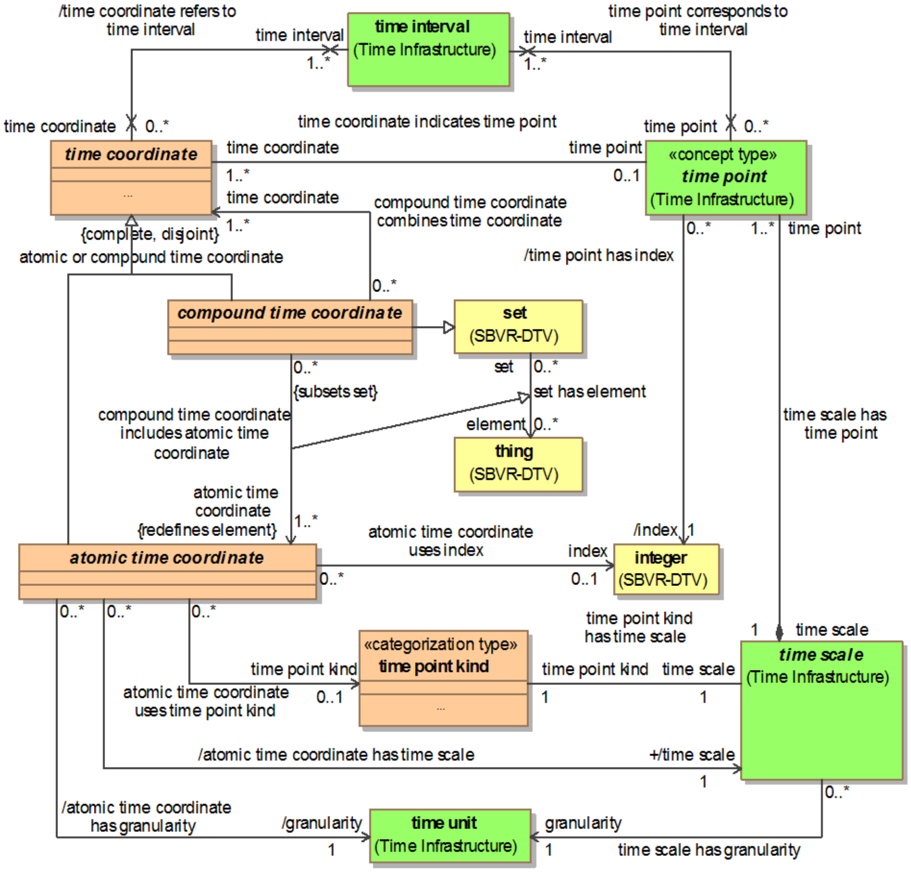


**Figure 10.9 - Atomic and Compound Time Coordinates**

## atomic time coordinate

| | |
|---|---|
| Definition: | time coordinate that *is* a term *for* a time point or that *uses* the index *of* a time point and the time point kind *of* the time point. |
| Necessity: | No atomic time coordinate *is* a compound time coordinate. |
| Note: | The two possible forms for an atomic time coordinate are based on two of the reference schemes for a time point. Expressions of these forms directly represent time points. |
| Note: | In this specification, the syntax<br><time point kind term> <index number><br>indicates a time point by representing the atomic time coordinate that consists of the time point kind of the time point and the index of the time point. |
| Example: | Tuesday |
| Example: | ISO week of year 53 |
| Example: | 2010 (understood as a reference to the time point kind Gregorian year and index ‘2010’) |

## atomic time coordinate *uses* time point kind

| | |
|---|---|
| Synonymous Form: | time point kind *of* atomic time coordinate |
| Definition: | the time scale *of* the time point kind *is* the time scale *of* the time point that the atomic time coordinate *indicates* |
| Necessity: | Each atomic time coordinate *uses* at most one time point kind. |
| Necessity: | Each atomic time coordinate that *uses* a time point kind *uses* an index. |
| Note: | Each time point kind is associated with exactly one time scale, and thus one set of time points and their indices. |

## index

| | |
|---|---|
| Concept Type: | role |
| General Concept: | integer |
| Definition: | integer that *is equal to* the index *of* the time point that a given atomic time coordinate *indicates* |

## atomic time coordinate *uses* index

| | |
|---|---|
| Synonymous Form: | index *of* atomic time coordinate |
| Definition: | the index *is* an integer that *is equal to* the index *of* the time point that *is indicated by* the atomic time coordinate |
| Necessity: | Each atomic time coordinate *uses* at most one index. |
| Necessity: | Each atomic time coordinate that *uses* an index *uses* a time point kind. |
| Note: | The time point kind specifies a time scale. The index origin value and index origin member of each time scale, which define the relationship of index values to time points, is specified in defining the time scale. In all relative time scales, the index origin member is the first member of the time scale. In the calendar time scales introduced in clauses 11, 12 and 13, the index origin value for Gregorian month of year, Gregorian day of month, Gregorian day of year, ISO week of year, and ISO day of week, use index origin value 1, while time-of-day scales (hour of day, minute of hour and second of minute) use index origin value 0. On the other hand, the index origin members and index origin values of absolute time scales are established by tradition or treaty, and related to events. |

## atomic time coordinate *indicates* time point

| | |
|---|---|
| General Concept: | time coordinate *indicates* time point |
| Necessity: | Each atomic time coordinate *indicates* exactly one time point. |
| Note: | The following rules define how the two forms of atomic time coordinate indicate time points. |
| Necessity: | Each atomic time coordinate that *is* a term *for* a time point *indicates* the time point. |
| Necessity: | Each atomic time coordinate that *uses* a time point kind and that *uses* an index *indicates* the time point that *is on* the time scale *of* the time point kind and that *has* an index that *is equal to* the index. |

## atomic time coordinate *has* time scale

| | |
|---|---|
| Synonymous Form: | time scale *of* atomic time coordinate |
| Definition: | the time point that *is indicated by* the atomic time coordinate *is on* the time scale |
| Necessity: | Each atomic time coordinate *has* exactly one time scale. |

## atomic time coordinate *has* granularity

| | |
|---|---|
| Synonymous Form: | granularity *of* atomic time coordinate |
| Definition: | the granularity *is* the granularity *of* the time scale *of* the time point that *is indicated by* the atomic time coordinate |
| Necessity: | Each atomic time coordinate *has* exactly one granularity. |

## compound time coordinate

| | |
|---|---|
| Definition: | time coordinate that *is* a set of atomic time coordinates |
| Necessity: | The cardinality *of* each compound time coordinate *is greater than* 1. |
| Necessity: | No compound time coordinate *is* an atomic time coordinate. |
| Necessity: | A compound time coordinate *refers to* a time interval$_1$ if and only if each time point that *is indicated by* an atomic time coordinate *of* the compound time coordinate *corresponds to* some time interval that *includes* time interval$_1$ and exactly one atomic time coordinate *of* the compound time coordinate *indicates* a time point that *corresponds to* time interval$_1$. |
| Note: | Each atomic time coordinate indicates one time point; and each time interval that the compound time coordinate refers to is an instance of the time point with the smallest granularity and is a part of some instance of each other time point. |
| Note: | The set of time intervals to which a compound time coordinate refers may or may not be the extension of some time point. "March 3 at noon" uses a compound time coordinate to refer to time intervals, but there is no corresponding time point. It refers to instances of noon that are part of a March and part of a day of month 3. |
| Example: | "January 2010" represents 'January' on the Gregorian year of months scale, and '2010' on the Gregorian years scale, combined to *indicate* a particular Gregorian month on the Gregorian months scale. |
| Example: | "'1 February' is the first day of February" mentions (rather than uses) "1 February". The mention means 'February' on the Gregorian year of months scale, 'day 1' on the Gregorian month of days scale, combined to *indicate* Gregorian day 32 on the Gregorian year of days scale. |
| Example: | "'1 March' is the first day of March" mentions (rather than uses) "1 March". The mention means 'March' on the Gregorian year of months scale, 'day 1' on the Gregorian month of days scale, combined to *indicate* the time set {Gregorian day 60, Gregorian day 61} on the Gregorian year of days scale. The time set models the idea that the meaning of "1 March" depends upon whether it is a common year or a leap year. |
| Example: | "Tax returns are due each 15 April." The quantifier and the use (rather than mention) of "15 April" mean a set of Gregorian days, one in each Gregorian year. |

## compound time coordinate *includes* atomic time coordinate

| | |
|---|---|
| Synonymous Form: | atomic time coordinate *of* compound time coordinate |
| Synonymous Form: | compound time coordinate *has* atomic time coordinate |
| General Concept: | set *includes* thing |
| Definition: | the atomic time coordinate *is* an element *of* the compound time coordinate |
| Necessity: | If a compound time coordinate *includes* an atomic time coordinate$_1$ and an atomic time coordinate$_2$ that *is* not atomic time coordinate$_1$, the time scale *of* atomic time coordinate$_1$ *is* not the time scale *of* atomic time coordinate$_2$. |
| Note: | That is, no two elements of a compound time coordinate indicate time points on the same time scale. |
| Example: | "2010 month 3" includes {Gregorian year 2010, Gregorian month of year 3} to indicate Gregorian month 24111 and refer to its unique instance. |

Compound time coordinates are constructed using the *combines* verb concept, which specifies a combination of time coordinates. The atomic time coordinates that are combined, and the atomic time coordinates that are elements of any compound time coordinates that are combined, together compose the set that is the compound time coordinate.

## compound time coordinate *combines* time coordinate

| | |
|---|---|
| Definition: | if the time coordinate *is* an atomic time coordinate, the time coordinate *is* an element *of* the compound time coordinate;<br>and if the time coordinate *is* a compound time coordinate, each atomic time coordinate *of* the time coordinate *is* an element *of* the compound time coordinate |

Example: A date time coordinate combines a date coordinate and a time of day coordinate. The date coordinate is a compound time coordinate that includes Gregorian year, month of year and day of month atomic time coordinates. The time of day may be given as hour of day and minute of hour atomic time coordinates. The set that is the date time coordinate includes exactly the year, month, day, hour, and minute atomic time coordinates.

## compound time coordinate *indicates* time point

Synonymous Form: time point *indicated by* compound time coordinate
General Concept: time coordinate *indicates* time point
Definition: the compound time coordinate *refers to* each instance *of* the time point, and each time interval that the compound time coordinate *refers to is* an instance *of* the time point
Note: This definition says that a compound time point indicates any time point that is coextensive with the category of time interval that the compound time coordinate characterizes. In practice, some algorithm relates the set of atomic time coordinates to a specific time point on an entirely different time scale.
Possibility: A time point *is indicated by* more than one compound time coordinate.
Note: See sub clauses 11.6, 12.4, and 13.3 for details about how atomic time coordinates *are combined* in the compound time coordinates that are defined by standard calendars.
Example: "January 4, 2010" *indicates* Gregorian day 733778

The meaning of every time coordinate is defined with respect to a particular time scale. For example, year time coordinates are defined on the Gregorian years scale. Commonly-used time coordinates are specified earlier in this document. Less commonly-used time coordinates are defined here.

A time coordinate can be absolute or relative, and atomic or compound. This yields four combinations.

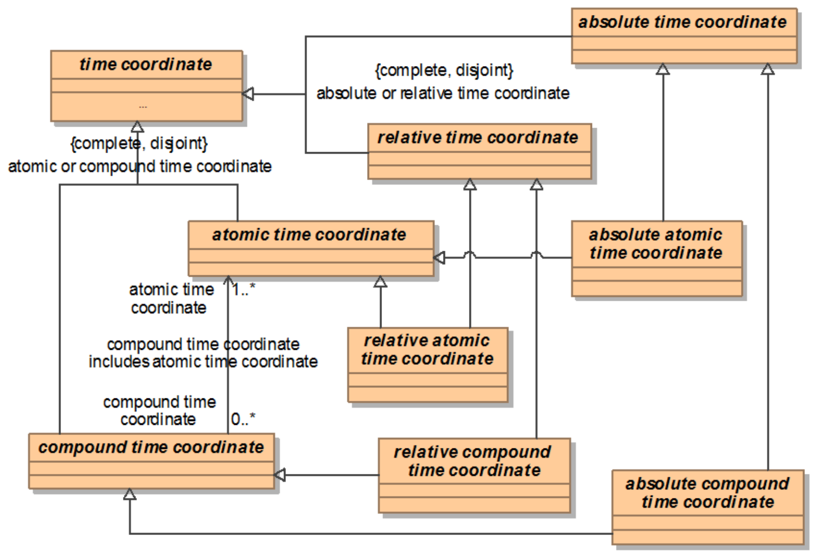


**Figure 10.10 - Time Coordinate Types**

## absolute atomic time coordinate

Definition: absolute time coordinate that *is* an atomic time coordinate
Example: 2010

## absolute compound time coordinate

Definition: absolute time coordinate that *is* a compound time coordinate
Example: 5 April 2010

## relative atomic time coordinate

Definition: relative time coordinate that *is* an atomic time coordinate
Example: January

## relative compound time coordinate

Definition: relative time coordinate that *is* a compound time coordinate
Example: 10:00

### 10.5.4 Time Coordinate Equivalence

Equivalence of time coordinates captures the idea that they can mean the same thing though given differently. For example, "February 15" and "day 46" *are equivalent*.

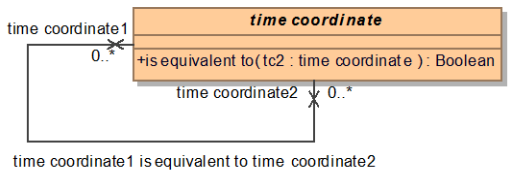


**Figure 10.11 - Time Coordinate Equivalence**

## time coordinate$_1$ *is equivalent to* time coordinate$_2$

Definition: time coordinate$_1$ *refers to* each time interval that time coordinate$_2$ *refers to* and time coordinate$_2$ *refers to* each time interval that time coordinate$_1$ *refers to*
Necessity: If time coordinate$_1$ *indicates* some time point$_1$ and time coordinate$_2$ *indicates* some time point$_2$ then time point$_1$ *is* time point$_2$.
Example: "2010 day 3" *is equivalent to* "January 3, 2010"
Example: "March" *is equivalent to* "month 3"
Example: "March 1" refers to the set {Gregorian day of year 60 in common years, Gregorian day of year 61 in leap years}. Therefore March 1 is not equivalent to Gregorian day of year 61.

## 10.6 Time Sets

A time set represents a choice of one or more possible time point sequences on a given time scale. Each time point sequence may contain one or more time points. This concept models the idea that a relative time point may *convert to* one of several different time point sequences on a related relative time scale., depending on the absolute time point that the relative time scales subdivide.

In particular, every Gregorian month of year *converts to* a time set on the Gregorian year of days scale, which depends upon whether the Gregorian year is a common year or a leap year. The time set concept may be needed for other calendars with variable-length subdivisions.

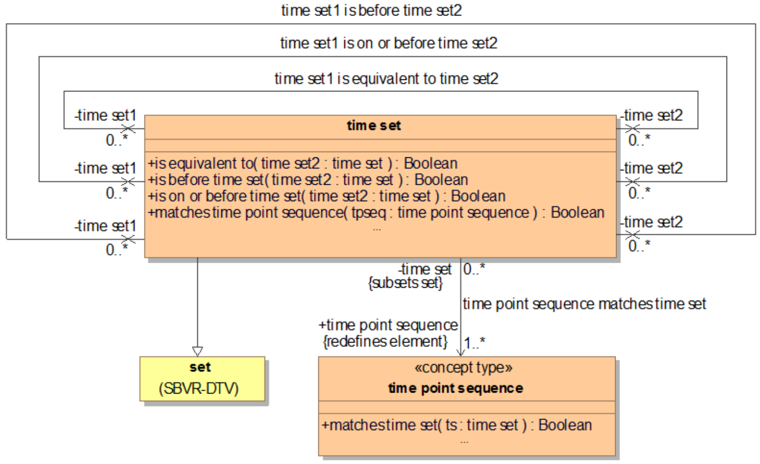


**Figure 10.12 - Time Sets**

## time set

| | |
|---|---|
| Definition: | set *of* time point sequences |
| Necessity: | the cardinality *of* a time set *is greater than* 0 |
| Necessity: | Some time scale₁ *is* the time scale *of* each time point sequence that *is in* a given time set |
| Example: | the time set {Gregorian day of year 59, Gregorian day of year 60} |

## time set$_1$ *is equivalent to* time set$_2$

| | |
|---|---|
| Synonymous Form: | time set$_1$ *equals* time set$_2$ |
| Synonymous Form: | time set$_1$ = time set$_2$ |
| Definition: | each time point sequence$_1$ *of* time set$_1$ *is* some time point sequence$_2$ *of* time set$_2$ and each time point sequence$_2$ *of* time set$_2$ *is* some time point sequence$_1$ *of* time set$_1$ |
| Example: | {Gregorian day of year 59 *through* Gregorian day of year 60, Gregorian day of year 60 *through* Gregorian day of year 61} *is equivalent to* {Gregorian day of year 60 *through* Gregorian day of year 61, Gregorian day of year 59 *through* Gregorian day of year 60} |

## time point sequence *matches* time set

| | |
|---|---|
| Synonymous Form: | time set *matches* time period |
| General Concept: | thing *is in* set |
| Definition: | time point sequence *is* some time point sequence *of* time set |
| Example: | Gregorian day of year 60 *matches* March 1 because March 1 is either Gregorian day of year 60 or Gregorian day of year 61 |

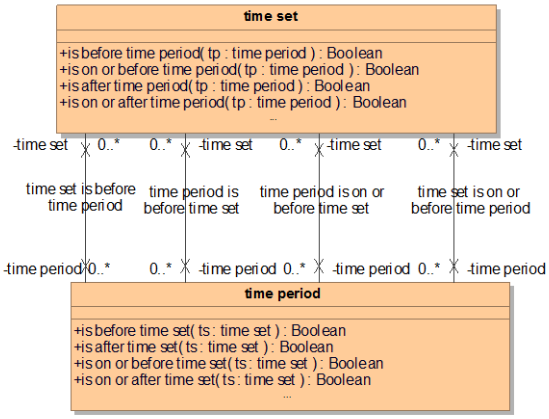


**Figure 10.13 - Time Set Relations**

## time set$_1$ *is on or before* time set$_2$

| | |
|---|---|
| Synonymous Form: | time set$_2$ *is on or after* time set$_1$ |
| Synonymous Form: | time set$_1$ ≤ time set$_2$ |
| Synonymous Form: | time set$_2$ ≥ time set$_1$ |
| Definition: | each time point sequence$_1$ *of* time set$_1$ *corresponds to* a time interval$_1$ that *is before* the time interval$_2$ that *instantiates* each time point sequence$_2$ *of* time set$_2$ |
| Example: | {Gregorian day of year 100 *through* Gregorian day of year 101} *is on or before* {Gregorian day of year 101 *through* Gregorian day of year 102} |

## time period *is on or before* time set

| | |
|---|---|
| Synonymous Form: | time set *is on or after* time period |
| Synonymous Form: | time period ≤ time set |
| Synonymous Form: | time set ≥ time period |
| Definition: | time period *is before* the time interval that *instantiates* each time point sequence *of* time set |
| Example: | Gregorian day of year 102 *is on or before* {Gregorian day of year 102, Gregorian day of year 103} |
| Example: | "January" *is on or before* {Gregorian day of year 102 *through* Gregorian day of year 103} |

## time set *is on or before* time period

| | |
|---|---|
| Synonymous Form: | time period *is on or after* time set |
| Synonymous Form: | time set ≤ time period |
| Synonymous Form: | time period ≥ time set |
| Definition: | the time interval that *instantiates* each time point sequence *of* time set *is before* time period |
| Example: | {Gregorian day of year 102, Gregorian day of year 103} *is on or before* Gregorian day of year 103 |

## time set$_1$ *is before* time set$_2$

| | |
|---|---|
| Synonymous Form: | time set$_2$ *is after* time set$_1$ |
| Synonymous Form: | time set$_1$ < time set$_2$ |
| Synonymous Form: | time set$_2$ > time set$_1$ |

Definition: the time interval$_1$ that *instantiates* each time point sequence$_1$ *of* time set$_1$ < the time interval$_2$ that *instantiates* each time period$_2$ *of* time set$_2$

Example: {Gregorian day of year 100 *through* Gregorian day of year 101} *is before* {Gregorian day of year 102 *through* Gregorian day of year 103}

## time period *is before* time set

Synonymous Form: time set *is after* time period
Synonymous Form: time period < time set
Synonymous Form: time set > time period
Definition: time period *precedes* the time interval that *instantiates* each time point sequence *of* time set
Example: Gregorian day of year 101 *is before* {Gregorian day of year 102 *through* Gregorian day of year 103}

## time set *is before* time period

Synonymous Form: time period *is after* time set
Synonymous Form: time set < time period
Synonymous Form: time period > time set
Definition: the time interval that *instantiates* each time point sequence *of* time set *precedes* time period
Example: {Gregorian day of year 102 *through* Gregorian day of year 103} *is before* Gregorian day of year 104

# 10.7 Dates and Times of Day

The most common references to specific time intervals are to specific days (calendar days) and to specific times of day. This section introduces the general concepts calendar date (coordinate), which refers to a calendar day, and time of day coordinate, which refers to a specific time period within a calendar day. A calendar date may be combined with a time of day coordinate to produce a date time coordinate.

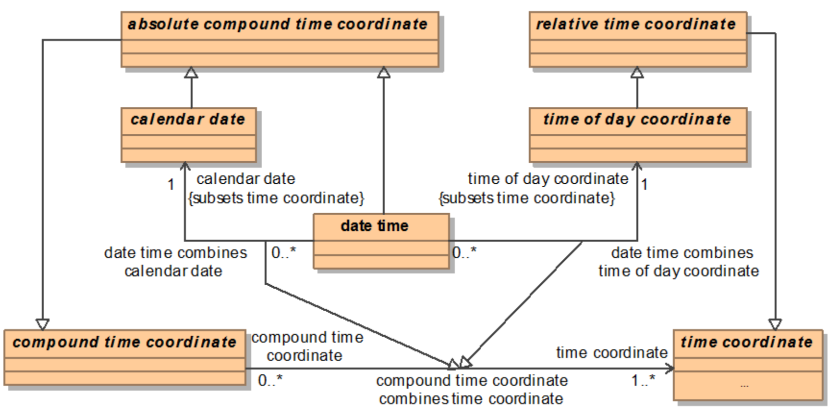


**Figure 10.14 - Date and time coordinates**

## calendar date

Synonym: date
Synonym: date coordinate

| | |
|---|---|
| Synonym: | calendar date coordinate |
| Definition: | absolute time coordinate that *indicates* a calendar day |
| Note: | Most calendar dates are compound time coordinates. |
| Example: | The Gregorian date coordinate "January 25, 2012" is a calendar date. |

### time of day coordinate

| | |
|---|---|
| Definition: | relative time coordinate that *indicates* a time of day |
| Note: | Each time of day coordinate indicates a time point on a finite time scale whose granularity is smaller than 1 day. That is, a time of day coordinate refers to time intervals that are within a calendar day. |
| Example: | The standard time coordinate "15:00" is a time of day coordinate. |

### date time

| | |
|---|---|
| Synonym: | date time coordinate |
| Synonym: | date and time |
| Definition: | absolute compound time coordinate that *combines* a calendar date and that *combines* a time of day coordinate |
| Necessity: | Each date time ***refers to*** exactly one time interval. |
| Necessity: | Each date time ***refers to*** the time interval that the time of day coordinate ***of*** the date time ***refers to*** and that ***is during*** the time interval that the calendar date ***of*** the date time ***refers to***. |
| Note: | That is, the date time refers to the unique time interval that is at that time of day and on that date. |
| Example: | June 9, 1990 5:49:03 p.m. |
| Example: | 13:00 on 1949 day 53 |
| Example: | 6 p.m. on 2010 August 6 |

### date time *combines* calendar date

| | |
|---|---|
| Synonymous Form: | calendar date ***of*** date time |
| General Concept: | compound time coordinate *combines* time coordinate |
| Note: | This verb concept wording provides a term for the date coordinate that the date time combines. |
| Necessity: | Each date time *combines* exactly one calendar date. |

### date time *combines* time of day coordinate

| | |
|---|---|
| Synonymous Form: | time of day coordinate ***of*** date time |
| General Concept: | compound time coordinate *combines* time coordinate |
| Note: | This verb concept wording provides a term for the time of day coordinate that the date time combines. |
| Necessity: | Each date time *combines* exactly one time of day coordinate. |

## 10.8 Time Scale Comparison and Conversion

Two time points are commensurable (comparable) if and only if they are on the same time scale, or can both be converted to a common time scale. For example, "hour 10" is commensurable with "11:30" because "hour 10" can be converted to a minute of day on the day of minutes scale, and "11:30" is on already that time scale. "hour 10" is not commensurable with "March" because they cannot be converted to any common time scale.

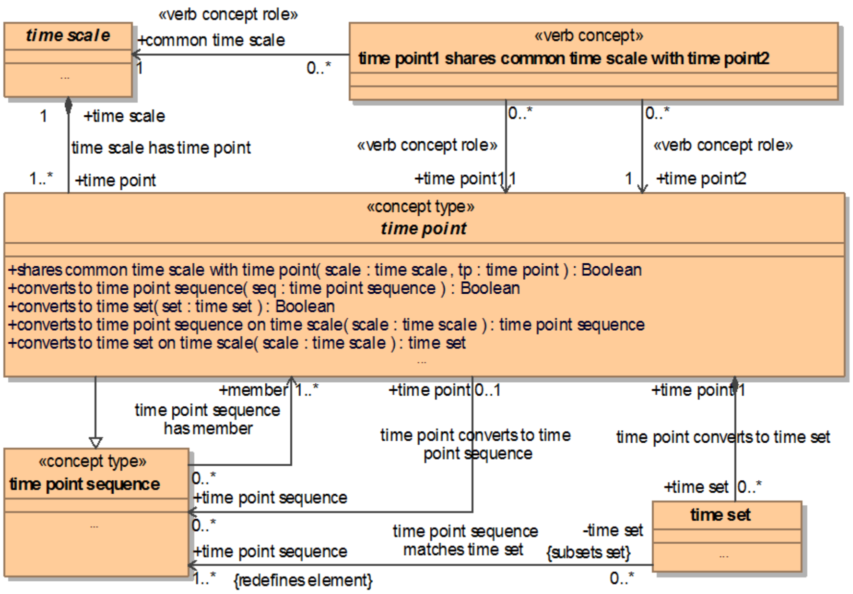


**Figure 10.15 - Time Scale Commonality and Conversion**

The concept "time point$_1$ *shares* common scale *with* time point$_2$" is used below to declare specific combinations of time points that can be compared if they are converted to particular common time scales. Other combinations are not commensurable.

## common time scale

Concept Type: role
General Concept: time scale

## time point$_1$ *shares* common time scale *with* time point$_2$

Definition: some time point sequence$_1$ *on* the common time scale *corresponds to* each time period that *instantiates* time point$_1$ and some time point sequence$_2$ *on* the common time scale *corresponds to* each time period that *instantiates* time point$_2$

The concept "time point *converts to* time point sequence" describes conversion of a time point on some time scale$_1$, to a time point sequence on some time scale$_2$. The time point and the time point sequence correspond to the same time intervals. The target time scale$_2$ always has a granularity that is less than or equal to the granularity of time scale$_1$. For example, the Gregorian month of year that *is indicated by* "January" (on the Gregorian year of months scale) converts to the time point sequence from Gregorian day of year 1 *through* Gregorian day of year 31 on the Gregorian year of days scale.

Clause 11.8 uses this concept to define specific conversions for Gregorian calendar time points. The concept applies to absolute time points and relative time points.

## time point *converts to* time point sequence

Definition: the time point *is coextensive with* the time point sequence

Necessity: The granularity *of* the time scale *of* a time point *is greater than* the granularity *of* the time scale *of* each time point sequence that the time point *converts to*.

Possibility: A time point that *converts to* a time point sequence *is* an absolute time point or *is* a relative time point.

Description: The time point and the time point sequence are two different ways to identify the same time intervals.

Note: The method _"time point"._"converted to time scale" returns the time point sequence(s) that the time point converts to on the given time scale. It is provided for convenience in formulating OCL rules.

Note: The specific conversions supported by this document are defined below in verb concepts that specialize "time coordinate *converts to* time point sequence *on* time scale."

Example: The time point that *is indicated by* the time coordinate January 2012 *converts to* the time point sequence 2012 day 1 through 2012 day 31 on the Gregorian days scale.

## time point *converts to* time set

Definition: each instance *of* the time point *is* an instance *of* at least one time point sequence *of* the time set

Necessity: The granularity *of* the time scale *of* a time point *is greater than* the granularity *of* the time scale *of* each time point sequence that *is in* a time set that the time point *converts to*.

Possibility: A time point that *converts to* a time set *is* an absolute time point or *is* a relative time point.

Description: The time scale is defined with discontinuities (e.g., leap days), such that the time set identifies several alternative time intervals that may correspond to the time point.

Example: The time point that *is indicated by* the time coordinate 'February' *converts to* the time set {Gregorian day of year 32 through Gregorian day of year 59, Gregorian day of year 32 *through* Gregorian day of year 60} on the Gregorian year of days scale.

## compound time coordinate *converts to* time set *on* time scale

Definition: each time point sequence *of* the time set *corresponds to* some time interval that the compound time coordinate *refers to*, and each time interval that the compound time coordinate *refers to is* an instance *of* exactly one time point sequence *of* the time set

Note: In most cases of interest, each of the time point sequences will consist of a single time point.

Example: "15 June" *refers to* one calendar day in each Gregorian year, but in common years it is Gregorian day of year 165 and in leap years it is Gregorian day of year 166. So, "15 June" *converts to* the time set {Gregorian day of year 165, Gregorian day of year 166} on the Gregorian year of days scale.

## 10.9 Mixed Base Time Arithmetic

Addition of a duration value to a time coordinate, subtraction of a duration value from a time coordinate, and subtraction of one time coordinate from another all employ "mixed-based time arithmetic." This is an extension of traditional mixed-base arithmetic as employed, for example, in the old-style English currency of pounds, shillings, and pence. The variation of mixed-base arithmetic that is described here accommodates the special issues raised by the nominal time units 'year' and 'month', and by the fact that 'week' is incommensurate with 'year'. This procedure is described in text, rather than as a set of SBVR concept definitions, because SBVR is not adapted to defining complex procedures.

Both addition and subtraction apply from the start of a time coordinate. For example,"9 April + 10 hours" is "9 April 10:00", while "9 April – 10 hours" is "8 April 14:00".

Mixed-base arithmetic is performed by separately adding or subtracting the individual components of a time coordinate, and then, if necessary, performing a "carry" (for addition) or "borrow" (for subtraction) from the number of the atomic time coordinate that has the next coarser time unit. The result may be either compound or atomic. For example, "9 days 20 minutes = 6 days 13 hours 27 minutes + 2 days 10 hours 53 minutes" by the following steps:

27 minutes + 53 minutes ➔ 80 minutes ➔ 20 minutes with 1 hour carry
13 hours + 10 hours + 1 hour carry ➔ 24 hours ➔ 0 hours with 1 day carry
6 days + 2 days + 1 day carry ➔ 9 days

The following list gives equivalences among most of the precise time units for use in determining when "carries" and "borrows" are needed. Equivalences between years and days, years and weeks, and months and days are discussed below because these are special cases.

Each minute *is equivalent to* 60 seconds.
Each hour *is equivalent to* 60 minutes.
Each day *is equivalent to* 24 hours.
Each week *is equivalent to* 7 days.
Each year *is equivalent to* 12 months.

A "carry" is applied if the number of an atomic time coordinate that is formed as an intermediate calculation result is greater than shown in the equivalences given above. To perform a "carry," divide the number of the intermediate result by the equivalence shown above, add the result to the number of the notional next coarser atomic time coordinate (which may be 0), and set the number of the finer component to the remainder. Note that "carries" may propagate across multiple components. See the example given above.

A "borrow" is performed if the number of an atomic time coordinate, formed as an intermediate calculation result, is negative. To apply a "borrow," divide the number of the absolute value of the intermediate result by the equivalence shown above, subtract the result for the number of the notional next coarser atomic time coordinate (which may be 0), and set the the number of the finer component to the remainder. Note that "borrows" may propagate across multiple components. For example, "22 minutes 15 seconds = 35 minutes – 12 minutes 45 seconds" by the following steps:

0 seconds – 45 seconds ➔ 15 seconds with 1 minute borrow
35 minutes – 12 minutes – 1 minute borrow 22 minutes

The procedure described above works even when an atomic duration value has a number that is greater than an equivalence shown above. For example, "23 hours = 2 days – 25 hours".

When adding or subtracting values of 'days' from time coordinates of the nominal time units 'year' and 'month', the interpretation of any "carries" or "borrows" depends upon the particular year or month coordinate. For a year coordinate, a "carry" occurs if the number of days exceeds 365 for a common year, and 366 for a leap year, and a "borrow" is made from 365 if the year is a common year, and from 366 otherwise. For example, "2007 day 331 = 2008 – 35 days" but "2006 day 330 = 2007 – 35 days".

For a month coordinate, "carries" and "borrow" are made according to the following number of days per particular calendar month:

**Table 10.1 - Number of Calendar Days per Gregorian Month**

| Gregorian Month | Equivalent Number of Calendar Days |
|---|---|
| **February** | **28** in common years, **29** in leap years |
| **April**, **June**, **September**, **November** | **30** |
| **January**, **March**, **May**, **July**, **August**, **October**, **December** | **31** |

Note that, in some cases, repeated "carries" or "borrows" may be required across multiple calendar months. For example, "2 March 2010 = 31 January 2010 + 30 days".

Subtraction is defined for most combinations of two time coordinates. For example, "30 days = 2 March 2010 – 31 January 2010". However, subtraction of date coordinates that span the end of February is not defined if the calendar years are not specified. For example, "3 days = 2 February – 30 January" but "2 March – 28 February" is either "2 days" or "3 days" depending upon whether these dates are in a leap year or not.

Arithmetic involving weeks and years presents a special problem - determine which concept of 'year' is intended. That is because the Gregorian year (clause 11) and the ISO week-based year (clause 12) are of different lengths and are only loosely aligned.

When the time coordinates are Gregorian time coordinates, additions and subtractions involving years, months, weeks, and days is done in Gregorian terms, treating each week as 7 days. For example: 20 December 2008 plus 1 year and 8 weeks is 20 December 2009 + 56 days = 14 February 2010.

When the time coordinates are ISO year week or ISO year week day coordinates, additions and subtractions involving years and weeks is done in terms of the ISO week-based year. That is, each 'year' that is added or subtracted is taken to be exactly 52 weeks or exactly 53 weeks, according to the "First Thursday Rule" (see 12.2). For example: "2008 week 50 plus 1 year and 8 weeks" is 2009 week 50 plus 8 weeks = 2010 week 5. Following the logic above, week 50 + 8 weeks gives 58 weeks, which causes a carry into the 'year' position. But Gregorian year 2009 started on a Thursday, so the ISO week-based year 2009 has 53 weeks, and the residue is 5 weeks. By comparison, 2010 week 50 plus 8 weeks is 2011 week 6, because the ISO week-based year 2010 has only 52 weeks.

The ISO day of week is not affected by variation in the duration of ISO week-based years. Every week has exactly 7 days. Carrying or borrowing out of the 'day' (of week) position modifies the ISO week of year value in the obvious way.

Additions or subtractions to relative ISO week of year coordinates and ISO week-day coordinates that carry or borrow into the 'years' position is not well-defined. Some ISO week-based years have 52 weeks and some have 53.

Explicit subtraction between Gregorian calendar time coordinates and ISO weeks calendar time coordinates is best accomplished by reducing both time coordinates to indices on the indefinite scale of Gregorian days. The difference is then an exact duration in days, which can be converted to any convenient compound duration value.

# 11 Gregorian Calendar (normative)

## 11.1 General

This clause provides terminology for the concepts in the Gregorian calendar.

## Gregorian Calendar Vocabulary

General Concept: terminological dictionary
Language: English
Included Vocabulary: Calendars Vocabulary
Included Vocabulary: Duration Values Vocabulary
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#GregorianCalendarVocabulary

---

The Gregorian calendar concepts depend on concepts and terminology introduced in the Calendars Vocabulary and the Duration Values Vocabulary.

## 11.2 Gregorian Calendar

The Gregorian calendar was standardized for international commerce by the Convention du Mètre, and is widely used in business and everyday activities.

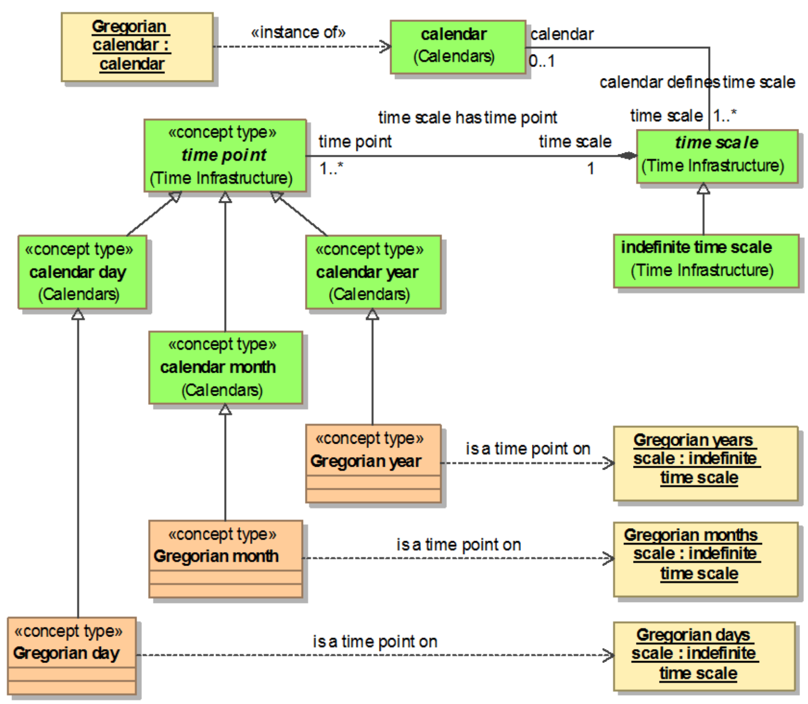


**Figure 11.1 - Gregorian Indefinite Time Scales and Time Points**

## Gregorian calendar

Source: ISO 8601 (2.2.15, 'Gregorian calendar')
Definition: calendar in general use, introduced in 1582 to define a calendar year that more closely approximated the tropical year than the Julian calendar

| | |
|---|---|
| Note: | The Gregorian Calendar was defined in 1582 in [Inter Gravissimas] and was adopted at various times by various countries. It is now the international standard calendar. |
| Note: | The interpretation of any date depends upon the calendar used. Caution should be used with historical dates because the standard calendar varied by locality as well as time. The Gregorian Calendar was adopted in 1582 in Italy and a few other countries, and at various times as late as 1926 in other countries. |

## Convention du Mètre

| | |
|---|---|
| Definition: | occurrence that is the signing of the Convention du Mètre |
| Necessity: | The Convention du Mètre *occurred within* 20 May 1875. |
| Note: | The particular Gregorian day on which the signing of the Convention du Mètre occurred establishes the index origin of the various Gregorian scales. |
| Note: | [ISO 8601] establishes the date of the signing of the Convention du Mètre, 20 May 1875, as the reference date for the Gregorian calendar. |

## Gregorian years scale

| | |
|---|---|
| Definition: | the indefinite time scale that *has* granularity year and that *has* time points that *are* Gregorian years |
| Necessity: | The index origin value *of* the Gregorian years scale *equals* 1875. |
| Necessity: | The time interval that Gregorian year 1875 *corresponds to is started by* the time interval that *is* the Gregorian day 684 467. |
| Note: | Gregorian day 684 467 is January 1, 1875. |
| Note: | The starting Gregorian day and the rules for the duration of Gregorian years define a unique time interval. |
| Note: | This definition applies to the Gregorian calendar as recognized at the Prime Meridian at Greenwich in England during Standard Time. Other Gregorian years scales may be obtained by adding or subtracting time offsets, as discussed in sub clause 13.5. |

## Gregorian months scale

| | |
|---|---|
| Definition: | the indefinite time scale that *has* granularity month and that *has* time points that *are* Gregorian months |
| Necessity: | The index origin value *of* the Gregorian months scale *equals* 22 493. |
| Note: | 22 493 is 12 * (1875 - 1) + 5 (for the month of May) |
| Necessity: | The time interval that Gregorian month 22 493 *corresponds to is* a May and *is started by* the time interval that *is* the Gregorian day 684 587. |
| Note: | Gregorian day 684 587 is May 1, 1875. |
| Note: | The starting Gregorian day, and the fact that the Gregorian month is a May (and therefore has 31 days) defines a unique time interval. |
| Note: | This definition applies to the Gregorian calendar as recognized at the Prime Meridian at Greenwich in England during Standard Time. Other Gregorian months scales may be obtained by adding or subtracting time offsets, as discussed in sub clause 13.5. |

## Gregorian days scale

| | |
|---|---|
| Definition: | the indefinite time scale that *has* granularity day and that *has* time points that *are* Gregorian days |
| Necessity: | The index origin value *of* the Gregorian days scale *equals* 684 606. |
| Note: | Gregorian day 684 606 is May 20, 1875. |
| Note: | The calendar reform instituted by Pope Gregory XIII and promulgated in the bull [Inter Gravissimas] started the use of the Gregorian calendar with the date 15 October 1582, which is the same as 05 October 1582 in the Julian calendar. That calendar day had index 577 738 on the Julian calendar, computed as 1581 years of 365 days plus 395 leap days from 1 January of year 1 (calendar day 1) to 1 January 1582 + 277 days from 1 January 1582 to 5 |

October 1582. From 15 October 1582 to the Convention du Mètre on 20 May 1875, there were 106 868 calendar days (including leap days). Therefore, the Convention happened on calendar day 684 606 of the Gregorian days scale.

Necessity: The Convention du Mètre *occurred within* the time interval that *is* the Gregorian day 684 606.

Necessity: The duration *of* the time interval that *is* the Gregorian day 684 606 *is* 1 day.

Necessity: The time interval that *is* the Gregorian day 684 606 *is started by* a time interval that *is the* 12 hours *preceding* an observation of noon at the Greenwich observatory.

Note: The combination of the above necessities identifies a unique time interval. The reference origin for the Gregorian months scale and the Gregorian years scale are defined in terms of that time interval.

Note: Noon at the Greenwich observatory was the reference point for Gregorian days until 1884. The [International Meridian Conference] of 1884 established the Greenwich Meridian as the international standard for zero degrees longitude. It also established a uniform international time standard called the 'universal day' – a mean solar day of 24 hours measured from midnight on the Greenwich Meridian. This time standard was formally replaced by Universal Coordinated Time in 1972.

Note: This definition applies to the Gregorian calendar as recognized at the Prime Meridian at Greenwich in England during Standard Time. Other Gregorian days scales may be obtained by adding or subtracting time offsets, as discussed in sub clause 13.5.

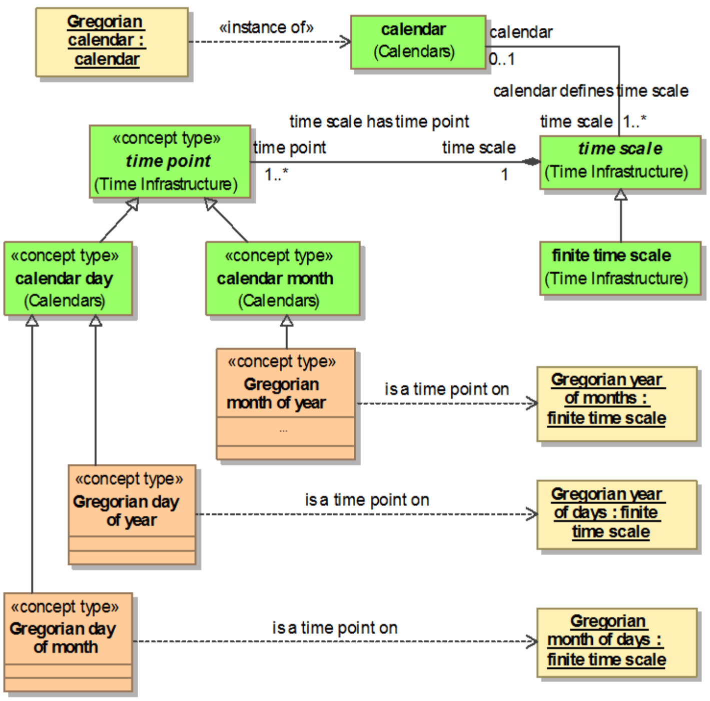


**Figure 11.2 - Gregorian Finite Time Scales and Time Points**

## Gregorian year of months scale

| | |
|---|---|
| Definition: | the finite time scale that *has* granularity 1 month and that *has* cardinality 12 and that *exactly subdivides* each Gregorian year |
| Necessity: | Each Gregorian year *subdivides into* exactly 12 Gregorian months of year. |
| Necessity: | The index origin value *of* the Gregorian year of months scale *equals* 1. |
| Necessity: | The first member *of* the Gregorian year of months scale *is* the index origin member *of* the Gregorian year of months scale. |

## Gregorian year of days scale

| | |
|---|---|
| Definition: | the finite time scale that *has* granularity 1 day and that *has* cardinality 366 and that *subdivides* 'Gregorian year' |
| Note: | Each leap year is subdivided into 366 Gregorian day of year time points. Each common year is subdivided into 365 Gregorian day of year time points. |
| Necessity: | The index origin value *of* the Gregorian year of days scale *equals* 1. |
| Necessity: | The first member *of* the Gregorian year of days scale *is* the index origin member *of* the Gregorian year of days scale. |
| Note: | This time scale has 366 Gregorian days of year in order to accommodate leap years. |

## Gregorian month of days scale

| | |
|---|---|
| Definition: | the finite time scale that *has* granularity 1 day and that *has* cardinality 31 and that *subdivides* 'Gregorian month of year' |
| Note: | Each Gregorian month of year is subdivided into a specific number of Gregorian day of month time points. The subdivision of February is a set of two time sequences. |
| Necessity: | The index origin value *of* the Gregorian month of days scale *equals* 1. |
| Necessity: | The first member *of* the Gregorian month of days scale *is* the index origin member *of* the Gregorian month of days scale. |
| Note: | This time scale has 31 Gregorian days of month in order to accommodate the longest Gregorian month. |

## 11.3 Gregorian Time Points

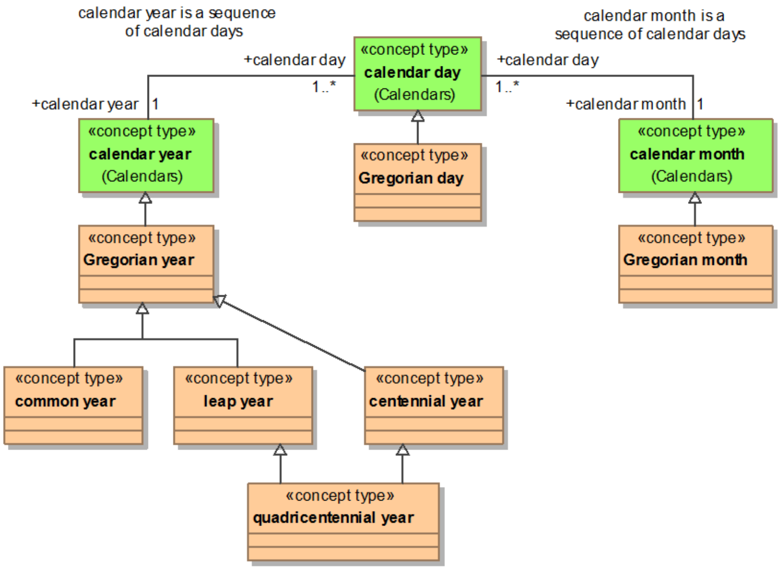


**Figure 11.3 - Gregorian Time Points**

### common year

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | calendar year that *is on* the Gregorian years scale and the number *of* the calendar year, when divided by 4, generates a remainder that is not zero, or that *is* a centennial year |
| Necessity: | Each common year *subdivides into* exactly 365 Gregorian days of year. |
| Note: | This is an absolute time point because it is on an indefinite time scale. |

### leap year

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | calendar year that *is on* the Gregorian years scale and the number *of* the calendar year, when divided by 4, generates a remainder that is zero, and that *is* not a centennial year |
| Necessity: | Each leap year *subdivides into* exactly 366 Gregorian days of year. |
| Note: | The rules for leap years were established by Pope Gregory XIII in [Inter Gravissimas]. These rules were eventually adopted by various civil governments and incorporated into [ISO 8601]. |
| Note: | This is an absolute time point because it is on an indefinite time scale. |

### centennial year

| | |
|---|---|
| Source: | ISO 8601 (2.2.18, 'centennial year') |
| Concept Type: | concept type |
| Definition: | calendar year that *is on* the Gregorian years scale that *is* not a quadricentennial year, and the number *of* the calendar year, when divided by 100, generates a remainder that is zero |
| Note: | This is an absolute time point because it is on an indefinite time scale. |

## quadricentennial year

Source: ISO 8601 (2.2.18, 'centennial year')
Concept Type: concept type
Definition: calendar year that *is on* the Gregorian years scale and the number *of* the calendar year, when divided by 400, generates a remainder that is zero
Note: This is an absolute time point because it is on an indefinite time scale.

## Gregorian year

Concept Type: concept type
Definition: common year or leap year that *is on* the Gregorian years scale
Note: This is an absolute time point because it is on an indefinite time scale.

## Gregorian month

Concept Type: concept type
Definition: calendar month that *is on* the Gregorian months scale
Note: This is an absolute time point because it is on an indefinite time scale.

## Gregorian month of year

Concept Type: concept type
Definition: calendar month that *is on* the Gregorian year of months scale
Note: This is a relative time point because it is on a finite time scale.

## Gregorian calendar month

Definition: Gregorian month or Gregorian month of year
Concept Type: concept type

## Gregorian day

Concept Type: concept type
Definition: calendar day that *is on* the Gregorian days scale
Note: This is an absolute time point because it is on an indefinite time scale.

## Gregorian day of year

Concept Type: concept type
Definition: calendar day that *is on* the Gregorian year of days scale
Note: This is a relative time point because it is on a finite time scale.
Necessity: Each Gregorian day of year *corresponds to* a set *of* Gregorian days.
Note: In general each Gregorian day of year corresponds to one calendar day in each Gregorian year but Gregorian day of year 366 occurs only in leap years.

## Gregorian day of month

Concept Type: concept type
Definition: calendar day that *is on* the Gregorian month of days scale
Note: This is a relative time point because it is on a finite time scale.
Necessity: Each Gregorian day of month *corresponds to* a set *of* Gregorian days.
Note: In general each Gregorian day of month corresponds to one calendar day in each Gregorian month but Gregorian day of month 29, 30, and 31 do not occur in every Gregorian month.

## Gregorian calendar day

Definition: Gregorian day or Gregorian day of year or Gregorian day of month
Concept Type: concept type

## 11.4 Gregorian Months of Year

Because of the cyclic usage of the finite time scales associated with calendars, the names of months, days of the week, and holidays designate many time intervals and so are general concepts. However, these names are traditionally capitalized like proper names and also seem like individual concepts. Using this specification, when the name of a time point is used, it designates the general concept (the time point) and denotes the corresponding time intervals, in the same way that a term for any general concept denotes its instances. Such usage commonly involves quantifiers and qualifiers, such as "every April" or "next April" or "April (in) 2001", which select specific time intervals. When the intent is to refer to the time point itself – the individual thing that appears on the calendar – the name is qualified as referring to a time point, as in "the time point 'April'," or "the Gregorian month of year 'April'," which is a short form of "the Gregorian month of year that is designated by 'April'."

All named time points are treated in this way, including the Gregorian months of year, the days-of-week, and recurring holidays and anniversaries.

Some holidays, like Easter and Ramadan, recur irregularly, so additional information, such as an ephemeris, is required to resolve the name to particular Gregorian calendar dates. Formalizing such definitions is beyond the scope of this specification.

The following defines the common names for Gregorian month of years as individual concepts because they identify specific months on the Gregorian year of months scale.

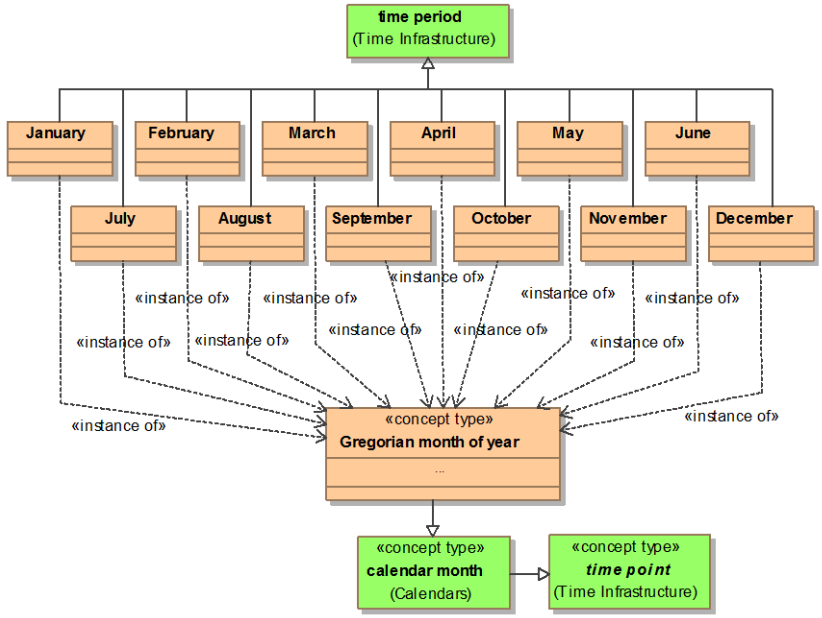


**Figure 11.4 - Gregorian Months**

## January

Source: ISO 8601 (Table 1)
Definition: time interval that *has* duration 31 days and that *starts* an instance *of* a Gregorian year
Necessity: The concept 'January' *is* the Gregorian month of year that *is in* sequence position 1 *of* the Gregorian year of months scale.
Necessity: The time point 'January' *subdivides into* exactly 31 Gregorian days of month.
Necessity: Each January *is met by* a December.
Note: "January 2008" and "2008 month 01" are expressions for a calendar date
Note: "January, 2008" is an expression for a calendar date for a Gregorian month of year using a reference scheme involving a Gregorian month and a calendar year.

## February

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* a January and that *has* a duration that *is* 28 days if the time interval *is part of* an instance *of* a common year, or that *is* 29 days if the time interval *is part of* an instance *of* a leap year
Necessity: The time point 'February' *subdivides into* exactly 28 Gregorian days of month or exactly 29 Gregorian days of month.
Necessity: The time point sequence that *is* Gregorian day of month 1 *through* Gregorian day of month 28 *corresponds to* each February that *is during* a common year.
Necessity: The time point sequence that *is* Gregorian day of month 1 *through* Gregorian day of month 29 *corresponds to* each February that *is during* a leap year.
Note: The set of these two time point sequences is how Gregorian month of days subdivides February.
Note: The subdivision of the time point is fixed. day-of-month 29 is part of the sequence, but not every February has 29 day-of-month subintervals.
Note: The rules for leap years were established by Pope Gregory XIII in [Inter Gravissimas]. These rules were eventually adopted by various civil governments and incorporated into [ISO 8601].

## March

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* a February and that *has* a duration that *is* 31 days
Necessity: The concept 'March' *is* the Gregorian month of year that *is in* sequence position 3 *of* the Gregorian year of months scale.
Necessity: The time point 'March' *subdivides into* exactly 31 Gregorian days of month.

## April

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* a March and that *has* a duration that *is* 30 days
Necessity: The concept 'April' *is* the Gregorian month of year that *is in* sequence position 4 *of* the Gregorian year of months scale.
Necessity: The time point 'April' *subdivides into* exactly 30 Gregorian days of month.

## May

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* an April and that *has* a duration that *is* 31 days
Necessity: The concept 'May' *is* the Gregorian month of year that *is in* sequence position 5 *of* the Gregorian year of months scale.
Necessity: The time point 'May' *subdivides into* exactly 31 Gregorian days of month.

## June

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* a May and that *has* a duration that *is* 30 days
Necessity: The concept 'June' *is* the Gregorian month of year that *is in* sequence position 6 *of* the Gregorian year of months scale.
Necessity: The time point 'June' *subdivides into* exactly 30 Gregorian days of month.

## July

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* a June and that *has* a duration that *is* 31 days
Necessity: The concept 'July' *is* the Gregorian month of year that *is in* sequence position 7 *of* the Gregorian year of months scale.
Necessity: The time point 'July' *subdivides into* exactly 31 Gregorian days of month.

## August

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* a July and that *has* a duration that *is* 31 days
Necessity: The concept 'August' *is* the Gregorian month of year that *is in* sequence position 8 *of* the Gregorian year of months scale.
Necessity: The time point 'August' *subdivides into* exactly 31 Gregorian days of month.

## September

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* an August and that *has* a duration that *is* 30 days
Necessity: The concept 'September' *is* the Gregorian month of year that *is in* sequence position 9 *of* the Gregorian year of months scale.
Necessity: The time point 'September' *subdivides into* exactly 30 Gregorian days of month.

## October

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* a September and that *has* a duration that *is* 31 days
Necessity: The concept 'October' *is* the Gregorian month of year that *is in* sequence position 10 *of* the Gregorian year of months scale.
Necessity: The time point 'October' *subdivides into* exactly 31 Gregorian days of month.

## November

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* an October and that *has* a duration that *is* 30 days
Necessity: The concept 'November' *is* the Gregorian month of year that *is in* sequence position 11 *of* the Gregorian year of months scale.
Necessity: The time point 'November' *subdivides into* exactly 30 Gregorian days of month.

## December

Source: ISO 8601 (Table 1)
Definition: time interval that *is met by* a November and that *has* a duration that *is* 31 days
Necessity: The concept 'December' *is* the Gregorian month of year that *is in* sequence position 12 *of* the Gregorian year of months scale.
Necessity: The time point 'December' *subdivides into* exactly 31 Gregorian days of month.
Necessity: Each December *finishes* an instance *of* a Gregorian year.

## 11.5 Gregorian Year Values

This sub clause defines the meaning of nominal atomic duration values that use the nominal time unit 'year'. It accounts for the varying numbers of calendar days in Gregorian years, due to leap years, centennial years, and quadricentennial years.

Note: this sub clause defines some concepts, such as 'year remainder', that are only needed to support the concept 'year value *specifies* duration value set'. These supporting concepts need not be explicitly defined in versions of this specification in other modeling systems.

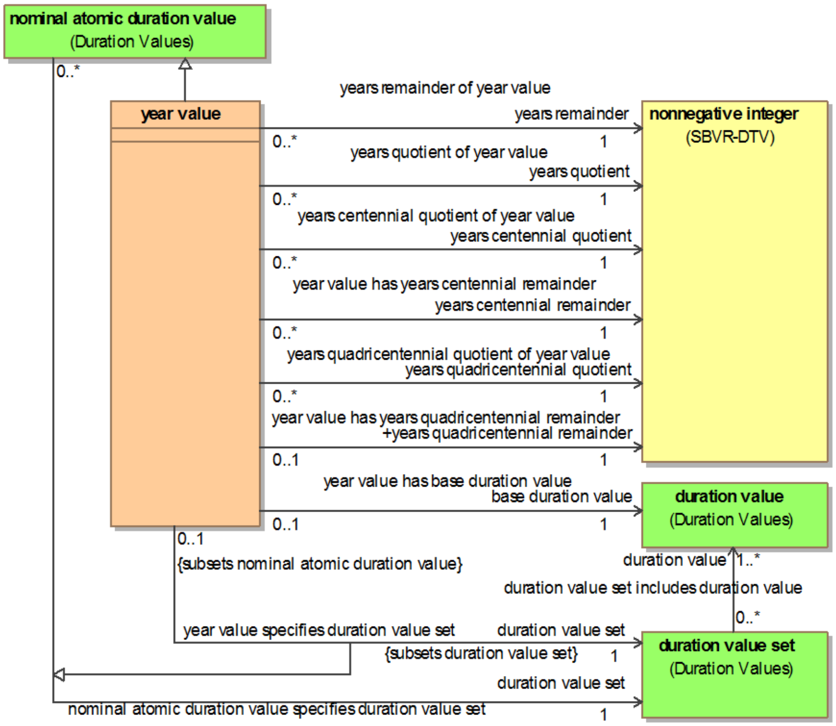


**Figure 11.5 - Year Values**

### year value

Definition: nominal atomic duration value that *has* the time unit 'year'

### years remainder

Concept Type: role
General Concept: nonnegative integer

## years remainder *of* year value

Definition: the years remainder is the remainder produced by dividing the number *of* the year value by 4
Note: Each 4-year cycle includes exactly 1 leap day.
Example: the years remainder of '5 years' is 1

## years quotient

Concept Type: role
General Concept: nonnegative integer

## years quotient *of* year value

Definition: the years quotient is the quotient produced by dividing the number *of* the year value by 4
Note: Each 4-year cycle includes exactly 1 leap day.
Example: the years quotient of '11 years' is 2

## years centennial quotient

Concept Type: role
General Concept: nonnegative integer

## years centennial quotient *of* year value

Definition: the years centennial quotient is the quotient produced by dividing the number *of* the year value by 100
Note: According to [Inter Gravissimas], a leap day is omitted for each centennial year that is not a quadricentennial year.
Example: the years centennial quotient of '5 years' is 0
Example: the years centennial quotient of '301 years' is 3

## years centennial remainder

Concept Type: role
General Concept: nonnegative integer

## year value *has* years centennial remainder

Definition: the years centennial remainder *is* the remainder produced by dividing the number of the year value by 100
Example: the years centennial remainder of 601 years is 1

## years quadricentennial remainder

Concept Type: role
General Concept: nonnegative integer

## year value *has* years quadricentennial remainder

Definition: the years quadricentennial remainder *is* the remainder produced by dividing the number of the year value by 400
Example: the years quadricentennial remainder of 601 years is 201

## years quadricentennial quotient

Concept Type: role
General Concept: nonnegative integer

## years quadricentennial quotient *of* year value

Definition: the years quadricentennial quotient is the quotient produced by dividing the number *of* the year value by 400
Note: According to [Inter Gravissimas], a leap day is included for each quadricentennial year even though it is a centennial year.
Example: the years quadricentennial quotient of '301 years' is 0
Example: the years quadricentennial quotient of '401 years' is 1

## base duration value

Concept Type: role
General Concept: duration value

## year value *has* base duration value

Definition: the base duration value *is* the number of the year value times 365 days *plus* 1 day times (the years quotient *of* the year value – the years centennial quotient *of* the year value + the years quadricentennial quotient *of* the year value).
Note: That is, if Y is the year value, the base duration value B(Y) is given by:
B(Y) = Y * 365 + Y/4 – Y/100 + Y/400 days,
where the / denotes the quotient of the integer division.
Note: [Inter Gravissima] specifies that a leap day occurs every 4 years, except every 100 years, with a further exception that a leap day does occur every 400 years.
Note: The base duration value is the number of days in a number of years that does not involve two conditions:
- the year value is not a multiple of 4 and the particular calendar years involved include one more leap year; or
- the year value is not a multiple of 100 and the particular calendar years involved include one more centennial year (which may be a quadricentennial year)
Example: The base duration value of 400 years is 146 000 + 100 – 4 + 1 = 146 097 days, and neither of the two conditions can apply. 400 years is actually a precise duration value.
Example: The base duration value of 111 years is 40 515 + 27 – 1 + 0 = 40 541, but it is possible that either of the two conditions above is met.

## years duration value set

Concept Type: role
Definition: duration value set

## year value *specifies* years duration value set

General Concept: nominal atomic duration value *specifies* duration value set
Definition: the years duration value set *is* the duration value set that consists of the following duration values:
- the base duration value *of* the year value,
- the base duration value *of* the year value *minus* 1 day, only if the years centennial remainder *of* the year value is greater than zero,
- the base duration value *of* the year value *plus* 1 day, only if the years remainder *of* the year value is greater than zero or the years quadricentennial remainder *of* the year value is greater than zero,

| | |
|---|---|
| Note: | If Y is the year value, and B(Y) is the base duration value, the duration set specified by Y is given by:<br>S(Y) = {B(Y)}<br>∪ {B(Y) – 1}, if Y mod 100 > 0,<br>∪ {B(Y) + 1}, if Y mod 4 > 0 or Y mod 400 > 0.<br>where 'mod' denotes the remainder of the integer division. |
| Note: | The duration value set includes only the base duration value when the year value is exactly divisible by 400 (none of the remainders is greater than zero). |
| Example: | The duration value set for 400 years is {146 097 days}. |
| Example: | The duration value set for 111 years is {40 541 days, 40 540 days, 40 542 days}. |
| Example: | For example, the 111 years could be between 1903 and 2014, which includes the 27 leap years between 1903 and 2011 (including the quadricentennial year 2000), but also the leap year in 2012. So the actual value is 40 542. By comparison, the 111 years between July 1,1796 and July 1, 1907 includes two centennial years and thus only 25 leap years, so the actual value is 40 540. |
| Example: | The duration value set for 100 years is {36 524 days, 36 525 days}. For example, the 100 years from 1814 and 1914 is 36 524 days. The 100 years from 1914 to 2014 is 36 525 days. |

## 11.6 Gregorian Month Values

This sub clause defines the meaning of nominal atomic duration values that use the nominal time unit 'month.' It accounts for the varying numbers of calendar days in the calendar months of the Gregorian calendar. It accounts for leap years by considering that 48 months (4 years of 12 months) includes one leap day (February 29). The computation adjusts for the fact that centennial years have no leap days, but quadricentennial years have one leap day.

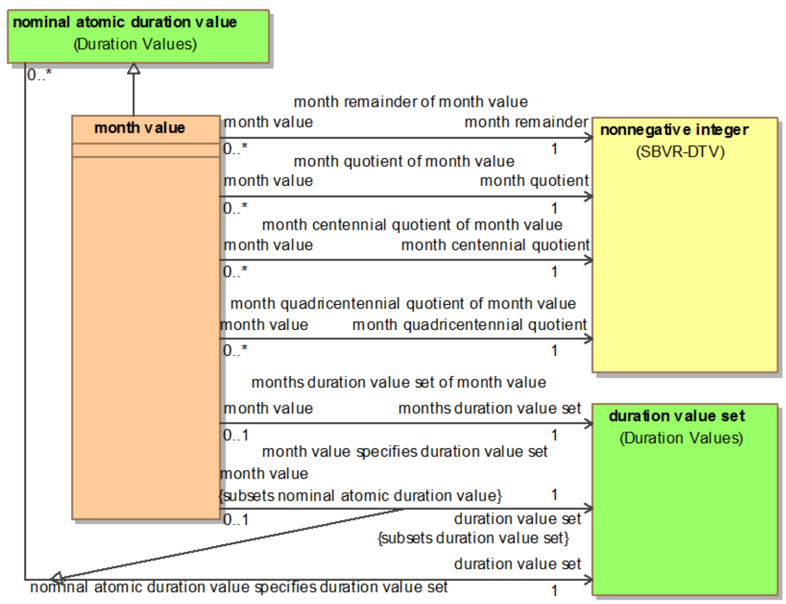


**Figure 11.6 - Month Values**

Note: this sub clause defines some concepts, such as 'month remainder', that are only needed to support the concept 'month value *specifies* duration value set'. These supporting concepts need not be explicitly defined in versions of this specification in other modeling systems.

## month value

Definition: nominal atomic duration value that *has* the time unit 'month'

## months remainder

Concept Type: role
General Concept: nonnegative integer

## months remainder *of* month value

Definition: the months remainder is the remainder produced by dividing the number *of* the month value by 48
Note: **48 is the number of months in a 4-year cycle that includes one leap day.**
Example: the months remainder of '50 months' is 2

## months quotient

Concept Type: role
General Concept: nonnegative integer

## months quotient *of* month value

Definition: the months quotient is the quotient produced by dividing the number *of* the month value by 48
Note: **48 is the number of months in a 4-year cycle that includes one leap day.**
Example: the months quotient of '50 months' is 1

## months centennial quotient

Concept Type: role
General Concept: nonnegative integer

## months centennial quotient *of* month value

Definition: the months centennial quotient is the remainder produced by dividing the number *of* the month value by 1 200
Note: 1 200 **is** 100 years **of** 12 months. According to [Inter Gravissimas], a leap day is omitted for each centennial year that is not a quadricentennial year.
Example: the months centennial quotient of '60 months' is 0
Example: the months centennial quotient of '2405 months' is 2

## months quadricentennial quotient

Concept Type: role
General Concept: nonnegative integer

## months quadricentennial quotient *of* year value

Definition: the months quadricentennial quotient is the remainder produced by dividing the number *of* the month value by 4 800
Note: 4 800 **is** 400 years **of** 12 months. According to [Inter Gravissimas], a leap day is included for each quadricentennial year even though it is a centennial year.

Example: the months quadricentennial quotient of '10 months' is 0
Example: the months quadricentennial quotient of '4 805 months' is 1

### months duration value set

Concept Type: role
Definition: duration value set

### months duration value set *of* month value

Definition: the months duration value set is specified by the following table, according to the months remainder *of* the month value

## 11.7 Gregorian Time Coordinates

This sub clause defines several Gregorian time coordinates and their meaning in terms of time scales. It also "anchors" the Gregorian calendar on the Time Axis per the signing of the Convention du Mètre.

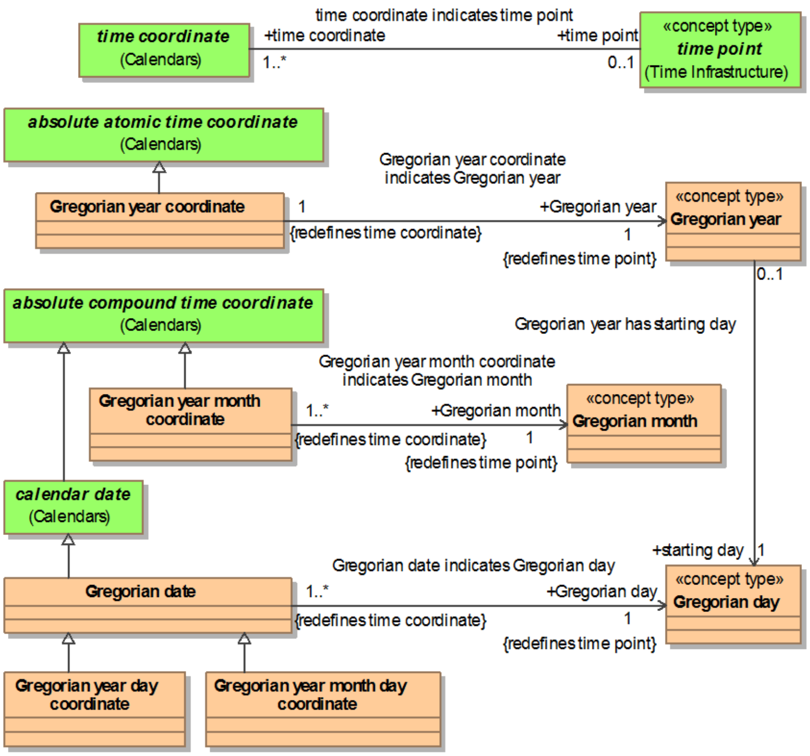


**Figure 11.7 - Gregorian Absolute Time Coordinates**

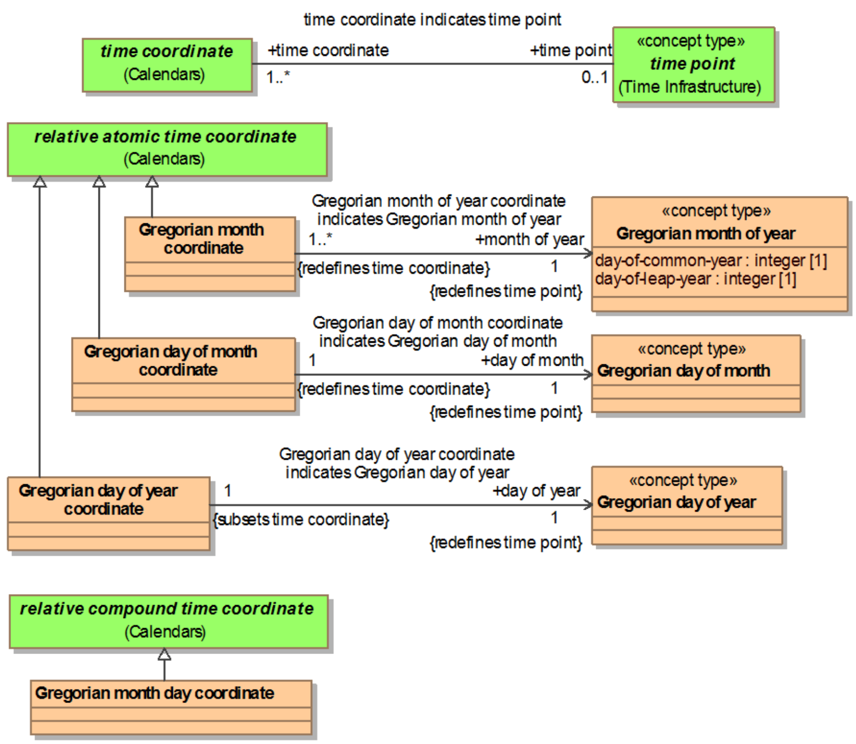


**Figure 11.8 - Gregorian Relative Time Coordinates**

- A Gregorian year coordinate *indicates* a Gregorian year, for example "2010"
- A Gregorian month coordinate *indicates* a Gregorian month, for example "January"
- A Gregorian day of year coordinate *indicates* a Gregorian day of year, for example "Gregorian day of year 360"
- A Gregorian day of month coordinate *indicates* a Gregorian day of month, for example "Gregorian day of month 14"
- A Gregorian year month coordinate *combines* a Gregorian year and a Gregorian month of year, *to indicate* a Gregorian month, for example "July 2010"
- A Gregorian year month day coordinate *combines* a Gregorian year, a Gregorian month of year, and a Gregorian day of month *to indicate* a Gregorian day, for example "9 July 2010"
- A Gregorian year day coordinate *combines* a Gregorian year and a Gregorian day of year *to indicate* a Gregorian day, for example "2010 day 33"
- A Gregorian month day coordinate *combines* a Gregorian month of year and a Gregorian day of month *to refer to* one time interval in each Gregorian year, for example "9 July", but only the first 60 of them (January 1 to February 29) indicate Gregorian day of year time points

## Gregorian year coordinate

Definition: absolute atomic time coordinate that *indicates* a Gregorian year

| | |
|---|---|
| Necessity: | Each Gregorian year coordinate *indicates* a Gregorian year that *has* an index *equal to* the index *of* the Gregorian year coordinate |
| Description: | A Gregorian year coordinate directly gives the Gregorian year number. |
| Example: | 2010 |

## Gregorian month coordinate

| | |
|---|---|
| Definition: | relative atomic time coordinate that *indicates* a Gregorian month of year |
| Necessity: | Each Gregorian month coordinate *indicates* a Gregorian month of year that *has* an index *equal to* the index *of* the Gregorian month coordinate. |
| Description: | A Gregorian month coordinate directly gives the index *of* a calendar month within a calendar year. |
| Necessity: | Each Gregorian month coordinate *is greater than or equal to* 1. |
| Necessity: | Each Gregorian month coordinate *is less than or equal to* 12. |
| Example: | "January" and "month 1" *indicate* the same Gregorian month of year |

## Gregorian day of year coordinate

| | |
|---|---|
| Definition: | relative atomic time coordinate that *indicates* a Gregorian day of year |
| Necessity: | Each Gregorian day of year coordinate *indicates* a Gregorian day of year that *has* an index *equal to* the index *of* the Gregorian day of year coordinate. |
| Description: | A Gregorian day of year coordinate directly gives the index *of* a calendar day within a calendar year. |
| Necessity: | Each Gregorian day of year coordinate *is greater than or equal to* 1. |
| Necessity: | Each Gregorian day of year coordinate *is less than or equal to* 366. |
| Example: | "day 45" and "14 February" *indicate* the same Gregorian day of year |

## Gregorian day of month coordinate

| | |
|---|---|
| Definition: | relative atomic time coordinate that *indicates* a Gregorian day of month |
| Necessity: | Each Gregorian day of month coordinate *indicates* a Gregorian day of month that *has* an index *equal to* the index *of* the Gregorian day of month coordinate. |
| Description: | A Gregorian day of month coordinate directly gives the index *of* a calendar day within a calendar month. |
| Necessity: | Each Gregorian day of month coordinate *is greater than or equal to* 1. |
| Necessity: | Each Gregorian day of month coordinate *is less than or equal to* 31. |
| Example: | "Gregorian day of month 14" *indicates* the Gregorian day of month that has index 14 |

These absolute compound time coordinates support various combinations of Gregorian years, Gregorian months of year, and calendar days.

## Gregorian year month coordinate

| | |
|---|---|
| Definition: | absolute compound time coordinate that *combines* a Gregorian year coordinate and that *combines* a Gregorian month coordinate and that *indicates* a Gregorian month |
| Necessity: | Each Gregorian year month coordinate *indicates* a Gregorian month that *has* index 12 times (the index *of* the Gregorian year coordinate minus 1) plus (the index *of* the Gregorian month coordinate minus 1). |
| Description: | The Gregorian year coordinate and the Gregorian month coordinate *of* the Gregorian year month coordinate jointly identify the Gregorian month on the infinite Gregorian months scale. |
| Note: | The definition subtracts 1 from the indices of the Gregorian year coordinate and Gregorian month coordinate because these are index origin value 1. |
| Example: | "2010 month 3" *combines* the set *of* {2010, month 3}, and *indicates* the Gregorian month that *has* index 24123 |

## starting day

Concept Type: role
Definition: Gregorian day that is the first calendar day of some Gregorian year

## Gregorian year *has* starting day

Definition: the starting day *is* the Gregorian day that *corresponds to* the time interval that *is part of* the Gregorian year and that *is* an instance *of* day-of-year 1
Necessity: Each Gregorian year *has* exactly one starting day.
Necessity: The index *of* the starting day *of* each Gregorian year that *follows* Gregorian year 1600 equals 584 391
plus 365 times (index *of* the Gregorian year minus 1601)
plus ((index *of* the Gregorian year minus 1601) divided by 4)
minus ((index *of* the Gregorian year minus 1601) divided by 100)
plus ((index *of* the Gregorian year minus 1601) divided by 400).
Necessity: The index *of* each Gregorian year *is greater than* 1581.
Note: The Gregorian calendar was adopted in different places at different times between 1582 and 1918. The formula is only valid for Gregorian dates.
Note: In mathematical form, the definition above is:
sd = 584 391 + (365 * y) + (y/4) - (y/100) + (y/400)
where:
sd is the index of the starting day
y is the index of a Gregorian year – 1601
y >= zero
/ is integer division
Note: 584 391 is the index of 1 January 1601, computed as 577 738 (index of 15 October 1582) plus 6653 days from 15 October 1582 through 1 January 1601.
Note: 1 January 1601 is used as the basis for this formula because the pattern of leap days is consistent since 1601. It is the first day after the first quadricentennial year after [Inter Gravissimas]. This day is picked because the first day of a Gregorian year does not include any leap day that occurs during that Gregorian year.
Note: The definition compensates for leap days by adding 1 for each 4th year, subtracting 1 for each 100th year (because most centurial years are not leap years), and adding 1 for each 400th year (because quadricentennial years are leap years), per [Inter Gravissimas].
Note: This formula is valid only for Gregorian calendar years after 1600.
Example: The first calendar day of 2010 is Gregorian day 733775.

## day-of-common-year

Concept Type: role
General Concept: nonnegative integer
Definition: the number of days between the beginning of a Gregorian year and the beginning of a given Gregorian month of year in a common year

## Gregorian month of year *has* day-of-common-year

Definition: the day-of-common-year *is* the number of days between the beginning of a common year and the instance *of* the Gregorian month of year that *is part of* the common year
Note: The day-of-common-year for each Gregorian month-of-year is given in Table 11.1.
Example: The day-of-common-year for April is 90 (days). The duration of the time period from January to April in a common year is 31 days (of January) + 28 days (of February) + 31 days (of March).

## day-of-leap-year

| | |
|---|---|
| Concept Type: | role |
| General Concept: | nonnegative integer |
| Definition: | the number of days between the beginning of a Gregorian year and the beginning of a given Gregorian month of year in a leap year |

## Gregorian month of year *has* day-of-leap-year

| | |
|---|---|
| Definition: | the day-of-leap-year *is* the number of days between the beginning of a leap year and the instance *of* the Gregorian month of year that *is part of* the leap year |
| Note: | The day-of-leap-year for each Gregorian month-of-year is given in Table 11.1. |
| Example: | The day-of-leap-year for April is 91. The duration of the time period from January to April in a leap year is 31 days (of January) + 29 days (of February) + 31 days (of March). |

## Gregorian year month day coordinate

| | |
|---|---|
| Definition: | Gregorian date that *combines* a Gregorian year coordinate and that *combines* a Gregorian month coordinate and that *combines* a Gregorian day of month coordinate, and that *indicates* a Gregorian day |
| Necessity: | Each Gregorian year month day coordinate *indicates* the Gregorian day that equals the starting day *of* the Gregorian year that *is indicated by* the Gregorian year coordinate, plus the value taken for the start of each month from the table of calendar days (below) as indexed by the index of the Gregorian month coordinate and whether the Gregorian year coordinate *indicates* a leap year, plus the index of the Gregorian day of month coordinate minus 2. |
| Description: | The index of the Gregorian day on the Gregorian days scale is computed from the three components of the Gregorian year coordinate. |
| Example: | "2010 month 3 day 15" *combines* the set *of* {2010, month 3, day 15}, and *indicates* the Gregorian day that *has* index 733 848. The index is 149 457 calendar days after January 1, 1601, which has index 584 391 (the reference point for the formula).<br>The 149 457 days is calculated as:<br>365*(2010 – 1601) + (2010 – 1601)/4 – (2010-1601)/100 + (2010 – 1601)/ 400 (number of calendar days from Jan 1, 1601 to Jan 1, 2010)<br>plus 59 (to day 1 of month 3, from the table) plus 14 (from day 1 to day 15). |

**Table 11.1 - Index of the First Gregorian Day of Year of Each Gregorian Month of Year**

| Gregorian month of year index | Gregorian month of year term | Gregorian month of year day-of-common-year | Gregorian month of year day-of-leap-year |
|---|---|---|---|
| 1 | January | 1 | 1 |
| 2 | February | 32 | 33 |
| 3 | March | 60 | 61 |
| 4 | April | 91 | 92 |
| 5 | May | 121 | 122 |
| 6 | June | 152 | 153 |
| 7 | July | 182 | 183 |
| 8 | August | 213 | 214 |
| 9 | September | 244 | 245 |
| 10 | October | 274 | 275 |
| 11 | November | 305 | 306 |
| 12 | December | 335 | 336 |

The table shown above is derived from Table 1 of [ISO 8601].

## Gregorian year day coordinate

Definition: Gregorian date that *combines* a Gregorian year coordinate and that *combines* a Gregorian day of year coordinate and that *indicates* a Gregorian day

Necessity: Each Gregorian day year coordinate *indicates* a Gregorian day that equals the index *of* the starting day *of* the Gregorian year that *is indicated by* the Gregorian year coordinate, plus the index of the Gregorian day of year coordinate minus 1.

Description: A Gregorian day year coordinate combines a Gregorian year coordinate and a Gregorian day of year coordinate to identify a particular Gregorian day.

Example: "2010 day 45" *combines* the set *of* {2010, day 45}, and *indicates* the Gregorian day that *has* index 733 819. The index is 149 428 calendar days after January 1, 1601, which has index 584 391 (the reference point for the formula). The 149428 days is calculated as: 365*(2010 – 1601) + (2010 – 1601)/4 – (2010-1601)/100 + (2010 – 1601)/ 400 (number of calendar days from Jan 1, 1601 to Jan 1, 2010) plus 44 (from day 1 to day 45).

## Gregorian month day coordinate

Definition: relative compound time coordinate that *combines* a Gregorian month coordinate and a Gregorian day of month coordinate, and that *refers to* one instance *of* Gregorian day *in* a given Gregorian year

Necessity: Each Gregorian month day coordinate *converts to* the time set {Gregorian day of year from the start of the calendar year to the calendar month that has the index of the Gregorian month coordinate in common years, Gregorian day of year from the start of the calendar year to the calendar month that has the index of the Gregorian month coordinate in leap years } *plus* the index *of* the Gregorian day of month coordinate *minus* 1 day

Note: A Gregorian month day coordinate does not include a year number, so there is no way to know whether a March date follows a 28-day or 29-day February. For this reason, every Gregorian month coordinate after February 28 does not consistently *indicate* either of two possible Gregorian days of year. But it *converts to* the time set that includes both of them.

Example: "15 June" *combines* "June" and "(Gregorian day of month) 15". It *refers to* one calendar day in each Gregorian year, but in common years it is Gregorian day of year 165 and in leap years it is Gregorian day of year 166. So, "15 June" *converts to* the time set {Gregorian day of year 165, Gregorian day of year 166}.

Example: "15 January" *combines* "January" and "(Gregorian day of month) 15". It always *indicates* Gregorian day of year 15.

## Gregorian date

Synonym: Gregorian date coordinate

Definition: calendar date that *indicates* a Gregorian day

Dictionary Basis: ISO 8601 (2.1.9, 'calendar date')

Note: Gregorian date coordinates may be combined with time offsets, see clause 10.3.

Example: 1989 September 3

Example: 2005 day 49

## 11.8 Gregorian Indefinite Scale Comparisons and Conversions

These verb concepts enable comparison of time points that are on different indefinite time scales. These are absolute time points, meaning that each *corresponds to* exactly one time interval.

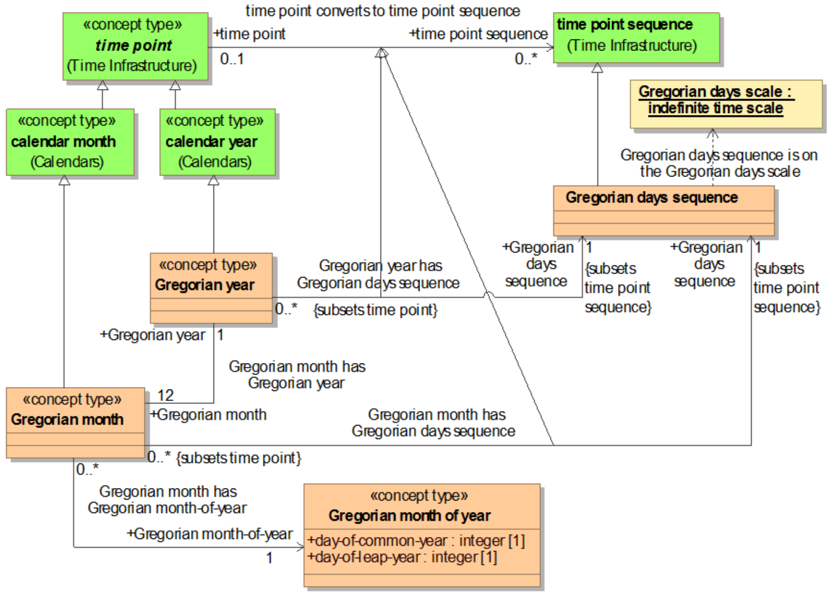


**Figure 11.9 - Gregorian Year Conversions**

The following Necessities identify the Gregorian calendar time points that can be converted to a common 'shared' time scale. Conversions to other time scales, such as International Atomic Time, are possible.

| | |
|---|---|
| Necessity: | Each Gregorian year *shares* the Gregorian days scale *with* each Gregorian month. |
| Example: | 1979 *shares* the Gregorian days scale *with* June 1990 |
| Necessity: | Each Gregorian year *shares* the Gregorian days scale *with* each Gregorian day. |
| Example: | 1949 *shares* the Gregorian days scale *with* 23 June 1990 |
| Necessity: | Each Gregorian month *shares* the Gregorian days scale *with* each Gregorian day. |
| Example: | June 1990 *shares* the Gregorian days scale *with* 23 June 1990 |

Conversions to other time scales, such as International Atomic Time, are possible.

The following concepts relate Gregorian years to the Gregorian days scale.

## Gregorian days sequence

| | |
|---|---|
| Definition: | time point sequence that *is on* the Gregorian days scale |

## Gregorian year *has* Gregorian days sequence

| | |
|---|---|
| General Concept: | time point *converts to* time point sequence |
| Definition: | the Gregorian year *converts to* the Gregorian days sequence |
| Necessity: | Each Gregorian year *converts to* exactly one Gregorian days sequence. |
| Necessity: | The first time point *of* the Gregorian days sequence *of* a Gregorian year *is* the starting day *of* the Gregorian year. |
| Necessity: | The Gregorian days sequence *of* a Gregorian year *has* cardinality 365 if the Gregorian year *is* a common year, and *has* cardinality 366 if the Gregorian year *is* a leap year. |
| Note: | The Gregorian year converts to the time point sequence whose first time point is the first Gregorian day of the year (the starting day) and that has as many Gregorian day time |

points as the year has days. The last time point will be the starting day plus length of year minus 1.

Example: The Gregorian year that *is indicated by* "2010" *converts to* Gregorian day 733 775 *through* Gregorian day 734 140.

The following concepts support conversion of Gregorian months to the Gregorian days scale. Note that the first two concepts associate the Gregorian month with "month-of-year" and "year" time points, and thus with common time coordinates.

### Gregorian month *has* Gregorian year

| | |
|---|---|
| Definition: | the Gregorian month *is part of* the Gregorian year |
| Necessity: | Each Gregorian month *has* exactly one Gregorian year. |
| Necessity: | The index *of* the Gregorian year *of* a Gregorian month *equals* 1 plus the integer quotient of dividing the index *of* the Gregorian month by 12. |
| Example: | Gregorian month 24108 *has* Gregorian year 2010 |

### Gregorian month *has* Gregorian month of year

| | |
|---|---|
| Definition: | the Gregorian month of year *corresponds to* the time interval that *is* the instance *of* the Gregorian month |
| Necessity: | Each Gregorian month *has* exactly one Gregorian month of year. |
| Necessity: | The index *of* the Gregorian month-of-year *of* a Gregorian month equals 1 plus the integer remainder of dividing the index *of* the Gregorian month by 12 |
| Example: | Gregorian month 24108 *has* Gregorian month of year 1, i.e., January. |

### Gregorian month *has* Gregorian days sequence

| | |
|---|---|
| General Concept: | time point *converts to* time point sequence |
| Definition: | the Gregorian month *converts to* the Gregorian days sequence |
| Necessity: | Each Gregorian month *has* exactly one Gregorian days sequence. |
| Necessity: | The duration *of* the Gregorian days sequence *of* a Gregorian month *is equal to* the duration *of* the time period that *is* the instance *of* the Gregorian month. |
| Necessity: | The index *of* the first time point *of* the Gregorian days sequence *of* a Gregorian month is equal to the index *of* the starting day *of* the Gregorian year *of* the Gregorian month minus 1 plus the day-of-common-year *of* the Gregorian month-of-year *of* the Gregorian month if the Gregorian year *is* a common year, or the day-of-leap-year *of* the Gregorian month-of-year if the Gregorian year *is* a leap year. |
| Description: | the Gregorian month *converts to* a sequence of Gregorian days on the indefinite Gregorian days scale, using the starting day of the Gregorian year of the Gregorian month and the day-of-common-year or day-of-leap-year of the Gregorian month-of-year of the Gregorian month. |
| Note: | The day-of-common-year and the day-of-leap-year for a Gregorian month-of-year are given in Table 11.1 |
| | The Gregorian month that *is indicated by* "June 2010" *converts to* Gregorian day 733 926 *through* Gregorian day 733 955 *on* the Gregorian days scale. The starting day of 2010 has index 733775, the day of common year of June is 152, and 2010 is a common year. |

## 11.9 Gregorian Month of Year Comparisons and Conversions

These verb concepts enable comparison of time points that are on the Gregorian year of days scale and the Gregorian year of months scale.

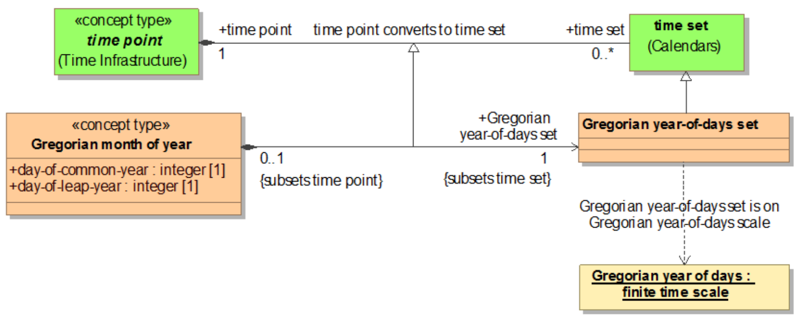


**Figure 11.10 - Gregorian Month of Year Conversion**

The following Necessity identifies the fact that Gregorian months of year and Gregorian days of year can be compared by conversion of the former to the Gregorian year of days scale:

Necessity: Each Gregorian month of year *shares* the Gregorian year of days scale *with* each Gregorian day of year.

Example: "May" can be compared with "day 33" on the Gregorian year of days scale

Because of leap days, a Gregorian month of year *converts to* a time set on the Gregorian year of days scale, rather than to an individual time point sequence. The following concepts characterize these conversions.

## Gregorian year of days set

Definition: time set *on* the Gregorian year of days scale

## Gregorian month of year *has* Gregorian year of days set

General Concept: time point *converts to* time set

Definition: the Gregorian month of year *converts to* the Gregorian year of days set

Necessity: Each Gregorian month of year *converts to* exactly one Gregorian year of days set.

Necessity: Each Gregorian month of year *converts to* the time set *on* the Gregorian year of days scale that is given for the Gregorian month of year in Table 11.2.

Note: The time set for January has only one member. All of the others have one time point sequence for common years and one time point sequence for leap years.

Note: These time sets could be formulated "intensionally" in much the same way as the Gregorian day time point sequences are formulated for Gregorian months, but since there are only 12, it is simpler to enumerate the extension of the verb concept.

Example: The Gregorian month of year that *is indicated by* 'August' *converts to* the time set {Gregorian day of year 213 *through* Gregorian day of year 243, Gregorian day of year 214 *through* Gregorian day of year 244}

**Table 11.2 - Time sets for Gregorian Months**

| Gregorian month of year index | Gregorian month of year term | Gregorian year of days set |
|---|---|---|
| 1 | January | {Gregorian day of year 1 *through* Gregorian day of year 31} |
| 2 | February | {Gregorian day of year 32 *through* Gregorian day of year 59, Gregorian day of year 32 *through* Gregorian day of year 60} |
| 3 | March | {Gregorian day of year 60 *through* Gregorian day of year 90, Gregorian day of year 61 *through* Gregorian day of year 91} |
| 4 | April | {Gregorian day of year 91 *through* Gregorian day of year 120, Gregorian day of year 92 *through* Gregorian day of year 121} |
| 5 | May | {Gregorian day of year 121 *through* Gregorian day of year 151, Gregorian day of year 122 *through* Gregorian day of year 152} |
| 6 | June | {Gregorian day of year 152 *through* Gregorian day of year 181, Gregorian day of year 153 *through* Gregorian day of year 182} |
| 7 | July | {Gregorian day of year 182 *through* Gregorian day of year 212, Gregorian day of year 183 *through* Gregorian day of year 213} |
| 8 | August | {Gregorian day of year 213 *through* Gregorian day of year 243, Gregorian day of year 214 *through* Gregorian day of year 244} |
| 9 | September | {Gregorian day of year 244 *through* Gregorian day of year 273, Gregorian day of year 245 *through* Gregorian day of year 274} |
| 10 | October | {Gregorian day of year 274 *through* Gregorian day of year 304, Gregorian day of year 275 *through* Gregorian day of year 305} |
| 11 | November | {Gregorian day of year 305 *through* Gregorian day of year 334, Gregorian day of year 306 *through* Gregorian day of year 335} |
| 12 | December | {Gregorian day of year 335 *through* Gregorian day of year 365, Gregorian day of year 336 *through* Gregorian day of year 366} |

The table shown above is derived from Table 1 of [ISO 8601].

________________________________________________
________________________________________________

# 12 ISO Week Calendar (normative)

## 12.1 General

The week calendar has been used for centuries, separate from and in combination with the Gregorian calendar, even though they are incommensurate. This calendar supports human discourse using weekday names such as "Monday", "Tuesday", and so forth.

This specification follows [ISO 8601] in defining "Monday" as the first day of the week. Various cultures and religions define other initial week days. Users of this specification are welcome to redefine the weekday concepts according to their preferences.

We define January 3, 2000 to be a Monday, and thereby define an indefinite sequence of time intervals that are ISO weeks. That is the basis for the ISO weeks time scale.

These ISO weeks are further gathered into ISO week-based years – time periods of exactly 52 or 53 ISO weeks that correspond roughly to Gregorian years. (The correspondence algorithm is given in ISO 8601. It is based on the 'first Thursday rule' – the first week of an ISO week-based year is the ISO week that contains the first Thursday in the Gregorian year, and it may contain days from the prior Gregorian year.) The ISO week-based year forms the basis for the ISO year of weeks time scale and the ISO year of weekdays time scale, which number the weeks and days, respectively, within an ISO week-based year. These scales then provide the basis for time coordinates of the ISO year-week form, such as "year 2000 week 6", and the ISO year-week-day form, such as "2004 week 37 day 2" or "Tuesday of 2004 week 37".

---

ISO Week Calendar Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Language: | English |
| Included Vocabulary: | Gregorian Calendar Vocabulary |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#ISOWeekCalendarVocabulary |

---

## 12.2 ISO Week Time Scales

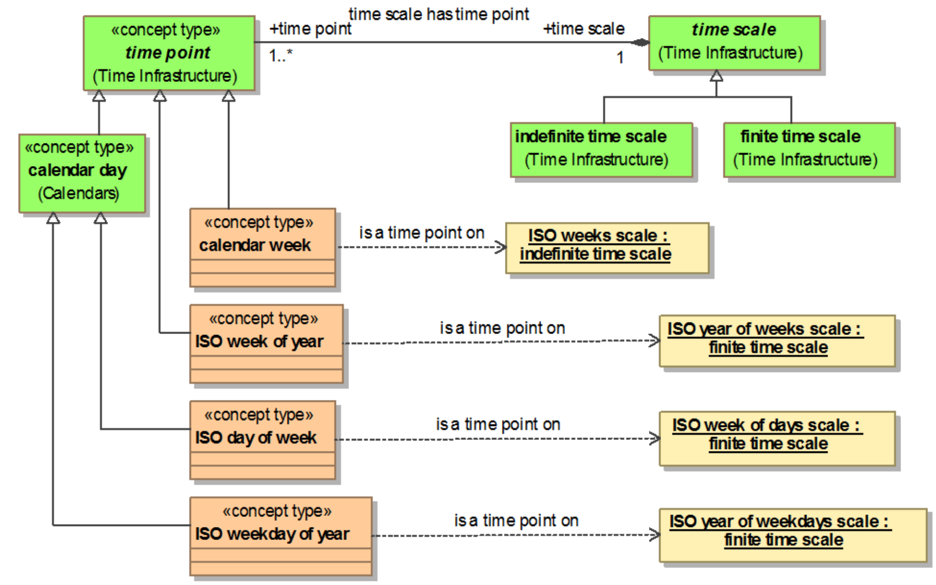


**Figure 12.1 - ISO Week Calendar time scales and time points**

### ISO weeks scale

| | |
|---|---|
| Definition: | indefinite time scale that *has* granularity week and that *has* time points that *are* ISO weeks |
| Necessity: | The index origin value *of* the ISO weeks scale *is* 104 304. |
| Necessity: | Calendar |
| Note: | Gregorian day 730 124 is Monday, January 3, 2000. This date was chosen for consistency with ISO 8601, which defines the origin of the ISO weeks calendar as Saturday, January 1, 2000, but that date is part of the last week of Gregorian year 1999 according to the algorithm in ISO 8601. |
| Note: | ISO week 104 303 ended on Gregorian day 730 123 and not 730 121 (a multiple of 7), because Gregorian day 1 was a Saturday, and ISO week 1 began the following Monday (per ISO 8601). |
| Note: | A more convenient reference for the ISO weeks scale is that January 1, 1601 was the Monday of calendar week 83 485. |

### ISO week of days scale

| | |
|---|---|
| Definition: | the finite time scale that *has* granularity 1 day and that *has* cardinality 7 and that *exactly subdivides* 'ISO week' |
| Necessity: | Each ISO week *subdivides into* exactly 7 ISO days of week. |
| Necessity: | The index origin value *of* theISO week of days scale *equals* 1. |
| Necessity: | The first time point *of* the ISO week of days scale *is* the index origin member *of* the ISO week of days scale |
| Necessity: | The index origin member *of* the ISO week of days scale *is* Monday. |

ISO year of weeks scale

| | |
|---|---|
| Definition: | the finite time scale that *has* granularity 1 week and that *has* cardinality 53 and that *repeats over* the ISO weeks scale |
| Description: | From [ISO 8601] clause 3.2.2: A calendar year has 52 or 53 weeks of year. |
| Note: | The ISO year of weeks scale repeats over the indefinite scale of ISO weeks and renumbers the weeks within a year from 1 to 52 or 53. |
| Necessity: | The index origin value *of* the ISO year of weeks scale *equals* 1. |
| Necessity: | The first time point *of* the ISO year of weeks scale *is* the index origin member *of* the ISO year of weeks scale. |
| Necessity: | The first time point *of* the ISO year of weeks scale *corresponds to* each time interval that *is* the instance *of* the starting week *of* a Gregorian year. |
| Note: | From the definition of the starting week, it follows that the Thursday of a first ISO week of year is one of the first 7 days of the year, but the Monday, Tuesday and Wednesday might be part of the previous year. |
| Note: | Any Gregorian year that begins on Thursday, and any leap year that begins on Wednesday, has 53 ISO week of year time intervals. Any other year has 52 ISO week of year time intervals. |

ISO year of weekdays scale

| | |
|---|---|
| Definition: | the finite time scale that *has* granularity 1 day and that *has* cardinality 371 and that *subdivides* each ISO week-based year |
| Note: | The ISO year of weekdays scale subdivides the ISO week-based year in parallel to the way the Gregorian year of days subdivides the Gregorian year. But the two kinds of year are of different lengths and are only loosely aligned. |
| Necessity: | The index origin value *of* the ISO year of weekdays scale *equals* 1. |
| Necessity: | The first time point *of* the ISO year of weekdays scale *is* the index origin member *of* the ISO year of weekdays scale. |
| Necessity: | Each *instance of* the first time point *of* the ISO year of weekdays *is* a Monday and *is part of* the instance *of* the starting week *of* a Gregorian year. |
| Note: | An instance of ISO weekday of year 1 may be as late as January 4 of the Gregorian year or as early as December 29 of the previous Gregorian year. |

## ISO week

| | |
|---|---|
| Dictionary Basis: | ISO 8601 (2.2.8, 'calendar week') |
| Concept Type: | concept type |
| Definition: | calendar week that *is on* the ISO weeks scale and that *corresponds to* a time interval that *is started by* a Monday |
| Note: | The ISO weeks scale is an indefinite time scale; so each ISO week corresponds to exactly one time interval. |
| Note: | This is an absolute time point because it is on an indefinite time scale. |
| Example: | The third ISO week of 2009. |

## ISO week-based year

| | |
|---|---|
| Definition: | time period that *has* duration 52 weeks or 53 weeks and that *is started by* an ISO week of year 1 and that *meets* an ISO week of year 1 |
| Necessity: | The ISO year of weeks scale *subdivides* each ISO week-based year. |
| Necessity: | Each ISO week-based year *is* an instance *of* a time point sequence *of* ISO weeks. |
| Note: | There is an indefinite sequence of ISO week-based years that covers the Time Axis in parallel to the indefinite sequence of Gregorian years. But it was not necessary to model it. ISO week-based years are identified by Gregorian year numbers and the 'first Thursday rule'. |

| | |
|---|---|
| Necessity: | First Thursday Rule: The first Thursday in an ISO week-based year is the first Thursday of a Gregorian year. |
| Necessity: | Each ISO week-based year *is started by* a time interval that *is the* 3 days *preceding* the first Thursday *of* a Gregorian year. |
| Note: | The last Thursday of a Gregorian year is part of the last week of the corresponding ISO week-based year. That determines whether the ISO week-based year has 52 weeks or 53 weeks. |
| Note: | Any Gregorian year that begins on Thursday, and any leap year that begins on Wednesday, relates to an ISO week-based year that has 53 ISO week of year time intervals and 371 ISO weekday of year time intervals. The first ISO week of year includes 2 or 3 days from the prior year (from the Monday to the start of the year), and the 53rd ISO week of year includes 2 or 3 days from the following year (from the Thursday or Friday that is December 31st through the following Sunday). Any other year has 52 ISO week of year time intervals and 364 ISO weekday of year time intervals, but it may include 1 or 2 days of the prior year or 1 or 2 days from the following year, while losing 1 to 3 days to the other of them. |

## ISO day of week

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | calendar day that *is on* the ISO week of days scale |
| Note: | This is a relative time point because it is on a finite time scale. |
| Necessity: | ISO day of week 1 *is* the concept 'Monday'. |
| Necessity: | ISO day of week 2 *is* the concept 'Tuesday'. |
| Necessity: | ISO day of week 3 *is* the concept 'Wednesday'. |
| Necessity: | ISO day of week 4 *is* the concept 'Thursday'. |
| Necessity: | ISO day of week 5 *is* the concept 'Friday'. |
| Necessity: | ISO day of week 6 *is* the concept 'Saturday'. |
| Necessity: | ISO day of week 7 *is* the concept 'Sunday'. |
| Source: | ISO 8601 (Table 2) |
| Note: | Other day of week time scales may choose a different numbering. |

## ISO week of year

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | time point that *is on* the ISO year of weeks scale |
| Necessity: | Each ISO week of year *renumbers* at least 1 ISO week. |
| Note: | This is a relative time point because it is on a finite time scale. |

## ISO weekday of year

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | calendar day that *is on* the ISO year of weekdays scale |
| Necessity: | Each ISO weekday of year *renumbers* at least 1 Gregorian day. |
| Note: | Each ISO weekday of year time point is a calendar day of each ISO week-based year. The usual time coordinate has "week and day" form, i.e., an ISO week of year coordinate and an ISO day of week coordinate. See clause 18. |

The following concepts were created to support the formal definition of the ISO year of weeks and ISO year of weekdays time scales.

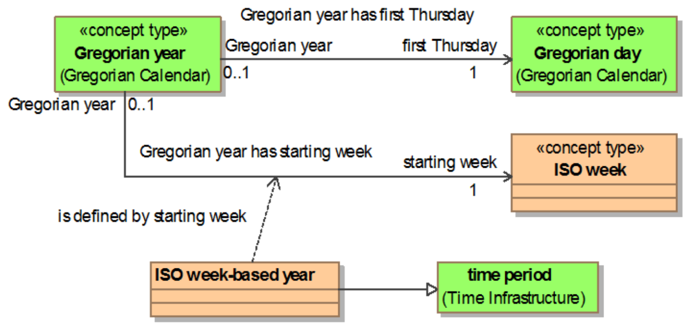


Figure 12.2 Starting week

## first Thursday

| | |
|---|---|
| Concept Type: | role |
| General Concept: | Gregorian day |
| Description: | the Gregorian day that is the first Thursday in a given Gregorian year |

## Gregorian year *has* first Thursday

| | |
|---|---|
| Definition: | the first Thursday *is* the Gregorian day that *corresponds to* the time interval that *is* a Thursday and that *is part of* the Gregorian year and that *is* not *after* a Thursday that *is part of* the Gregorian year |
| Necessity: | The index *of* the first Thursday *of* a Gregorian year equals the index *of* the starting day *of* the Gregorian year plus 6 minus the remainder of dividing the index *of* the starting day *of* the Gregorian year by 7. |
| Note: | If the remainder of dividing the index of the starting day by 7 is 0, the starting day is a Friday, if the remainder is 1, it is a Saturday, and so on. So, 6 minus the remainder is the number of days from the starting day to the first Thursday. |
| Note: | This concept is introduced only to define the starting week concept. |

## starting week

| | |
|---|---|
| Concept Type: | role |
| General Concept: | ISO week |
| Definition: | the ISO week that *includes* the first Thursday *of* a given Gregorian year |
| Note: | This definition follows the specifications in ISO 8601. |
| Note: | It is possible that the Monday, Tuesday, and Wednesday of the starting week are part of the previous Gregorian year. It is also possible that January 1st, 2nd, and 3rd, are not part of the starting week and are part of the last week of the previous year. |
| Example: | January 1, 2000 is a Saturday. So the first Thursday of 2000 is January 6 and the starting week of 2000 begins on Monday, January 3. Thus January 1, 2000 and January 2, 2000 are part of the last week of 1999. |
| Example: | January 1, 2002 is a Tuesday. So the first Thursday of 2002 is January 4, and the starting week of 2002 begins on Monday, December 31, 2001. |

## Gregorian year *has* starting week

| | |
|---|---|
| Definition: | the starting week *is* the ISO week that *includes* the first Thursday *of* the Gregorian year |

Necessity: The index *of* the starting week *of* a Gregorian year equals the index *of* the starting day *of* the Gregorian year divided by 7 plus 1.

Note: This formula works because Gregorian day 1 was a Saturday. The quotient is the number of complete weeks though a Friday that is on or before the starting day. So the quotient is greater by 1 exactly when January 1 falls on a Friday, Saturday, or Sunday, and the *following* Monday begins the starting week. Otherwise the starting week begins *on or before* the starting day, and there is one less complete week before it.

## 12.3 Days of the week

The concepts in this clause are the traditional days of the week, each of which is treated as a concept that corresponds to time intervals. They are defined to correspond to specific Gregorian days, by requiring that January 1, 2000 is a Saturday.

The days of the week are not 'time points' as defined. They become time points when they are chosen to be members of a time scale. This allows different time scales to make different choices for the first day of the week, without changing the relationship between the day of week and the actual time intervals.

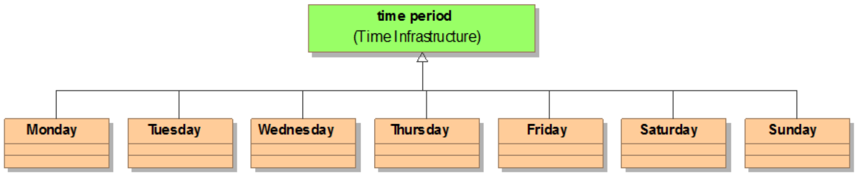


**Figure 12.2 – Week days**

### Monday

Definition: time interval that *has* duration 1 day and that *meets* a Tuesday

### Tuesday

Definition: time interval that *has* duration 1 day and that *meets* a Wednesday

### Wednesday

Definition: time interval that *has* duration 1 day and that *meets* a Thursday

### Thursday

Definition: time interval that *has* duration 1 day and that *meets* a Friday

### Friday

Definition: time interval that *has* duration 1 day and that *meets* a Saturday

### Saturday

Definition: time interval that *has* duration 1 day and that *meets* a Sunday

Necessity: One Saturday *is* the time interval that *has* duration 1 day and that *starts* Gregorian year 2000.

Note: This requirement anchors the repeating sequence of days of week to specific Gregorian days. It requires that January 1, 2000 is a Saturday. It follows that January 2, 2000 must be the Sunday that it meets, and so on.

## Sunday

Definition: time interval that *has* duration 1 day and that *meets* a Monday

## 12.4 Week Time Coordinates

This sub clause supports the following time coordinates based on weeks:

- An ISO day of week coordinate *indicates* an ISO day of week, for example "Tuesday"
- An ISO week of year coordinate *indicates* an ISO week of year, for example "week 15"
- An ISO week day coordinate *combines* an ISO week of year coordinate and an ISO day of week to *indicate* an ISO weekday of year, for example "Tuesday week 15"
- An ISO year week coordinate *combines* a Gregorian year and an ISO week of year coordinate *to indicate* an ISO week, for example "2010 week 15."
- An ISO year week day coordinate *combines* a Gregorian year, an ISO week of year coordinate, and an ISO day of week *to indicate* a calendar day, for example "Tuesday 2010 week 15."

The detailed definitions of these time coordinates follow.

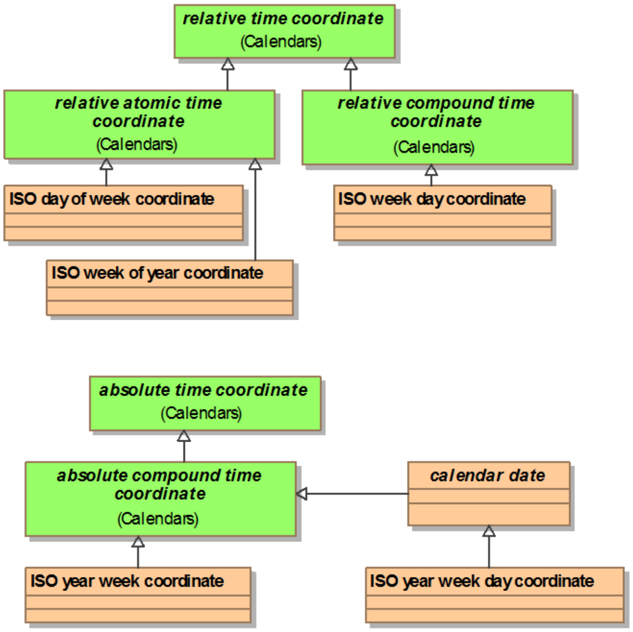


**Figure 12.3 - Week Coordinates**

## ISO day of week coordinate

Definition: relative atomic time coordinate that *indicates* an ISO day of week

Necessity: Each ISO day of week coordinate *indicates* an ISO day of week that *has* the index equal to the index *of* the ISO day of week coordinate

Description: An ISO day of week coordinate directly identifies an ISO day of week.

| | |
|---|---|
| Necessity: | Each ISO day of week coordinate *is greater than or equal to* 1. |
| Necessity: | Each ISO day of week coordinate *is less than or equal to* 7. |
| Example: | Wednesday |

## ISO week of year coordinate

| | |
|---|---|
| Synonym: | ISO week number |
| Definition: | relative atomic time coordinate that *indicates* an ISO week of year |
| Necessity: | Each ISO week of year coordinate *indicates* the ISO week of year that *has* an index equal to the index *of* the ISO week of year coordinate. |
| Description: | A ISO week of year coordinate gives the number of an ISO week within a calendar year. |
| Description: | Number which *identifies* an ISO week within its calendar year according to the rule that the first ISO week of a calendar year is that which includes the first Thursday of that calendar year and that the last ISO week of a calendar year is the ISO week immediately preceding the first ISO week of the next calendar year. See [ISO 8086] clause 2.2.10 for details. |
| Necessity: | Each ISO week of year coordinate *is greater than or equal to* 1. |
| Necessity: | Each ISO week of year coordinate *is less than or equal to* 53. |
| Example: | week 35 |

## ISO week day coordinate

| | |
|---|---|
| Definition: | relative compound time coordinate that *combines* an ISO week of year coordinate and that *combines* an ISO day of week coordinate and that *indicates* an ISO weekday of year |
| Necessity: | Each ISO weekday coordinate *indicates* the ISO weekday of year that *has* an index equal to 7 times (the index *of* the ISO week of year coordinate – 1) plus the index *of* the ISO day of week coordinate. |
| Description: | An ISO week day coordinate combines an ISO week of year coordinate and an ISO day of week coordinate to identify an ISO weekday of year, i.e., a calendar day within an ISO week-based year. |
| Note: | The first ISO week of year may start up to 3 days before the first calendar day of a Gregorian year, and the last ISO week of year may include up to 3 calendar days from the following Gregorian year. See [ISO 8601] clause 3.2.2 for details. |
| Example: | Wednesday week 35 *indicates* ISO weekday of year 241 |
| Example: | Sunday week 1 *indicates* ISO weekday of year 7 |

## ISO year week coordinate

| | |
|---|---|
| Definition: | absolute compound time coordinate that *combines* a Gregorian year coordinate and that *combines* an ISO week of year coordinate, and that *indicates* an ISO calendar week |
| Necessity: | Each ISO year week coordinate *indicates* the ISO calendar week that *has* the index that *equals* the index *of* the ISO week of year coordinate minus 1 plus the index *of* the starting week *of* the Gregorian year that *is indicated by* the Gregorian year coordinate. |
| Description: | An ISO year week coordinate identifies a calendar week time interval by the Gregorian year in which it occurs and its relative position within the Gregorian year. Note that the relationship between week of year 1 and January 1 is complex. |
| Example: | 2010 week 35 *indicates* the ISO calendar week 104860. January 1, 2010 is a Friday. So the starting week of 2010 begins on the following Monday, and is calendar week 104826. Calendar week 104860 = 104826 + 35 – 1. |

### ISO year week day coordinate

| | |
|---|---|
| Definition: | Gregorian date that *combines* a Gregorian year coordinate and that *combines* an ISO week of year coordinate and that *combines* an ISO day of week coordinate |
| Description: | An ISO year week day coordinate *indicates* a calendar day by a combination of a Gregorian year coordinate, an ISO week of year coordinate, and a ISO day of week coordinate. |
| Necessity: | Each ISO year week day coordinate *indicates* the Gregorian day that *has* an index that *equals* 7 times the index *of* the ISO week of year coordinate *of* the ISO year week day coordinate<br>plus the index *of* the ISO day of week coordinate *of* the ISO year week day coordinate minus 11<br>plus the index *of* the first Thursday *of* the Gregorian year that *is indicated by* the Gregorian year coordinate *of* the ISO year week day coordinate. |
| Note: | That is, the ISO year week day coordinate (y, w, d) indicates the Gregorian day whose index is<br>7 * (w – 1) + (d – 1) + firstThursday(y) – 3, or<br>7 * w + d + firstThursday(y) – 11.<br>The beginning day of the starting week is the Monday before the first Thursday, so its index is the index of the first Thursday minus 3. |
| Example: | Wednesday 2010 week 35 indicates Gregorian day 834 647 (starting weekday *of* 2010) + 238 (7 * (35 – 1)) + 3 – 1 □ Gregorian day 834 887. |

# 13 Time of Day (normative)

## 13.1 General

### Time of Day Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Language: | English |
| Included Vocabulary: | Calendars Vocabulary |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#TimeofDayVocabulary |

## 13.2 Time of Day Time Scales

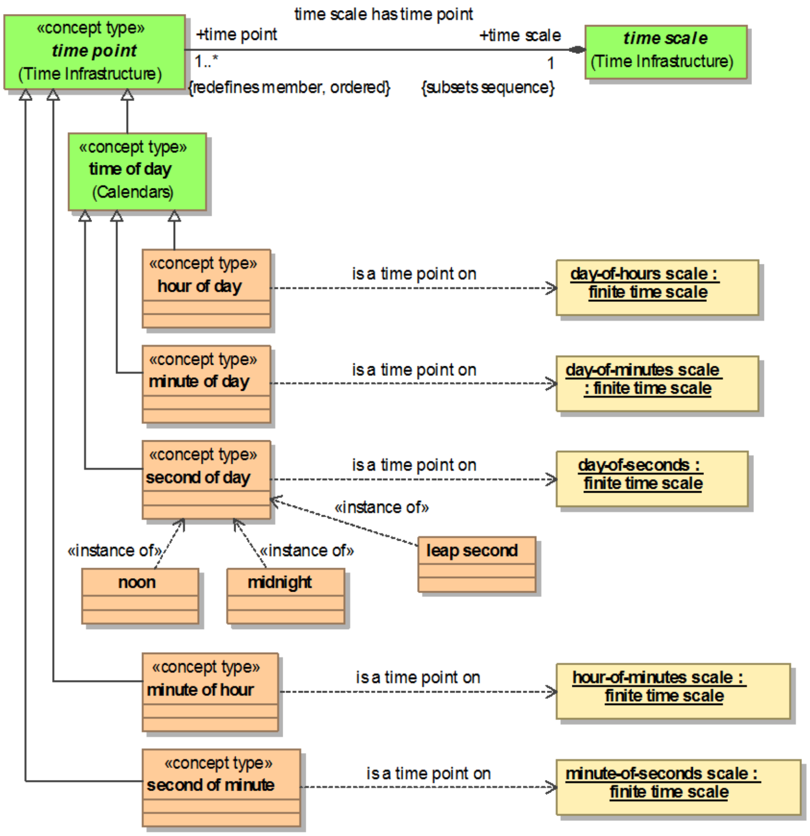


**Figure 13.1 - Time of Day Time Scales, Time Points, and Time Periods**

### day of hours scale

| | |
|---|---|
| Definition: | the finite time scale that *has* granularity 1 hour and that *has* cardinality 24 and that *exactly subdivides* 'Gregorian calendar day' |
| Necessity: | Each calendar day *subdivides into* exactly 24 hours of day. |
| Necessity: | The index origin value *of* the day of hours scale *equals* 0. |
| Necessity: | The first position *of* the day of hours scale *is* the index origin member *of* the day of hours scale. |

### day of minutes scale

| | |
|---|---|
| Definition: | the finite time scale that *has* granularity 1 minute and that *has* cardinality 1440 and that *exactly subdivides* 'Gregorian calendar day' |
| Necessity: | Each calendar day *subdivides into* exactly 1440 minutes of day. |
| Necessity: | The index origin value *of* the day of minutes scale *equals* 0. |

Necessity: The first position of the day of minutes scale is the index origin member of the day of minutes scale.

### day of seconds scale

Definition: the finite time scale that has granularity 1 second and that has cardinality 86400 and that exactly subdivides 'Gregorian calendar day'
Necessity: Each calendar day subdivides into exactly 86400 seconds of day.
Necessity: The granularity of the day of seconds scale is 'second'.
Necessity: The index origin value of the day of seconds scale equals 0.
Necessity: The first position of the day of seconds scale is the index origin member of the day of seconds scale.

### hour of minutes scale

Definition: the finite time scale that has granularity 1 minute and that has cardinality 60 and that exactly subdivides 'minute of hour'
Necessity: Each hour of day subdivides into exactly 60 minutes of hour.
Necessity: The index origin value of the hour of minutes scale equals 0.
Necessity: The first position of the hour of minutes scale is the index origin member of the hour of minutes scale.

### minute of seconds scale

Definition: the finite time scale that has granularity 1 second and that has cardinality 60 and that exactly subdivides 'minute of hour'
Necessity: Each minute of hour subdivides into exactly 60 seconds of minute.
Necessity: The index origin value of the minute of seconds scale equals 0.
Necessity: The first position of the minute of seconds scale is the index origin member of the minute of seconds scale.

## 13.3 Time of Day Time Points

### midnight

Definition: second of day 0
Necessity: time point 0 on the day of seconds time scale corresponds to time intervals that have duration 1 second and start an instance of a calendar day

### noon

Definition: second of day 43 200
Note: 43 200 = 12 hours * 60 minutes * 60 seconds

### hour of day

Dictionary Basis: ISO 8601 (3.2.3)
Concept Type: concept type
Definition: time point that is on the day of hours scale where the index of the time point represents the number of full hours that have elapsed since midnight at the start of each time interval that the time point corresponds to
Necessity: Each time interval is an instance of hour of day 0 if and only if the time interval has duration 1 hour and starts an instance of a calendar day.
Necessity: For each hour of day$_1$ that has an index that is greater than 0, each time interval is an instance of hour of day$_1$ if and only if the time interval has duration 1 hour and is

| | |
|---|---|
| | *met by* an instance *of* the hour of day that *precedes* hour of day$_1$ *on* the day of hours scale. |
| Note: | The standard that the hour of day is counted since midnight was established by the International Meridian Conference of 1884 [International Meridian]. |
| Note: | This is a relative time point because it is on a finite time scale. |

## minute of day

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | time point that *is on* the day of minutes scale where the index of the time point represents the number of full minutes that have elapsed since midnight at the start of each time interval that the time point *corresponds to* |
| Necessity: | Each time interval *is* an instance *of* minute of day 0 if and only if the time interval *has* duration 1 minute and *starts* an instance *of* a calendar day. |
| Necessity: | For each minute of day$_1$ that *has* an index that *is greater than* 0, each time interval *is* an instance *of* minute of day$_1$ if and only if the time interval *has* duration 1 minute and *is met by* an instance *of* the minute of day$_2$ that *precedes* minute of day$_1$ *on* the day of minutes scale. |
| Note: | This is a relative time point because it is on a finite time scale. |
| Example: | "03:15" is the minute-of-day that has index 195 |

## second of day

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | time point that *is on* the day of seconds scale where the index of the time point represents the number of full seconds that have elapsed since midnight at the start of each time interval that the time point *corresponds to* |
| Necessity: | Each time interval *is* an instance *of* second of day 0 if and only if the time interval *has* duration 1 second and *starts* an instance *of* a calendar day. |
| Necessity: | For each second of day$_1$ that *has* an index that *is greater than* 0, each time interval *is* an instance *of* second of day$_1$ if and only if the time interval *has* duration 1 second and *is met by* an instance *of* the second of day$_2$ that *precedes* second of day$_1$ *on* the day of seconds scale. |
| Note: | This is a relative time point because it is on a finite time scale. |
| Example: | "03:15:48" is the second-of-day that has index 11748 |

## leap second

| | |
|---|---|
| Concept Type: | concept type |
| Definition: | second of day that *is* used to adjust UTC to ensure appropriate agreement with the rotation of the Earth |
| Dictionary Basis: | ISO 8601 (2.2.2, 'leap second') |
| Note: | Leap seconds are added or deleted at 23:59:59 on specific calendar days of UTC. These intercalary seconds of day adjust midnight of the next calendar day to match Earth's rotation. The International Earth Rotation and Reference Systems Service [IERS] announces leap seconds whenever the difference between UTC and the Earth's rotation exceeds 0.6 seconds. |
| Note: | As of 2012, there is a proposal to drop the 'leap second' concept. This proposal will be formally considered at the World Radio Conference in 2015. |

## minute of hour

| | |
|---|---|
| Dictionary Basis: | ISO 8601 (3.2.3) |
| Concept Type: | concept type |

| | |
|---|---|
| Definition: | time point that *is on* the hour of minutes scale where the index of the time point represents the number of full minutes that have elapsed since the last full hour at the start of each time interval that the time point *corresponds to* |
| Necessity: | Each time interval *is* an instance *of* minute of hour 0 if and only if the time interval *has* duration 1 minute and *starts* an instance *of* an hour of day. |
| Necessity: | For each minute of hour$_1$ that *has* an index that *is greater than* 0, each time interval *is* an instance *of* minute of hour$_1$ if and only if the time interval *has* duration 1 minute and *is met by* an instance *of* the minute of hour$_2$ that *precedes* minute of hour$_1$ *on* the hour of minutes scale. |
| Note: | This is a relative time point because it is on a finite time scale. |

## second of minute

| | |
|---|---|
| Dictionary Basis: | ISO 8601 (3.2.3) |
| Concept Type: | concept type |
| Definition: | time point that *is on* the minute of seconds scale where the index of the time point represents the number of full seconds that have elapsed since the last full minute at the start of each time interval that the time point *corresponds to* |
| Necessity: | Each time interval *is* an instance *of* second of minute 0 if and only if the time interval *has* duration 1 second and *starts* an instance *of* a minute of day. |
| Necessity: | For each second of minute$_1$ that *has* an index that *is greater than* 0, each time interval *is* an instance *of* second of minute$_1$ if and only if the time interval *has* duration 1 second and *is met by* an instance *of* the second of minute$_2$ that *precedes* second of minute$_1$ *on* the minute of seconds scale. |
| Note: | This is a relative time point because it is on a finite time scale. |
| Note: | Business Calendar Concepts |

## hour period

| | |
|---|---|
| Definition: | time period that begins and ends at the same minute of hour on consecutive hours of day |
| Example: | 1:05 *to* 2:05 |

## 13.4 Time of Day Time Coordinates

This sub clause defines the following relative time coordinates and time scales for these combinations of time of day time units:

- An hour coordinate *indicates* an hour of day, for example "hour 10" or "10 a.m."
- A minute coordinate *indicates* a minute of hour, for example "minute 33"
- A second coordinate *indicates* a second of minute, for example "second 27"
- An hour minute coordinate *combines* an hour of day and a minute of hour, to *indicate* a minute of day, for example "10:33"
- An hour minute second coordinate *combines* an hour of day, a minute of hour, and a second of minute, to *indicate* a second of day, for example "10:33:27"

This specification does not define time coordinates and time scales for fractions of seconds (e.g., milliseconds). Business vocabularies may extend this specification as needed to address fractional seconds.

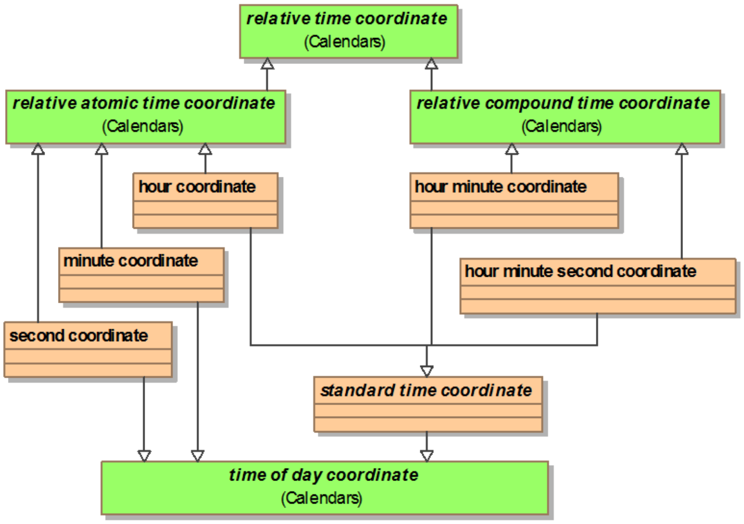


**Figure 13.2 - Time of Day Coordinates**

## hour coordinate

| | |
|---|---|
| Definition: | relative atomic time coordinate that *indicates* an hour of day |
| Necessity: | Each hour coordinate *indicates* an hour of day that *has* the index equal to the index *of* the hour coordinate. |
| Description: | An hour coordinate directly *indicates* an hour of day. |
| Necessity: | Each hour coordinate *is greater than or equal to* 0. |
| Necessity: | Each hour coordinate *is less than or equal to* 23. |
| Example: | "11 p.m." and "23:00" *indicate* the same hour of day |

## minute coordinate

| | |
|---|---|
| Definition: | relative atomic time coordinate that *indicates* a minute of hour |
| Necessity: | Each minute coordinate *indicates* a minute-of-hour that *has* the index equal to the index *of* the minute coordinate. |
| Description: | A minute coordinate directly *indicates* a minute of hour. |
| Necessity: | Each minute coordinate *is greater than or equal to* 0. |
| Necessity: | Each minute coordinate *is less than or equal to* 59. |
| Example: | minute 23 |
| Note: | This type of time coordinate is not common in everyday use, but is defined here to support the concepts 'hour minute coordinate' and 'hour minute second coordinate' |

## second coordinate

| | |
|---|---|
| Definition: | relative atomic time coordinate that *indicates* a second of minute |
| Necessity: | Each second coordinate *indicates* a second of minute that *has* the index equal to the index *of* the second coordinate. |
| Necessity: | Each second coordinate *is greater than or equal to* 0. |
| Necessity: | Each second coordinate *is less than or equal to* 59. |
| Example: | second 45 |

Note: This type of time coordinate is not common in everyday use, but is defined here to support the concept ‘hour minute second coordinate’

## hour minute coordinate

Definition: relative compound time coordinate that *combines* an hour coordinate and that *combines* a minute coordinate, and that *indicates* a minute of day

Necessity: Each hour minute coordinate *indicates* a minute of day that *has* index 60 times the index *of* the hour coordinate plus the index *of* the minute coordinate.

Description: An hour minute coordinate *combines* an hour coordinate and a minute coordinate to *indicate* a minute of day.

Example: “11:23 a.m.” *combines* the set *of* {11 a.m, minute 23}, and *indicates* the minute of day that *has* index 683

## hour minute second coordinate

Definition: relative compound time coordinate that *combines* an hour coordinate and that *combines* a minute coordinate and that *combines* a second coordinate and that *indicates* a second of day

Necessity: Each hour minute second coordinate *indicates* a second of day that *has* index 3 600 times the index *of* the hour coordinate plus 60 times the index *of* the minute coordinate plus the index *of* the second coordinate.

Example: “11:23:49 a.m.” *combines* the set *of* {11 a.m, minute 23, second 49}, and *indicates* the second of day that *has* index 36 432

## standard time coordinate

Definition: time of day coordinate that *is* an hour coordinate or hour minute coordinate or hour minute second coordinate

Dictionary Basis: ISO 8601 (2.1.9, ‘calendar date’)

Note: standard time coordinates may be combined with time offsets, see clause 10.3.

Example: 3 p.m.

Example: 15:00

Example: 15:00:35

## 13.5 Time of Day Comparisons and Conversions

Hours of day, minutes of day, and seconds of day may be compared with each other.

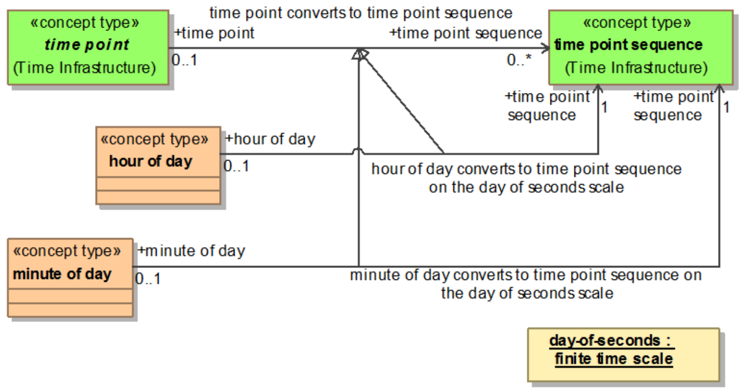


**Figure 13.3 - Time of Day Conversions**

The following Necessities identify which time of day time points can be compared by conversion to a common 'shared' time scale:

| | |
|---|---|
| Necessity: | Each hour of day *shares* the day of minutes scale *with* each minute of day. |
| Example: | "10 a.m." can be compared with "10:39" on the day of minutes scale |
| Necessity: | Each hour of day *shares* the day of seconds scale *with* each second of day. |
| Example: | "10 a.m." can be compared with "10:39:42" on the day of seconds scale |
| Necessity: | Each minute of day *shares* the day of seconds scale *with* each second of day. |
| Example: | "10:39" can be compared with "10:54:48" on the day of seconds scale |

Hours of day and minutes of day can *be converted to* the day of seconds scale.

## hour of day *converts to* time point sequence *on the day of seconds scale*

| | |
|---|---|
| General Concept: | time point *converts to* time point sequence *on* time scale |
| Definition: | the time point sequence *is on* the day of seconds scale and the index *of* the first time point *of* the time point sequence *equals* 3 600 times the index *of* the hour of day, and the index *of* the last time point *of* time point sequence *is* the index *of* the first time point plus 3 599 |
| Description: | The hour of day *converts to* a sequence of seconds of day whose indices are computed by the formula. |
| Example: | The hour of day that *is indicated by* "hour 0" *converts to* second of day 0 *through* second of day 3 599 *on* the day of seconds scale |

## minute of day *converts to* time point sequence *on the day of seconds scale*

| | |
|---|---|
| General Concept: | time point *converts to* time point sequence *on* time scale |
| Definition: | the time point sequence *is on* the day of seconds scale and the index *of* the first time point *of* time point sequence *equals* 60 times the index *of* minute of day, and the index *of* the last time point *of* time point sequence *is* the index *of* the first time point plus 59 |
| Description: | The minute of day *converts* to a sequence of seconds of day whose indices are computed by the formula. |
| Example: | The minute of day that *is indicated by* "1:48" *converts to* second of day 6 480 *through* second of day 6 539 *on* the day of seconds scale. |

______________________________________________________
______________________________________________________

## 13.6 Time Zones

In order to make local noon (12:00) coincide approximately with the Sun's zenith at the locale, authorities in each locale specify one or more local calendars to be used, during different seasons of a year, for commerce in the locale. A locale in which a standard calendar is used is called a "time zone." The governing authority over time zones is the national or state government of the locale. Many local calendars are named. For example, Pacific Daylight Time, Eastern Standard Time, British Summer Time. Two or more time zones may have the same name, e.g., there is an Eastern Standard Time in the U.S. and another in Australia, and they are different time zones.

A local calendar is UTC with a characteristic time offset from UTC by up to ±12 hours. These offsets are usually an integer number of hours or half hours. The nominal offset is zero at the Prime Meridian, +1 hour for each 15º of longitude east of the Prime Meridian, and –1 hour for each 15º of longitude west of the Prime Meridian. '+' means a particular reading of a clock set to the time of the local calendar occurs before a clock that is set to UTC has the same reading; '–' means the local reading occurs after the UTC reading. The duration between corresponding readings is the time offset. The 180º meridian is nominally the International Date Line: a date in locales west of the International Date Line (e.g., longitude 179ºE) is one day ahead of the date in locales east of the International Date Line (e.g., longitude 179ºW).

The time offset from UTC affects more than time of day for a local calendar: At any UTC time there is some locale that has a different local date that is one day before or after the UTC date: the date can be different as well as the hour and minute. For example, during periods when standard time is used in Australia (early April to early October), 18:00 UTC (19:00 BST in London) is 04:00 local time the next day in Sydney (UTC+10 hours); 04:00 UTC is 18:00 local time the previous day in Honolulu (UTC–10 hours); Honolulu and Sydney, being 20 hours apart, are on different dates for all but four hours each day (10:00 – 14:00 UTC that day). The approach adopted in this specification is to consider that each time zone has one or two distinguished local calendars.

A complete literal specification of a time interval includes a calendar specification as part of the time coordinate; otherwise there is a 24 hour ambiguity. For example, compare "July 4, 2010 12:00 PDT" to "July 4, 2010 12:00" or "July 4, 2010 PDT" to "July 4, 2010." Note the 24-hour ambiguity when the calendar specification is left out, not knowing where in the world the time is meant.

The intended calendar is often implied by the locale of the utterance of a time coordinate, or by the locale of the associated event, or by other context, but a calendar specification should be provided explicitly when necessary to remove all doubt. This is especially important in discourses that involve multiple time zones. When time coordinates are used in a discourse without specifying the time zone, it is assumed for purposes of comparison and date-time arithmetic that they are on the same calendar. Time references without calendar specifications in different discourses also without locale references are not prima facie comparable to within less than 24 hours.

A representation of a time offset may be combined with a date coordinate, a time of day coordinate, or a date time coordinate to indicate that the time coordinate is specified according to a local calendar that has that offset. The effect of the time offset is to shift the interpretation of the time coordinate with respect to UTC.

### 13.6.1 Calendar Offsets

This subclause defines the basic relationship between calendars that use the same nominal time scales and time points but use the time points to refer to different time intervals.

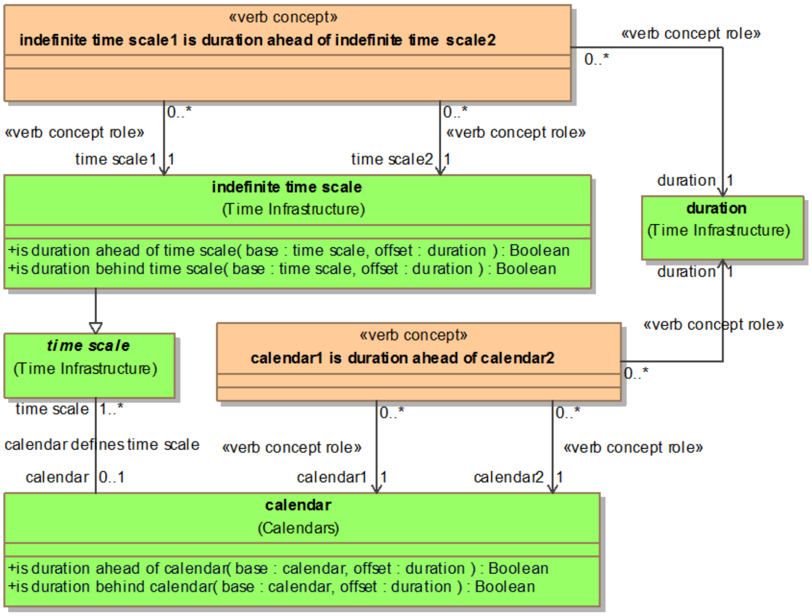


**Figure 13.4 - Calendars and Time Offsets**

## indefinite time scale$_1$ *is* duration *ahead of* indefinite time scale$_2$

Synonymous Form: indefinite time scale$_2$ *is* duration *behind* indefinite time scale$_1$
Synonymous Form: indefinite time scale$_1$ = indefinite time scale$_2$ + duration
Synonymous Form: indefinite time scale$_2$ = indefinite time scale$_1$ - duration
Definition: the granularity *of* indefinite time scale$_1$ *is* the granularity *of* indefinite time scale$_2$ and each time point$_1$ *of* indefinite time scale$_1$ *corresponds to* a time interval$_1$ that *starts* duration *before* the time interval that *is* the instance *of* the time point$_2$ that *is* a time point *of* indefinite time scale$_2$ and that *has* an index that *is equal to* the index *of* time point$_1$
Note: That is, the time scales have the same nominal time points but the correspondence to time intervals is adjusted by the time offset.
Note: In particular, the time point on indefinite time scale$_2$ that has the same index as the time point that defines the reference time interval for the indefinite time scale$_1$ corresponds to a time interval that starts duration before the reference time interval.

## calendar$_1$ *is* duration *ahead of* calendar$_2$

Synonymous Form: calendar$_2$ *is* time offset *behind* calendar$_1$
Synonymous Form: calendar$_1$ = calendar$_2$ + duration
Synonymous Form: calendar$_2$ = calendar$_1$ - duration
Definition: each indefinite time scale$_1$ that *is defined by* calendar$_1$ *is* duration *ahead of* the indefinite time scale$_2$ that *is defined by* calendar$_2$ and that *has* the granularity *of* indefinite time scale$_1$
Description: The two calendars have the same time scales, and the time scales *correspond to* time intervals that are duration apart.

| | |
|---|---|
| Note: | All of the time scales defined by calendar$_1$ are considered to be the same duration ahead of the corresponding time scales of calendar$_2$, because the finite time scales are defined relative to time points on the indefinite time scales. |
| Example: | Eastern Standard Time (EST) is UTC - 5 hours, and Pacific Standard Time (PST) is UTC - 8 hours. Therefore, EST *is* 3 hours *ahead of* PST, and PST *is* 3 hours *behind* EST. |
| Example: | India Standard Time (IST) = UTC + 5 hours 30 minutes. Therefore IST *is* 13 hours 30 minutes *ahead of* PST. And each Gregorian day in India begins 13 hours 30 minutes before the same Gregorian day in California. |

### 13.6.2 Time Zones and Standard Time

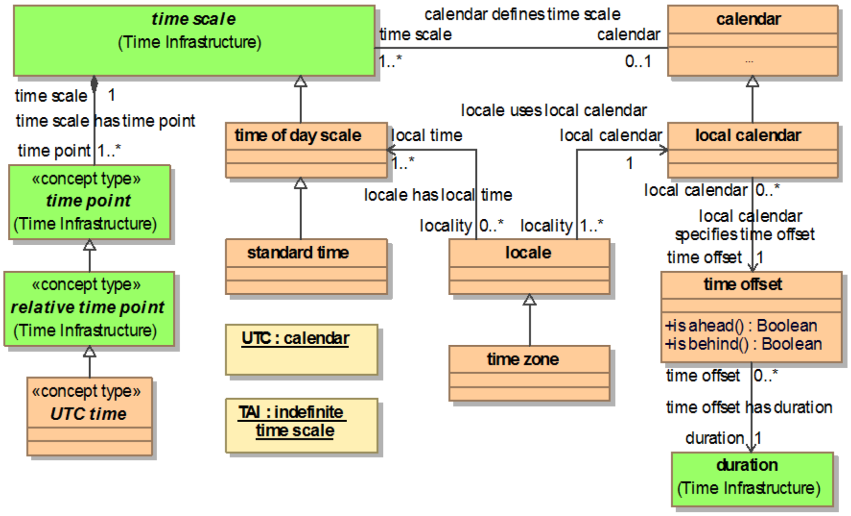


**Figure 13.5 - Calendars and time of day**

## time of day scale

| | |
|---|---|
| Definition: | time scale that *has* members that *are* times of day |
| Necessity: | Each time point *of* each time of day scale *is* a time of day. |

## UTC

| | |
|---|---|
| Synonym: | Coordinated Universal Time |
| Source: | International Bureau of Weights and Measures (BIPM) |
| Source: | ISO 8601 (2.1.12) |
| Source: | IEC 60050-713 |
| Dictionary Basis: | time scale which forms the basis for the coordinated dissemination of standard frequencies and time signals; it corresponds exactly in rate with International Atomic Time, but differs from it by an integral number of seconds |
| Definition: | calendar that combines the Gregorian Calendar with a day of seconds scale based on TAI, to identify time intervals by date and time of day |
| Necessity: | UTC *defines* the day-of-hours scale, the hour-of-minutes scale, and the minute-of-seconds scale |

| | |
|---|---|
| Note: | UTC is defined to be a calendar, because it defines the relationship of the Gregorian day time points to time intervals, as well as defining time of day scales. |
| Note: | All time zone calendars are correlated to UTC. |
| Note: | UTC is officially maintained by the BIPM in cooperation with national metrology institutes or observatories around the world. See http://www.bipm.org/en/scientific/tai/time_server.html. |
| Note: | The UTC day of seconds scale differs from TAI by the insertion of leap seconds (about every 18 months) to ensure approximate agreement with the time derived from the rotation of the Earth to within one second. The leap second adjustments make UTC a discontinuous time scale, because the Gregorian days in which the leap seconds occur have 86401 seconds. Thus, the UTC day of seconds scale *is* the current number of leap seconds *behind* TAI. Businesses that are sensitive to elapsed seconds of day may prefer to use TAI instead. |

## TAI

| | |
|---|---|
| Synonym: | Temps Atomique International |
| Synonym: | International Atomic Time |
| Definition: | indefinite time scale that is defined in a geocentric reference frame with the SI second as realized on the rotating geoid as the scale unit |
| Source: | SI |
| Note: | SI cites the " declaration of the CCDS, BIPM Com. Cons. DŽf. Seconde, 1980, 9, S 15 and Metrologia, 1981, 17, 70". |
| Necessity: | The granularity *of* the TAI Scale *is* second. |
| Necessity: | The index origin of TAI is midnight Gregorian day 2443145 (1 January 1977 00:00:00), Julian Date 2443144.5 |
| Note: | TAI is a continuous time scale of seconds, maintained by the Bureau international des poids et mesures (BIPM) as the average of over 200 hundred atomic clocks located in over 50 national laboratories. |
| Note: | Time coordinates for TAI are given as Julian date and time of day, where each Julian day is exactly 86 400 seconds. Businesses that are sensitive to the discontinuities of UTC should instead use TAI. |

## UTC time

| | |
|---|---|
| Source: | ISO 8601 (2.1.13) |
| Concept Type: | concept type |
| Definition: | time point within a calendar day in accordance with UTC |

## standard time

| | |
|---|---|
| Source: | ISO 8601 (2.1.14) |
| Source: | IEC 60050-111 |
| Definition: | time scale derived from Coordinated Universal Time, UTC, by a time offset established in a given location by the competent authority |

## locale

| | |
|---|---|
| Definition: | A place or region whose time of day is specified by a competent authority |

## local time

| | |
|---|---|
| Synonym: | local time of day |
| Concept Type: | role |
| Source: | ISO 8601 (2.1.16) |
| Dictionary Basis: | locally applicable time of day based on standard time, or a non-UTC based time of day |
| Definition: | time of day scale that is applicable to a given locale |

## locale *has* local time

Definition: the local time is the time of day scale that is applicable for the locale at a given time

## time offset

Definition: specification of the difference between a local calendar and UTC

Description: A time offset involves a direction – whether the local calendar is ahead of UTC or behind UTC – and the duration by which the local calendar is ahead of or behind UTC.

Example: Difference between a given indication (e.g., 12:00:00.000) on a clock set to local time and the same indication on a clock set to UTC time, where both of the clocks change at the same rate.

Note: Conventionally, a time offset is prefixed + to indicate that the local clock indication occurs before (is ahead of) the UTC indication, and – to indicate the local clock indication occurs after (is behind) the UTC indication. These are noun forms of '$calendar_1$ *is* duration *ahead of* $calendar_2$' (above). The number of a duration is always non-negative.

Example: Indian Standard Time – UTC = +5½ hours.

## time offset *has* duration

Definition: the local calendar that *has* the time offset *is* the duration *ahead of* UTC or *is* the duration *behind* UTC

Description: The duration is the amount of time between the local calendar time intervals and the corresponding UTC time intervals without regard to the direction.

## time offset *is ahead*

Definition: the local calendar that *has* the time offset *is* the duration *of* the time offset *ahead of* UTC

## time offset *is behind*

Definition: the local calendar that *has* the time offset *is* the duration *of* the time offset *behind* UTC

## local calendar

Definition: calendar that *is* exactly one duration *ahead of* UTC or that *is* exactly one duration *behind* UTC

Reference Scheme: the time offset *of* the local calendar

Example: Pacific Daylight Time (UTC–7 hours) , Eastern Standard Time (UTC–5 hours), British Summer Time (UTC+1 hour), Indian Standard Time (UTC+5½ hours)

Note: Many, but not all, local calendars are named. Calendar names are not unique, e.g., EST in the US and Australia. Many named local calendars may have the same time offset. For example, both Central European Standard Time and Algeria Standard Time are UTC+1 hour. A local calendar does not need to be named; it is identified by its time offset from UTC.

Note: ISO 8601 abbreviates time offsets by using only a signed four-digit number representing hours and minutes, omitting the "UTC" and "hours". Thus, IST is "+0530".

Note: Time references that are intended to be independent of changes to local calendars should be specified as UTC and a time offset.

Example: Most locations in the United States change between daylight time and summer time twice a year, and the specifications for when the changes happen have themselves changed on occasion. To specify noon in standard time in NY independent of local calendar, use '12:00-5:00'.

## local calendar *specifies* time offset

Synonymous Form: time offset *of* local calendar
Definition: the time offset is the difference between the local calendar and UTC
Necessity: Each local calendar *specifies* exactly one time offset.

## locale *uses* local calendar

Necessity: Each locale *uses* exactly one local calendar at any given time.

## time zone

Definition: locale in which one or two local calendars is used
Note: When there are two calendars for a time zone, one is standard time and the other is daylight savings time. The dates and time of day for changing between them is determined by local authorities for each time zone.
Note: The Time Zone Database [Zoneinfo] documents the history of local time for many locations. It is updated periodically to reflect changes made by political bodies to time zone boundaries, UTC offsets, and daylight-saving rules.

### 13.6.3 Time Coordinates with Time Offsets

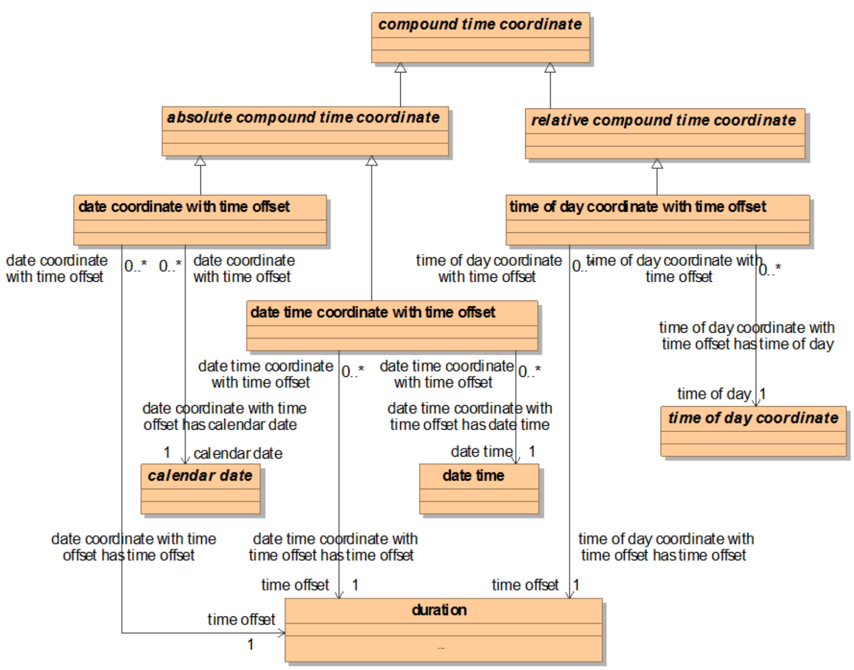


**Figure 13.6 – Time coordinates with a time offset**

**date coordinate with time offset**

| | |
|---|---|
| Definition: | time coordinate that combines a date coordinate and a time offset and that *indicates* the time point that *is indicated by* the date coordinate and that *is on* the calendar that *specifies* the time offset |
| Note: | Time offsets affect the meaning of dates because they change the relationship of midnight to time intervals. |
| Example: | "July 9 -5:00" means "July 9" on the calendar specified by time offset "*is behind* 5 hours", that is, UTC -5 hours. |
| Example: | "July 9 +11:00" is 22 hours before "July 9 – 11:00". |

**time of day coordinate with time offset**

| | |
|---|---|
| Definition: | time coordinate that combines a time of day coordinate and a time offset and that *indicates* the time point that *is indicated by* the time of day coordinate and that *is on* the calendar that *specifies* the time offset |
| Example: | "10:00 -5:00" means "10:00" on the calendar specified by time offset "*is behind* 5 hours", that is, UTC -5 hours. |
| Example: | "10:00 +11:00" is 22 hours before "10:00 – 11:00". |

**date time coordinate with time offset**

| | |
|---|---|
| Definition: | time coordinate that combines a date time coordinate and a time offset and that *indicates* the time point that *is indicated by* the date time coordinate and that *is on* the calendar that *specifies* the time offset |
| Example: | "July 9 10:00 -5:00" means "July 9 10:00" on the calendar specified by time offset "*is behind* 5 hours", that is, UTC -5 hours. |
| Example: | "July 9 10:00 +11:00" is 22 hours before "July 9 10:00 – 11:00". |

# 14 Internet Time (normative)

## 14.1 General

Internet Time is the calendar of the Network Time Protocol (NTP), published by the Internet Engineering Task Force (IETF); see http://www.rfc-editor.org/info/rfc5905. Virtually all computers and cell phones are synchronized with the NTP.

---

Internet Time Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Language: | English |
| Included Vocabulary: | Calendars Vocabulary |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#InternetTimeVocabulary |

---

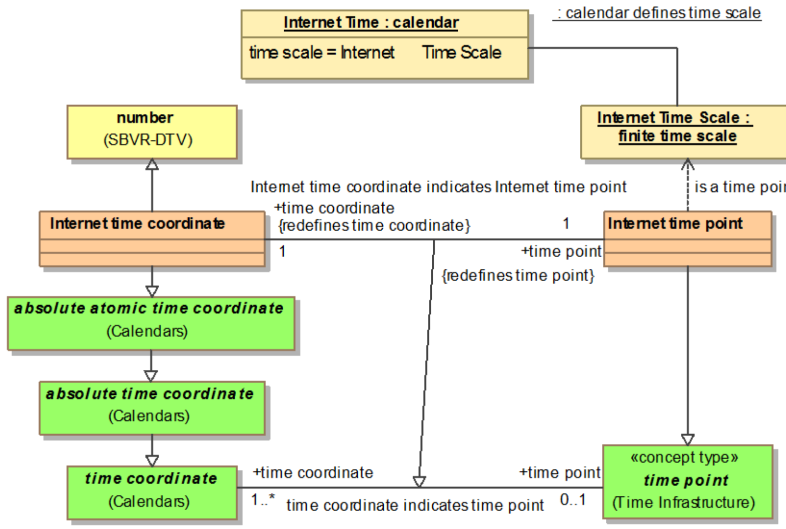


**Figure 14.1 - Internet Calendar**

## 14.2 Internet Calendar

### Internet Time

| | |
|---|---|
| Definition: | calendar that keeps UTC time and that *uses* the Internet Time Scale |
| Source: | [NTP] |
| Note: | Internet Time is based on UTC but is not necessarily always coincident with it (see [NTP] Appendix E.8 for a fuller explanation of reckoning the Internet Time Scale with UTC). Internet Time accounts for UTC's leap seconds, with a small uncertainty around the time of insertion of a leap second. |
| Necessity: | Accuracy of Internet Time relative to UTC is on the order of 1 millisecond. Stated precision is 200 picoseconds. |

### Internet Time Scale

| | |
|---|---|
| Definition: | finite time scale whose granularity *is* $2^{32}$ seconds and *whose* cardinality *is* $2^{64}$ and *whose* first $2^{32}$ time points *correspond to* January 1, 1900 00:00:00 UTC |
| Note: | The data format of NTP is defined in [NTP] section 3.1 and Appendix A. The Internet Time Scale will overflow the 64 bits after about 136 years, in 2036. The IETF is considering a revision of NTP (RFC 5905) that may likely extend its lifetime considerably. |

### Internet time coordinate

| | |
|---|---|
| Definition: | time coordinate that *is* a 64-bit unsigned fixed-point number having a 32 bit integer part and 32 bit fractional part and that *indicates* the Internet time point that *is* the number of seconds since January 1, 1900 00:00:00 UTC. |

### Internet time point

| | |
|---|---|
| Concept Type: | concept type |

Definition: time point that *is on* the Internet time scale and that *is* the number *of* seconds since January 1, 1900 00:00:00 UTC.

________________________________________________
________________________________________________

# 15 Indexical Time Concepts (normative)

## 15.1 General

"Indexical" is a linguistic concept that refers to terms that make implicit reference to the speaker or the context of the communication. It includes words like "now," "here," "we," etc. This clause defines indexical terms for time periods that are in common business use.

The use of indexical terms in business vocabularies and rules can be ambiguous, and the practice is generally deprecated, but these concepts are needed for some use cases.

________________________________________________

Indexical Time Vocabulary

General Concept: terminological dictionary
Language: English
Included Vocabulary: Time of Day Vocabulary
Included Vocabulary: ISO Week Calendar Vocabulary
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#IndexicalTimeVocabulary
________________________________________________

## 15.2 Indexical Characteristics

These unary fact types locate time intervals relative to the fundamental concept 'time interval *is past*'. An alternative design choice would be to specify a fundamental concept 'current time' as a kind of 'time interval', and then define 'time interval *is past*', 'time interval *is future*', etc., in terms of 'current time'. One of them must be defined; otherwise the definitions are circular. But every time interval has a duration, and defining 'current time' implies specifying its duration. The advantage of making 'time interval *is past*' fundamental is that we need not give a duration for current time.

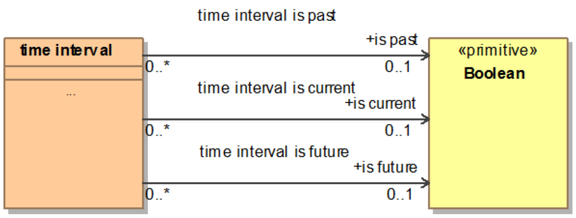


**Figure 15.1 - Indexical Characteristics**

## time interval *is past*

| | |
|---|---|
| Definition: | time interval that *is before* some reference time interval that is defined by context |
| Note: | The reference time interval is the time interval in which a rule is evaluated or applied. That is, any time interval that *is past* is always *before* the time interval at which the rule is used. |
| Example: | The time interval *identified by* "January 1, 1900" *is past* with respect to a reference time interval in 2012. |

## time interval *is current*

| | |
|---|---|
| Synonymous Form: | time interval *is present* |
| Synonymous Form: | time interval *is now* |
| Definition: | time interval that *includes* a time interval that *is past* and a time interval that *is* not *past* |
| Example: | If the contract deadline *is current* … |

## time interval *is future*

| | |
|---|---|
| Definition: | time interval that *includes* no time interval that *is in the past* |
| Necessity: | Each time interval that *is future*, *is after* each time interval that *is past*. |
| Example: | The supplier may respond to the RFP only if the due date of the RFP *is future*. |

These definitions of 'time interval *is past*', 'time interval *is current*', and 'time interval *is future*' are under-specified in the sense that many time intervals (of different durations) fit them. In particular, the verb concept 'time interval *is future*' includes the 'current time' reference time interval of the verb concept 'time interval *is past*'. Rules that compare time against 'current time' may be stated more precisely by referencing the indexicals given in sub clause 15.3, below. For example "if the contract due date *is* a future day ..." clearly tests the time interval given by the contract due date against a time interval that has a duration of 1 day and an alignment against the Gregorian calendar, whereas "if the contract due date *is future*" may be interpreted with any "comparison granularity," such as 'second' or 'hour'.

## 15.3 Indexical Time Intervals

Indexical time concepts are noun concepts that are indexical references to time. To minimize confusion, the indexical time intervals defined in this clause follow a consistent designation pattern. These time intervals are distinguished by whether they define the immediate previous or subsequent time point of a given kind, any past or future time point of a given kind, or a time period of a specific duration that ends or begins at a reference time.

Table 15.1 summarizes the designation patterns for the indexical time intervals. The patterns may be combined with the designations of any time units. In the table, the symbol '…' stands for the designation of a time unit, such as 'day', or 'second'.

**Table 15.1 - Naming Pattern for Indexical Time Intervals**

| time intervals relative to 'current time' | Description | Examples |
|---|---|---|
| current … | Time intervals of a specific time point kind that are current. | current time |
| last …<br>previous … | Time intervals of a specific time point kind that *meet* the reference time. | last day |
| next …<br>subsequent … | Time intervals of a specific time point kind that *are met by* the reference time. | next week |

| | | |
|---|---|---|
| past …<br>prior …<br>earlier … | Time intervals of a specific time point kind that are *before* the reference time. | past hour<br>earlier month |
| future …<br>later … | Time intervals of a specific time point kind that *are after* the reference time. | future month |
| preceding … | Time periods of a specified duration that *meet* the reference time. | preceding year |
| following …<br>upcoming … | Time periods of a specified duration that *are met by* the reference time. | following day |

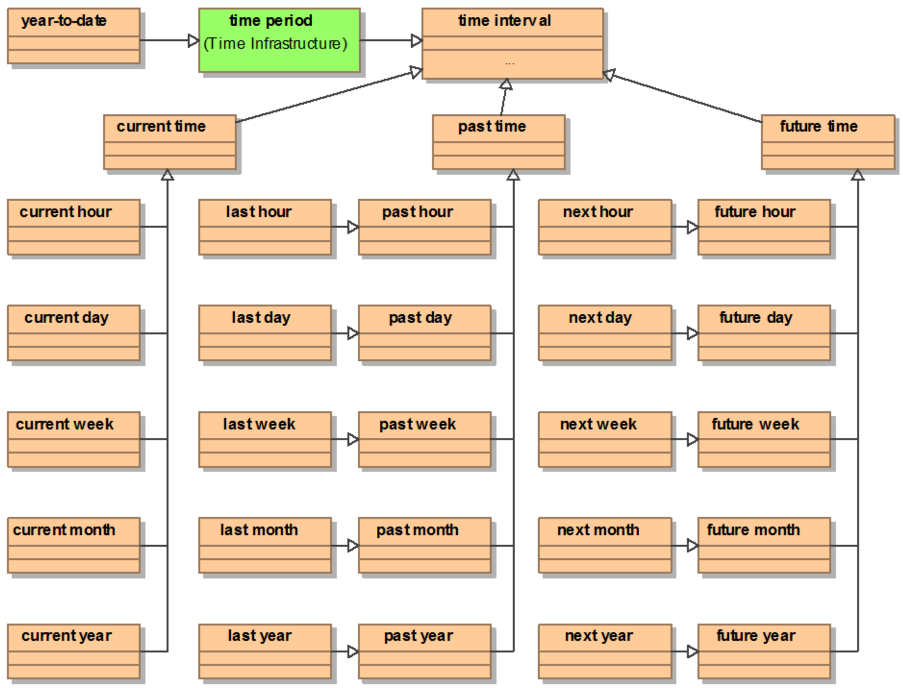


**Figure 15.2 - Indexical Time Intervals Relative to 'Current Time'**

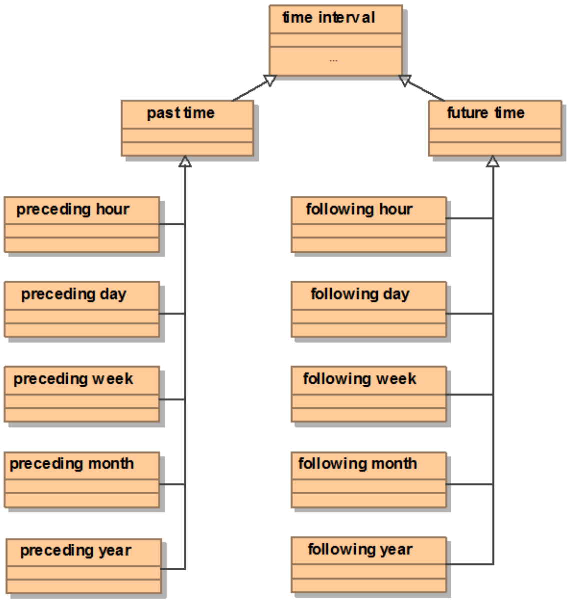


**Figure 15.3 - Indexical Time Periods Relative to 'Current Time'**

## current time

Synonym: present time
Definition: time interval that *is current*
Note: Every time interval that overlaps the "reference time interval" for 'time interval *is past*' is a current time (one of many).
Example: If the reference time interval is the current hour, then the calendar day, calendar week, calendar month, calendar year (etc.) that overlap the current hour are all current times.

## past time

Synonym: prior time
Synonym: earlier time
Definition: time interval that *is past*
Example: In any given calendar, if the reference time interval is denoted by "2012", then past time is any time interval that is before 2012.

## future time

Synonym: later time
Definition: time interval that *is future*
Example: If the reference time interval is the calendar day of the title closing of a real estate transaction, then future time is that calendar day and any later time interval.

## current hour

Concept Type: unitary concept
General Concept: current time

Definition: the time interval that *instantiates* an hour of day and that *is current*
Example: If the reference time interval is 10:32, then the current hour is a time interval denoted as hour of day 10.

## last hour

Synonym: previous hour
Concept Type: unitary concept
General Concept: past hour
Definition: the time interval that *instantiates* an hour of day and that *meets* the current hour
Example: If the reference time interval is 10:32, then the last hour is a time interval denoted as hour of day 9.

## next hour

Synonym: subsequent hour
Concept Type: unitary concept
General Concept: future hour
Definition: the time interval that *instantiates* an hour of day and that *is met by* the current hour
Example: If the reference time interval is 10:32, then the next hour is a time interval denoted as hour of day 11.

## past hour

Synonym: prior hour
Synonym: earlier hour
General Concept: past time
Definition: time interval that *instantiates* an hour of day and that *is before* the current hour
Definition: time interval that *instantiates* an hour of day that *is past*
Example: If the reference time interval is 10:32, then one past hour is a time interval denoted as hour of day 9. Another past hour is a time interval denoted as hour of day 8.

## future hour

Synonym: later hour
General Concept: future time
Definition: time interval that *instantiates* an hour of day and that *is after* the current hour
Definition: time interval that *instantiates* an hour of day that *is future*
Example: If the reference time interval is 10:32, then one future hour is a time interval denoted as hour of day 11. Another future hour is a time interval denoted as hour of day 12.

## preceding hour

Concept Type: unitary concept
General Concept: past time
Definition: the hour period that meets a time interval that instantiates a minute of hour and that *is current*
Example: If the reference time interval is 10:32, then the preceding hour is an hour period from 9:32 through 10:31.

## following hour

Synonym: upcoming hour
Concept Type: unitary concept
General Concept: future time

| | |
|---|---|
| Definition: | the hour period that is met by a time interval that instantiates a minute of hour and that *is current* |
| Example: | If the reference time interal is 10:32, then the following hour is an hour period from 10:33 through 11:32. |

## current day

| | |
|---|---|
| Concept Type: | unitary concept |
| General Concept: | current time |
| Definition: | the time interval that *instantiates* some calendar day and that *is current* |
| Example: | If the reference time interval is July 7 10:32, then the current day is a time interval denoted as July 7. |

## last day

| | |
|---|---|
| Synonym: | previous day |
| Concept Type: | unitary concept |
| General Concept: | past day |
| Definition: | the time interval that *instantiates* a calendar day and that *meets* the current day |
| Example: | If the reference time interval is July 7 10:32, then the last day is a time interval denoted as July 6. |

## next day

| | |
|---|---|
| Synonym: | subsequent day |
| Concept Type: | unitary concept |
| General Concept: | future day |
| Definition: | the time interval that *instantiates* a calendar day and that *is met by* the current day |
| Example: | If the reference time interval is July 7, then the next day is July 8. |

## past day

| | |
|---|---|
| Synonym: | prior day |
| Synonym: | earlier day |
| General Concept: | past time |
| Definition: | time interval that *instantiates* a calendar day and that *is before* the current day |
| Definition: | time interval that *instantiates* a calendar day that *is past* |
| Example: | If the reference time interval is July 7, then one past day is a time interval denoted by July 6 and another is a time interval denoted by July 5. |

## future day

| | |
|---|---|
| Synonym: | later day |
| General Concept: | future time |
| Definition: | time interval that *instantiates* a calendar day and that *is after* the current day |
| Definition: | time interval that *instantiates* a calendar day that *is future* |
| Example: | If the reference time interval is July 7, then one future day is a time interval that is denoted by July 8, and another future day is a time interval that is denoted by July 9. |

## preceding day

| | |
|---|---|
| Concept Type: | unitary concept |
| General Concept: | past time |
| Definition: | the day period that meets a time interval that instantiates a minute of hour and that *is current* |

Example: If the reference time interval is July 7 10:32, then the preceding day is a day period from July 6 10:32 through July 7 10:31.

## following day

Synonym: upcoming day
Concept Type: unitary concept
General Concept: future time
Definition: the day period that is met by a time interval that instantiates a minute of hour and that *is current*
Example: If the reference time interval is July 7 10:32, then the following day is a day period from July 7 10:33 through July 8 10:32.

## current week

Concept Type: unitary concept
General Concept: current time
Definition: the time interval that *instantiates* some calendar week and that *is current*
Example: If the reference time interval is week 15 day 3, then the current week is a time interval that instantiates week 15.

## last week

Synonym: previous week
Concept Type: unitary concept
General Concept: past week
Definition: the time interval that *instantiates* a calendar week and that *meets* the current week
Example: If the reference time interval is week 15 day 3, then the last week is a time interval that instantiates week 14.

## next week

Synonym: subsequent week
Concept Type: unitary concept
General Concept: future week
Definition: the time interval that *instantiates* a calendar week and that *is met by* the current week
Example: If the reference time interval is week 15 day 3, then the next week is a time interval that instantiates week 16.

## past week

Synonym: prior week
Synonym: earlier week
General Concept: past time
Definition: time interval that *instantiates* a calendar week and that *precedes* the current week
Definition: time interval that *instantiates* a calendar week that *is past*
Example: If the reference time interval is week 15 day 3, then one past week is a time interval that instantiates week 14, and another past week is a time interval that instantiates week 13.

## future week

Synonym: later week
General Concept: future time
Definition: time interval that *instantiates* a calendar week and that *is after* the current week
Definition: time interval that *instantiates* a calendar week that *is future*

Example: If the reference time interval is week15 day3, then one future week is a time interval that instantiates week16 and another future week is a time interval that instantiates week17.

## preceding week

Concept Type: unitary concept
General Concept: past time
Definition: the week period that meets a time interval that instantiates a minute of hour and that *is current*
Example: If the reference time interval is week15 day3, then the preceding week is a week period that is from week14 day3 through week15 day2.

## following week

Concept Type: unitary concept
General Concept: future time
Definition: the week period that is met by a time interval that instantiates a minute of hour and that *is current*
Example: If the reference time interval is week15 day3, then the following week is a week period that is from week15 day4 *through* week16 day3.

## current month

Concept Type: unitary concept
General Concept: current time
Definition: the time interval that *instantiates* some calendar month and that *is current*
Example: If the reference time interval is July 7, then the current month is a time interval that instantiates July.

## last month

Synonym: previous month
Concept Type: unitary concept
General Concept: past month
Definition: the time interval that *instantiates* a calendar month and that *meets* the current month
Example: If the reference time interval is July 7, then the last month is a time interval that instantiates June.

## next month

Synonym: subsequent month
General Concept: future month
Concept Type: unitary concept
Definition: the time interval that *instantiates* a calendar month and that *is met by* the current month
Example: If the reference time interval is July 7, then the next month is a time interval that instantiates August.

## past month

Synonym: prior month
Synonym: earlier month
General Concept: past time
Definition: time interval that *instantiates* a calendar month and that *precedes* the current month
Definition: time interval that *instantiates* a calendar month that *is past*

| | |
|---|---|
| Example: | If the reference time interval is July 7, then one past month is a time interval that instantiates June, and another past month is a time interval that instantiates May. |

## future month

| | |
|---|---|
| Synonym: | later month |
| General Concept: | future time |
| Definition: | time interval that *instantiates* a calendar month and that *is after* the current month |
| Definition: | time interval that *instantiates* a calendar month that *is future* |
| Example: | If the reference time interval is July 7, then one future month is a time interval that instantiates August, and another future month is a time interval that instantiates September. |

## preceding month

| | |
|---|---|
| Concept Type: | unitary concept |
| General Concept: | past time |
| Definition: | the month period that meets a time interval that instantiates a Gregorian day of year and that *is current* |
| Necessity: | The duration *of* the preceding month *is* the duration *of* the last month. |
| Note: | The previous Necessity addresses the varying duration of calendar months. |
| Example: | If the reference time interval is July 7, then preceding month is a month period from June 7 *through* July 6. |
| Example: | If the reference time interval is June 7, then preceding month is a month period from May 7 *through* June 6. |

## following month

| | |
|---|---|
| Concept Type: | unitary concept |
| General Concept: | future time |
| Definition: | the month period that is met by a time interval that instantiates a Gregorian day of year and that *is current* |
| Necessity: | The duration *of* the following month *is* the duration *of* the current month. |
| Note: | The previous Necessity addresses the varying duration of calendar months. |
| Example: | If the reference time interval is July 7, then following month is a month period from July 8 through August 7. |
| Example: | If the reference time interval is June 7, then following month is a month period from June 8 through July 7. |

## current year

| | |
|---|---|
| Concept Type: | unitary concept |
| General Concept: | current time |
| Definition: | the time interval that *instantiates* some calendar year and that *is current* |
| Example: | If the reference time interval is July 11, 2011, then the current year is the time interval that instantiates 2011. |

## last year

| | |
|---|---|
| Synonym: | previous year |
| Concept Type: | unitary concept |
| General Concept: | past year |
| Definition: | the time interval that *instantiates* a calendar year and that *meets* the current year |
| Example: | If the reference time interval is July 11, 2011, then the last year is the time interval that instantiates 2010. |

## next year

Synonym: subsequent year
Concept Type: unitary concept
General Concept: future year
Definition: the time interval that *instantiates* a calendar year and that *is met by* the current year
Example: If the reference time interval is July 11 2011, then the next year is the time interval that instantiates 2010.

## past year

Synonym: prior year
Synonym: earlier year
General Concept: past time
Definition: time interval that *instantiates* a calendar year and that *precedes* the current year
Definition: time interval that *instantiates* a calendar year that *is past*
Example: If the reference time interval is July 11 2011, then one past year is the time interval that instantiates 2010 and another past year is the time interval that instantiates 2009.

## future year

Synonym: later year
General Concept: future time
Definition: time interval that *instantiates* a calendar year and that *is after* the current year
Definition: time interval that *instantiates* a calendar year that *is future*
Example: If the reference time interval is July 7 2011, then one future year is the time interval denoted by 2012 and another future year is the time interval denoted by 2013.

## preceding year

Concept Type: unitary concept
General Concept: past time
Definition: the year period that meets a time interval that instantiates a day of year and that *is current*
Necessity: The duration *of* the preceding year *is* the duration *of* the last year.
Note: The previous Necessity addresses the varying duration of calendar years.
Example: If the reference time interval is July 11 2011, then the preceding year is the year period from July 11 2010 *through* July 10 2011.

## following year

Synonym: upcoming year
Concept Type: unitary concept
General Concept: future time
Definition: the year period that is met by a time interval that instantiates a day of year and that *is current*
Necessity: The duration *of* the following year *is* the duration *of* the current year.
Note: The previous Necessity addresses the varying duration of calendar years.
Example: If the reference time interval is July 7 2011, then the following year is the year period from July 8 2011 through August 7 2012.

## year to date

Definition: the time period that *starts on* calendar day 1 *of* the current year and that *ends on* the current day

Example: If the reference time interval is July 7, 2011, then year to date is July 1, 2011 through July 7, 2011.

______________________________________________
______________________________________________

# 16 Situations (normative)

## 16.1 General

______________________________________________

Situations Vocabulary

General Concept: terminological dictionary
Language: English
Included Vocabulary: Indexical Time Vocabulary
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#SituationsVocabulary

______________________________________________

This clause provides a vocabulary for relating situations to time intervals and durations; that is, it provides the basic vocabulary for writing rules or facts about the relationship between situations, events or activities and time. This treatment is motivated by the discussion in [Parsons] and [Menzel].

This specification relies on the idea of 'possible world' that is introduced in SBVR and derived from [Plantinga] – a specific collection of things and relationships that could be described by a set of consistent assertions (an SBVR 'fact model'), regardless of how that world relates to what we perceive as reality. Further, this specification uses the term 'universe of discourse' (or 'world of interest') to refer to the particular possible world that is chosen as the basis for determining what is 'true' or 'actual' with respect to a use of the ontology for reasoning and decision making. The conventional first-order logic treatment of time is: a different time is a different (possible) world. This specification treats time as an aspect of every possible world, so that any possible world can have a present, a past, and a future.

Consider the following rule that could exist in EU-Rent:

It is prohibited that a renter *has possession of* more than one rental car.

Rules are evaluated with respect to possible worlds, each of which has a particular current time. The prohibition is of a renter possessing more than one rental car in any possible world, that is, at any particular current time. Rationale clause 7.15 further discusses the meaning of rules with respect to time.

SBVR defines the concepts 'state of affairs' and 'state of affairs *is actual*' as the basis for determining the truth of propositions in terms of the facts of a universe of discourse. Sub clause 16.2 defines 'situation kind' and 'occurrence' as specializations of 'state of affairs' in order to distinguish potential situations from actual happenings, which have different relationships to time. Sub clauses 16.3 through 16.7 specify these temporal relationships. Sub clause 16.8 integrates the Date-Time Vocabulary concepts with 'state of affairs' and 'proposition'. Sub clause 16.9 introduces concepts that support tense and aspect as used in human languages.

## 16.2 Situation Kinds and Models

Figure 16.1 describes two principal concepts – situation kind, and occurrence – and the definitive relationships among them. 'Situation kind' and its specializations are types of events, activities and situations – the elements of process and activity models. They represent potential states of affairs that may be instantiated, perhaps many times, in the real business environment. These potential states of affairs may be planned for, budgeted for, dreamed of, feared, etc. 'Occurrences' are real happenings in the business environment. Each situation kind may have multiple occurrences. For example, a business may plan for the situation "power failure that shuts down production", which may have multiple occurrences. These concepts are parallel to the BPMN ideas of an activity/event model element (situation kind) and an activity/event instance (occurrence).

'Situation kind' is further specialized as 'general situation kind' (a situation kind that may have multiple occurrences) and 'individual situation kind' (a situation kind that has most one occurrence). Typically, individual situation kinds *refine* general situation kinds by adding distinguishing characteristics. For example, the "power failure that shut down production on Friday" refers to an individual situation kind that *refines* the general situation kind "power failure that shuts down production". Ordinary English usage blurs the distinction between an individual situation kind and its occurrence. The Date-Time Vocabulary supports that typical usage by providing verb concepts that access the time of the single occurrence of an individual situation kind.

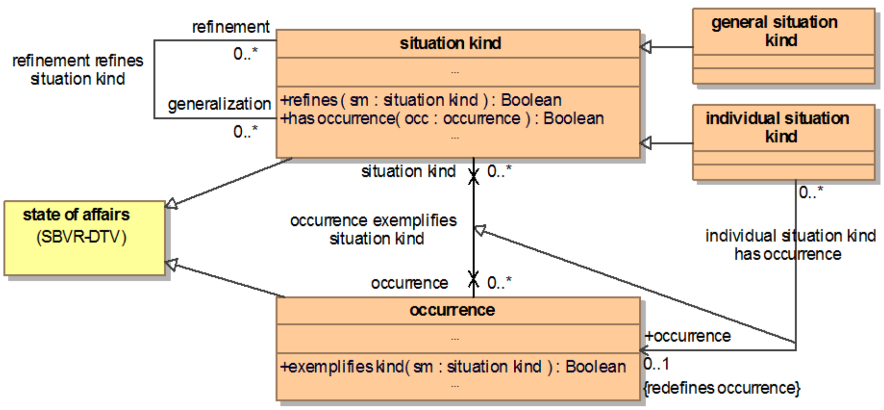


**Figure 16.1 - Situation Kinds and Occurrences**

### situation kind

| | |
|---|---|
| Synonym: | occurrence kind |
| Definition: | state of affairs that may or may not happen in some possible world |
| Note: | A situation kind may be seen as a type of situation, event or activity that may occur, perhaps more than once, or as a potential state of affairs that may be planned for, budgeted for, feared, dreamed about, etc. |
| Example: | Building codes often require special accommodations for building fires, understood as a situation kind. Some buildings may have one or more fires, others may never have a fire, but the requirements are not specific to individual fires. |
| Necessity: | Each situation kind *is* either a general situation kind or an individual situation kind. |

### occurrence

| | |
|---|---|
| Definition: | state of affairs that is a happening in the universe of discourse |
| Note: | An occurrence is an actual situation at some place and time in the possible world chosen for the universe of discourse. |

| | |
|---|---|
| Note: | This is a primitive concept. |
| Example: | An occurrence of 'fire' can burn you. |
| Example: | If a possible world includes all of December 2010, the physical flight of an aircraft from Washington to Minneapolis on December 20, 2010 from 7:00 to 9:00 EST is an occurrence in that world. In a possible world that is described by a fact model that includes flights, the flight of the aircraft exists. Any statement about all flights includes the particular flight. It *occurs within* December 2010 and *within* December 20, 2010, but it *occurs for* only the specified 2-hour time interval. It *occurs throughout* every time interval that is within that 2-hour time interval. |

## occurrence *exemplifies* situation kind

| | |
|---|---|
| Synonymous Form: | situation kind *has* occurrence |
| Definition: | the occurrence is a realization of the situation kind |
| Note: | This is a primitive concept. |
| Possibility: | Each occurrence *exemplifies* zero or more situation kinds. |
| Possibility: | Each situation kind *has* zero or more occurrences. |
| CLIF Axiom: | (forall (s occ)<br>(if ("situation kind has occurrence" s occ)<br>(and ("situation kind" s) (occurrence occ)) )) |
| Example: | The proposition "EU-Rent rents car 123 to customer abc" corresponds to a situation kind that may have an occurrence. |

## individual situation kind

| | |
|---|---|
| Definition: | situation kind that *has* at most one occurrence in each possible world |
| Necessity: | Each individual situation kind *has* at most one occurrence. |
| Example: | The situation kind that *is described by* the proposition "EU-Rent was incorporated on January 1, 2003" is an individual situation kind because it has just one occurrence. |
| Note: | The distinction between an individual situation kind and its occurrence is often blurred in ordinary English. |

## general situation kind

| | |
|---|---|
| Definition: | situation kind that *is* not an individual situation kind |
| Note: | This concept is defined in contrast to 'individual situation kind' not because there is any characteristic that distinguishes 'general situation kind' from 'situation kind'. |
| Note: | A situation kind is a general situation kind if it can *be exemplified by* more than one occurrence in some possible world, even when it cannot have more than one occurrence in the possible world chosen to be the universe of discourse. |
| Possibility: | Each general situation kind *has* more than one occurrence. |
| Example: | The situation kind that *is described by* "EU-Rent rents a car to a customer" is a general situation kind if and only if there are multiple occurrences described by this situation kind. |

## refinement

| | |
|---|---|
| Definition: | situation kind that *has* no occurrence that *does* not *exemplify* a given situation kind |
| Concept Type: | role |

## refinement *refines* situation kind

| | |
|---|---|
| Synonymous Form: | situation kind *has* refinement |
| Definition: | Each occurrence *of* the refinement *exemplifies* the situation kind |

Example: The individual situation kind *described by* "flight 123 from Washington to Minneapolis on December 20, 2010 arrives at 2pm" *refines* the general situation kind *described by* "flight from Washington to Minneapolis arrives at 2pm."

Note: The *refines* fact type defines a partial ordering relationship among situation kinds that is analogous to the specialization/subtype relationship among concepts.

### generalization

Definition: situation kind that *is exemplified by* each occurrence *of* a given situation kind

Concept Type: role

### situation kind *has* generalization

Definition: Each occurrence *of* the situation kind *exemplifies* the generalization

Note: This is the inverse relationship to situation kind *has* 'refinement'.

## 16.3 Occurrences and Time

An occurrence is an actual happening in the world of interest. This sub clause provides a vocabulary for relating occurrences to time intervals and durations.

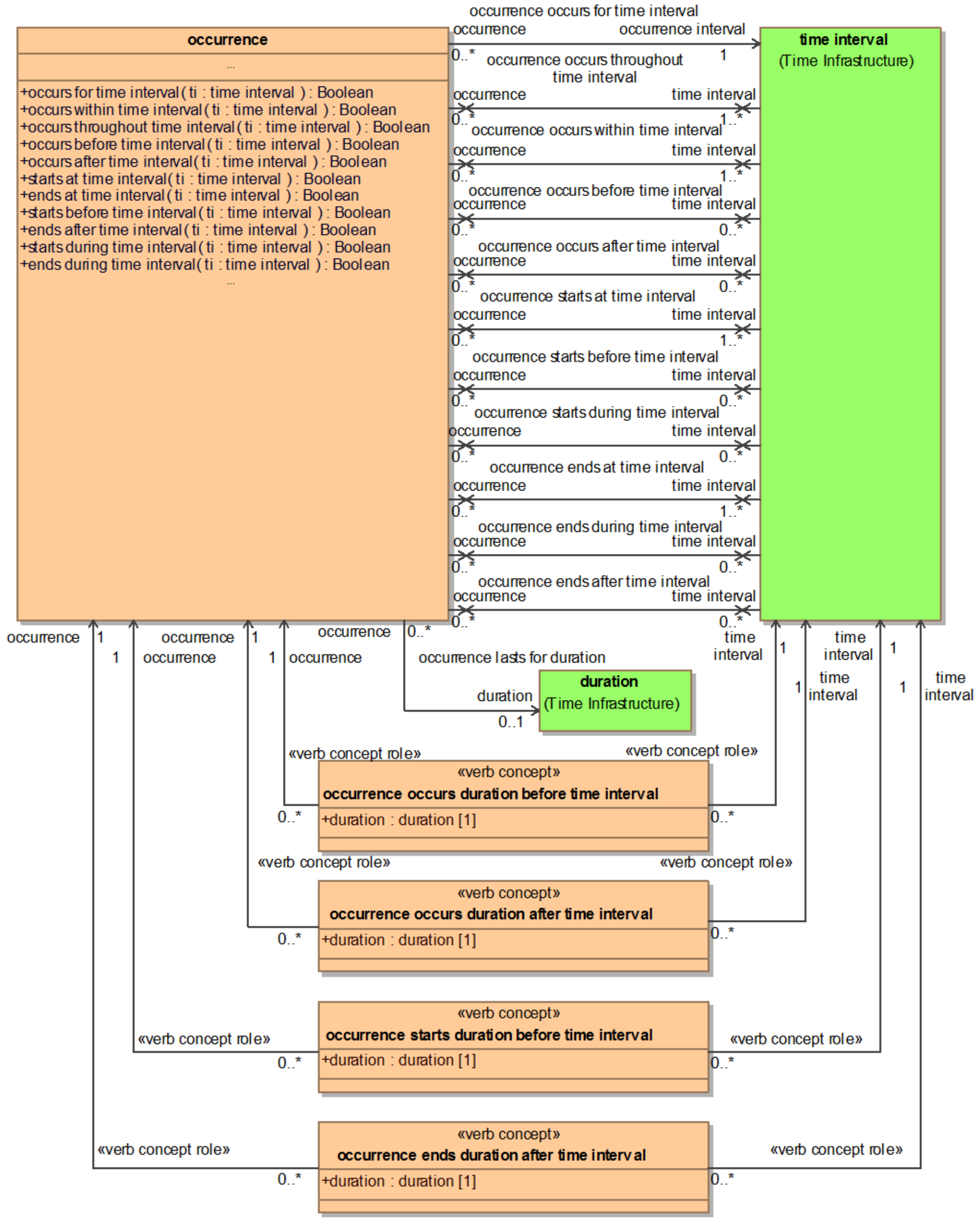


**Figure 16.2 - Occurrences and Time**

## occurrence *occurs throughout* time interval

Synonymous Form: occurrence *throughout* time interval

Definition: the occurrence happens continuously, without interruption, in each time interval$_2$ that *is part of* the time interval

Note: This is a "primitive concept" – the fundamental relationship between occurrences and time. It cannot be defined in terms of other concepts. The idea is that an occurrence occurs at all times in some sufficiently small time interval.

Possibility: The occurrence may *occur throughout* some time interval$_2$ that *is* not *part of* the time interval.

Note: That is, the occurrence could occur throughout a longer time interval that includes other time intervals.

CLIF Axiom:
```
(forall (occ ti)
  (if ("occurrence occurs throughout time interval" occ ti)
      (and (occurrence occ) ("time interval" ti)) ))
```

OCL Constraint:
```
context occurrence
inv: _'time interval'->allInstances(one t |
   self._'occurrence interval' = t)
```

Example: The occurrence of "Barack Obama is President of the U.S." *occurred throughout* March, 2009.

## occurrence *occurs within* time interval

Synonymous Form: occurrence *within* time interval

Synonymous Form: occurrence *in* time interval

Synonymous Form: occurrence *occurs at* time interval

Synonymous Form: occurrence *at* time interval

Synonymous Form: occurrence *during* time interval

Synonymous Form: time interval *covers* occurrence

Definition: the occurrence *occurs throughout* some time interval$_2$ that *is part of* the time interval

CLIF Definition:
```
(forall (occ t1)
  (iff ("occurrence occurs within time interval" occ t1)
    (and
      (occurrence occ) ("time interval" t1)
      (exists (t2)
        (and
          ("time interval1 is part of time interval2" t2 t1)
          ("occurrence occurs throughout time interval"
              occ t2))) )))
```

OCL Definition:
```
context _'occurrence'
def: _'occurrence occurs within time interval'
     (t: _'time interval') : Boolean
     t._'part of'->exists(t2 |
       self._'occurrence occurs throughout
        time interval'(t2))
```

Example: The occurrence "William the Conqueror defeats Harold Godwineson in battle" *occurs within* the time interval that *has* the time coordinate "14 October 1066".

## occurrence interval

Concept Type: role

General Concept: time interval

Definition: the time interval that a given occurrence *occurs for*, i.e., the time span from the start of the occurrence to the end of the occurrence

## occurrence *occurs for* occurrence interval

Synonymous Form: occurrence *occurs over* occurrence interval
Synonymous Form: occurrence *for* occurrence interval
Synonymous Form: occurrence *over* occurrence interval
Synonymous Form: occurrence *has* occurrence interval
Definition: the occurrence *occurs throughout* the occurrence interval and the occurrence *does* not *occur within* some time interval$_2$ that *meets* the occurrence interval and the occurrence *does* not *occur within* some time interval$_3$ that *is met by* the occurrence interval

CLIF Definition:

```
(forall (occ t1)
  (iff ("occurrence occurs for occurrence interval" occ t1)
    (and
      ("occurrence occurs throughout time interval" occ t1)
      (exists (t2 t3)
        (and
          ("time interval1 meets time interval2" t2 t1)
          (not ("occurrence occurs within time interval" occ t2))
          ("time interval1 meets time interval2" t1 t3)
          (not ("occurrence occurs within time interval" occ t3))
)) )))
```

OCL Definition:

```
context _'occurrence'
def: _'occurrence occurs for time interval'
    (t: _'time interval') : Boolean
    self._'occurrence occurs throughout time interval'
      (t)
    and self._'is met by'->forAll(t2 |
        not self._'occurrence occurs throughout
              time interval'(t2))
    and self._'meets"->forAll(t3 |
        not self._'occurrence occurs throughout
              time interval'(t3))
```

Note: The occurrence interval is the maximal time interval in which the individual occurrence occurs. The occurrence interval is immediately preceded and followed by time intervals when the occurrence does not happen.
Necessity: Each occurrence *occurs for* exactly one occurrence interval.
CLIF Axiom:

```
(forall (occ) (exists (t)
  (and
    ("occurrence occurs for occurrence interval" occ t)
    (forall (t2)
      (if (occurrence occurs for occurrence interval" occ t2)
          (= t2 t)))
)))
```

Possibility: Zero or more occurrences that *exemplify* a given general situation kind *occur for* a given occurrence interval.
Example: The occurrence that is a specific flight of a specific aircraft *occurs for* the occurrence interval from the airplane's takeoff to the airplane's landing.
Note: No occurrence "recurs." An occurrence is an individual event; a "recurrence" is a different event, being distinguished by *occurring for* different time interval. What "recurs" is the common situation kind.
Note: A former occurrence is an occurrence that *occurs over* some occurrence interval that *is in the past*. A planned occurrence is usually an occurrence that occurs over some future occurrence interval. A goal is a situation kind that may have an occurrence at some future time.
Note: The occurrence interval is an essential intrinsic property of an occurrence, but it may not be known or specified, and it may not be relevant to every business model. For some uses,

it may only be important that an occurrence happens *within* some time period, or that the situation kind *occurs throughout* some time period.

## occurrence *lasts for* duration

Synonymous Form: duration *of* occurrence

Definition: the occurrence *occurs for* some occurrence interval and the duration *is* the duration *of* the occurrence interval

CLIF Definition:

```
(forall (occ d)
 (iff ("occurrence lasts for duration" occ d)
  (and
   (occurrence occ) (duration d)
   (exists (t)
    (and
     ("occurrence occurs for time interval" occ t)
     ("time interval has duration" t d))))))
```

OCL Definition:

```
context _'occurrence'
def: _'occurrence lasts for duration'(d: duration): Boolean
   self._'occurrence occurs for time interval'.duration = d
```

Example: The duration of yesterday's meeting was 2 hours.

The following fact types are used primarily to enable us to talk about the beginning and end of occurrences in time.

## occurrence *occurs before* time interval

Synonymous Form: occurrence *ends before* time interval

Definition: the occurrence interval *of* the occurrence *is before* the time interval

CLIF Definition:

```
(forall (occ ti) (iff
("occurrence occurs before time interval" occ ti)
(and
 (occurrence occ)
 ("time interval" ti)
 ("time interval is before time interval"
   ("occurrence interval" occ) ti) )))
```

OCL Definition:

```
context occurrence
def: _'occurs before time interval'(t: time interval): Boolean =
  self._'occurrence interval'._'is before'(t)
```

## occurrence *occurs after* time interval

Synonymous Form: occurrence *starts after* time interval

Definition: the occurrence interval *of* the occurrence *is after* the time interval

CLIF Definition:

```
(forall (occ ti) (iff
("occurrence occurs after time interval" occ ti)
(and
 (occurrence occ)
 ("time interval" ti)
 ("time interval is before time interval"
   ti ("occurrence interval" occ)) )))
```

OCL Definition:

```
context occurrence
def: _'occurs after time interval'(t: time interval): Boolean =
  t._'is before'(self._'occurrence interval')
```

## occurrence *starts at* time interval

Definition: the time interval *starts* the occurrence interval *of* the occurrence or the occurrence interval *of* the occurrence *starts* the time interval or the occurrence interval *of* the occurrence *equals* the time interval

Note: '*Starts*' is the Allen relation (sub clause 8.2.3) between time intervals.

Note: The idea here is that the time intervals start together, but we know nothing about when they finish.

## occurrence *starts before* time interval

Definition: the occurrence interval *of* the occurrence *precedes* the time interval or the occurrence interval *of* the occurrence *properly overlaps* the time interval

Note: '*Properly overlaps*' is the Allen relation (sub clause 8.2.3) between time intervals.

## occurrence *ends at* time interval

Definition: the time interval *finishes* the occurrence interval *of* the occurrence or the occurrence interval *of* the occurrence *finishes* the time interval or the occurrence interval *of* the occurrence *equals* the time interval

Note: '*Finishes*' is the Allen relation (see 8.2.3) between time intervals.

Note: The idea here is that the time intervals finish together, but we know nothing about when they started. For example: "We should have a decision on the XYZ matter about the time that the contract review completes" means that the time interval at which the decision occurs will finish jointly with the contract review, irrespective of the times they started.

## occurrence *ends after* time interval

Definition: the occurrence interval *of* the occurrence *follows* the time interval or the occurrence interval *of* the occurrence *is properly overlapped by* the time interval.

Note: 'Is properly overlapped by' is the Allen relation (sub clause 8.2.3) between time intervals

## occurrence *occurs* duration *before* time interval

Synonymous Form: occurrence *ends* duration *before* time interval

Synonymous Form: time interval *is* duration *after* occurrence

Synonymous Form: time interval *starts* duration *after* occurrence

Definition: the occurrence interval *of* the occurrence *is* duration *before* the time interval

Description: The end of the occurrence is duration before the time interval.

## occurrence *occurs* duration *after* time interval

Synonymous Form: occurrence *starts* duration *after* time interval

Synonymous Form: time interval *is* duration *before* occurrence

Synonymous Form: time interval *ends* duration *before* occurrence

Definition: the occurrence interval *of* the occurrence *is* duration *after* the time interval

Description: The start of the occurrence is duration after the time interval.

## time interval *starts* duration *before* occurrence

Definition: time interval *starts* the duration *before* the occurrence interval *of* the occurrence

Description: The start of the time interval is duration before the occurrence.

Note: This says nothing about the relationship between the occurrence and the end of the time interval

### time interval *ends* duration *after* occurrence

Definition: time interval *ends* the duration *after* the occurrence interval *of* the occurrence
Description: The end of the time interval is duration after the occurrence.
Note: This says nothing about the relationship between the occurrence and the start of the time interval

### occurrence *starts during* time interval

Synonymous Form: occurrence *starts within* time interval
Definition: the occurrence interval *of* the occurrence *starts during* the time interval
Description: The occurrence begins sometime within the time interval.
CLIF Definition:
```
(forall (occ ti)
  (iff ("occurrence starts during time interval" occ ti)
    (exists (ti2)
      (and
        ("occurrence occurs for occurrence interval" occ ti2)
        ("time interval1 starts during time interval2" ti2 ti) ))))
```
OCL Definition:
```
context occurrence
def: _'starts during'(t2: _'time interval'): Boolean =
  self._'occurrence interval'._'starts during'(t2)
```
Example: The report must include all contracts undertaken during the reporting period.

### occurrence *ends during* time interval

Synonymous Form: occurrence *ends within* time interval
Definition: the occurrence interval *of* the occurrence *ends during* the time interval
Description: The occurrence ends sometime within the time interval.
CLIF Definition:
```
(forall (occ ti)
  (iff ("occurrence ends during time interval" occ ti)
    (exists (ti2)
      (and
        ("occurrence occurs for occurrence interval" occ ti2)
        ("time interval1 ends during time interval2" ti2 ti) ))))
```
OCL Definition:
```
context occurrence
def: _'ends during'(t2: _'time interval'): Boolean =
  self._'occurrence interval'._'ends during'(t2)
```
Example: The building will be completed within 2015.

## 16.4 Temporal Ordering of Occurrences

Business processes and many rules constrain the time order of activities and events without specifying the actual times. And in general, these rules refer to activities and events as situation kinds. But only individual occurrences can occur in temporal order. So, in fact, only occurrences are ordered. The following verb concepts facilitate careful specification of such usages.

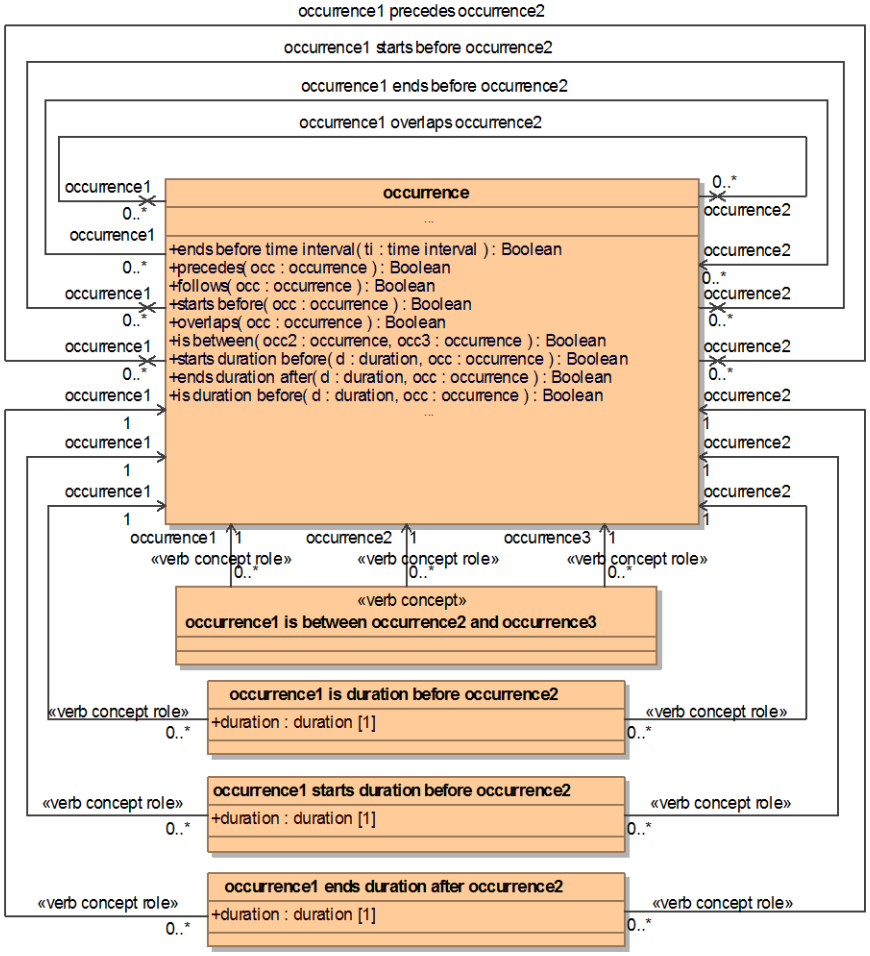


**Figure 16.3 - Temporal Ordering of Occurrences**

## occurrence$_1$ *precedes* occurrence$_2$

Synonymous Form: occurrence$_2$ *follows* occurrence$_1$

Definition: the occurrence interval *of* occurrence$_1$ *precedes* the occurrence interval *of* occurrence$_2$

CLIF Definition:
```
(forall (o1 o2)
  (iff ("occurrence1 precedes occurrence2" o1 o2)
    (and
      (occurrence o1) (occurrence o2)
      (forall (t1 t2)
        (if
          (and
            ("occurrence occurs for time interval" o1 t1)
            ("occurrence occurs for time interval" o2 t2))
          ("time interval1 precedes time interval2" t1 t2)))
)))
```
OCL Definition:
```
context _'occurrence'
def: _'occurrence1 precedes occurrence2'
    (o2: _'occurrence') : Boolean
    self._'occurs for' < o2._'occurs for'
```
Necessity: If some occurrence$_1$ *precedes* some occurrence$_2$, and if the occurrence$_2$ *precedes* some occurrence$_3$, then occurrence$_1$ *precedes* occurrence$_3$.

CLIF Axiom:
```
(forall (o1 o2 o3)
  (if (and
        ("occurrence1 precedes occurrence2" o1 o2)
        ("occurrence1 precedes occurrence2" o2 o3))
      ("occurrence1 precedes occurrence2" o1 o3)))
```
OCL Constraint:
```
context _'occurrence'
    inv: self._'precedes'->exists(o2 |
           o2._'precedes'->exists(o3 | implies
            self._'precedes'->contains(o3)))
```
Note: This verb concept permits comparing the time order of two occurrences.

Example: On each airplane flight, the airplane takes off before the airplane lands.

## occurrence$_1$ *starts before* occurrence$_2$

Synonymous Form: occurrence$_2$ *starts after* occurrence$_1$

Definition: the occurrence interval *of* occurrence$_1$ *starts before* the occurrence interval *of* occurrence$_2$

CLIF Definition:
```
(forall (o1 o2)
  (iff ("occurrence1 starts before occurrence2" o1 o2)
    (and
      (occurrence o1) (occurrence o2)
      (forall (t1 t2)
        (if
          (and
            ("occurrence occurs for time interval" o1 t1)
            ("occurrence occurs for time interval" o2 t2))
          ("time interval1 starts before time interval2" t1 t2)))
)))
```
OCL Definition:
```
context _'occurrence'
def: _'occurrence1 starts before occurrence2'
    (o2: _'occurrence') : Boolean
  self._'occurs for'._'time interval starts before time interval'(o2._'occurs for')
```
Note: This verb concept permits comparing the starting times of two occurrences.

Example: The procession must not start before the band plays.

## occurrence$_1$ *ends before* occurrence$_2$

Synonymous Form: occurrence$_2$ *ends after* occurrence$_1$

Definition: the occurrence interval *of* occurrence$_1$ *ends before* the occurrence interval *of* occurrence$_2$

CLIF Definition:
```
(forall (o1 o2)
 (iff ("occurrence1 ends before occurrence2" o1 o2)
   (and
     (occurrence o1) (occurrence o2)
     (forall (t1 t2)
       (if
         (and
           ("occurrence occurs for time interval" o1 t1)
           ("occurrence occurs for time interval" o2 t2))
         ("time interval1 ends before time interval2" t1 t2)))
)))
```

OCL Definition:
```
context _'occurrence'
def: _'occurrence1 ends before occurrence2'
    (o2: _'occurrence') : Boolean
 self._'occurs for'._'time interval ends before time interval'(o2._'occurs for')
```

Note: This verb concept permits comparing the ending times of two occurrences without regard to their start times.

Example: The delivery must be completed before the contract expires.

## occurrence$_1$ *overlaps* occurrence$_2$

Synonymous Form: occurrence$_1$ *while* occurrence$_2$

Synonymous Form: occurrence$_1$ *occurs while* occurrence$_2$

Definition: the occurrence interval *of* occurrence$_1$ *overlaps* the occurrence interval *of* occurrence$_2$

CLIF Definition:
```
(forall (o1 o2)
 (if ("o1 overlaps o2")
   (and
     (occurrence o1)
     (occurrence o2)
     (forall ((t1 "time interval") (t2 "time interval"))
       (if (and
             ("occurrence occurs for time interval" o1 t1)
             ("occurrence occurs for time interval" o2 t2))
           ("time interval1 overlaps time interval2" t1 t2)) ))))
```

OCL Definition:
```
context _'occurrence'
def: _'occurrence1 overlaps occurrence2'
     (o2: _'occurrence') : Boolean
     self._'occurs for'._overlaps(o2._'occurs for')
```

## occurrence$_1$ *is between* occurrence$_2$ *and* occurrence$_3$

Synonymous Form: occurrence$_1$ *between* occurrence$_2$ *and* occurrence$_3$

Synonymous Form: occurrence$_1$ *occurs between* occurrence$_2$ *and* occurrence$_3$

Synonymous Form: occurrence$_1$ *between* occurrence$_2$ *to* occurrence$_3$

Definition: occurrence$_1$ *follows* occurrence$_2$ and occurrence$_1$ *precedes* occurrence$_3$

CLIF Definition:
```
(forall (o1 o2 o3)
 (iff ("occurrence1 is between occurrence2 and occurrence3"
         o1 o2 o3)
   (and
     ("occurrence precedes occurrence" o2 o1)
     ("occurrence precedes occurrence" o1 o3)) ))
```

Example: The ship "Mauretania" crossed the equator *between* the ship leaving Hawaii *and* the ship arriving in Sydney.

### occurrence$_1$ *is* duration *after* occurrence$_2$

Synonymous Form: occurrence$_1$ *starts* duration *after* occurrence$_2$
Synonymous Form: occurrence$_2$ *is* duration *before* occurrence$_1$
Synonymous Form: occurrence$_2$ *ends* duration *before* occurrence$_1$
Definition: the occurrence interval *of* occurrence$_1$ *is* duration *after* the occurrence interval *of* occurrence$_2$
Description: The time between the two occurrences is the given duration.

### occurrence$_1$ *starts* duration *before* occurrence$_2$

Definition: the occurrence interval *of* occurrence$_1$ *starts* duration *before* the occurrence interval *of* occurrence$_2$
Description: One occurrence starts duration before the other occurrence starts.
Note: This says nothing about the relationship between occurrence$_2$ and the end of occurrence$_1$

### occurrence$_1$ *ends* duration *after* occurrence$_2$

Definition: the occurrence interval *of* occurrence$_1$ *ends* duration *after* the occurrence interval *of* occurrence$_2$
Description: One occurrence ends duration after the other occurrence ends.
Note: This says nothing about the relationship between occurrence$_2$ and the start of occurrence$_1$

## 16.5 Situation Kinds and Time

This sub clause provides the basic vocabulary for writing rules or facts about the relationship between situation Kinds and time.

Business processes and many rules constrain the timing of activities and events. In general, these rules refer to activities and events using situation kinds. A process specification assumes that what is being described is the sequencing of occurrences in an individual instance of the process. That is, the individual occurrences are described by the nature of the happening (the situation kind) and whatever information identifies the process instance. The fundamental notion here is that a situation kind 'occurs' at any time it is exemplified by an actual occurrence in the world of interest, as discussed in 16.3.

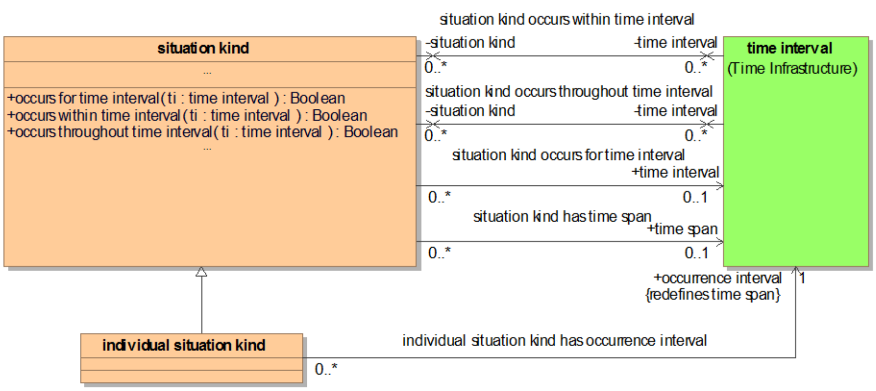


Figure 16.4 - Situation Kinds and Time

## situation kind *occurs throughout* time interval

Synonymous Form: situation kind *throughout* time interval
Definition: some occurrence *of* the situation kind *occurs throughout* the time interval
Possibility: A situation kind may *occur throughout* no time interval.

## situation kind *occurs within* time interval

Synonymous Form: situation kind *within* time interval
Synonymous Form: situation kind *in* time interval
Synonymous Form: situation kind *at* time interval
Synonymous Form: situation kind *during* time interval
Definition: some occurrence *of* the situation kind *occurs within* the time interval
Example: The situation kind "soldiers are engaged in battle" *occurred within* the time interval that *has* the time coordinate "14 October 1066".
Example: " Flight 70 landed in Minneapolis at 9:12 on May 13, 2011."

## situation kind *occurs for* time interval

Definition: some occurrence *of* the situation kind *occurs for* the time interval
Necessity: Each individual situation kind *occurs for* at most one time interval.
Possibility: A general situation kind *occurs for* more than one time interval.
Note: For an individual situation kind, the time interval is unique. For a general situation kind, the model and the time interval may uniquely identify an occurrence.

## time span

General Concept: time interval
Concept Type: role

## situation kind *has* time span

Definition: the occurrence interval *of* each occurrence *of* situation kind *is part of* time span and no time interval that *is part of* time span *is before* the occurrence interval *of* each occurrence *of* situation kind and no time interval that *is part of* time span *is after* the occurrence interval *of* each occurrence *of* situation kind
Description: The time span is the smallest time interval that contains the occurrence intervals of all the occurrences in a given situation kind.
Note: A general situation kind may specify a constraint on the time interval of all of its occurrences, by stating the time span for the general situation kind, or stating a constraint on it. Individual situation kinds that refine the general situation kind each resolve the time down to a particular occurrence interval that must be within the time span.
Example: " The meetings will be weekly for the next three months" describes a general situation kind whose time span is the specified time interval of the next three months. There can be a schedule of these meetings, giving the particular time for each meeting, which is an individual situation kind.
Example: The time span of all the discount offers (a general situation kind) is within July 2011. A particular discount (an individual situation kind) offer occurs for July 13 from 2-3pm.
Example: The proposition "the meetings are scheduled for each Monday of July 2011" *describes* a general situation kind whose time span *is within* the time interval "July 2011." If the individual meetings are held, then they *occur within* the Mondays of July 2011.

## individual situation kind *has* occurrence interval

Definition: the occurrence interval *is* the time span *of* the individual situation kind

Necessity: Each individual situation kind *has* at most one occurrence interval.

Note: The time span of an individual situation kind is exactly the occurrence interval of its only occurrence.

Example: The occurrence interval of the Great Fire of London was 2 September 1666 through 5 September 1666 (English old style calendar).

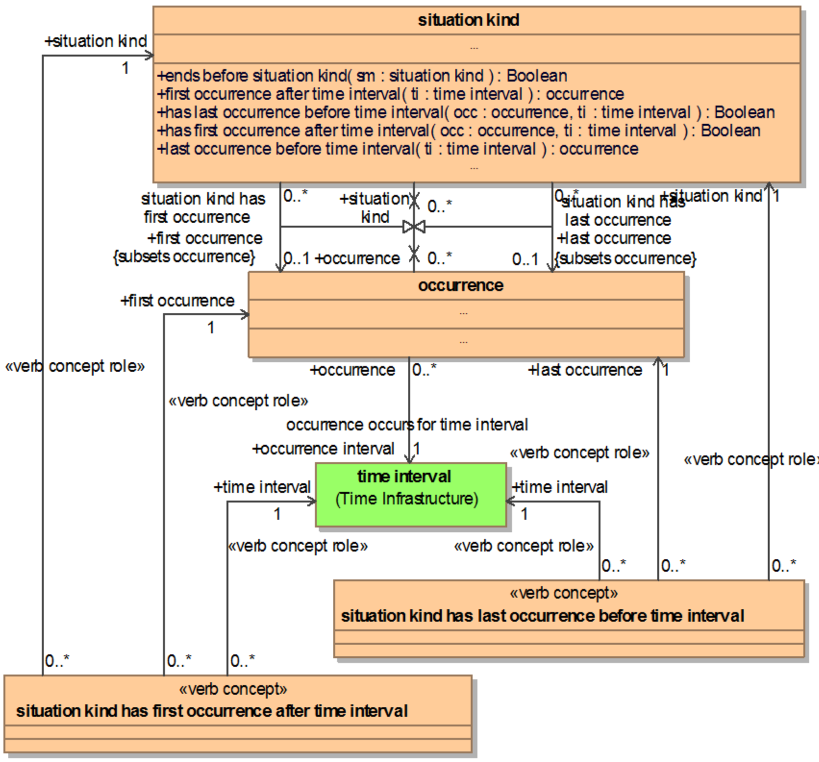


**Figure 16.5 - First and last occurrences of situation kinds**

## first occurrence

General Concept: occurrence

Concept Type: role

## situation kind *has* first occurrence *after* time interval

Synonymous Form: first occurrence *of* situation kind *after* time interval

Definition: the first occurrence *exemplifies* the situation kind and the first occurrence *occurs after* the time interval and no occurrence that *exemplifies* the situation kind and that *occurs after* the time interval *starts before* the first occurrence

CLIF Definition:
```
(forall (sk fo ti) (iff
  ('situation kind has first occurrence after time interval' sk fo ti)
  (and
    ('occurrence exemplifies situation kind' fo sk)
    ('occurrence occurs after time interval' fo ti)
    (not (exists (occ) (and
        ('occurrence exemplifies situation kind' occ sk)
        ('occurrence occurs after time interval' occ ti)
        ('occurrence1 starts before occurrence2' occ fo)
    ))) )))
```
OCL Definition:
```
context _'situation kind'
def: _'has first occurrence after time interval'(ti: _'time interval'): occurrence =
     occurrence->allInstances(fo |
     fo.exemplifies(sk) and fo._'occurs after'(ti) and
     not occurrence->allInstances(exists occ |
         occ.exemplifies(sk) and occ._'occurs after'(ti)
         and occ._'starts before'(fo)))
```

## situation kind *has* first occurrence

Definition: the first occurrence *exemplifies* the situation kind and no occurrence that *exemplifies* the situation kind *starts before* the first occurrence

CLIF Definition:
```
(forall (sk fo) (iff
  ('situation kind has first occurrence' sk fo)
  (and
    ('occurrence exemplifies situation kind' fo sk)
    (not (exists (occ) (and
        ('occurrence exemplifies situation kind' occ sk)
        ('occurrence1 starts before occurrence2' occ fo)
    ))) )))
```
OCL Definition:
```
context _'situation kind'
def: self._'first occurrence': occurrence) =
    occurrence->allInstances(fo |
     fo.exemplifies(sk) and
     not occurrence->allInstances(exists occ | occ._'starts before'(fo)))
```
Example: The first occurrence of the situation kind ‘landing of a human on the moon’ had the occurrence interval 20 July 1969 through 21 July 1969.

## last occurrence

General Concept: occurrence
Concept Type: role

## situation kind *has* last occurrence

Definition: the last occurrence *exemplifies* the situation kind and no occurrence that *exemplifies* the situation kind *ends after* the last occurrence

CLIF Definition:
```
(forall (sk fo) (iff
  ('situation kind has last occurrence' sk lo)
  (and
    ('occurrence exemplifies situation kind' lo sk)
    (not (exists (occ) (and
        ('occurrence exemplifies situation kind' occ sk)
        ('occurrence1 ends before occurrence2' lo occ)
    ))) )))
```

OCL Definition:
```
context _'situation kind'
def: self._'last occurrence': occurrence) =
   occurrence->allInstances(lo |
    lo.exemplifies(sk) and
    not occurrence->allInstances(exists occ | lo._'ends before'(occ)))
```

### situation kind *has* last occurrence *before* time interval

Synonymous Form: last occurrence *of* situation kind *before* time interval

Definition: the last occurrence *exemplifies* the situation kind and the last occurrence *occurs before* the time interval and no occurrence that *exemplifies* the situation kind and that *occurs before* the time interval *ends after* the last occurrence

CLIF Definition:
```
(forall (sk lo ti) (iff
  ('situation kind has last occurrence before time interval' sk lo ti)
  (and
    ('occurrence exemplifies situation kind' lo sk)
    ('occurrence occurs before time interval' lo ti)
    (not (exists (occ) (and
        ('occurrence exemplifies situation kind' occ sk)
        ('occurrence occurs before time interval' occ ti)
        ('occurrence1 ends before occurrence2' lo occ)
    ))) )))
```

OCL Definition:
```
context _'situation kind'
def: _'has last occurrence before time interval'(ti: _'time interval'): occurrence =
    occurrence->allInstances(lo |
    lo.exemplifies(sk) and lo._'occurs before'(ti) and
    not occurrence->allInstances(exists occ |
        occ.exemplifies(sk) and occ._'occurs before'(ti)
        and lo._'ends before'(occ)))
```

Example: The last occurrence of the situation kind 'landing of a human on the moon' before December 2012 occurred over the time interval 21 April 1972 *through* 24 April 1972.

## 16.6 Temporal Ordering of Situation Kinds

Business processes and many rules constrain the time order of activities and events without specifying the actual times. And in general, these rules refer to activities and events as situation kinds. Only individual occurrences actually have temporal ordering, but assigning such an ordering to the situation kinds themselves constrains the ordering of the actual occurrences. The following verb concepts facilitate careful specification of such usages.

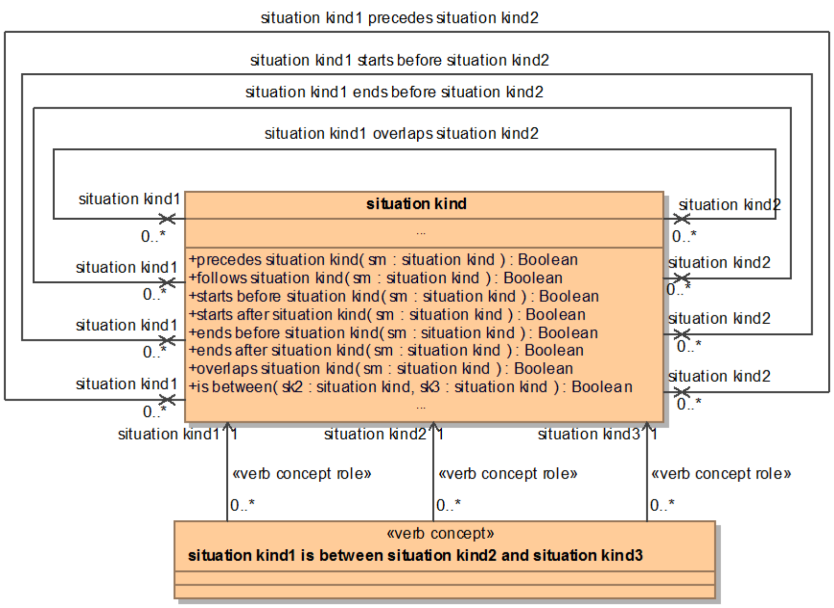


**Figure 16.6 - Temporal Ordering of Situation Kinds**

## situation kind$_1$ *precedes* situation kind$_2$

Synonymous Form: situation kind *follows* situation kind$_1$
Definition: each occurrence *of* situation kind$_1$ *precedes* each occurrence *of* situation kind$_2$
CLIF Definition:
```
(forall (s1 s2)
  (iff ("situation kind1 precedes situation kind2" s1 s2)
     (forall (o1 o2)
       (if
         (and
            ("situation kind has occurrence" s1 o1)
            ("situation kind has occurrence" s2 o2))
         ("occurrence1 precedes occurrence2" o1 o2)) )))
```
OCL Definition:
```
context _'situation kind'
def: _'situation kind1 precedes situation kind2'
   (s2: _'situation kind') : Boolean =
   self._'occurrence'.precedes(s2._'occurrence')
```
Note: This verb concept permits comparing the time order of two situation kinds. This is most useful in comparing individual situation kinds, but it has broader use.

Example: On each airplane flight, the airplane takes off before the airplane lands. (This compares two individual situation kinds.)

Example: The bank failures of the Great Depression (a general situation kind) preceded World War II (an individual situation kind).

## situation kind$_1$ *starts before* situation kind$_2$

Synonymous Form: situation kind$_2$ *starts after* situation kind$_1$
Definition: each occurrence *of* situation kind$_1$ *starts before* each occurrence *of* situation kind

CLIF Definition:
```
(forall (s1 s2)
  (iff ("situation kind1 starts before situation kind2" s1 s2)
    (and
      ("situation kind" s1) ("situation kind" s2)
      (forall (o1 o2)
        (if
         (and
           ("situation kind has occurrence" s1 o1)
           ("situation kind has occurrence" s2 o2))
         ("occurrence1 starts before occurrence2" o1 o2)))
)))
```
OCL Definition:
```
context _'situation kind'
def: _'situation kind1 starts before situation kind2'
    (s2: _'situation kind') : Boolean
    self.occurrence._'starts before'(s2.occurrence)
```
Note: This verb concept permits comparing the starting times of two situation kinds. This is primarily used for individual situation kinds.

Example: The procession must not start before the band plays.

## situation kind₁ *ends before* situation kind₂

Synonymous Form: situation kind₂ *ends after* situation kind₁

Definition: each occurrence *of* situation kind₁ *ends before* each occurrence *of* situation kind₂

CLIF Definition:
```
(forall (s1 s2)
  (iff ("situation kind1 ends before situation kind2" s1 s2)
    (and
      ("situation kind" s1) ("situation kind" s2)
      (forall (o1 o2)
        (if
         (and
           ("situation kind has occurrence" s1 o1)
           ("situation kind has occurrence" s2 o2))
         ("occurrence1 ends before occurrence2" o1 o2)))
)))
```
OCL Definition:
```
context _'situation kind'
def: _'situation kind1 ends before situation kind2'
    (s2: _'situation kind') : Boolean
    self.occurrence._'ends before'(s2.occurrence)
```
Note: This verb concept permits comparing the ending times of two situation kinds without regard to their start times. This is primarily used for individual situation kinds.

Example: The delivery must be completed before the contract expires.

## situation kind₁ *overlaps* situation kind₂

Synonymous Form: situation kind₁ *while* situation kind₂

Synonymous Form: situation kind₁ *occurs while* situation kind₂

Definition: each occurrence *of* situation kind₁ *overlaps* some occurrence *of* situation kind₂

CLIF Definition:

```
(forall (s1 s2)
 (iff ("situation kind1 overlaps situation kind2" s1 s2)
  (and
    ("situation kind" s1) ("situation kind" s2)
    (forall (o1 o2)
      (and
        (occurrence o1) (occurrence o2)
        (if (and
              ("situation kind has occurrence" s1 o1)
              ("situation kind has occurrence" s2 o2))
          ("occurrence1 overlaps occurrence2" o1 o2) )) ))))
```

OCL Definition:

```
context _'situation kind'
def: _'situation kind1 overlaps situation kind2'
    (s2: _'situation kind') : Boolean
    self._'occurrence'.overlaps (s2._'occurrence')
```

### situation kind[1] *is between* situation kind[2] *and* situation kind[3]

Synonymous Form: situation kind[1] *between* situation kind[2] and situation kind[3]
Synonymous Form: situation kind[1] *is between* situation kind[2] *to* situation kind[3]
Synonymous Form: situation kind[1] *between* situation kind[2] *to* situation kind[3]
Definition: situation kind[1] *follows* situation kind[2] and situation kind[1] *precedes* situation kind[3]
Note: This verb concept permits comparing the time order of three situation kinds. This is most
Note: useful in ordering individual situation kinds, but it has broader use.
Example: When heading south, one crosses the equator *between* leaving Hawaii *and* arriving in Sydney.

## 16.7 Specification of Time Intervals Using Situations

This sub clause defines concepts related to the use of occurrences and individual situation kinds to specify time intervals.

### 16.7.1 Specifying time intervals using occurrences

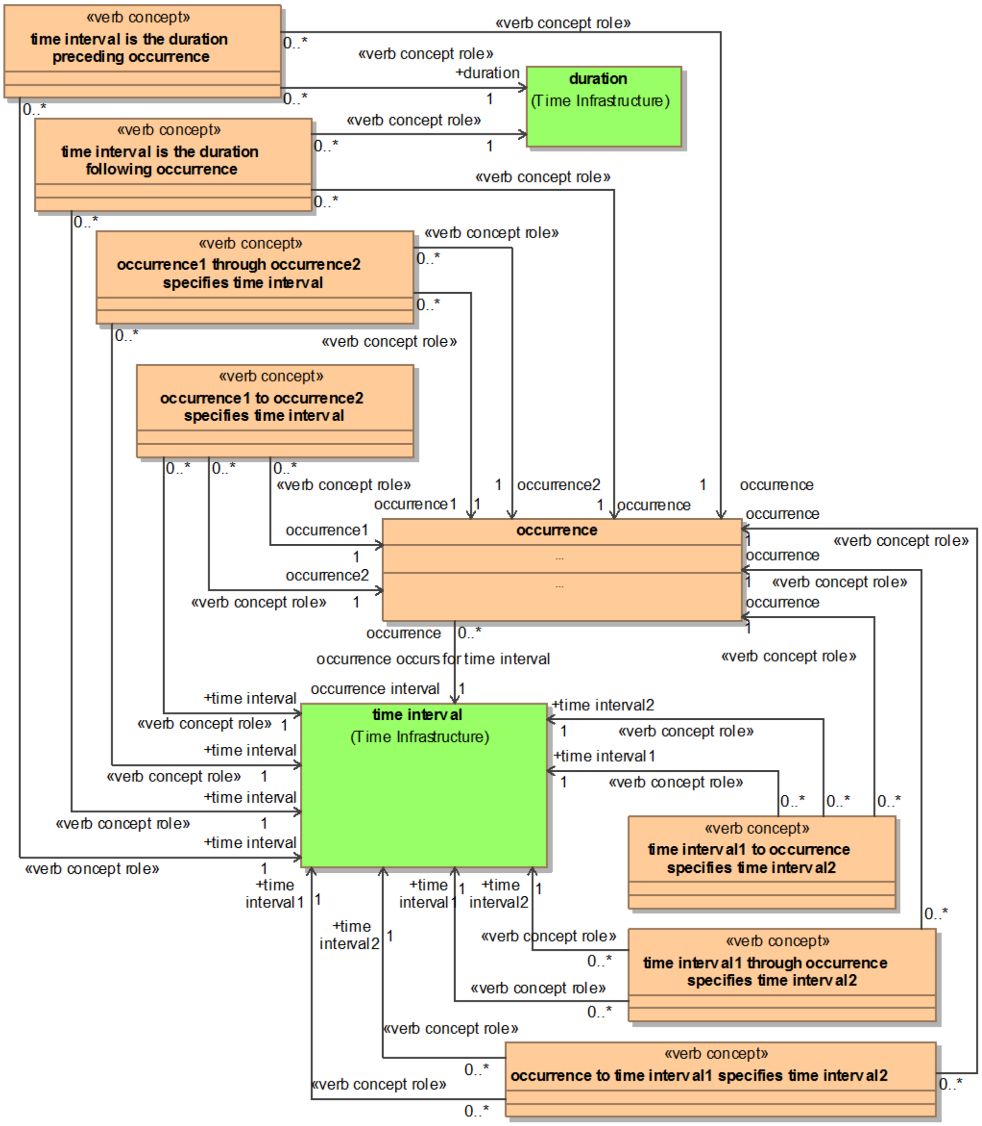


**Figure 16.7 - Time intervals specified by occurrences**

## time interval₁ *through* occurrence *specifies* time interval₂

Synonymous Form: time interval₁ *through* occurrence

Synonymous Form: time interval$_2$ *is* time interval$_1$ *through* occurrence
Synonymous Form: occurrence *through* time interval$_1$ *specifies* time interval$_2$
Synonymous Form: occurrence *through* time interval$_1$
Synonymous Form: time interval$_2$ *is* occurrence *through* time interval$_1$
Definition: the time interval$_2$ *is* the time interval$_1$ *plus* the occurrence interval *of* the occurrence
Description: The time interval extends from the start of time interval$_1$ through the end of the occurrence.
Note: The definition is correct for both the 'time interval$_1$ *through* occurrence' and 'occurrence *through* time interval$_1$' forms.
Example: The contract signing through 2012.

## occurrence$_1$ *through* occurrence$_2$ *specifies* time interval

Synonymous Form: occurrence$_1$ *through* occurrence$_2$
Synonymous Form: time interval *is* occurrence$_1$ *through* occurrence$_2$
Definition: the time interval *is* the occurrence interval *of* the occurrence$_1$ *plus* the occurrence interval *of* the occurrence$_2$
Description: The time interval extends from the start of occurrence$_1$ through the end of occurrence$_2$.
Example: The contract signing through the termination of the contract.

## time interval$_1$ *to* occurrence *specifies* time interval$_2$

Synonymous Form: time interval$_1$ *to* occurrence
Synonymous Form: time interval$_2$ *is* time interval$_1$ *to* occurrence
Synonymous Form: time interval$_1$ *until* occurrence *specifies* time interval$_2$
Synonymous Form: time interval$_1$ *until* occurrence
Synonymous Form: time interval$_2$ *is* time interval$_1$ *until* occurrence
Definition: the time interval$_2$ *is* the time interval$_1$ *to* the occurrence interval *of* the occurrence
Description: Time interval$_2$ extends from the start of time interval$_1$ up to, but not including, the start of the occurrence.
Example: Primordiality to the inauguration of the President.

## occurrence *to* time interval$_1$ *specifies* time interval$_2$

Synonymous Form: occurrence *to* time interval$_1$
Synonymous Form: occurrence *to* time interval$_1$ *is* time interval$_2$
Synonymous Form: occurrence *until* time interval$_1$ *specifies* time interval$_2$
Synonymous Form: occurrence *until* time interval$_1$
Synonymous Form: occurrence *until* time interval$_1$ *is* time interval$_2$
Definition: the time interval$_2$ *is* the occurrence interval *of* the occurrence *to* the time interval$_1$
Description: Time interval$_2$ extends from the start of the occurrence up to, but not including, the start of the time interval$_1$.
Example: The rise of the human species to perpetuity.

## occurrence$_1$ *to* occurrence$_2$ *specifies* time interval

Synonymous Form: occurrence$_1$ *to* occurrence$_2$
Synonymous Form: time interval *is* occurrence$_1$ *to* occurrence$_2$
Synonymous Form: occurrence$_1$ *until* occurrence$_2$ *specifies* time interval
Synonymous Form: occurrence$_1$ *until* occurrence$_2$
Synonymous Form: time interval *is* occurrence$_1$ *until* occurrence$_2$
Definition: the time interval *is* the occurrence interval *of* the occurrence$_1$ *to* the occurrence interval *of* the occurrence$_2$
Description: The time interval extends from the start of occurrence$_1$ up to, but not including, the start of occurrence$_2$.
Example: The contract signing to the contract termination.

## time interval *is the* duration *preceding* occurrence

| | |
|---|---|
| Synonymous Form: | duration *preceding* occurrence |
| Definition: | time interval *is the* duration *preceding* the occurrence interval *of* the occurrence |
| Description: | The time interval has the duration and is immediately before the occurrence. |

## time interval *is the* duration *following* occurrence

| | |
|---|---|
| Synonymous Form: | duration *following* occurrence |
| Definition: | time interval *is the* duration *following* the occurrence interval *of* the occurrence |
| Description: | The time interval has the duration and is immediately after the occurrence. |

### 16.7.2 Specifying time intervals using situation kinds

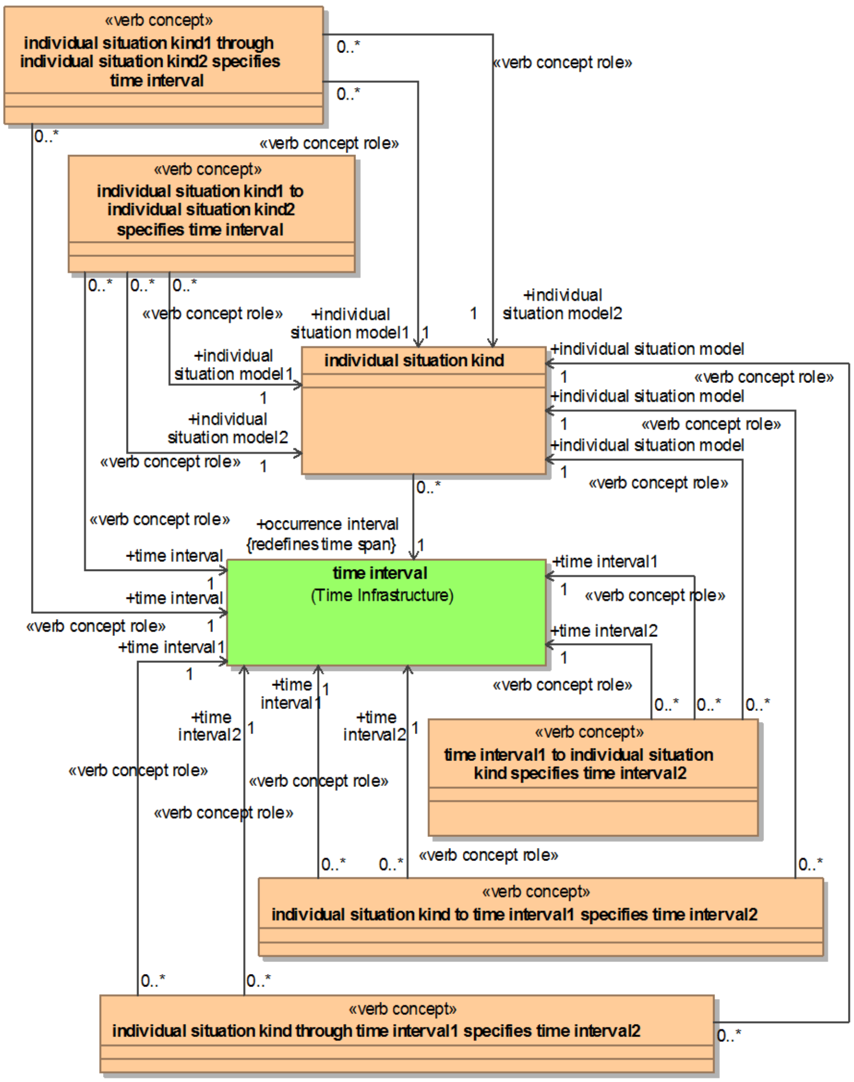


Figure 16.8 - Time intervals specified by situation kinds

**time interval₁ *through* individual situation kind *specifies* time interval₂**

Synonymous Form: time interval₁ *through* individual situation kind
Synonymous Form: time interval₂ *is* time interval₁ *through* individual situation kind
Synonymous Form: individual situation kind *through* time interval₁ *specifies* time interval₂

Synonymous Form: individual situation kind *through* time interval$_1$

Synonymous Form: time interval$_2$ *is* individual situation kind *through* time interval$_1$

Definition: the individual situation kind *has* exactly one occurrence and the time interval$_2$ *is* the time interval$_1$ *through* the occurrence interval *of* the individual situation kind

Description: Time interval$_2$ extends from the start of time interval$_1$ through the end of the occurrence of the individual situation kind.

Note: The definition is correct for both the individual situation kind' and 'individual situation kind *through* time interval$_1$' forms.

Example: Primordiality through the rise of the human race.

Example: The coronation of Queen Elizabeth II through 1972.

## individual situation kind$_1$ *through* individual situation kind$_2$ *specifies* time interval

Synonymous Form: individual situation kind$_1$ *through* individual situation kind$_2$

Synonymous Form: time interval *is* individual situation kind$_1$ *through* individual situation kind$_2$

Definition: the individual situation kind$_1$ *has* exactly one occurrence and the individual situation kind$_2$ *has* exactly one occurrence and the time interval *is* the occurrence interval *of* the individual situation kind$_1$ *through* the occurrence interval *of* the individual situation kind$_2$

Description: The time interval extends from the start of occurrence *of* individual situation kind$_1$ through the end of the occurrence of individual situation kind$_2$.

Example: The inception of a contract through the termination of the contract.

## time interval$_1$ *to* individual situation kind *specifies* time interval$_2$

Synonymous Form: time interval$_1$ *to* individual situation kind

Synonymous Form: time interval$_2$ *is* time interval$_1$ *to* individual situation kind

Synonymous Form: time interval$_1$ *until* individual situation kind *specifies* time interval$_2$

Synonymous Form: time interval$_1$ *until* individual situation kind

Synonymous Form: time interval$_2$ *is* time interval$_1$ *until* individual situation kind

Definition: the individual situation kind *has* exactly one occurrence and the time interval$_2$ *is* the time interval$_1$ *to* the occurrence interval *of* the individual situation kind

Description: Time interval$_2$ extends from the start of time interval$_1$ up to just before the occurrence of individual situation kind.

Example: 2010 to the termination of employment.

## individual situation kind *to* time interval$_1$ *specifies* time interval$_2$

Synonymous Form: individual situation kind *to* time interval$_1$

Synonymous Form: time interval$_2$ *is* individual situation kind *to* time interval$_1$

Synonymous Form: individual situation kind *until* time interval$_1$

Synonymous Form: individual situation kind *until* time interval$_1$ *is* time interval$_2$

Definition: the individual situation kind *has* exactly one occurrence and the time interval$_2$ *is* the occurrence interval *of* the individual situation kind *to* the time interval$_1$

Definition: Time interval$_2$ extends from the first occurrence of situation kind up to just before the first time interval$_1$.

Example: Hiring to 2010.

## individual situation kind$_1$ *to* individual situation kind$_2$ *specifies* time interval

Synonymous Form: individual situation kind$_1$ *to* individual situation kind$_2$

Synonymous Form: time interval *is* individual situation kind$_1$ *to* individual situation kind$_2$

Synonymous Form: individual situation kind$_1$ *until* individual situation kind$_2$ *specifies* time interval

Synonymous Form: individual situation kind$_1$ *until* individual situation kind$_2$

Synonymous Form: time interval *is* individual situation kind$_1$ *until* individual situation kind$_2$

Definition: the individual situation kind$_1$ *has* exactly one occurrence and the individual situation kind$_2$ *has* exactly one occurrence and the time interval *is* the occurrence interval *of* the individual situation kind$_1$ *to* the occurrence interval *of* the individual situation kind$_2$

Description: The time interval extends from the start of the occurrence of individual situation kind$_1$ up to, but not including, the occurrence of individual situation kind$_2$.

Example: Hiring to termination.

## 16.8 Propositions, Situation Kinds, and Occurrences

The Date-Time Vocabulary builds on SBVR's state of affairs and related concepts. Clause 16.8.1 examines the relevant aspects of SBVR as background for clause 16.8.2, which discusses the truth of propositions, and for clause 16.8.3, which suggests how situation kinds, occurrences, and states of affairs should be used with verb concepts and verb concept objectifications.

### 16.8.1 'State of Affairs' in SBVR

The following glossary entries are excerpted from sub-clause 8.5 'Extensions' of SBVR. See the SBVR specification for the Notes, Examples, and other related material.

#### state of affairs

Definition: res that is an event, activity, situation, or circumstance

#### proposition *corresponds to* state of affairs

General Concept: 'meaning *corresponds to* thing'

Definition: the state of affairs is posited by the proposition and if the state of affairs were *actual*, the proposition would *be true*

#### state of affairs *is actual*

Definition: the state of affairs happens (i.e., takes place, obtains)

#### actuality

Definition: state of affairs that *is actual*

SBVR sub clause 8.5.2 "Necessities Concerning Extension" defines several Necessities that are relevant to the Date-Time Vocabulary. Two of these are quoted verbatim here because an understanding of the relationship of states of affairs to time depends upon these constraints, and because the applicability of the second Necessity is narrowed by the Date-Time Vocabulary in this subclause.

Necessity: Each instance *of* a verb concept *is* an actuality.

Necessity: Each proposition that *is true corresponds to* exactly one actuality.

SBVR sub clause 8.5.2 also contains a Necessity that reads "Each proposition *corresponds to* exactly one state of affairs." As discussed below, this Necessity is unacceptable for the Date-Time Vocabulary because it requires a proposition such as "the United States elects a president" to correspond to only one state of affairs; i.e., only one event. The goal of the Date-Time Vocabulary is to provide concepts that are sufficient to represent real states of affairs, such as elections that occur multiple times. The Date-Time Vocabulary replaces this Necessity with a close alternative, "Each proposition *corresponds to* exactly one situation kind." This alternative is discussed in detail, below.

The Date-Time Vocabulary extends the concepts outlined above to address the following concerns.

1. The Necessity "Each proposition *corresponds to* exactly one state of affairs" fails to acknowledge that many propositions correspond to states of affairs that recur. As stated in an Example in the 'state of affairs' glossary

entry in the SBVR specification, the proposition "EU-Rent owns 10,000 rental cars" *corresponds to* the state of affairs "EU-Rent owning 10,000 rental cars". The Necessity requires that this state of affairs only happens once. What if it happens in 2009 and also in 2012, but not in 2010 or 2011?

2. The same Necessity also interferes with verb concept objectifications. An example in clause 11.1.5.3 of the SBVR specification, under the glossary entry for 'general concept *objectifies* verb concept', reads "The general concept 'sponsorship' objectifies the verb concept 'company sponsors publication'. Each sponsorship is an actuality that a given company sponsors a given publication." The Necessity that a proposition *corresponds to* exactly one state of affairs means that there can be only one sponsorship. That contradicts the observed business situations, in which many real companies support multiple sponsorships.

3. Any proposition can be interpreted in two different ways: (i) as a possible state of affairs that may be planned, budgeted for, feared, considered, etc., and (ii) as an occurrence. For example, many building codes require builders to plan for the possibility of building fires, whereas fire departments fight actual fires. Possible fires may or may not be actual (in the sense of SBVR's 'state of affairs *is actual*' characteristic). Occurrences – such as actual fires – are *actual* if the universe of discourse contains current facts about them. Even future events (e.g., the election of a U.S. President in the years 2024 and 2028) are occurrences if they are facts ("propositions that are taken as true") at the current time.

The Date-Time Vocabulary addresses these concerns by building on the SBVR state of affairs concept as described in sub clause 16.8.2.

### 16.8.2 Propositions and States of Affairs

In a static world that has no notion of change, there is a 1-to-1 relationship between propositions and states of the possible worlds: A proposition *is true* if the state it describes is the state of that world, and it is false if the state it describes is not the state of that world. (The SBVR model of states of affairs reflects this model.)

When temporal concepts are introduced into the formal logic model, a distinction must be made between two aspects of the SBVR concept 'proposition' – the truth or falsity of the proposition, and a 'meaning' in terms of a situation. This is because many propositions *correspond to* a single situation (a 'situation kind') that may have multiple occurrences. Such propositions are also said to *describe* the occurrences of the situation kind. For example, the proposition "each payment must precede delivery" is an SBVR way to state an obligation about the sequencing of payment and delivery, as might be given in a BPMN process model. In a given possible world, there may be many occurrences of payment and delivery, and thus many occurrences of payment preceding delivery.

SBVR sub clause 8.1.2 says that a proposition is true if "the state of affairs that the proposition *corresponds to is actual*". The Date-Time Vocabulary specifies that each proposition *corresponds to* exactly one situation kind, and the situation kind *is actual* if and only if the situation kind has at least one occurrence that *is current* in the universe of discourse. This clause specifies what it means for a situation kind to *be actual*, and thus for the corresponding proposition to *be true*.

| | |
|---|---|
| Necessity: | Each situation kind *is actual* if and only if the situation kind *has* at least one *occurrence* that *is current*. |
| Note: | In SBVR, a proposition *is true* if it *corresponds to* a state of affairs that *is actual*. The Necessity above establishes the basis for determining whether a proposition *is true* in a given universe of discourse that contains time. |
| Note: | The rule "Each factory manager must budget for situations where machines break down" states an obligation with respect to a situation kind that is the instance of the proposition "machines break down". The situation kind may or may not turn out to *be actual* at some time because the situation kind may or may not have any occurrences. |

Each proposition may or may not reference time, and if it does reference time, then it may reference the past, the present, or the future. Regardless, a proposition *is true* if it *corresponds to* a situation kind that has an occurrence that *is current* in the universe of discourse. Each case is discussed and illustrated with an example, in the following text.

Most propositions do not mention time (i.e., are "atemporal"). For example, the proposition "the building is on fire" does not mention time. The truth of this example depends upon whether the proposition corresponds to an occurrence that *occurs for* the current time in the universe of discourse. The occurrence may be directly given by a fact in the universe of discourse, or may be inferred from facts in the universe of discourse.

Some propositions are stated using past, present, or future tense, or contain explicit references to past time, current time, or future time. These propositions are true if and only if the universe of discourse contains facts ("propositions taken as true") that specify or imply current occurrences of the propositions. For example, the proposition "the contract was signed" is true if and only if there is an occurrence of "a signing of the contract" and that occurrence is in the past. The occurrence may exist as a fact or can be inferred from facts of the universe of discourse. Similarly, propositions about the present or the future are true if they exist as facts or are implied by facts of the universe of discourse. The proposition "the contract will expire" is a true proposition about the future if an occurrence of the proposition can be inferred from the facts of the universe of discourse.

Propositions may mention an explicit time, either as a time coordinate or as a definite description. For example, "an election is held in 2012" mentions the time coordinate "2012". The proposition "the contract will expire 2 years from the date the contract is signed" specifies a time via a definite description. Such propositions are true if the universe of discourse contains facts that specify or imply their occurrence – even if they are in the future.

Occurrences *are actual* if they *are current*:

Necessity: Each occurrence *is actual* if and only if the occurrence *is current*.

The Date-Time Vocabulary takes the position that propositions do not *correspond to* occurrences, even though occurrence is a specialization of state of affairs:

Necessity: It is not the case that some proposition *corresponds to* an occurrence.

When a proposition *corresponds to* a situation kind, the proposition *describes* any occurrences of the situation kind.

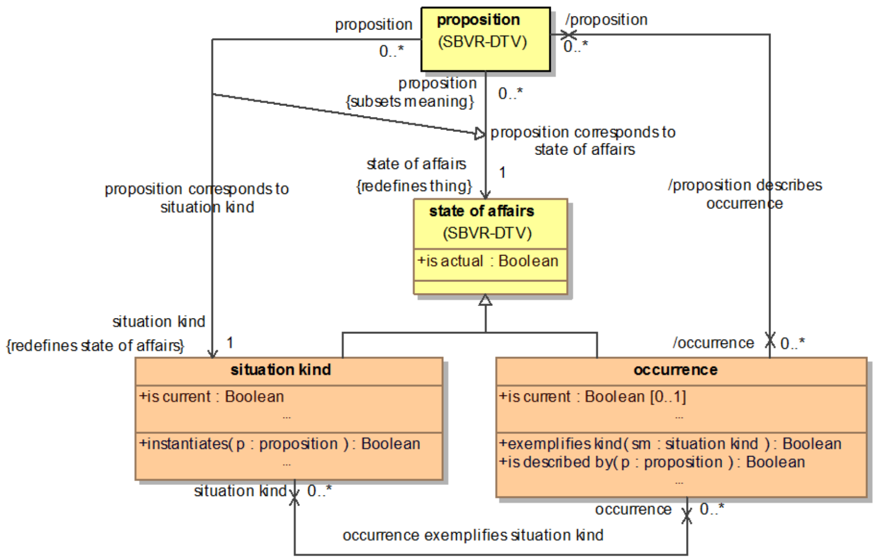


**Figure 16.9 - Propositions, Situation Kinds, and Occurrences**

### proposition *corresponds to* situation kind

General Concept: proposition *corresponds to* state of affairs
Necessity: Each proposition *corresponds to* exactly one situation kind.
Note: In the Date-Time Vocabulary, the Necessity immediately above replaces the SBVR Necessity "Each proposition *corresponds to* exactly one state of affairs".
Note: The instances of propositions are situation kinds, which may or may not be actual. Propositions may be planned, feared, budgeted for, etc., whether or not they *correspond to* situation kinds that are actual. A proposition may refer to the past, present, or future without implying that the corresponding situation kind has been, is, or will be actual.

### proposition *describes* occurrence

Definition: The proposition *corresponds to* a situation kind that *has* the occurrence.
Note: That is, the occurrence *exemplifies* the proposition in the sense of Plantinga (see [Menzel]).
Necessity: A proposition *is true* if and only if the proposition *describes* an occurrence that *is current*.
Note: In a temporal world, the same proposition can describe several different occurrences, even when all the roles in the proposition are played by exactly the same things in all occurrences. What distinguishes the occurrences are the things that are not mentioned in the proposition. In particular, a proposition that does not mention time may describe different occurrences that have different occurrence intervals.
Example: Brazil wins the FIFA World Cup. That was true in 1994 and 2002, but false in 1992, 1998, 2006, and 2010. So the proposition "Brazil wins the FIFA World Cup" *describes* two occurrences in the period 1992 to 2012.
Example: The proposition "Brazil won the FIFA World Cup in 1994" describes an occurrence that *is current* in 2012. Thus, the proposition "Brazil won the FIFA World Cup in 1994" *is true* in the world of 2012.
Possibility: A proposition *describes* zero or more occurrences (in a given possible world).
Possibility: An occurrence *is described by* zero or more propositions.

## 16.8.3 Verb Concepts, Verb Concept Objectification, and States of Affairs

The Date-Time Vocabulary distinction between situation kinds and occurrences enables verb concepts to be explicit about whether they range over potential states of affairs or real happenings. For example, an '*insures*' verb concept might be defined as 'person *insures against* situation kind' to mean that the verb ranges over potential events, activities, situations, or circumstances. A '*reports*' verb concept might be specified as 'person *reports* occurrence' to mean that what gets reported are real events, etc. One insures against fires that may never happen, but one should only report actual fires.

Business vocabularies should not define verb concepts that range over 'state of affairs' because the meaning is unclear.

SBVR sub clause 11.1.5.3 "Verb Concept Objectification" formalizes the idea that a general concept may be coextensive with a verb concept, the way many English gerunds (e.g., "planning") are coextensive with some verbs (e.g., "plan"). Verb concept objectifications that may or may not be *actual* should specialize either 'state of affairs' or 'situation kind'. Verb concept objectifications that are specifically about occurrences should specialize 'occurrence'.

Verb concept objectifications that specialize 'state of affairs' have the advantage that they may fill verb concept roles that range over 'situation kind' and also verb concept roles that range over 'occurrence'. For example, the verb concept objectification 'machine breakdown' defined as 'state of affairs that machine *is broken down*' may be used with the verb concept 'manager *plans for* situation kind' and also with the verb concept 'manager *reports* occurrence'. With this approach, a single verb concept objectification can be used with slightly different meanings associated with each verb concept that ranges over the verb concept objectification. This is possible because both 'situation kind' and 'occurrence' are specializations of 'state of affairs'. The advantage of this technique is that it better matches typical business English usage.

## 16.9 Language Tense and Aspect

As discussed in sub clause 7.12, human languages use past, present, and future tenses and incorporate simple, progressive, and perfect aspects. This sub clause provides concepts that enable all these tenses and aspects, in any combination. They extend the relationships between situation kinds, occurrences, and time that are defined in this clause.

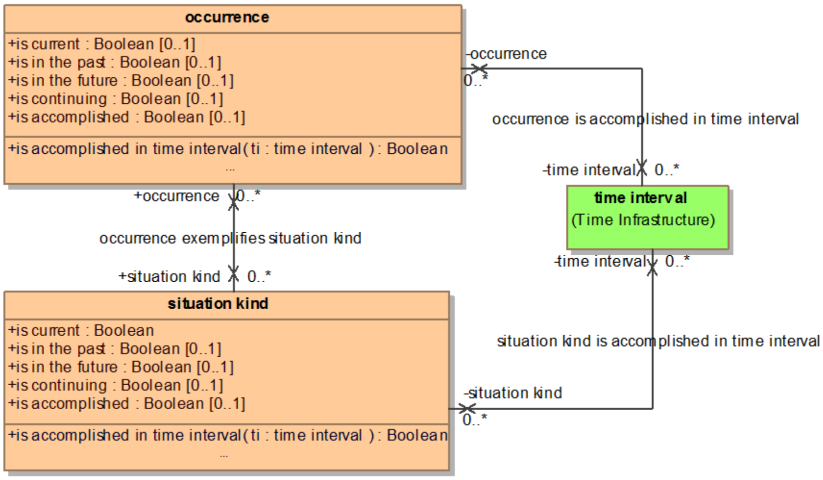


**Figure 16.10 - Language Tense and Aspect**

The following verb concepts formalize the progressive and perfect language aspects. The concepts are provided for both 'situation kind' and 'occurrence'; the former are normally used in guidance statements, while the latter are most useful in facts.

### situation kind *is continuing*

| | |
|---|---|
| Definition: | the situation kind is unfinished at some reference time interval |
| Note: | The reference time interval is when a fact is evaluated or a rule is being applied. |
| Note: | 'situation kind *is continuing*' indicates the progressive aspect of natural language. It is sometimes called the "continuous aspect." |
| Example: | If company x is going bankrupt.... |
| Note: | 'Situation kind *is continuing*' is not the negation of 'situation kind *is accomplished*' because a situation kind may end without being accomplished. Consider that the situation kind 'John writes book' in the partial rule "if John writes a book…" may end without John ever completing the book. |
| Note: | A situation kind may be *is continuing* or *is accomplished* or both or neither, and may also be in the past, present, or future tense. (See Table 16.1). |

### situation kind *is accomplished*

| | |
|---|---|
| Definition: | the situation kind has reached a point of completion or perfection at with respect to the "reference time interval" associated with the concept 'time interval *is past*' |
| Note: | The reference time interval is when a fact is evaluated or a rule is being applied. |
| Example: | If company x has gone bankrupt.... |
| Note: | 'Situation kind *is accomplished*' is not the negation of 'situation kind *is continuing*' because a situation kind may end without being accomplished. Consider that the |

| | |
|---|---|
| | situation kind 'John writes book' in the partial rule "if John writes a book…" may end without John ever completing the book. |
| Note: | A situation kind may be *is continuing* or *is accomplished* or both or neither, and may also be in the past, present, or future tense. (See Table 16.1). |

## situation kind *is accomplished in* time interval

| | |
|---|---|
| Definition: | the situation kind reaches a point of completion or perfection at some time interval$_2$ that *is part of* the time interval |
| Example: | If the contract is completed within this year …. |

## occurrence *is continuing*

| | |
|---|---|
| Definition: | the occurrence is unfinished at some reference time interval |
| Note: | The reference time interval is when a fact is evaluated or a rule is being applied. |
| Note: | 'occurrence *is continuing*' indicates the progressive aspect of natural language. It is sometimes called the "continuous aspect." |
| Example: | Company x is going bankrupt. |
| Note: | 'Occurrence *is continuing*' is not the negation of 'occurrence *is accomplished*' because an occurrence may end without being accomplished. Consider that the occurrence 'John writes book' may end without John ever completing the book. |
| Note: | An occurrence may be *is continuing* or *is accomplished* or both or neither, and may also be in the past, present, or future tense. (See Table 16.1). |

## occurrence *is accomplished*

| | |
|---|---|
| Definition: | the occurrence has reached a point of completion or perfection at with respect to the "reference time interval" associated with the concept 'time interval *is past*' |
| Note: | The reference time interval is when a fact is evaluated or a rule is being applied. |
| Example: | Company x has gone bankrupt. |
| Note: | 'Occurrence *is accomplished*' is not the negation of 'occurrences *is continuing*' because a state of affairs may end without being accomplished. Consider that the state of affairs 'John writes book' may end without John ever completing the book. |
| Note: | An occurrence may be *is continuing* or *is accomplished* or both or neither, and may also be in the past, present, or future tense. (See Table 16.1). |

## occurrence *is accomplished in* time interval

| | |
|---|---|
| Definition: | the occurrence reaches a point of completion or perfection at some time interval$_2$ that *is part of* the time interval |
| Example: | The occurrence "Columbus reaches the new world" *is accomplished in* the 15th Century. |

These verb concepts enable formulation of past, present, and future tense propositions. As above, the 'situation kind' versions of these concepts are most useful in guidance statements, while the 'occurrence' versions are intended for use in facts.

## situation kind *is in the past*

| | |
|---|---|
| Definition: | the situation kind *occurs throughout* some time interval that *is in the past* |
| Example: | If the customer has previously failed to pay his bill …. |
| Note: | Whether a situation kind *is in the past* may be inferred when a situation kind is located in time via any of the verb concepts given above, such as "situation kind$_1$ *is before* situation kind$_2$." |

## situation kind *is current*

| | |
|---|---|
| Definition: | the situation kind *occurs for* some time interval that *is current* |

Example: "If the bill is currently due" (which might be formulated as "if the bill is due is current").

## situation kind *is in the future*

Definition: the situation kind *occurs throughout* some time interval that *is in the future*

Example: "If President Obama will write his memoirs," which might be formulated as "If President Obama writes his memoirs *in the future*."

Note: Whether a situation kind *is in the future* may be inferred when a situation kind is located in time via any of the verb concepts given above, such as "situation kind$_1$ *is before* situation kind$_2$."

## occurrence *is in the past*

Definition: the occurrence *occurs throughout* some time interval that *is in the past*

Example: The reign of Alexander the Great *is in the past*.

Note: Whether an occurrence *is in the past* may be inferred when an occurrence is located in time via any of the verb concepts given in this clause, such as "occurrence$_1$ *is before* occurrence$_2$".

## occurrence *is current*

Definition: the occurrence *occurs for* some time interval that *is current*

Example: That EU-Rent is in business *is current* (which means the same as "EU-Rent is currently in business").

## occurrence *is in the future*

Definition: the occurrence *occurs throughout* some time interval that *is in the future*

Example: "President Obama writes his memoirs" *is in the future*.

Note: Whether a state of affairs *is in the future* may be inferred when an occurrence is located in time via any of the verb concepts given in this clause, such as "occurrence$_1$ *is before* occurrence$_2$."

This specification defines vocabulary fact types in the present tense. Table 16.1 gives examples of how other tenses and aspects can be formulated. To show the range of expression supported by this vocabulary, some examples reference specific time intervals, while others leave unstated the time interval that an occurrence *is continuing* or *is accomplished*.

Table 16.1 assumes a domain vocabulary verb concept "John *writes* book". The examples are given as facts, and hence are formulated using the 'occurrence' version of the verb concepts listed above.

The text "(that John *writes* a book)" is short-hand for "the proposition 'John *writes* a book', corresponds to a situation kind". Nesting is used for some combinations. For example, "(that (that John *writes* a book) *is in the future*) *is accomplished*" means that the characteristic '*is accomplished*' is applied to a situation kind of "the characteristic '*is in the future*', which itself is applied to a situation kind of the proposition 'John *writes* a book' ".

**Table 16.1 – Examples of tense and aspect formulation**

| Simple Aspect | | |
|---|---|---|
| **Tense** | **Example** | **Formulation** |
| past | John wrote a book | (that John writes a book) is in the past |
| present | John writes a book | John writes a book |
| future | John will write a book | (that John writes a book) is in the future |
| **Progressive Aspect** | | |
| **Tense** | **Example** | **Formulation** |
| past | John was writing a book | (that (that John writes a book) is continuing) is in the past |
| present | John is writing a book | (that John writes a book) is continuing |
| future | John will be writing a book | (that (that John writes a book) is continuing) is in the future |
| **Perfect Aspect** | | |
| **Tense** | **Example** | **Formulation** |
| past | John had written a book before 2009 | (that (that John writes a book) is accomplished) occurs before 2009 |
| present | John has written a book | (that John writes a book) is accomplished |
| future | John will have written a book by 2030 | (that (that John writes a book) is accomplished) occurs before 2030 |
| **Progressive and Perfect** | | |
| **Tense** | **Example** | **Formulation** |
| past | John had been writing a book before 2009 | (that (that (that John writes a book) is continuing) is accomplished) occurs before 2009 |
| present | John has been writing a book | (that (that John writes a book) is continuing) is accomplished |
| future | John will have been writing a book by 2030 | (that (that (that John writes a book) is continuing) is accomplished) occurs before 2030 |

At the time of writing this document, the example "John will be writing a book during January 2021 through June 2022" is in the future. Nevertheless, the formulation includes the apparently redundant "is in the future" to express the future tense of the statement even after 2022.

The formulation of "John was writing a book last year" excludes "is in the past" because "last year" applies at all times.

______________________________________________
______________________________________________

# 17 Schedules (normative)

## 17.1 General

An important element of business activity and contracts is schedules: plans for situation kinds to occur at specific times.

---

Schedules Vocabulary

General Concept: terminological dictionary
Language: English
Included Vocabulary: Situations Vocabulary
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#SchedulesVocabulary

---

## 17.2 Schedules

Schedules model relationships between time intervals and situation kinds that are planned to occur at the time intervals. Time intervals of schedules can be sequential or overlapping, and at regular or irregular intervals. Schedules with non-overlapping sequential time intervals that repeat regularly are called regular schedules. Most mortgage loans call for payment according to regular schedules. Schedules with irregular time intervals are called ad hoc schedules. A conference schedule is usually ad hoc.

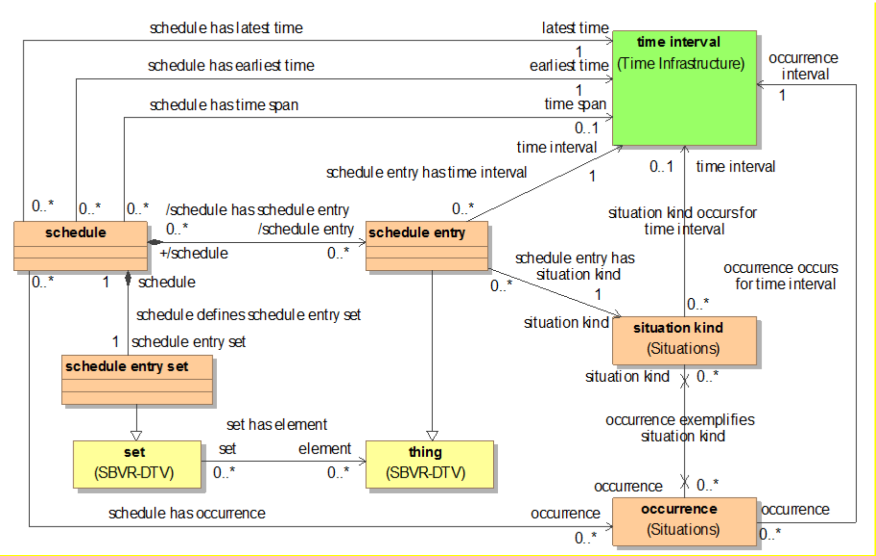


**Figure 17.1 - Schedules**

## schedule

Definition: a plan for carrying out situation kinds at each of multiple time intervals

Each schedule is composed of an explicit (for ad hoc schedules and schedule stubs of regular schedules) or implicit (for regular schedules) set of schedule entries.

## schedule entry

Definition: proposition that the situation kind happens on a time interval

Note: The situation kind should define its precise relationship with the time interval: whether the situation kind *occurs for*, *within*, etc., the time interval.

## schedule *has* schedule entry

Definition: the schedule entry *is in* the schedule entry set *of* the schedule.

CLIF Definition:

```
(forall (s se)
(iff ("schedule has schedule entry" s se)
     (exists (ses)
       (and
         ("schedule entry set of schedule" ses s)
         ("thing is in set" se ses) ))))
```

OCL Definition:

```
context schedule
def: _'schedule has schedule entry'(se: _'schedule entry') : Boolean =
  self._'schedule entry set'.includes(se)
```

## schedule entry *has* situation kind

Necessity: Each schedule entry *has* exactly one situation kind.

CLIF Axiom:

```
(forall (se) (exists (sk1)
  (and ("schedule entry has situation kind" se sk1)
       (forall (sk2)
         (if ("schedule entry has situation kind" se sk2)
             (= sk1 sk2) )))))
```

OCL Constraint:

```
context _'schedule entry'
     inv: self._'situation kind'->size() = 1
```

## schedule entry *has* time interval

Necessity: Each schedule entry *has* exactly one time interval.

CLIF Axiom:

```
(forall (se) (exists (t1)
  (iff ("schedule entry has time interval" se t1)
       (forall (sk2)
         (if ("schedule entry has time interval" se t2)
             (= t1 t2) )))))
```

OCL Constraint:

```
context _'schedule entry'
     inv: self._'time interval'->size() = 1
```

## schedule entry set

Definition: set that *is of* 'schedule entry'

Necessity: Each schedule entry set *includes* at least one schedule entry.

CLIF Axiom:

```
(forall (seset) (exists (se)
  ("schedule entry set includes schedule entry" seset se)))
```

OCL Constraint:

```
context _'schedule entry set'
     inv: self.includes->size()>0
```

## schedule *defines* schedule entry set

Description: The schedule entry set is explicit in an ad hoc schedule, and implicit in a regular schedule. The schedule entry set models the situation kinds and corresponding time intervals of the schedule.

Note: This verb concept is refined, below, by 'regular schedule *defines* regular entry set'. 'Ad hoc schedule' uses this verb concept as-is.

Necessity: Each schedule *defines* exactly one schedule entry set.

CLIF Axiom:
```
(forall (se) (exists (ses)
  (and ("schedule entry has schedule entry set" se ses)
        (forall (ses2)
          (if ("schedule entry has schedule entry set" se ses2)
              (= ses1 ses2) )))))
```

OCL Constraint:
```
context _'schedule entry'
     inv: self._'schedule entry set'->size() = 1
```

Schedules of all types share several attributes:

## schedule *has* occurrence

Definition: the occurrence *exemplifies* the situation kind *of* a schedule entry *of* the schedule and the occurrence interval *of* the occurrence *overlaps* the time interval *of* the schedule entry

Note: The occurrence may be in the past or may be planned for the future.

CLIF Definition:
```
(forall (s o)
  (iff ("schedule has occurrence" s o)
      (exists (("schedule entry" se) ("situation kind" sk))
          (and
            ("schedule has schedule entry" s se)
            ("schedule entry has situation kind" s sk)
            ("occurrence exemplifies situation kind" o sk)
            ("time interval1 overlaps time interval2"
              ("occurrence interval" o) ("time interval" se))
          ))))
```

OCL Definition:
```
context schedule
def: _'schedule has occurrence'(o: occurrence) : Boolean =
    self._'schedule entry' ->exists(se |
        o.exemplifies(se._'situation kind')
        and o._'occurrence interval'.overlaps(se._'time interval))
```

## earliest time

Concept Type: role

General Concept: time interval

Description: The earliest scheduled time of a schedule.

## schedule *has* earliest time

Definition: the earliest time *is* the time interval *of* some schedule entry *of* the schedule and the earliest time *does* not *start after* the time interval *of* each schedule entry *of* the schedule

CLIF Definition:
```
(forall (s et)
  (iff ("schedule has earliest time" s et)
       (and
         (exists (se1)
            (and
               ("schedule has schedule entry" s se1)
               ("schedule entry has time interval" se1 et)))
         (forall (se2 ti2)
            (if (and
                   ("schedule has schedule entry" s se2)
                   ("schedule entry has time interval" se2 ti2) )
               (not ("time interval1 starts after time interval2" et ti2)) )
))))
```

OCL Definition:
```
context schedule
def: _'earliest time'(et: _'time interval') : Boolean =
  self._'schedule entry' -> exists (se1 | se1._'time interval'.equals(et))
  and self._'schedule entry' -> forAll(se2 |
          not et._'starts after'(se2._' time interval'))
```

Synonymous Form: earliest time *of* schedule

CLIF Definition:
```
(forall ((s schedule) (et "time interval"))
  (iff (= et ("earliest time of schedule" s)
      ("schedule has earliest time" s et) ))
```

OCL Definition:
```
context schedule
def: _'schedule has earliest time'() : _'time interval' =
  self._'schedule entry'._'time interval'->
    select(ti |self._'earliest time'(ti))
```

## latest time

Concept Type: role

General Concept: time interval

Description: The latest scheduled time of a schedule.

## schedule *has* latest time

Definition: the latest time *is* the time interval *of* some schedule entry *of* the schedule and the latest time *ends after* the time interval *of* each schedule entry *of* the schedule

CLIF Definition:
```
(forall (s lt)
  (iff ("schedule has latest time" s lt)
       (and
          (exists (se1)
             (and
               ("schedule has schedule entry" s se1)
               ("schedule entry has time interval" se1 lt)))
          (forall (se2 ti2)
               (if (and
                      ("schedule has schedule entry" s se2)
                      ("schedule entry has time interval" se2 ti2) )
                 (not ("time interval1 ends after time interval2" ti2 lt)) )
          ))))
```

OCL Definition:
```
context schedule
def: _'schedule has latest time'(lt: _'time interval') : Boolean =
  self._'schedule entry'-> exists(se1 | lt.equals(se1._'time interval'))
 and self._'schedule entry'->forAll(se2: |
          lt._'ends after'(se2._'time interval'))
```

Synonymous Form: latest time *of* schedule

CLIF Definition: (forall ((s schedule) (lt "time interval"))
  (iff (= lt ("latest time of schedule" s)
        ("schedule has latest time" s lt) ))

OCL Definition: context schedule
def: _'latest time of schedule'() : _'time interval' =
  self._'schedule entry'._'time interval'->
    select(ti |self._'schedule has latest time'(ti))

## schedule *has* time span

Definition: the time span *equals* the earliest time *of* the schedule *through* the latest time *of* the schedule

Description: the time span is the smallest time interval that *includes* the time intervals of all planned occurrences of the schedule

Description: The time span is the “ convex hull” of a schedule.

CLIF Definition:
```
(forall (s ts)
  (iff ("schedule has time span" s ts)
    (and
      ("time interval" ts)
      ("time interval1 plus time interval2 is time interval3"
         ("earliest time of schedule" s)
         ("latest time of schedule" s)
          ts ) )))
```

OCL Definition:
```
context schedule
def: _'schedule has time span'(ts: _'time interval') : Boolean =
   ts.equals(self._'earliest time'.plus(self._'latest time'))
```

Synonymous Form: time span *of* schedule

CLIF Definition:
```
(forall ((ts "time interval") (s schedule))
  (iff (= ts ("time span of schedule" s))
       ("schedule has time span" s ts) ))
```

OCL Definition:
```
context schedule
def: _'time span of schedule'() : _'time interval' =
  self._'earliest time'.plus(self._'latest time')
```

Necessity: Each schedule *has* exactly one time span.

CLIF Axiom:
```
(forall (s) (exists (t1)
  (and ("schedule has time span" s t1)
        (forall (t2)
          (if ("schedule has time span" s t2) (= t1 t2))
        ))))
```

OCL Constraint:
```
context schedule
      inv: schedule._'time span'->size() = 1
```

Note: The verb concept ‘occurrence *occurs for* time interval’ can be used to say that an occurrence happens for the entire time span of a schedule.

Example: A conference meeting might *occur at* a particular time interval of an ad hoc schedule, while the entire conference *occurs for* the time span of the entire schedule.

## 17.3 Regular Schedules

Regular schedules define a single situation kind that recurs at each time interval of the regular schedule. The verb concept ‘regular schedule *is for* situation kind’ means that the situation kind *occurs at* each time interval of the regular schedule.

This definition requires further extension to address what might be called 'complex regular schedules': regular schedules in which the scheduled time interval is defined according to a calendar to be one or more proper parts (rather than the whole) of the recurrence duration. For example, this definition does not support schedules such as or "the first Tuesday of each calendar month" or "the first and last calendar day of each calendar month".

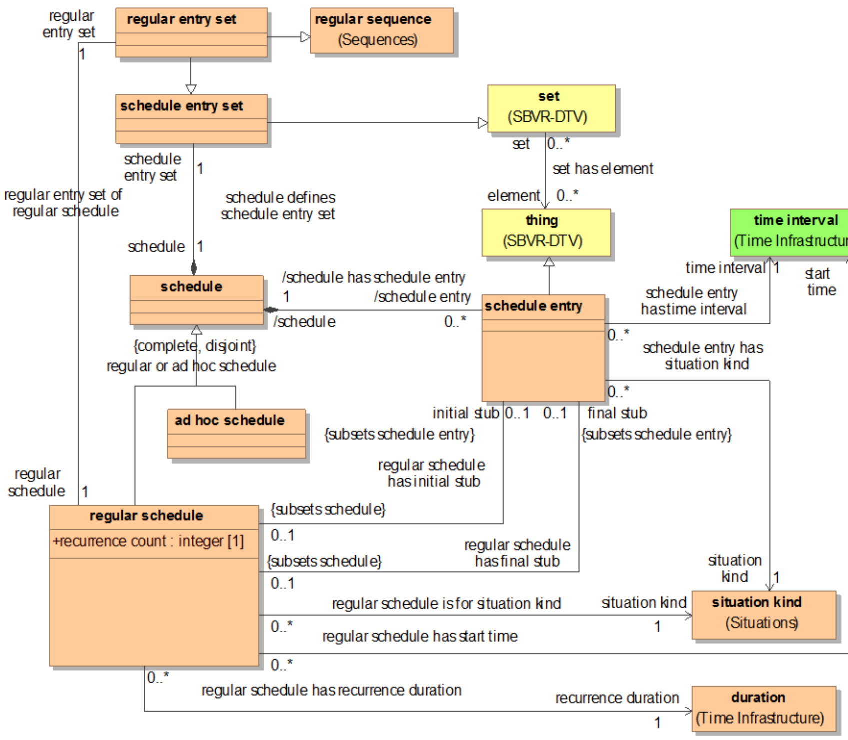


**Figure 17.2 - Regular Schedules**

## regular schedule

Definition: schedule that a single situation kind occurs at the earliest time of the regular schedule, and thereafter once each recurrence duration, for the recurrence count number of recurrence durations, with optional initial stub and final stub

Necessity: No regular schedule *is* an ad hoc schedule.

CLIF Axiom: (forall ((rs "regular schedule"))
(not ("ad hoc schedule" rs)) )

OCL Constraint: context _'regular schedule'
inv: not self.oclIsTypeOf(_'ad hoc schedule')

Example: A mortgage is payable monthly.

## regular schedule *is for* situation kind

Synonymous Form: situation kind *according to* regular schedule

Synonymous Form: situation kind *has* regular schedule

Definition: the occurrence *of* each schedule entry *of* the regular entry set *of* the regular schedule *exemplifies* the situation kind

Necessity: A regular schedule *is for* exactly one situation kind.

CLIF Axiom:
```
(forall (rs sk1)
  (if ("regular schedule is for situation kind" rs sk1)
    (forall (sk2)
      (if ("regular schedule is for situation kind" rs sk2) (= sk1 sk2))
    )))
```
OCL Constraint:
```
context _'regular schedule'
    inv: _'regular schedule'._'situation kind'->size() = 1
```
Example: An airline flies daily from NY to Dubai *according to* a flight schedule. The situation kind is ‘fly from NY to Dubai.’

## start time

Concept Type: role

Definition: time interval of the start of the recurring portion of a regular schedule

## regular schedule *has* start time

Definition: the start time is the time interval of the first recurrence of the regular schedule

Necessity: Each regular schedule *has* exactly one start time.

CLIF Axiom:
```
(forall (rs st1)
  (if ("regular schedule has start time" rs st1)
    (forall (st2)
      (if ("regular schedule has start time" rs st2) (= st1 st2))
    )))
```
OCL Constraint:
```
context _'regular schedule'
    inv: _'regular schedule'._'start time'->size() = 1
```

## recurrence duration

Synonym: repeat duration

Concept Type: role

Definition: duration that is between the occurrence intervals *of* the occurrences *of* consecutive schedule entries *of* the regular entry set *of* a regular schedule

## regular schedule *has* recurrence duration

Definition: the occurrence interval *of* an occurrence *of* the regular schedule *starts* recurrence duration *before* the occurrence interval *of* the next occurrence *of* the regular schedule

Necessity: Each regular schedule *has* exactly one recurrence duration.

CLIF Axiom:
```
(forall (rs rd1)
  (if ("regular schedule has recurrence duration" rs rd1)
    (forall (rd2)
      (if ("regular schedule has recurrence duration" rs rd2)
          (= rd1 rd2)) )))
```
OCL Constraint:
```
context _'regular schedule'
    inv: _'regular schedule'._'recurrence duration'->size() = 1
```

## recurrence count

Synonym: repeat count

Concept Type: role

Definition: number of occurrences *of* a regular schedule

## regular schedule *has* recurrence count

Definition: the recurrence count *is* the cardinality *of* the regular entry set *of* the regular schedule

| | |
|---|---|
| Necessity: | Each regular schedule *has* at most one recurrence count. |
| CLIF Axiom: | (forall (rs rc1)<br>(if ("regular schedule has recurrence count" rs rc1)<br>(forall (rc2)<br>(if ("regular schedule has recurrence count" rs rc2)<br>(= rc1 rc2)) ))) |
| OCL Constraint: | context _'regular schedule'<br>inv: _'regular schedule'._'recurrence count'->size() = 1 |
| Note: | This Necessity disallows unlimited regular schedules. |

To support financial contracts, regular schedules may have an initial stub and/or a final stub that identify special situations that come before or after the schedule's repeating component. For example a home mortgage is payable monthly, at the start of each calendar month, for 30 years. Because the mortgage is finalized in the middle of a calendar month, an initial payment is due for the period up to the due date of the first monthly payment. Similarly, a final payment is due for several remaining days after the last monthly payment. The initial stub and final stub of a regular schedule can capture the details of these initial and final payments.

## initial stub

| | |
|---|---|
| Concept Type: | role |
| General Concept: | schedule entry |
| Description: | An initial stub identifies special business treatment that should happen before the start of the recurring portion of a regular schedule. |

## regular schedule *has* initial stub

| | |
|---|---|
| Necessity: | Each regular schedule *has* at most one initial stub. |
| CLIF Axiom: | (forall (rs is1)<br>(if ("regular schedule has initial stub" rs is1)<br>(forall (is2)<br>(if ("regular schedule has initial stub" rs is2)<br>(= is1 is2)) ))) |
| OCL Constraint: | context _'regular schedule'<br>inv: _'regular schedule'._'initial stub'->size() <= 1 |

## final stub

| | |
|---|---|
| Concept Type: | role |
| General Concept: | schedule entry |
| Description: | A final stub identifies special business treatment that should happen after the end of the recurring portion of a regular schedule. |

## regular schedule *has* final stub

| | |
|---|---|
| Necessity: | Each regular schedule *has* at most one final stub. |
| CLIF Axiom: | (forall (rs fs1)<br>(if ("regular schedule has final stub" rs fs1)<br>(forall (fs2)<br>(if ("regular schedule has final stub" rs fs2)<br>(= fs1 fs2)) ))) |
| OCL Constraint: | context _'regular schedule'<br>inv: _'regular schedule'._final stub'->size() <= 1 |

The following glossary entries "expand" 'regular schedule' to an implicit schedule entry set, including any initial stub and final stub. This enables the generic treatment (above) of regular schedules and ad hoc schedules.

## regular entry set

Definition: schedule entry set that *is* a regular sequence

## regular entry set *of* regular schedule

Definition: the cardinality *of* the regular entry set *is* the recurrence count *of* the regular schedule and the situation kind *of* each schedule entry *of* the regular entry set *is* the situation kind *of* the regular schedule and the time interval *of* the first member *of* the regular entry set *is* the start time *of* the regular schedule and the time interval *of* the schedule entry that *is next after* a given schedule entry *of* the regular entry set *is* the recurrence duration *of* the regular schedule *plus* the time interval *of* the schedule entry

Description: The regular entry set is defined inductively as follows:
- The recurrence count specifies the number of schedule entries.
- Each schedule entry has the situation kind of the regular schedule.
- The first schedule entry has the start time of the regular schedule.
- The time interval of each subsequent entry is computed from the time interval of the previous entry plus the recurrence duration.

Note: The following Necessity describes the construction of the (complete) schedule entry set of a regular schedule:

Necessity: The schedule entry set *of* a regular schedule *is* the regular entry set *of* the regular schedule *plus* each initial stub *of* the regular schedule *plus* each final stub *of* the regular schedule.

CLIF Axiom:
```
(forall (rs ses res)
  (if (and
        ("regular schedule" rs)
        ("schedule defines schedule entry set" rs ses)
        ("regular schedule has regular entry set" rs res)
        (exists (init) ("regular schedule has initial stub" rs init))
        (exists (fin) ("regular schedule has final stub" rs fin)) )
      (= ses (setplus (setplus res init) fin))
  ))
```

CLIF Axiom:
```
(forall (rs ses res)
  (if (and
        ("regular schedule" rs)
        ("schedule defines schedule entry set" rs ses)
        ("regular schedule has regular entry set" rs res)
        (exists (init) ("regular schedule has initial stub" rs init))
        (not (exists (fin) ("regular schedule has final stub" rs fin))) )
      (= ses (setplus res init))
  ))
```

CLIF Axiom:
```
(forall (rs ses res)
  (if (and
        ("regular schedule" rs)
        ("schedule defines schedule entry set" rs ses)
        ("regular schedule has regular entry set" rs res)
        (not (exists (init) ("regular schedule has initial stub" rs init)))
        (exists (fin) ("regular schedule has final stub" rs fin)) )
      (= ses (setplus res fin))
  ))
```

CLIF Axiom:
```
(forall (rs ses res)
  (if (and
      ("regular schedule" rs)
      ("schedule defines schedule entry set" rs ses)
      ("regular schedule has regular entry set" rs res)
      (not (exists (init) ("regular schedule has initial stub" rs init)))
      (not (exists (fin) ("regular schedule has final stub" rs fin))) )
    (= ses res)
  ))
```

OCL Definition:
```
context _'regular schedule'
inv: self._'schedule entry set' = self._'regular entry set'
  .plus(self._'initial stub').plus(self._'final stub')
```

## 17.4 Ad Hoc Schedule

Ad hoc schedules associate a situation kind with each time interval because (in the general case) different events happen at each time interval.

### ad hoc schedule

Definition: schedule that does not *have* a recurrence duration or a recurrence count

Note: An ad hoc schedule is a set, not a sequence, because the time intervals of the ad hoc schedule may not be unique and may not be ordered.

Necessity: No ad hoc schedule *is* a regular schedule.

CLIF Axiom:
```
(forall (ahs "ad hoc schedule")
  (not ("regular schedule" ahs)) )
```

OCL Constraint:
```
context _'ad hoc schedule'
    inv: not self.oclIsTypeOf(_'regular schedule')
```

___________________________________________________
___________________________________________________

# 18 Interchange of Duration Values and Time Coordinates (normative)

## 18.1 General

The foregoing parts of this specification provide a formal terminology for expressing facts and rules involving time concepts in business communications. The expressions for time intervals that are commonly used in business communications are based on time coordinates, duration values, references to occurrences, and on the verb concepts defined in clause 8.2 and clauses 16 and 17. Further discussions of this can be found in Annex C.

Where those business communications are implemented by data exchanges, the terminology used in the formal exchange forms, such as XML, can be derived from the SBVR forms above, as specified in [SBVR] clause XXX, or from the corresponding UML model elements, as specified in [XMI].

The instances of <term>duration value</term> and <term>time coordinate</term>, and of the corresponding UML classes, however, have standard computational representations. The implementations of those concepts are said to be *datatypes*. This clause specifies the datatype representation of duration values and time coordinates in data exchanges.

There are two significantly different standards for the representation of duration values and time coordinates:

- ISO 8601 "Representation of dates and times", which standardizes character string representations
- IETF RFC 5905 "Network Time Protocol", which standardizes binary integer representations

To maximize compatibility with other standards, this specification proposes three compliance points:

- The XML Schema Compliance point requires support for the subset of ISO 8601 representations that is specified in [XML Schema Part 2 Datatypes]. Tools and documents that implement this compliance point can exploit the features of existing XML parsers and generators.
- The ISO 8601 Compliance point requires support for an extended subset of ISO 8601 that is sufficient to cover all of the duration value and time coordinate concepts specified in clauses 9, 11, 12, and 13 of this specification. Tools that implement this compliance level can use standard XML parsers and generators for the datatypes defined by XML Schema, but must implement additional support as described in sub clause 18.1.
- The Internet Time Compliance point requires support for the representations of duration values and time coordinates that are specified in IETF RFC 5905. These forms should be used for time-critical applications in which calculations of durations and comparisons of time coordinates are intrinsic to aspects of the application.

These compliance points are further detailed below. This specification recommends the use of ISO 8601 forms (and related standards) for most business purposes.

## 18.2 Datatype representation of duration values

[ISO 8601] clause 4.4.3 defines a lexical representation for duration values as a component of time intervals. [XML Schema Part 2] defines a datatype named "duration" to represent duration values in XML documents. The XML Schema representation is compatible with ISO 8601 for representing duration values whose time unit is year, month, day, hour, minute, or second, or some combination thereof. ISO 8601 specifies a similar representation for duration values whose time unit is week, but those representations are not permissible values of the XML Schema datatype 'duration'.

### XML Schema Compliance Point

Implementations of the XML Schema Compliance Point shall implement all of the duration value representations that are valid values of the XML Schema datatype 'duration'. The requirement for representations in these forms applies to all exchanges, not just XML-based exchanges.

The XML Schema Part 2 clause 3.2.6.2 "Order Relation on Duration" does not apply to representations of duration values. This specification describes a more comprehensive approach to ordering of duration values based on duration value sets, and mandates that interpretation of ordering for duration values. Therefore, implementations should not rely on standard XML software libraries for the order relation on "duration".

Tools that only implement this compliance point should convert duration values given in weeks to equivalent values given in days.

### ISO 8601 Compliance Point

Implementations of the ISO 8601 Compliance point shall support all valid values of the XML Schema datatype 'duration'. In addition, implementations of this compliance point shall implement representation of duration values that include the time unit 'week' using the general form "PnYnWnDTnHnMnS", where the term "nW" denotes a duration value whose time unit is 'week'. In this representation, the year, day, and time of day components must conform to the rules defined in XML Schema Part 2 clause 3.2.6.1 for number of digits, value range, use of leading minus sign, reduced precision, and truncation. The number of weeks must be greater than 1. If the number of years, days, hours, minutes, or seconds equals zero, the number and corresponding designator may be omitted. Thus, the following examples are all legitimate:

| | |
|---|---|
| P3W | -- three weeks |
| P3W4D | -- three weeks and 4 days |

P1Y3W4D -- 1 year and 3 weeks and 4 days
P1Y3W4DT5H -- 1 year and 3 weeks and 4 days and 5 hours

XML elements that are used to interchange duration values that may include the 'week' time unit should have the "extendedDuration" XML element type defined as:

```
<xs:simpleType name="extendedDuration" >
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:element name="extendedDuration" type="extendedDuration"/>
```

Conforming tools shall accept all 'duration' values as valid values of this "extendedDuration" type. Conforming tools shall also accept the standard XML Schema "duration" datatype as a representation for duration values.

#### Internet Time Compliance Point

Implementations of the Internet Time Compliance Point shall represent all duration values as 64-bit integer multiples of the base time unit for Internet Time (equal to 2^-32 seconds, approximately 200 picoseconds), as specified in IETF RFC 5905. The actual representation of the (nominally binary) integer value depends on the nature of the exchange specification (e.g., JSON vs. XML).

## 18.3 Datatype representation of time coordinates

Table 18.1 below shows all of the time coordinate types that are defined in this specification, and the corresponding time coordinate format specifications from ISO 8601 and XML Schema Part 2 Datatypes. Where both standards specify a representation for the same time coordinate type, the XML Schema form is identical to the ISO 8601 form. The XML Schema forms for the additional time coordinate types it supports are consistent with the overall approach in ISO 8601. In a similar way, this specification mandates the ISO 8601 forms and the XML Schema datatypes that support time coordinates specified herein, and extends the representation set in a way that is consistent with the ISO 8601 approach.

#### XML Schema Compliance Point

Implementations of the XML Schema Compliance Point shall implement all of the time coordinate representations that are valid values of the XML Schema datatypes specified in Table 18.1. The requirement for representations in these forms applies to all exchanges, not just XML-based exchanges.

The XML Schema Part 2 clause 3.2.7.4 "Order Relation on dateTime" does not apply. This specification describes a more comprehensive approach to ordering of time coordinates based on time sets, and mandates that interpretation of ordering for time coordinates. Therefore, implementations should not rely on standard XML software libraries for the order relation on "dateTime".

For tools that conform only to this compliance point, the handling of time coordinates that have no XML Schema form is not specified. No support for such time coordinates is required, although conversion of Gregorian year day coordinates to Gregorian year month day coordinates is recommended.

**Table 18.1 - Relationship between Date-Time time coordinates and standard forms**

| category of time coordinate | ISO 8601 type | XML Schema datatype |
|---|---|---|
| date time | date and time of the day (4.3) | dateTime |
| time of day coordinate | time of the day (4.2 generally) | time |
| Gregorian year month day coordinate | Calendar date (complete representation 4.1.1.1) | date |
| Gregorian year month coordinate | Calendar date (reduced precision 4.1.1.2 a) | gYearMonth |
| Gregorian year coordinate | year (reduced precision 4.1.1.2 b) | gYear |
| Gregorian month day coordinate | Calendar date (truncated representation 4.1.1.3 d) | gMonthDay |
| Gregorian month coordinate | month (truncated representation 4.1.1.3 e) | gMonth |
| Gregorian day of month coordinate | day of the month (truncated representation 4.1.1.3 f) | gDay |
| Gregorian day of year coordinate | day of the year (truncated representation 4.1.3.2 b) | |
| Gregorian year day coordinate | Ordinal date (complete representation 4.1.3.1) | |
| ISO day of week coordinate | week date (truncated representation 4.1.4.3 g) | |
| ISO week of year coordinate | calendar week (truncated representation 4.1.4.3 f) | |
| ISO week day coordinate | week date (truncated representation 4.1.4.3 e) | |
| ISO year week coordinate | week date (reduced precision 4.1.4.2) | |
| ISO year week day coordinate | week date (complete representation 4.1.4.1) | |

## ISO 8601 Compliance Point

Implementations of the ISO 8601 Compliance point shall support all valid values of the XML Schema datatypes that appear in Table 18.1. In addition, implementations of this compliance point shall support the additional representations for the time coordinate types listed below. These additional lexical representations are, or are variants of, the formats already defined in ISO 8601. The design goal is to build upon ISO 8601 in as simple a manner as possible.

Table 18.2 specifies lexical representations for time coordinate types that are not supported by XML Schema datatypes. Several of these representations are specified in ISO 8601, as shown in the table. Tools shall generate and/or accept these representations using the "Extended format" described in ISO 8601.

In the representation formats specified in Table 18.2,

- "yyyy" represents a year number that should have four digits;
- "ddd" is a one- to three-digit number that indicates the day within the year (the 'day of year');
- "W" is the character 'W' – the week designator;
- "ww" is a one- or two-digit number that indicates te ISO week of year;
- "d" is a single-digit that indicates the ISO day of week number (where 1 represents Monday).

**Table 18.2 - Interchange Representations for Time Coordinates**

| time coordinate type | Lexical Representation | Source |
|---|---|---|
| Gregorian year day coordinate | yyyy-ddd | [ISO 8601] clause 4.1.3 |
| ISO year week coordinate | yyyy-Www | [ISO 8601] clause 4.1.4 |
| ISO year week day coordinate | yyyy-Www-d | [ISO 8601] clause 4.1.4 |
| Gregorian day of year coordinate | ----ddd | [ISO 8601] clause 4.1.3 |
| ISO day of week coordinate | W-d | [ISO 8601] clause 4.1.3 |
| ISO week of year coordinate | Www | [ISO 8601] clause 4.1.3 |
| ISO week day coordinate | Www-d | [ISO 8601] clause 4.1.3 |

An ISO day of week coordinate is represented by the week designator "W" (without an ISO week number), followed by one dash, followed by a single ISO day of week number.

An ISO week of year coordinate is represented by the week designator "W", followed by a one- or two-digit ISO week of year number.

An ISO week day coordinate is represented by the week designator "W", followed by a one- or two-digit ISO week of year number, followed by one dash and a single-digit ISO day of week number.

XML elements that are used to interchange time coordinates that may have any of the formats listed in Table 18.2 should have the "extendedDateTime" XML element type defined as:

```
<xs:simpleType name="extendedDateTime" >
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

```
<xs:element name="extendedDateTime" type="extendedDateTime"/>
```

Conforming tools shall accept all 'dateTime' values as valid values of this "`extendedDateTime`" type. Conforming tools shall also accept the standard XML Schema "dateTime" datatypes as representations for the corresponding time coordinate types.

### Internet Time Compliance Point

Implementations of the Internet Time Compliance Point shall represent all absolute time coordinate instances as an amount of time since midnight, January 1, 1900. The amount of time is a duration value and shall be represented in the Internet Time form for duration values (see 1.2).

Internet Time cannot be used to represent any relative time coordinate. Internet Time cannot be used to represent any time point prior to January 1, 1900.

Implementations that support the Internet Time Compliance Point are encouraged to implement one of the other compliance points for more general uses of time coordinates.

# Annex A - Attachments

## (normative)

This annex lists the machine-readable attachments that are included in this specification, and identifies which are normative and which are informative. The latest version of these files can be found at: http://www.omg.org/spec/DTV.

**Table A.1 - Machine-readable Attachments**

| File | Type | Description | Status |
|---|---|---|---|
| **dtv-sbvr.xml** | SBVR XMI | SBVR interchange file derived from the text of this specification | normative |
| **dtv-uml.xml** | UML | UML model of the Date-Time vocabulary, in standard XMI form. Validated by the OMG UML validator. | normative |
| **dtv.ocl** | OCL | OCL constraints stripped out of the text of this specification. The plan is to eventually merge them into the UML model. | normative |
| **dtv.clif** | CLIF | CLIF axioms stripped out of the text of this specification. Consistency checked via the Kojeware CLIF Validation Service at **http://www.kojeware.com/clif-file-validator**. Not yet validated semantically. | normative |
| **dtv-owl.zip** | OWL | OWL models of parts of the specification, a ZIP of separate .owl ontology files. Validated using Pellet. | informative |
| **dtv-md.xml** | XMI | UML model of the Date-Time vocabulary, in MagicDraw native form, with diagrams. | ancillary |

# Annex B - References

## (informative)

The authors reviewed a number of standards documents and academic papers. These are listed here.


Allen — Allen, James, Maintaining Knowledge about Temporal Intervals, Communications of the ACM, Volume 26, Issue 11, November 1983, pp. 832-843, http://portal.acm.org/citation.cfm?id=358434.

Alspaugh — Alspaugh, Thomas, Allen's interval algebra, February 14 2008, http://www.ics.uci.edu/~alspaugh/foundations/allen.html.

BPDM — Object Management Group (OMG), Business Process Definition MetaModel, Beta 2, December 2007, http://www.omg.org/cgi-bin/doc?dtc/2007-12-05.

Casati — Casati, Roberto and Varzi, Achille C., *Parts and Places*, MIT Press, 1999.

Davidson — Davidson, Donald, The Logical Form of Action Sentences. In: Nicholas Rescher (ed.), *The Logic of Decision and Action*, Pittsburgh: The University Press, pp. 81-95. (1967). Cited in [Kamp, Reyle].

Enhanced Time — Object Management Group (OMG), Enhanced View of Time Specification, Version 2.0, January 2008, http://www.omg.org/technology/documents/formal/enhanced_time_service.htm.

Halpin 2007 — Halpin, Terry, Temporal Modeling, four-part series:

- Part 1, February 2007, http://www.brcommunity.com/b332.php.
- Part 2, June 2007, http://www.brcommunity.com/b351.php.
- Part 3, November 2007, http://www.brcommunity.com/b374.php.
- Part 4, April 2008, http://www.brcommunity.com/b411.php.

Halpin 2008 — Halpin, Terry, OWL Time, 2008, presentation received in private communication.

Haley — Haley, Paul, Understanding events and processes takes time, February 19 2008, http://haleyai.com/wordpress/2008/02/19/understanding-events-and-processes-takes-time/.

Hayes — Hayes, Pat, A Catalog of Temporal Theories, University of Illinois Technical Report UIUC-BI-AI-96-01, 1995-1996, http://www.ihmc.us/users/phayes/docs/TimeCat96.pdf.

Hobbs 2004 — Hobbs, Jerry, *An OWL Ontology of Time*, July 2004, http://www.isi.edu/~hobbs/time/owl-time-july04.txt.

Hobbs 2008 — Hobbs, Jerry, *Time Representation*, email on Ontolog-Forum, January 21, 2008, http://ontolog.cim3.net/forum/ontolog-forum/2008-01/msg00336.html.

iCalendar — Internet Engineering Task Force (IETF), Internet Calendaring and Scheduling Core Object Specification, RFC 2445, November 1998, http://www.ietf.org/rfc/rfc2445.txt.

IEC 60050-111 — International Electrotechnical Committee, *International Electrotechnical Vocabulary - Chapter 111: Physics and chemistry*, Edition 2.0, 1996-07, Available from IEC website.

IERS — International Earth Rotation and Reference Service. See www.iers.org

| | |
|---|---|
| IKL Guide | Hayes, Pat, Florida Institute for Human and Machine Cognition, IKL Guide. Available at www.ihmc.us/users/phayes/ikl/guide/guide.html. |
| Inter Gravissimas | Pope Gregory XIII, *Inter Gravissimas*, papal bull issued 24 February 1582, prepared in English, Latin, and French by R.T.Crowley for ISO TC154 on 27 December 2002. |
| International Meridian Conference | International Conference Held at Washington for the Purpose of Fixing a Prime Meridian and a Universal Day, October, 1884. Protocols of the Proceedings available at http://www.gutenberg.org/etext/17759. |
| International Time | Object Management Group (OMG), Internationalization, Time Operations, and Related Facilities, Version 1.0, January 2000, http://www.omg.org:80/technology/documents/formal/internationalization_and_time.htm. |
| ISO 31-1 | International Standards Organization (ISO), *Quantities and units – Part 1:Space and Time.* Replaced by ISO/IEC 80000-3:2007. |
| ISO 8601 | International Standards Organization (ISO), Data elements and interchange formats – Information interchange – Representation of Dates and Times, Third edition, December 1, 2004, http://www.iso.org/iso/catalogue_detail?csnumber=40874. |
| ISO 18026 | International Standards Organization (ISO), *Information technology — Spatial Reference Model (SRM) Information technology — Spatial Reference Model (SRM)*, 2009. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=54166 |
| ISO 18629 | International Standards Organization (ISO), Industrial Automation Systems and Integration – Process Specification Language (PSL), 2004, http://www.iso.org:80/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35431. |
| ISO 24617-1 | International Standards Organization (ISO), Language resource management – Semantic annotation framework (SemAF) – Part 1: Time and events – Committee Draft, August 4, 2008, http://www.tc37sc4.org/new_doc/iso_tc37_sc4_n269_ver10_wg2_24617-1_semaf-time_utf8.pdf. |
| ISO/IEC 80000-3 | International Standards Organization (ISO), Quantities and units -- Part 3: Space and time, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=31888 |
| Kamp, Reyle | Kamp, Hans and Reyle, Uwe, *From Discourse to Logic*, Kluwer Academic Publishers (1993). |
| KnowGravity | KnowGravity Inc., *CASSANDRA/xUML User's Guide*, April 2008. |
| Lee | Lee, Kiyong et. al., ISO-TimeML and its Applications, August 24, 2007, http://www.tc37sc4.org/new_doc/iso_tc37_sc4_N385_wg2_iso-timeml_provo2007_beamer_utf8.pdf. |
| MARTE | Object Management Group (OMG), UML Profile for Modeling and Analysis of Real-time and Embedded Systems, Beta 1, August 2007, *http://www.omg.org:80/cgi-bin/doc?ptc/2007-08-04*. |
| Menzels | Menzels, Chris, *Actualism*, article in the *Stanford Encyclopedia of Philosophy*, December 8, 2008. |
| Mueller | Mueller, Erik T., *Common Sense Reasoning*, Morgan Kaufmann, 2006, ISBN 978-0-12-369388-4. |
| NTP | Internet Engineering Task Force, RFC 1305 *Network Time Protocol* (Version 3) http://tools.ietf.org/pdf/rfc1305. (RFC 5905 is a proposed update of RFC 1305.) |

| | |
|---|---|
| OWL Time | World Wide Web Consortium (W3C), An OWL Ontology of Time, 27 September 2006, http://www.w3.org/TR/owl-time/. |
| OWL Time Home | *OWL Time (formerly DAML-Time)*, " home page", http://www.isi.edu/~hobbs/owl-time.html. |
| Pan | Pan, Feng, Representing Complex Temporal Phenomena for the Semantic Web and Natural Language, PhD Thesis, 2007, http://www.isi.edu/~hobbs/time/pub/pan-phdthesis.pdf. |
| Parsons | Parsons, Terence, Events in the Semantics of English: a study in subatomic semantics, MIT Press, 1990, ISBN 0-262-16120-6, http://www.humnet.ucla.edu/humnet/phil/faculty/tparsons/Event%20Semantics/download.htm |
| QUDV | Object Management Group (OMG), *SysML 1.2 Annex C.5, Quantities, Units, Dimensions, Values*, http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-qudv:quantities_units_dimensions_values_qudv |
| QUOMOS | OASIS *Quantities and Units of Measure Ontology Standard* (QUOMOS) Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=quomos |
| SBVR | Object Management Group (OMG), *Semantics of Business Vocabulary and Business Rules* (SBVR), v1.0, January 2008, OMG document formal/08-01-02, http://www.omg.org/spec/SBVR/1.0/. |
| Schedulability | Object Management Group (OMG), UML Profile for Schedulability, Performance, and Time Specification, Version 1, January 2005, http://www.omg.org:80/technology/documents/formal/schedulability.htm. |
| SI | Bureau International des Poids et Mesures (BIPM), *The International System of Units*, 8th edition, 2006, http://www.bipm.org/utils/common/pdf/si_brochure_8.pdf. |
| Simons | Simons, Peter, *Parts: A Study in Ontology*, Oxford University Press, 1987. |
| SysML | Object Management Group (OMG), Systems Modeling Language V1.0, September 2007, http://www.omg.org:80/technology/documents/formal/sysml.htm. |
| TimeML | TimeML Working Group, Semantic Annotation: A TimeMLCase StudyISO/, January 8, 2007, http://www.tc37sc4.org/new_doc/ISO_TC37_SC4_N337_WG2_ISO-TimeML_Tilburg2007.pdf. |
| Time Services | Object Management Group (OMG), Time Service Specification, V1.1, May 2002, http://www.omg.org:80/technology/documents/formal/time_service.htm. |
| UML Time | Object Management Group (OMG), CommonBehaviors::SimpleTime package of *UML SuperStructure*, V2.1.2, November 2007, pp , http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF. |
| VIM | International Standards Organization/International Electrotechnical Commission (ISO/IEC), International Vocabulary for Metrology – Basic and General Concepts and Associated Terms (VIM), 3rd edition, JCGM 200:2008 http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2008.pdf |
| XML Schema | World Wide Web Consortium (W3C) Recommendation, XML Schema Part 2: Datatypes Second Edition, 28 October 2004, http://www.w3.org/TR/xmlschema-2/. |

Zoneinfo Olson, Ted and the International Assigned Numbers Authority (IANA), *Time Zone Database*, available at http://www.iana.org/time-zones

# Annex C - Business Usage Guidelines

(informative)

Annex C is now published as a separate document:
http://www.omg.org/spec/DTV/20150301/dtv-guidelines.pdf.

# Annex D - Fundamental Concepts

(normative)

## D.1 General

International standards, for example [VIM], [ISO 80000:3], and [ISO 18026] define duration as just one of many quantity kinds, and time scales as one of many kinds of coordinate systems. This permits the formation of derived quantities based on durations (e.g., velocity, which is length / duration), and multi-dimensional coordinate systems that include time as one dimension. Coordinate systems themselves depend upon mathematical concepts, such as sequences. The axioms related to time intervals depend upon mereology concepts.

Unfortunately, there is no existing SBVR vocabulary or ODM ontology that addresses these concepts. The authors recognize that they are out-of-scope for this specification, but felt it necessary to imagine how this Date-Time Vocabulary would fit into a complete schema that addresses them. Annex D summarizes that schema in the form of several SBVR vocabularies.

There are a few existing OMG efforts covering this topic that are referenced in Annex B. The most recent of these is [QUDV], but it models the concept 'quantity' differently than here because of limitations of UML and SysML. In particular, QUDV does not model the distinction between 'quantity' and 'quantity value.'

There is one external group [QUOMOS] that is working in this area, and that is proposed as an OASIS Technical Committee effort called "Quantity and Unit of Measure Ontology Standard (QUOMOS)." As and when [QUOMOS] reaches completion, the contents of this section should be reviewed for possible alignment with [QUOMOS].

Subclauses D.2 "Sequences" and D.4 "Mereology" are complete and consistent models of their topics and are normative.

Subclause D.3 "Quantities Vocabulary" is informative because it addresses only the aspects of quantities and units of measure that are required by the Date-Time Vocabulary, and because the other groups mentioned above have the charter to fully address the topic.

## D.2 Sequences (normative)

The 'sequence' concept models ordered collections of things in which the things are ordered by assigning numbers (indices) to them within the collection, as distinct from any particular properties of the things themselves. The model does not preclude the use of properties in creating indices, and it does not require the indices to be consecutive in the general case.

Regular sequences provide the mathematical foundation of time scales.

There are two somewhat different models of sequence that are in common use. Using UML terminology, we may call them the "composite model" and the "aggregation model." In the composite model, the existence and conceptualization of the members is dependent on the existence and conceptualization of the sequence. In these sequences, the index of a member is intrinsic to the member – its meaning is bound up with its position in the sequence. This is the case with time concepts like months of year or hours of day: 2:00 is the hour of day that occurs immediately after 1:00; its definition depends on the sequence.

In the aggregation model, the members of the sequence have independent existence, with intrinsic properties that are independent of the sequence. The sequence conceptualizes (and imposes) an ordering on the members that is not intrinsic to the members themselves. In these sequences, the indices of the members are extrinsic – the member acquires the index by being included in the sequence, and it can have other indices in other sequences. In some such sequences, a given member can occur more than once. A common example is a list of authorized suppliers in order of preference or total order volume. Similarly, time intervals exist without clocks, and although they are intrinsically ordered, they only acquire indices when we impose a standard clock and a time offset on them.

The model presented here is general enough to support both models, but each actual sequence will use it differently, depending on the nature of its members. The model below distinguishes between the things that are by definition elements of the sequence – the sequence positions – and things that exist independently and are ordered by the sequence – the members. Time scales, such as clocks and calendars, are defined to be sequences whose members are time points, such as 'hour of day'. The sequence positions of each time scale (sequence) have indices that are used to number the time points that are their members. The application of time scales to the Time Axis, causes the assignment of time intervals as instances of the time points. Thus, one time interval in each day is an instance of the 'hour of day' with index 12, i.e., 12 o'clock.

---

## Sequences Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Included Vocabulary: | SBVR-DTV Vocabulary |
| Language: | English |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#SequencesVocabulary |

---

### D.2.1 General Sequence Concepts

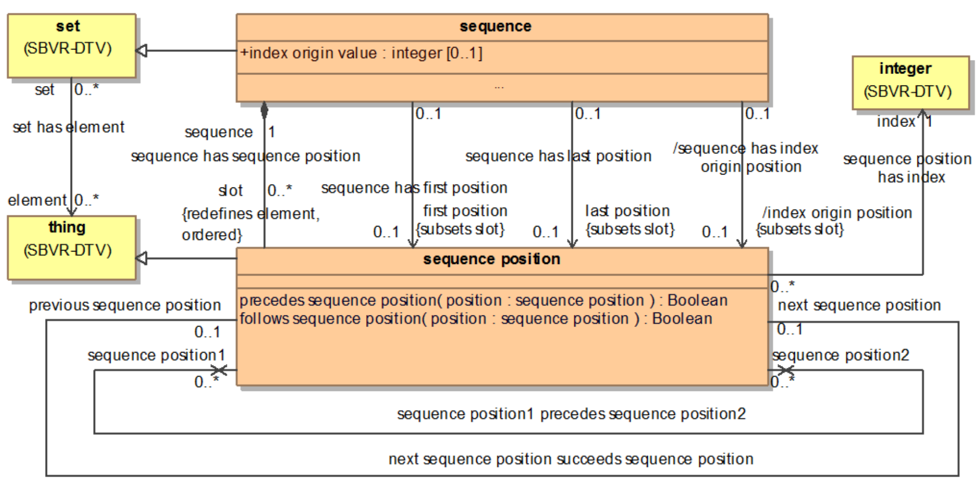


**Figure D.1 – Sequences**

## sequence

| | |
|---|---|
| Definition: | set whose elements are ordered by their indices |
| Note: | "Sequence" is a conceptual mechanism for ordering things. A sequence is made up of sequence positions (slots), each of which may have a member. These members are the things that *participate* in the sequence. For convenience, the things that are the members of the sequence positions of a sequence are also called the members of the sequence. |

| | |
|---|---|
| Note: | In the general case, a given thing may *participate* in a sequence more than once, i.e., as more than one member of the same sequence. See 'regular sequence' for a kind of sequence where a thing is constrained to *participate* at most once in the sequence. |
| Note: | Each sequence defines an ordering on its sequence positions, by assigning an integer index to each sequence position, and using the ordering of the integers to order the members. The index assignment may be based on some natural characteristics of the members, or it may be just sequential position numbers, or it may be some other numbering scheme associated with the meaning of the sequence. In general, the index assignments need not reflect any natural ordering of the members. That is, the ordering of the members of a sequence can be specific to the sequence concept. |

## sequence position

| | |
|---|---|
| Synonym: | slot |
| Definition: | element of a given sequence |
| Note: | A sequence is a set of sequence positions. Each sequence position is an element of the sequence that defines it, and no other. |
| Note: | Each sequence position has an integer index associated with it. The ordering on the sequence is induced on it by the natural ordering of the integers. |

## sequence *has* sequence position

Synonymous Form: sequence position *in* sequence

Necessity: Each sequence position *is of* exactly one sequence.

CLIF Axiom:

```
(forall (seq sp)
  (if ("sequence has sequence position" seq sp)
    (and
      (sequence seq)
      ("sequence position" sp)
      (forall (seq2)
        (if ("sequence has sequence position" seq2 sp)
            (= seq2 seq))) )))
```

Possibility: Some sequence *has* no sequence positions.

Note: This verb concept is a specialization of SBVR's 'thing *is in* set.'

## index

| | |
|---|---|
| Synonym: | indices |
| Concept Type: | role |
| General Concept: | integer |
| Note: | The basis for assigning a particular index to a given sequence position might be a characteristic of the member *of* the sequence position (such as weight, etc.). This technique would order the members by weight, or inversely by weight, depending on the index assignments. |
| Note: | Negative indices are meaningful for time scales of years that extend before year zero. |

## sequence position *has* index

| | |
|---|---|
| Synonymous Form: | index *indexes* sequence position |
| Definition: | the index is assigned to the sequence position and is used in ordering the sequence positions in the sequence |
| Definition: | Each sequence position *has* exactly one index. |
| Necessity: | If the index$_1$ *of* some sequence position$_1$ *of* some sequence *equals* the index$_2$ *of* some sequence position$_2$ *of* the sequence then sequence position$_1$ *is* sequence position$_2$. |

CLIF Axiom:
```
(forall (seq sp1 sp2 x1 x2)
  (if
    (and
      ("sequence has sequence position" seq sp1)
      ("sequence has sequence position" seq sp2)
      ("sequence position has index" sp1 x1)
      ("sequence position has index" sp2 x2)
      (= x1 x2) )
   (= sp1 sp2) ))
```

CLIF Axiom:
```
(forall ((sp "sequence position"))
  (exists ((x1 "integer"))
    (and
      ("sequence position has index" sp x1)
      (forall (x2)
        (if
          ("sequence position has index" sp x2)
          (= x1 x2)) ) )))
```

OCL Constraint:
```
context sequence:
    inv: self._'sequence position'->forAll(sp1 |
         self._'sequence position'->forAll(sp2 |
          indexOf(sp1) = indexOf(sp2) implies sp1 = sp2))
```

## sequence position$_1$ *precedes* sequence position$_2$

Synonymous Form: sequence position$_2$ *follows* sequence position$_1$

Definition: the index *of* sequence position$_1$ *is less than* the index *of* sequence position$_2$ .

CLIF Definition:
```
(forall (sp1 sp2 x1 x2)
  (if
    (and
      ("sequence position has index" sp1 x1)
      ("sequence position has index" sp2 x2) )
    (iff ("sequence position1 precedes sequence position2"
              sp1 sp2)
         (exists ((seq sequence))
           (and
             ("sequence has sequence position" seq sp1)
             ("sequence has sequence position" seq sp2)
             (< x1 x2) )) )))
```

Note: This is the ordering relation on the sequence positions.

## next sequence position

Definition: sequence position that *succeeds* a given sequence position

General Concept: sequence position

Concept Type: role

Note: In a finite sequence, the last position does not have a next sequence position.

## next sequence position *succeeds* sequence position

Synonymous Form: next sequence position *is next after* sequence position

Synonymous Form: sequence position *is just before* next sequence position

Synonymous Form: sequence position *has* next sequence position

Definition: next sequence position *follows* sequence position and the index *of* next sequence position *is less than or equal to* the index *of* each sequence position$_2$ that *follows* sequence position.

Necessity: Each sequence position *has* at most one next sequence position.

CLIF Definition:

```
(forall (sp nsp)
  (iff ("next sequence position succeeds sequence position"
            nsp sp)
    (and
      ("sequence position1 precedes sequence position2"
            sp nsp)
      (not (exists (sp2)
        (and
          ("sequence position1 precedes sequence position2"
            sp sp2)
          ("sequence position1 precedes sequence position2"
            sp2 nsp)) )) )))
```

OCL Definition:

```
context _'sequence position'
inv: self._'sequence position1 precedes sequence position2'
    ( self._'next sequence position')
   and self._'sequence position2'->forAll(sp2 |
      self._'next sequence position'.index <= sp2.index)
```

### first position

Concept Type: role
General Concept: sequence position

### sequence *has* first position

Definition: the index *of* the first position *is less than or equal to* the index *of* each sequence position *in* the sequence
Necessity: Each sequence *has* at most one first position.
Possibility: A sequence *has* no first position.
Necessity: No sequence position *precedes* the first position *of* each sequence.

### last position

Concept Type: role
General Concept: sequence position

### sequence *has* last position

Definition: the index *of* the last position *is greater than or equal to* the index *of* each sequence position *in* the sequence
Necessity: Each sequence *has* at most one last position.
Possibility: A sequence *has* no last position.
Necessity: No sequence position *succeeds* the last position *of* each sequence.

## D.2.2 Sequence Members

This sub clause extends the sequence model to accommodate situations in which the sequence position itself is artificial – it represents the role of some thing that exists independently from the sequence.

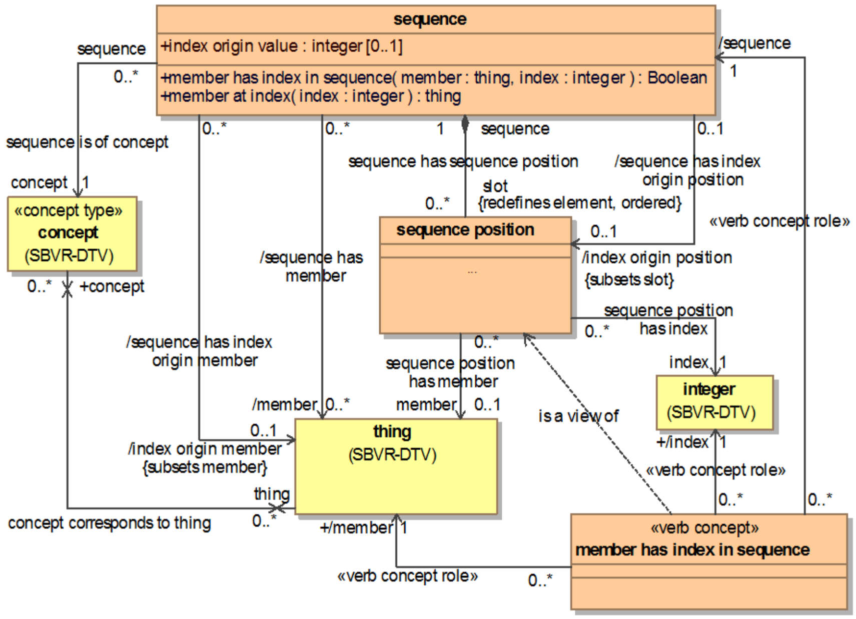


**Figure D.2 - Sequence Members**

## member

Concept Type: role
General Concept: thing
Definition: thing that *is in* a given sequence position, and by extension, any thing that participates in a given sequence

## sequence position *has* member

Synonymous Form: slot *contains* member
Synonymous Form: member *is in* sequence position
Synonymous Form: member *in* sequence position
Necessity: Each sequence position *has* at most one member.
Possibility: A sequence position *has* no member.
Possibility: Each thing *is* the member *of* zero or more sequence positions *in* zero or more sequences.
Note: For some sequences, the sequence positions have meaning in their own right, and may or may not have members. For example, the meaning of a scale point is a quantity.

## member *participates in* sequence

Synonymous Form: sequence *has* member
Synonymous Form: member *of* sequence
Synonymous Form: member *in* sequence
Definition: the member *is* the member *of* a sequence position *of* the sequence

CLIF Definition:
```
(forall ((s sequence) (member thing))
  (iff ("sequence has member" s member)
      (exists ((sp "sequence position"))
       (and ("sequence has sequence position" s sp)
           ("sequence position has member" sp m)))))
```
OCL Definition:
```
context sequence
def: _'member participates in sequence'
    (member: thing, s: sequence)
    : Boolean =
    self._'sequence position'->exists(sp |
     sp.member = member)
```
Note: Things are assigned as members of a sequence to induce a desired ordering relation among the things. Thus, a given set *of* things may be ordered differently in different sequences by their weight, height, arrival time in a queue, service priority, etc.

## member *has* index *in* sequence

Synonymous Form: sequence *has* member *with* index

Definition: The sequence *has* a sequence position that *has* an index that *equals* the index, and the sequence position *has* a member that *is* the member.

CLIF Definition:
```
(forall (member index s)
 (iff ("member has index in sequence" member index s)
  (and
     (sequence s) (integer index)
     (exists (sp)
      (and
        ("sequence has sequence position" s sp)
        ("sequence position has index" sp index)
        ("sequence position has member" sp member)) ))))
```
OCL Definition:
```
context sequence
def: _'member has index in sequence'
    (member: thing, i: integer, s: sequence)
    : Boolean =
    self._'sequence position'->exists(sp |
      sp.index = index and sp.member = member)
```
Note: This verb concept states that in a given sequence the position that is given by the index is occupied by the member. A given thing can have zero, one, or more than one indices in a given sequence.

Possibility: A thing *has* more than one index *in* the same sequence.

Note: The primary verb concept wording and the synonymous form given above are "sentential forms". Following the conventions described in clause 6, the corresponding CLIF predicate and OCL operation yield a Boolean result. In addition, this verb concept has a "noun form" (member *with* index *in* sequence), for which the corresponding CLIF and OCL functions return the thing that plays the member role in the relationship.

Synonymous Form: member *with* index *in* sequence

Note: The Synonymous Form given above is an SBVR "noun form" that yields a member given an index and a sequence.

CLIF Definition:
```
(forall (member index s)
 (iff (= member ("member with index in sequence" index s))
      ("member has index in sequence" member index s) ))
```
OCL Definition:
```
context sequence
def: _'member with index in sequence'
                 (index: integer, s sequence): thing =
    self._'sequence position'->select(sp | sp.index = index).member
```

## sequence *is of* concept

Definition: the concept *corresponds to* each member *of* the sequence

CLIF Definition:
```
(forall (s c)
  (iff ("sequence is of concept" s c)
    (and
      (sequence s) (concept c)
      (forall (member)
        (if ("member participates in sequence" member s)
            ("meaning corresponds to thing" c member)))
    )))
```

OCL Definition:
```
context sequence
def: _'sequence is of concept'(c: concept)
    : Boolean =
    sequence._'sequence position'.member->forAll(m |
            'concept corresponds to instance'(c m))
```

Necessity: Each sequence *is of* at least one concept.

Note: Constraints based on the verb concept ' sequence *is of* concept' limit each member to be an instance *of* the concept. If more than one such constraint is stated for the same sequence, every member must satisfy all such constraints.

Note: Such constraints can be relaxed as needed by specifying that a sequence *is of* any convenient more general concept of the members of the sequence. Since the concept 'thing' is a more general concept of all other object types, a sequence that '*is of* thing' permits members of any type.

### D.2.3 Index Origin

## index origin member

Concept Type: role

Definition: member that is assigned the index that *is* the index origin value

Note: This is a primitive definition. Either 'index origin member' or 'index origin value' must be defined in terms of the other.

Note: For sequences that have a first member, the first member is usually designated as the index origin member. In a sequence that has no first member, the index origin member is usually determined by association to some real world event or property.

Example: The member with index 1875 (the year of the Convention du Mètre) is the index origin member of the Gregorian years scale.

## index origin value

Concept Type: role

General Concept: integer

Note: The index origin value is most commonly either 0 or 1.

Example: The first member of time scales of hours, minutes, and seconds has index origin value 0 because these are counted from 0 by convention.

Example: The first member of time scales of years, months, weeks, and days has index origin value 1 because these are counted from 1 by convention.

## index origin position

Concept Type: role

General Concept: sequence position

### sequence *has* index origin member

Definition: The index origin member *of* the sequence *is* the member *of* the index origin position *of* the sequence.

Necessity: Each sequence *has* at most one index origin member.

CLIF Axiom:
```
(forall (seq) (forall (iom)
(if ("sequence has index origin member" seq iom)
    (and
      (forall (m)
        (if ("sequence has index origin member" seq m)
            (= m iom) )) ))))
```

OCL Constraint:
```
context sequence
inv: sizeOf(self._'index origin member') <= 1
```

### sequence *has* index origin value

Necessity: Each sequence *has* at most one index origin value.

CLIF Axiom:
```
(forall (seq) (forall (iov)
  (if ("sequence has index origin value" seq iov)
     (and
        (integer iov)
        (forall (iv2)
          (if ("sequence has index origin value"
               seq iv2)
               (= iov iv2)) )))))
```

OCL Constraint:
```
context sequence
inv: sizeOf(self._'index origin value') <= 1
```

### sequence *has* index origin position

Necessity: Each sequence *has* at most one index origin position.

Necessity: The index *of* the index origin position *equals* the index origin value *of* the sequence.

CLIF Axiom:
```
(forall (s p)
 (iff ("sequence has index origin position" s p)
    (exists (iov)
      (and
       ("sequence has index origin value" s iov)
       ("sequence position has index" p iov) ))))
```

OCL Constraint:
```
context sequence
      inv: sequence._'index origin value' =
            sequence._'index origin position'.index
```

### D.2.4 Kinds of Sequences

This clause defines various sequence types in order to clarify the distinctions among and meaning of each type.

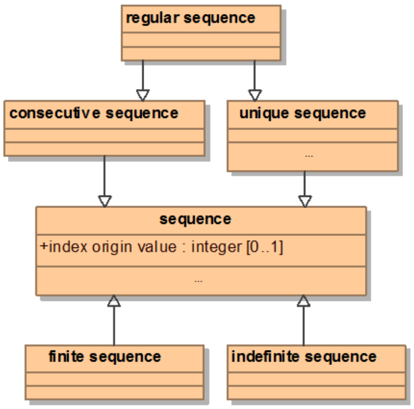


**Figure D.3 - Kinds of Sequences**

## consecutive sequence

Definition: sequence that each sequence position *of* the sequence that *is* not the first position *of* the sequence *is next after* a sequence position$_2$, and the index *of* the sequence position *equals* 1 plus the index *of* the sequence position$_2$

Description: A consecutive sequence is a sequence in which consecutive sequence positions have consecutive indices.

CLIF Definition:

```
(forall (s) (iff ("consecutive sequence" s)
    (and
     (sequence s)
     (forall (sp1 sp2 x1 x2)
      (if
       (and
        ("sequence has sequence position" s sp1)
        ("next sequence position succeeds sequence position"
            sp2 sp1)
        ("sequence position has index" sp1 x1)
        ("sequence position has index" sp2 x2))
       (= x2 (+ x1 1)) ))
   )))
```

OCL Definition:

```
context _'consecutive sequence'
inv: self._'sequence position'->forAll(sp |
    not (self._'first position'->exists()
     and sp = self._'first position')
     implies self._'sequence position'.indexOf(sp) =
       1 + self._'sequence position'
         ->indexOf(sp._'previous sequence position'))
```

Note: In a consecutive sequence, the indices of the members are consecutive integers.

## unique sequence

Definition: sequence that *has* no member that *is* the member *of* more than one sequence position *of* the sequence

CLIF Definition:
```
(forall (s)
 (iff ("unique sequence" s)
   (and
     (sequence s)
     (forall (sp1 sp2 t1 t2)
       (if
          (and
            ("sequence has sequence position" s sp1)
            ("sequence has sequence position" s sp2)
            ("sequence position has member" sp1 t1)
            ("sequence position has member" sp2 t2)
            (not (= sp1 sp2)) )
         (not (= t1 t2)) )))))
```
OCL Definition:
```
context sequence
inv: self.member->forall(m |
      self._'sequence position'.member->isUnique (m2 | m = m2))
```
Necessity: Each thing *has* at most one index *in* each unique sequence.

CLIF Axiom:
```
(forall (thing (x1 integer) (x2 integer) (us "unique sequence"))
   (if (and
          ("member has index in sequence" thing x1 us)
          ("member has index in sequence" thing x2 us) )
       (= x1 x2) ))
```
OCL Constraint:
```
context _'unique sequence'
      ???
```

## regular sequence

Definition: consecutive sequence that *is* a unique sequence

CLIF Definition:
```
(forall (s)
 (iff ("regular sequence" s)
  (and
    ("consecutive sequence" s)
    ("unique sequence" s))))
```
Note: Regular sequences are the basis of scales (clause D.3).

## finite sequence

Definition: sequence that *has* a cardinality

## indefinite sequence

Definition: sequence that *does* not *have* a cardinality

Note: This definition relies on the fact that 'set *has* cardinality' (in MRV) has the Necessity "Each set *has* at most one cardinality." An indefinite sequence has an unknown or unspecified number of elements, hence it does not have a 'cardinality'.

Note: 'Finite sequence' is used in this specification as the basis 'finite time scales', such as the 'Gregorian year of months scale'. 'Indefinite time sequence' is the basis of 'indefinite time scales' such as the 'Gregorian years scale'. The key distinction is that finite time scales have a specified number of sequence positions, whereas the number of sequence positions of indefinite time scales is not known.

Note: Different scientific, religious, and cultural traditions have varying views as to whether there is a first or last calendar year. This specification avoids taking a position about that by using the term 'indefinite sequence' rather than 'infinite sequence'.

### D.2.5 Sequence Member Relationships

The following concepts are relationships that a sequence imposes on its members.

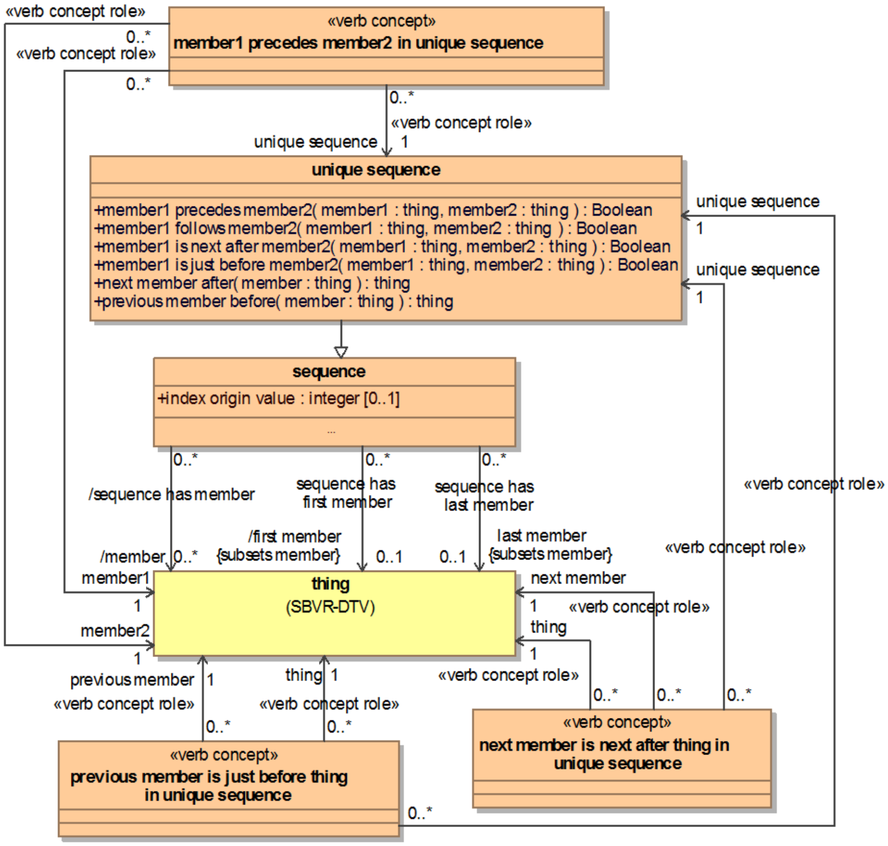


**Figure D.4 - Sequence Member Relationships**

## member$_1$ *precedes* member$_2$ *in* unique sequence

Synonymous Form: member$_2$ *follows* member$_1$ *in* unique sequence

Definition: member$_1$ *participates in* the unique sequence and member$_2$ *participates in* the unique sequence and each sequence position$_1$ *of* member$_1$ *in* the unique sequence *precedes* each sequence position$_2$ *of* member$_2$ *in* the unique sequence

CLIF Definition:

```
(forall ((s sequence)(m1 thing)(m2 thing))
 (iff ("member1 precedes member2 in unique sequence"
      m1 m2 s)
     (and
       ("member participates in sequence" m1 s)
       ("member participates in sequence" m2 s)
       (forall ((sp1 "sequence position")
               (sp2 "sequence position"))
         (if
           (and
             ("sequence position of member" sp1 m1)
             ("sequence position of member" sp2 m2))
          ("sequence position1 precedes
            sequence position2" sp1 sp2))))))
```

OCL Definition:

```
context sequence
def: _'member1 precedes member2 in unique sequence'
    (m1: thing, m2: thing) : Boolean =
   self.member->includes(m1)
     and self.member->includes(m2)
     and m1._'sequence position'->forall(sp1 |
          m2._'sequence position'->forall(sp2 |
           'sequence position1 precedes
              sequence position2'(sp1, sp2)))
```

## first member

Synonym: first
Concept Type: role
General Concept: thing
Definition: the member *of* the first position *of* a given sequence
Necessity: The concept 'first member' *specializes* the concept 'member'.

## sequence *has* first member

Definition: the first member *is the* member *of* the first position *of* the sequence

CLIF Definition:

```
(forall (s m)
 (iff ("sequence has first member" s m)
  (exists (first)
   (and
    ("sequence has first position" s first)
    ("sequence position has member" first m)))))
```

OCL Definition:

```
context _'sequence'
def: _'sequence has first member'
     (s: _'sequence') : thing = self._'first position'.member
```

Necessity: A sequence *has* at most one first member.
Note: An indefinite sequence has no first member or no last member.

## last member

Concept Type: role
General Concept: thing
Definition: the member *of* the last position *of* a given sequence
Necessity: The concept 'last member' *specializes* the concept 'member'.

## sequence *has* last member

Definition: the last member *is the* member *of* the last position *of* the sequence

CLIF Definition:
```
(forall (s m)
 (iff ("sequence has last member" s m)
  (exists (last)
   (and
    ("sequence has last position" s last)
    ("sequence position has member" last m)))))
```

OCL Definition:
```
context _'sequence'
def: _'sequence has last member
    (s: sequence) : thing = self._'last position'.member
```

Necessity: A sequence *has* at most one last member.

Note: An indefinite sequence has no first member or no last member.

## next member

Concept Type: role

General Concept: thing

## next member *is next after* thing *in* unique sequence

Synonymous Form: thing *has* next member *in* unique sequence

Synonymous Form: next member *after* thing *in* unique sequence

Definition: thing *is* the member *of* exactly one sequence position *in* the unique sequence and next member *is* the member *of* some sequence position *of* the unique sequence that *succeeds* the sequence position *of* the thing

CLIF Definition:
```
(forall (s nm m)
 (iff
  ("next member is next after thing in unique sequence" nm m s)
  (and
   ("unique sequence" s)
   (exists (sp nsp)
    (and
     ("sequence has sequence position" s sp)
     ("sequence position has member" sp m)
     ("next sequence position succeeds sequence position" nsp sp)
     ("sequence position has member" nsp nm) )))))
```

OCL Definition:
```
context 'unique sequence'
def: 'next member is next after thing in unique sequence'
    (nm: thing, m: thing) : Boolean =
    self._'sequence position'.member->count(m) = 1
    and self._'sequence position'->select(member = m).
      member._'next sequence position'.member = nm
```

Note: This fact type is meaningless if the thing does not appear in the unique sequence.

Necessity: The last member *of* each sequence *has* no next member.

CLIF Axiom:
```
(forall (s last)
 (if
  ("sequence has last member" s last)
  (not (exists (m)
   ("member is next after thing in sequence" m last s) ))))
```

OCL Constraint:
```
context sequence
    inv: self._'last member'->exists() implies
        not self._'last member'._'next member'->exists()
```

Necessity: Each member that *participates in* a unique sequence and that *has* no next member *in* the unique sequence *is* the last member *of* the unique sequence.

CLIF Axiom:
```
(forall (s m)
 (if
  (and
   ("sequence has member" s m)
   (not (exists (nm)
     ("next member is next after thing in sequence" nm m s) )))
  ("sequence has last member" s m)))
```
OCL Constraint:
```
context _'unique sequence'
    inv: self.member->forAll(m |
        not m._'next member'->exists()
            implies m = self._'last member')
```

## previous member

Concept Type: role
General Concept: thing

## previous member *is just before* thing *in* unique sequence

Synonymous Form: thing *has* previous member *in* unique sequence
Definition: thing *is* the member *of* exactly one sequence position *in* the unique sequence and previous member *is* the member *of* some sequence position *of* the unique sequence that *is just before* the sequence position *of* the thing
CLIF Definition:
```
(forall (s pm m)
  (iff
  ("previous member is just before thing in unique sequence"
              pm m s)
  (and
   ("unique sequence" s)
   (exists (sp psp)
    (and
     ("sequence has sequence position" s sp)
     ("sequence position has member" sp m)
     ("next sequence position succeeds sequence position"
              sp psp)
     ("sequence position has member" psp pm) )))))
```
OCL Definition:
```
context _'unique sequence'
def: _'previous member is just before member in unique
     sequence'(pm: thing, m: thing) : Boolean =
     self._'sequence position'.member->count(m) = 1
     and self._'sequence position'->select(member = m).
       'previous sequence position'.member = pm
```
Note: This fact type is meaningless if the thing does not appear in the unique sequence.
Necessity: The first member *of* each sequence *has* no previous member.
CLIF Axiom:
```
(forall (s first)
 (if
  ("sequence has first member" s first)
  (not (exists (m)
   ("previous member is just before thing in sequence" m first s) ))))
```
OCL Constraint:
```
context sequence
    inv: self._'first member'->exists() implies
        not self._'first member'._'previous member'->exists()
```
Necessity: Each member that *participates in* a unique sequence and that *has* no previous member *in* the unique sequence *is* the first member *of* the unique sequence.

CLIF Axiom:
```
(forall (s m)
 (if
  (and
   ("sequence has member" s m)
   (not (exists (nm)
    ("previous member is just before thing in sequence" nm m s)
   )))
  ("sequence has first member" s m)))
```
OCL Constraint:
```
context _'unique sequence'
    inv: self.member->forAll(m |
        not m._'previous member'->exists()
            implies m = self._'first member')
```

### D.2.6 Ordinals

These terms for ordinal numbers build on the definitions of 'unique sequence' and 'first member' above.

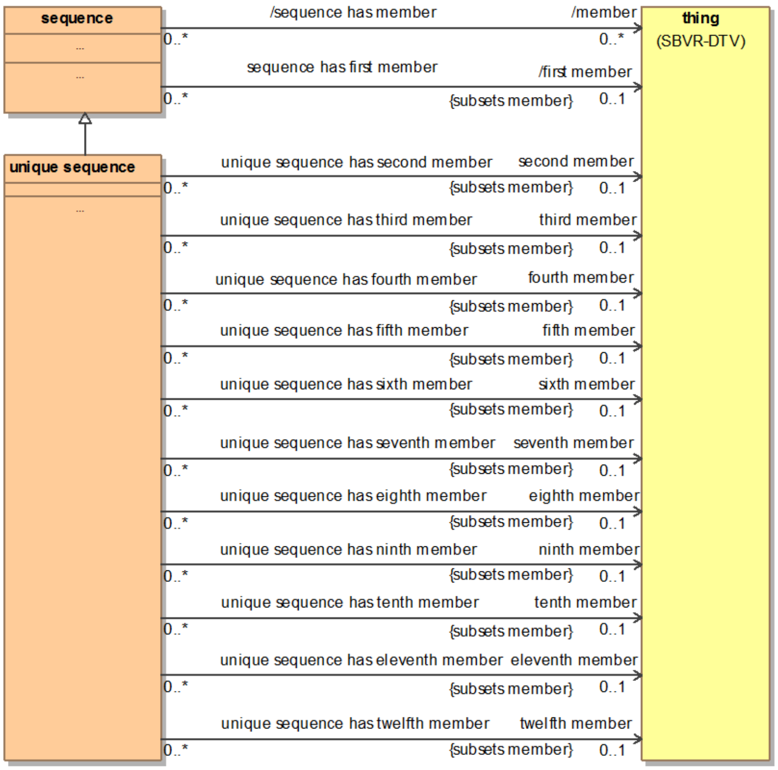


**Figure D.5 - Ordinals**

## second member

Synonym: second
Concept Type: role

General Concept: thing
Definition: the next member *after* the first member *in* a given unique sequence
Necessity: The concept 'second member' *specializes* the concept 'member'.

## unique sequence *has* second member

Definition: the unique sequence *has* a first member and the second member *is next after* the first member *in* the unique sequence

## third member

Synonym: third
Concept Type: role
General Concept: thing
Definition: the next member *after* the second member *in* a given unique sequence
Necessity: The concept 'third member' *specializes* the concept 'member'.

## unique sequence *has* third member

Definition: the unique sequence *has* a second member and the third member *is next after* the second member *in* the unique sequence

## fourth member

Synonym: fourth
Concept Type: role
General Concept: thing
Definition: the next member *after* the third member *in* a given unique sequence
Necessity: The concept 'fourth member' *specializes* the concept 'member'.

## unique sequence *has* fourth member

Definition: the unique sequence *has* a third member and the fourth member *is next after* the third member *in* the unique sequence

## fifth member

Synonym: fifth
Concept Type: role
General Concept: thing
Definition: the next member *after* the fourth member *in* a given unique sequence
Necessity: The concept 'fifth member' *specializes* the concept 'member'.

## unique sequence *has* fifth member

Definition: the unique sequence *has* a fourth member and the fifth member *is next after* the fourth member *in* the unique sequence

## sixth member

Synonym: sixth
Concept Type: role
General Concept: thing
Definition: the next member *after* the fifth member *in* a given unique sequence
Necessity: The concept 'sixth member' *specializes* the concept 'member'.

## unique_sequence *has* sixth_member

Definition: the unique sequence *has* a fifth member and the sixth member *is next after* the fifth member *in* the unique sequence

## seventh_member

Synonym: seventh
Concept Type: role
General Concept: thing
Definition: the next member *after* the sixth member *in* a given unique sequence
Necessity: The concept 'seventh member' *specializes* the concept 'member'.

## unique_sequence *has* seventh_member

Definition: the unique sequence *has* a sixth member and the seventh member *is next after* the sixth member *in* the unique sequence

## eighth_member

Synonym: eighth
Concept Type: role
General Concept: thing
Definition: the next member *after* the seventh member *in* a given unique sequence
Necessity: The concept 'eighth member' *specializes* the concept 'member'.

## unique_sequence *has* eighth_member

Definition: the unique sequence *has* a seventh member and the eighth member *is next after* the seventh member *in* the unique sequence

## ninth_member

Synonym: ninth
Concept Type: role
General Concept: thing
Definition: the next member *after* the eighth member *in* a given unique sequence
Necessity: The concept 'ninth member' *specializes* the concept 'member'.

## unique_sequence *has* ninth_member

Definition: the unique sequence *has* an eighth member and the ninth member *is next after* the eighth member *in* the unique sequence

## tenth_member

Synonym: tenth
Concept Type: role
General Concept: thing
Definition: the next member *after* the ninth member *in* a given unique sequence
Necessity: The concept 'tenth member' *specializes* the concept 'member'.

## unique_sequence *has* tenth_member

Definition: the unique sequence *has* a ninth member and the tenth member *is next after* the ninth member *in* the unique sequence

## eleventh member

Synonym: eleventh
Concept Type: role
General Concept: thing
Definition: the next member *after* the tenth member *in* a given unique sequence
Necessity: The concept 'eleventh member' *specializes* the concept 'member'.

## unique sequence *has* eleventh member

Definition: the unique sequence *has* a tenth member and the eleventh member *is next after* the tenth member *in* the unique sequence

## twelfth member

Synonym: twelfth
Concept Type: role
General Concept: thing
Definition: the next member *after* the eleventh member *in* a given unique sequence
Necessity: The concept 'twelfth member' *specializes* the concept 'member'.

## unique sequence *has* twelfth member

Definition: the unique sequence *has* an eleventh member and the twelfth member *is next after* the eleventh member *in* the unique sequence

### D.2.7 Set Concepts

This subclause defines additional verb concepts for SBVR 'set'.

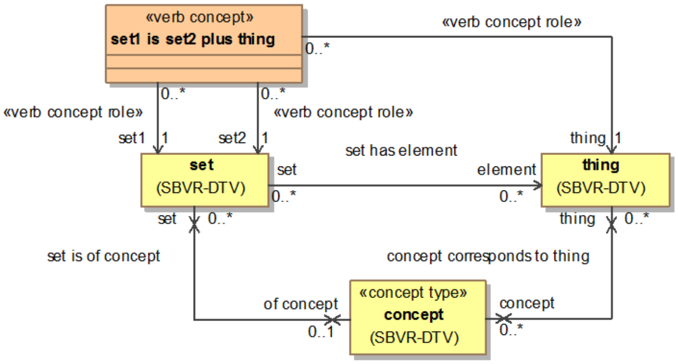


**Figure D.6 - Set concepts**

## set *is of* concept

Definition: each element *of* the set *is* an instance *of* the concept

CLIF Definition:
```
(forall (s c) (iff
  ("set is of concept" s c)
  (and
    (set s) (concept c)
    (forall (e)
      (if ("set includes element" s e) (c e) ))
  )))
```
OCL Definition:
```
context set
def: _'set is of concept'(c: concept) : Boolean =
    set.element->forAll(e |
      c._'concept corresponds to instance'(e))
```

### $set_1$ *is* $set_2$ *plus* thing

Synonymous Form: $set_2$ *plus* thing

Definition: $set_1$ *includes* the thing and $set_1$ *includes* each element *of* $set_2$, and each element *of* $set_1$ *is* the thing or an element *of* $set_2$

Description: $set_1$ is the combination of $set_2$ and thing

Note: This verb concept supports adding an element to a set.

CLIF Definition:
```
(forall (s1 s2 t) (iff
  ("set1 is set2 plus thing" s1 s2 t)
  (and
    (set s1) (set s2)
    ("thing is in set" t s1)
    (forall (e)
      (if ("thing is in set" e s2) ("thing is in set" e s2) ))
    (forall (e)
      (if ("thing is in set" e s1)
          (or ("thing is in set" e s2) (= e t) ) ))
  )))
```
CLIF Definition:
```
(forall (s1 s2 t) (iff
  (= s1 ("set plus thing" s2 t))
  ("set1 is set2 plus thing" s1 s2 t) ))
```
OCL Definition:
```
context set
def: _'set1 is set2 plus thing'(s2: set, t: thing) : Boolean =
      self.includes(t) and s2->forAll(t2: self.includes(t2))
      and self.element->forAll(e: e = t or s2.includes(e))
```
OCL Definition:
```
context set
def: _'plus thing'(t: thing) : set =
  self.element->union(t)
```
Example: {'a', 'b', 'c'} is {'a', 'b'} plus 'c'.

______________________________________________________

______________________________________________________

## D.3 Quantities Vocabulary (informative)

Quantities model many of the concepts in the International Vocabulary for Measures [VIM].

________________________________________________

## Quantities Vocabulary

General Concept: terminological dictionary
Included Vocabulary: SBVR-DTV Vocabulary
Language: English
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#QuantitiesVocabulary

________________________________________________

### D.3.1 Quantities

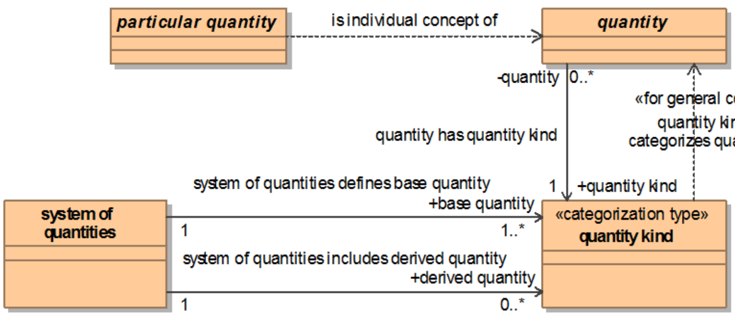


**Figure D.7 - Quantities**

## quantity

Definition: property of a phenomenon, body, or substance, to which a number can be assigned with respect to a reference

Dictionary Basis: VIM 1.1 'quantity'

Note: The term 'quantity' is used here to refer to the abstraction of the properties – the amount of measurable "stuff" that can be compared between particular quantities. The "height of the Washington Monument" refers to a 'particular quantity;' "555 ft 5 inches" refers to a 'quantity value'.

Note: This is not the SBVR concept 'quantity,' which is deprecated and not used in this model.

Example: second, kilogram, joule, meter. These are quantities in a general sense, which is what is meant here by 'quantity.'

Note: A quantity as defined here is said to be a "scalar" as distinct from a "vector." However, a vector or a tensor whose components are quantities is also considered to be a quantity.

## particular quantity

Definition: a property that is of an individual thing and is quantifiable as an instance of some quantity kind

Note: The weight of a given person, the mass of the Earth, the speed of light, and the distance between New York and Paris are said to be "particular quantities."

Note: A particular quantity is given by a definite description, which identifies the individual thing and the property. Particular quantities are properties of particular things and are generally expressed by a term for the property and a quantity value.

Note: Particular quantities appear in fact models as individual concepts that refer to instances of 'quantity.' Thus, a conceptual schema might have the fact type "meeting

| | |
|---|---|
| | *lasts* duration," where "duration" is a specialization of "quantity." (See the note about "duration" under "quantity kind.") A fact model might include the fact "last Monday's meeting lasted 2hr 20min." The definite description "the duration of last Monday's meeting" defines a particular quantity, an individual concept whose one instance is the quantity (thing) that is quantified by the quantity value "2 hr 20 min." "2hr 20min" is a compound quantity value of quantity kind "duration." |
| Reference Scheme: | A definite description *of* the particular quantity. |

## quantity kind

| | |
|---|---|
| Definition: | categorization type for 'quantity' that characterizes quantities as being mutually comparable |
| Concept Type: | categorization type |
| Dictionary Basis: | VIM 1.2 'kind of quantity' |
| Example: | duration, mass, energy, length |
| Note: | Every instance of 'quantity kind' is also a specialization of 'quantity'. So the concept 'duration' is an instance of 'quantity kind' and it is a specialization of 'quantity', i.e., it is a classifier of actual quantities. But *a given* duration (i.e., the duration of something) is an instance of 'duration' and thus a 'particular quantity,' not an instance of 'quantity kind'. For example, a 'year' is not an instance of quantity kind; it is an instance of quantity, but not a category of quantity. |
| Note: | The quantities "year" and "second" are instances of quantity, and they are both instances of the quantity kind 'duration' and are mutually comparable. Quantities of time given in years and seconds are comparable, although some transformation of quantity values (see below) is needed to compare them. Similarly 'metre' is an instance of 'length,' and 'foot' is another instance of 'length' that is comparable to 'metre,' although conversions are required when comparing values. But 'metre' is not comparable to 'second', because 'length' and 'duration' are disjoint quantity kinds. Only quantities of the same kind are mutually comparable. |
| Note: | All duration quantities are comparable regardless of the role they play – the particular properties they instantiate. The duration of the warranty on an automobile can be compared with the expected life of the battery, even though those are very different particular quantities. Similarly, the height of a tower can be compared to the distance one can see from the top, because they are both length quantities, even though they are unrelated properties. |
| Note: | The concept 'height' is a role of quantities of the quantity kind 'length'. In principle, 'height' could be considered a category of 'quantity' (a sub-category of 'length') and therefore its own 'quantity kind.' The concept 'range of a weapon' is a different role of length quantities. If we want to treat the height of a target as comparable to the range of a weapon, it is inadvisable to treat height and range as different quantity kinds. This idea is the basis for the 'system of quantities' concept. |

## quantity *has* quantity kind

| | |
|---|---|
| Definition: | quantity is an instance of the category of quantity that is the quantity kind |
| Necessity: | Each quantity *has* exactly one quantity kind. |
| CLIF Axiom: | (forall ((q quantity))<br>(exists ((qk "quantity kind"))<br>(and<br>("quantity has quantity kind" q qk)<br>(forall (qk2)<br>(if<br>("quantity has quantity kind" q qk2)<br>(= qk2 qk) )) ))) |

OCL Constraint: context quantity
inv: _'quantity kind'->allInstances(one qk |
self._'quantity kind' = qk)

Example: hour (the duration) is an instance of 'duration' – a specific quantity of time. So the quantity kind of 'hour' is 'duration'.

## system of quantities

Definition: set of quantities together with a set of non-contradictory equations relating those quantities

Dictionary Basis: VIM 1.3 'system of quantities'

## system of quantities *defines* base quantity

## base quantity

Definition: quantity kind in a conventionally chosen subset of a given system of quantities, where no subset quantity can be expressed in terms of the others

Concept Type: role

Dictionary Basis: VIM 1.4 'base quantity'

Example: The International System of Quantities (ISQ) comprises these base quantities (with their SI base measurement units):

- length (meter)
- mass (kilogram)
- duration (second)
- electric current (ampere)
- thermodynamic temperature (kelvin)
- amount of substance (mole)
- luminous intensity (candela)

These base quantities are not mutually comparable. All quantities of any one of these kinds are, however, mutually comparable. See also "quantity kind."

## system of quantities *includes* derived quantity

## derived quantity

Definition: quantity kind, in a system of quantities, that is not a base quantity of the system but may be defined in terms of base quantities of the system

Dictionary Basis: VIM 1.5 'derived quantity'

Example: velocity (`length/time`), mass density ($mass/length^3$)

## D.3.2 Measurement Units

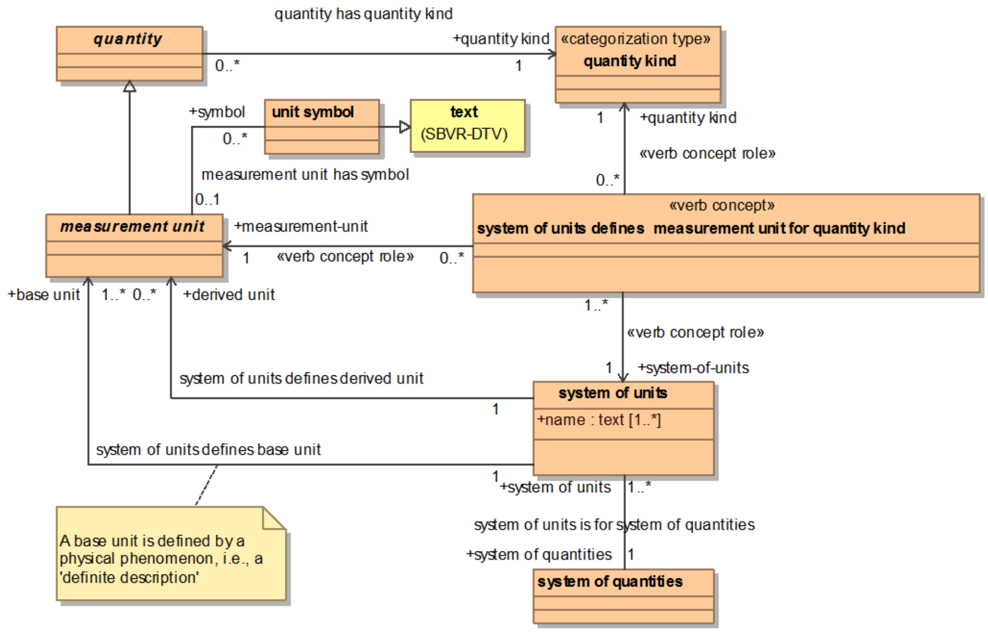


**Figure D.8 - Measurement Units**

### measurement unit

Definition: quantity, defined and adopted by convention, with which any other quantity of the same kind can be compared to express the ratio of the two quantities as a number

Dictionary Basis: VIM 1.9 'measurement unit'

Example: week, day, hour, minute, second, kilogram, joule, meter

### system of units

Definition: a set of measurement units associated with a system of quantities, together with a set of rules that assign one measurement unit to be the base unit for each base quantity in the system of quantities and a set of rules for the derivation of other units from the base units.

Example: The International System of Units (SI) is a system of units.

### system of units *is for* system of quantities

Necessity: Each system of units *is for* exactly one system of quantities.

CLIF Axiom:
```
(forall ((sou "system of units"))
 (exists ((soq "system of quantities"))
   (and
     ("system of units is for system of quantities" sou soq)
     (forall (soq2)
         (if ("system of units is for system of quantities" sou soq2)
             (= soq2 soq) ))
)))
```
OCL Constraint:
```
context _'system of units'
  self._'system of quantities'->size() = 1
```

## system of units *defines* measurement unit *for* quantity kind

Synonymous Form: measurement unit *is defined for* quantity kind *by* system of units

Definition: The system of units identifies the measurement unit as the reference measurement unit for the quantity kind.

Note: A system of units defines one base unit for each base quantity in the system of quantities that it is for. It may define additional measurement units (derived units) for the same quantity kinds. It may define derived units for derived quantities, or it may define a mechanism for expressing derived units.

## system of units *defines* base unit

## base unit

Definition: measurement unit that *is defined for* a base quantity *by* a system of units

Concept Type: role

Dictionary Basis: VIM 1.10 'base unit'

Note: Quantity units that are not base units are derived units.

Example: See the example SI units under "base quantity".

## system of units *defines* derived unit

## derived unit

Definition: measurement unit for a derived quantity

Dictionary Basis: VIM 1.11 'derived unit'

Note: Every derived unit is defined in terms of base units

Example: 1 minute = 60 seconds

Example: 1 stere = 1 metre$^3$

Example: 1 inch = 0.0254 metre

## International System of Units

Synonym: SI

Definition: The system of units that is defined for the International System of Quantities by the International Standard ISO 80000.

Source: VIM 1.16.

### D.3.3 Quantity values

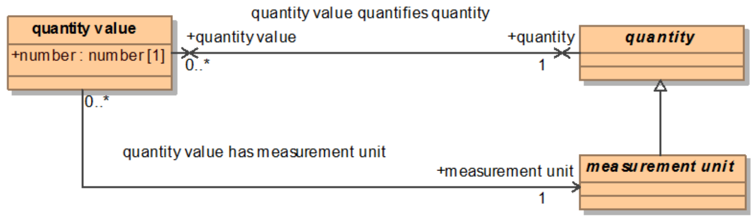


**Figure D.9 - Class Diagram for Quantity Values**

## quantity value

Definition: number and measurement unit together giving magnitude of a quantity
Dictionary Basis: VIM 1.19 'quantity value'
Note: The quantity expressed by a quantity value is the quantity whose ratio to the measurement unit is the number.
Example: 2 days, 3.5 hours, 150 lb, 45.5 miles

## quantity value *quantifies* quantity

Synonymous Form: quantity *is quantified as* quantity value
Synonymous Form: quantity value *of* quantity
Synonymous Form: quantity value *expresses* quantity
Definition: The quantity value gives the magnitude of the quantity.
Possibility: More than one quantity value may *quantify* a particular quantity.
Example: The duration of a meeting is a particular quantity that might *be quantified as* "1 hour" or as "6 minutes".

________________________________________________
________________________________________________

## D.4 Mereology (normative)

Mereology is the study [Simons] [Casati] of the relationships among whole things and their parts. This specification relies upon the following mereology axioms, among others, to define the properties of time intervals.

________________________________________________

## Mereology Vocabulary

General Concept: terminological dictionary
Included Vocabulary: SBVR-DTV Vocabulary
Language: English
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#MereologyVocabulary

________________________________________________

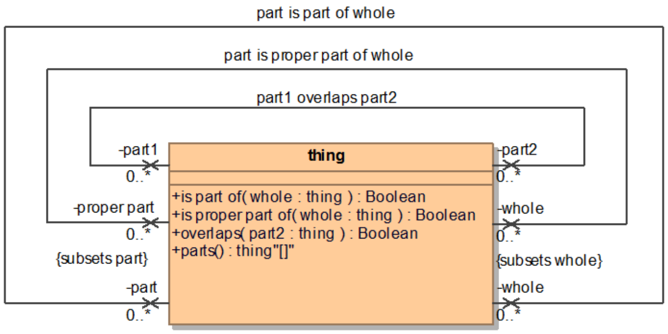


**Figure D.10 - Mereology**

## whole

Concept Type: role
General Concept: thing

## part

Concept Type: role
General Concept: thing

## part *is part of* whole

Synonymous Form: whole *includes* part
Definition: The part is a component of the whole.
Note: There are a number of axioms of mereology that apply to the concept 'part *is part of* whole.' The following 3 axioms specify only that subset of those axioms that are needed by this specification. This subset is needed to define the partial ordering relationship among time intervals.
Note: Axiom of *reflexivity*: every part is part of itself.
Necessity: Each part *is part of* the part.
CLIF Axiom:
```
(forall (part)
 ("part of" part part))
```
OCL Constraint:
```
context thing
     inv: self.part->exists(self)
```
Note: Axiom of *antisymmetry*: two distinct parts cannot be part of each other.
Necessity: If the part *is part of* the whole and the whole *is part of* the part then the part *is* the whole.
CLIF Axiom:
```
(forall ((part thing) (whole thing))
 (if (and ("part of" part whole)
         ("part of" whole part))
    (= part whole)))
```
OCL Constraint:
```
context thing
     inv: self.whole->exists(p |
                    p.whole ->exists(self))
         implies self = self.whole
```
Note: Axiom of *transitivity*
Necessity: If the part *is part of* some whole and the whole *is part of* some $part_3$ then the part *is part of* $part_3$.

| | |
|---|---|
| CLIF Axiom: | (forall ((part thing) (whole thing) (part3 thing))<br>(if (and ("part of" part whole)<br>("part of" whole part3))<br>("part of" part part3))) |
| OCL Constraint: | context thing<br>inv: self.whole->exists(whole \|<br>whole.whole->exists(part3 \|<br>part3 implies self._'part of'(part3))) |

The combination of the reflexivity, anti symmetry, and transitivity axioms define a partial ordering among things that have the '*part is part of whole*' relationship.

## thing₁ overlaps thing₂

| | |
|---|---|
| Definition: | there exists a thing that *is part of* thing$_1$ and that *is part of* thing$_2$ |
| CLIF Definition: | (forall (thing1 thing2)<br>(iff (overlaps thing1 thing2)<br>(exists (thing3)<br>(and<br>("part of" thing3 thing1)<br>("part of" thing3 thing2) )) )) |
| OCL Definition: | context thing<br>def: self.overlaps(thing2: thing): Boolean =<br>self.part->exists(thing3 \| thing2.part->exists(thing3) ) |
| Note: | Two things overlap if they have some part in common. |
| Note: | It is obvious from the definition that 'thing1 overlaps thing2' is symmetric. |
| Necessity: | If a thing$_1$ *overlaps* a thing$_2$, then thing$_2$ *overlaps* thing$_1$. |
| CLIF Axiom: | (forall (thing1 thing2)<br>(iff (overlaps thing1 thing2) (overlaps thing2 thing1) )) |
| OCL Constraint: | context thing<br>inv: self.overlaps(thing2) eqv thing2.overlaps(self) |

## part *is a proper part of* whole

| | |
|---|---|
| Definition: | the part *is part of* the whole and the whole *is* not *part of* the part |
| CLIF Definition: | (forall (whole part)<br>(iff ("proper part" part whole)<br>(and<br>("part of" part whole)<br>(not ("part of" whole part))))) |
| OCL Definition: | context thing<br>inv: self._'proper part'->forall(pp \| pp <> self) |
| Note: | A proper part is a part that is not the whole. |

Axiom of *supplementation*: If a whole has a proper part, then it has more than one proper part.

| | |
|---|---|
| Necessity: | If a part$_1$ *is a proper part of* a whole then there exists a part$_2$ that *is a proper part of* the whole and part$_2$ *does* not *overlap* part$_1$. |
| CLIF Axiom: | (forall (part1 whole)<br>(if ("proper part" part1 whole)<br>(exists (part2)<br>(and<br>("proper part" part2 whole)<br>(not (overlaps part2 part1)) )))) |
| OCL Constraint: | context thing<br>inv: self._'proper part'->forAll(part1 \|<br>self._'proper part'->exists(part2 \| not part2.overlaps(part1))) |

______________________________________________

______________________________________________

# Annex E - Formalizing English Tense and Aspect

(informative)

## E.1 General

The normative clauses of this specification deal the semantics of time as used natural languages. This Annex describes how propositions that are given in English language syntax may be formulated using the Date-Time Vocabulary.

## E.2 Syntax and Semantics of Time

Many natural languages have built-in syntactical mechanisms for expressing when an action occurs relative to the time of utterance or writing, or relative to the occurrence of another event. They also have standard ways of indicating whether and when an action is progressing or is accomplished. These mechanisms include the use of affixes with verbs, called *tense*, and the use of auxiliary verbs together with the main verb of a clause, called *aspect*. Not all languages have the same set of these kinds of mechanisms.

The terms 'tense' and 'modal' are used with somewhat different connotations when referring to syntax or semantics. In syntactic theory, 'tense' refers to different verb forms used to denote different times: past, present, or future. The term 'tense' can also be used to refer to the semantics of a temporal expression: the past tense, the present tense, the future tense. All languages incorporate mechanisms to express such semantics, but different languages have different syntactical mechanisms for doing so. Confusion sometimes arises in English, which has verb forms only for present tense and past tense. Consequently, it is common for some authors to say that English has only two tenses, past and present, and no future tense. At the same time, it is often said that the future tense in English is expressed using the auxiliary verb 'will'. In this annex, 'tense' refers to verb forms that express past or present time, and 'aspect' to the use of auxiliaries to generate different senses of past, present, and future time. In the normative clauses of the specification, 'tense' refers to the semantics of past, present, or future time, without regard to the syntactical mechanisms employed to express time in any language.

The term 'modal' can be confused with 'modality.' In this annex, 'modal' is a grammatical term that refers to a modal verb (*see 'modal'* below). 'Modality' is a logical term, used in SBVR, to refer to the mood of a proposition as involving the affirmation of either possibility, impossibility, necessity, contingency, obligation, or permission. SBVR includes a modal logic for these modalities, including modal formulae and modal negation rules. This specification does not provide a temporal logic for the temporal modality, rather temporal concepts are handled by the introduction of first order concepts and fact types defined in the normative clauses of this specification. No temporal logical operations are introduced in this specification. Negation of propositions involving time is treated conventionally as logical negation as specified in SBVR.

English syntax involving modal auxiliary verbs serves to denote both the tense and the logical mood of a proposition. The meaning depends on the particular auxiliary verbs used. A temporal connotation can be associated with each auxiliary verb, such that auxiliary verbs carry both a temporal connotation and a mood. The following table gives some examples.

**Table E.1 - Modalities for Auxiliary Verbs**

| Auxiliary Verb | Time Frame | Modality |
|---|---|---|
| can | present | possibility |
| can not | present | impossibility |
| could | past | possibility |
| do not | future | negation |

| | | |
|---|---|---|
| does not | present | negation |
| did not | past | negation |
| may | present | permission |
| might | past*, future | possibility |
| must | past*, future | obligation |
| need | always | necessity |
| shall | future | necessity |
| should | past*, future | contingency |
| used | past | -- |
| will | future | -- |
| would | past | -- |
| | * with *have* | |

Logical negation can be indicated by using *not* with an auxiliary verb; only a few examples are shown. *Always, never,* or *not ever* can be used with some modal auxiliary verbs to indicate *at all times*, or *not at any time*, as the case may be. Some words that serve as auxiliary verbs can have other grammatical roles as well. Time frame and modality can be expressed by means other than auxiliary verbs; this annex focuses on the behavior of English verbs in referring to time.

## E.3 Organization of This Annex

This specification includes fact types that accurately capture the meaning of relationships between states of affairs and time, but the fact type forms needed for precise definition are not idiomatic. This annex describes a way to accommodate idiomatic English expressions involving time, giving rules for mapping such expressions to concepts provided in this specification preparatory to creating closed logical formulations of the idiomatic expression. This treatment is informative, not normative; other approaches are possible. It is extensive but not exhaustive; the most common cases are treated, but not all possibilities. A formal grammar of the tense and aspect in English is provided, followed by a general algorithm for performing the syntax-to-semantics transformations for the twelve grammatical categories. Finally, a table of specific cases of the use of tense and aspect in English is provided.

This annex only describes formulations in which time is denoted by verbs. Other temporal constructs, such as the use of literal duration values and time coordinates and expressions involving relationships between time periods, are not discussed here.

This annex effectively extends the modal operations described in SBVR Annex F The RuleSpeak® Business Rule Notation, to include time, but stops short of being a full treatment of temporal modality.

## E.4 Definitions

The following definitions are excerpted from Sag, Wasow, and Bender, *Syntactic Theory, Second Edition*, Stanford University, Center for the Study of Language and Information (2003), Glossary.

**tense** Finite verbs come in different form depending on the time they denote; these forms are called 'tenses'. English has present and past tense, exemplified by the present tense forms *walk* and *walks*, and by the past tense form *walked*. Some languages also have future tenses, but English uses other means (e.g., the modal [q.v.] *will*) to express future time.

**aspect** Many language have special grammatical elements for locating in time the situation referred to. Among the temporal notions often expressed are whether situations are in process or completed and whether they occur repeatedly. These notions are often called 'aspect,' and words or affixes whose function is to express aspect are called 'aspectual markers.' *See also* perfective, progressive.

**finite verb** A finite verb is one that is marked for tense [q.v.] (present or past, in English).

**modal** The English verbs *can, could, may, might, must, shall, should, will,* and *would*, along with their negated forms (*can't,* etc.) are referred to as 'modals' or 'modal verbs.' They share the following properties: they function only as finite verbs [q.v.]; they exhibit auxiliary behavior (negation, inversion, contraction, and ellipsis); they take base VP [verb phrase] compliments; and they show no agreement [q.v.] (i.e., no third-person singular *–s* suffix). Some other languages have similar syntactically distinctive classes of words expressing necessity, possibility, obligation, and permission; these are also known as modals.

**agreement** In many languages, the form of certain elements can vary to indicate such properties such as person [referring to the speaker, the hearer, or third parties], number [referring to single entities or multiple entities], gender, etc. Often, these variations are marked with affixes. Some grammatical relationships between pairs of linguistic elements require they agree on these properties. In English, for example, present tense verbs are marked to indicate whether the subjects are third-person singular (with the suffix *–s*), and nouns indicate plurality (also with a suffix *–s*). The systematic covariation of the forms of the subject and verb is called 'subject-verb agreement'. Similarly, pronouns must agree with their antecedents in person, number, and (if third-person) gender.

**perfective** Many languages have special verb forms or constructions used to indicate that the event denoted by the verb is completed. These are referred to as 'perfective' (or just 'perfect') in aspect. The English perfective involves the combination of *have* with a past participle [q.v.], as in *The dog has eaten the cake. See also* aspect.

**progressive** Special verb forms or construction used to indicate that the event denoted by the verb is in progress are referred to as 'progressive' aspect. The English progressive involves combination of *be* with a present participle [q.v.], as in *The dog is eating the cake. See also* aspect.

**participle** Certain nonfinite verbs – usually ones that share some properties with adjectives – are referred to as 'participles.' English has three types of participles: present participles, which end in *–ing* and usually follow some form of *be*; past participles, which usually end in *–ed* or *–en* and follow some form of *have*; and passive participles, which look exactly like past participles but indicate the passive voice [q.v.]. The three participles of *eat* are illustrated in the following sentences:

(i) Termites are eating the house.
(ii) Termites have eaten the house.
(iii) The house was eaten by termites.

## E.5 English Grammar of Tense and Aspect

English grammar for tense and aspect can be defined as follows, using Extended Backus Nauer Form notation (ISO/IEC 14977 Information technology – Syntactic metalanguage - Extended BNF). '::=" means 'is defined as.' Each '::=' statement is a production rule. Each production rule is terminated by ';'. The order of the symbols on the right hand side of each production rule is significant, unless delimited by '|'. '|' means 'or', a choice. Brackets '[]' indicate the element is optional. Quoted words are literals. Comments are included between '(*' and '*)'.

S ::= NP AUX VP; (* S–sentence, NP–noun phrase, VP–verb phrase *)

AUX ::= [MODAL] [PERF] [PROG]; (* auxiliary verb *)

MODAL ::= 'can' | 'could' | 'may' | 'might' | 'must' | 'shall' | 'should' | 'used' | 'will' | 'would';

PERF ::= 'have' | 'has' | 'had'; (* perfective *)

PROG ::= 'is' | 'are' | 'was' | 'were' | 'be' | 'been'; (* progressive *)

#### Additional Rules for Auxiliaries (AUX)

1. Auxiliaries are optional.
2. Auxiliaries precede any non-auxiliary verb.
3. Auxiliaries determine the form of the following verb.
4. Auxiliaries can co-occur with each other, but only in a fixed order.
5. Auxiliaries of any given type cannot iterate.

The modals all indicate future time. They have the additional property of expressing necessity, possibility, obligation, or permission, as discussed in SBVR.

Not all combinations generated by the above grammar are valid English. Other rules apply, not given, such as subject-verb agreement. *Not, never, always,* or *not ever* can be used with some modals; these grammatical details are outside the scope of this annex, but the methods of this annex can be extended to include them. The table in E.6 gives a listing of grammatical constructs that appear regularly in English.

Reference: Sag, Wasow, and Bender (ibid.), pp.392-394.

## E.6 Formulating Tense and Aspect

The general approach used here to formulate a sentence involving tense or aspect is as follows:

1. Transform the sentence into a proposition based on the applicable fact type form in the conceptual schema, noting the original tense and aspect.
2. Identify the situation kind that the base proposition describes.
3. Restrict the situation kind by instantiating one or more of the fact types defined in this specification involving states of affairs and time, as noted in 1.
4. Create closed logical formulations that mean the base proposition and its restrictions, as described in SBVR.

#### Transform to a base proposition

All propositions in SBVR are considered to be true or false when considered with respect to a given fact model. A proposition might be true when considered with respect to one fact model, and false when considered with respect to another. Each fact model is taken to be a snapshot of the state of the universe of discourse at some time. The fact model is tantamount to a database, and the veracity of each proposition is based on the facts in the database at the time of the snapshot, which time may or may not be stated. This is standard SBVR.

Propositions in standard SBVR are expressed preferably in the simple present tense when finite verbs are used. Such propositions are considered untensed, as they apply to any fact model representing the state of the universe at the snapshot time of the fact model. Propositions involving non-finite verbs are also considered untensed in standard SBVR.

This specification includes the concept now, which is the current time, or present. When evaluating propositions using this specification, now is the snapshot time of the fact model with respect to which the propositions are being evaluated.

Transforming a proposition into an base form involves changing the verb to the tense of the applicable fact type in the conceptual schema, maintaining subject-verb agreement.

For example, the present perfect progressive sentence "Acme has been trading with Xycore" transforms to untensed "Acme trades with Xycore," with the notation that the original is present perfect progressive. These sentences are both based on the fact type company$_1$ *trades with* company$_2$.

The guidance is generally not to encode tense or aspect into fact type forms unless the domain model specifically requires a particular tense or aspect for that fact type. Consider this example, "Six tasks have completed on May 5, 2010" may be based on the fact type "task *completed on* time point." This fact type has an intransitive past tense verb. The conceptual schema has already restricted facts of this type to past or perfected. The example transforms to "Six tasks completed on May 5, 2010" with a notation that the original is present perfect. A different conceptual schema might include the fact type "task *completes on* time point" instead. The proposition then transforms to "Six tasks complete on May 5, 2010" with the same present perfect notation. The "*completes on*" fact type, unlike the "*completed on*" version, could be used for facts about future planned completions (*will complete on*). This illustrates that there is a certain economy in using simple-present fact type forms in domain models: every different tense and aspect variation of these sentences is based on the same fact type and transforms to the same untensed form.

#### Identify the situation kind

The situation kind of interest is the one that is described by the transformed sentence, the base proposition.

#### Restrict the situation kind

The situation kind is restricted by involving it in a role in an instance of appropriate fact type(s) from this specification. Which fact types to use depends on the tense and aspect of the original sentence, as noted at the time the base proposition was created. Create a fact instance of each of the appropriate fact types.

#### Create closed logical formulations

A closed logical formulation is created for the conjunction of the base proposition and the restricting facts. This constitutes the closed logical formulation of the original sentence.

## E.7 Mapping Tense and Aspect to the Date-Time Vocabulary

This table is extensive but not exhaustive. Different modals can be substituted for 'will,' with other restrictions in the logical formulation (e.g., obligatory). In some of the examples, the 'now' time is apparently in the past, to accord with the history of James Joyce.

**Table E.2 - Mapping Tense and Aspect to the Date-Time Vocabulary**

| MODAL | PERF sg/pl | PROG sg/pl | Verb Form | Grammatical Term | Example: person *writes* book | Date-Time Vocabulary Fact Type |
|---|---|---|---|---|---|---|
| | | | present | present simple | Joyce writes Ulysses. | None. This is the base situation kind (s)<br>s: "Joyce writes Ulysses" |
| | | | past | past simple | Joyce wrote Ulysses. | s *is in the past* |
| used | | | infinitive | past simple | Joyce used to write Ulysses. | s *is in the past* |
| will | | | present | future simple | Joyce will not write Ulysses. | s *is in the future* and<br>s *is* not *an actuality* |
| | | is/are | present participle | present progressive | Joyce is writing Ulysses in 1919. | s *holds within* 1919 |
| | | was/were | present participle | past progressive, imperfective | Joyce was writing Ulysses in 1919. | s *is in the past* and<br>s *holds within* 1919 |
| will | | be | present participle | future progressive | Joyce will not be writing Ulysses in 2012. | s *is in the future* and<br>s does not *hold within* 2012 |
| | has/have | | past participle | present perfective | Joyce has written Ulysses. | s *is accomplished* |
| | had/had | | past participle | past perfective, pluperfect | Joyce had written Ulysses by 1922. | s *is in the past* and<br>s *is accomplished* and<br>s *occurs before* 1922 |
| will | have | | past participle | future perfective | Joyce will have written Ulysses by 1922. | s *is in the future* and<br>s *is accomplished* and<br>s *occurs before* 1922 |
| | has/have | been | present participle | present perfect progressive | Joyce has been writing Ulysses in 1919. | s *holds within* 1919 and<br>s *is accomplished* |
| | had | been | present participle | pluperfect progressive | Joyce had been writing Ulysses in 1919. | s *is in the past* and<br>s *holds within* 1919 and<br>s *is accomplished* |
| will | have | been | present participle | future perfect progressive | By the end of 1920, Joyce will have been writing Ulysses for 33 months. | s *is in the future* and<br>s *holds during* December 1920 – 33 months |

# Annex F - Vocabulary Registration Vocabulary

(normative)

## F.1 Vocabularies Presented in this Document

This annex formally lists the vocabularies provided by the Date-Time Vocabulary specification.

______________________________________________________

### Date-Time Vocabulary Registration Vocabulary

General Concept: terminological dictionary
Language: English
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#DTVRegistrationVocabulary

______________________________________________________

### Date-Time Vocabulary Registration Vocabulary

General Concept: terminological dictionary
Language: English
Note: This vocabulary formally registers all the vocabularies specified in this document.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#DTVRegistrationVocabulary

### Time Infrastructure Vocabulary

General Concept: terminological dictionary
Language: English
Description: The primary purpose of this vocabulary is to enable the definition of various kinds of calendars, such as fiscal, lunar, or religious calendars. Most end users will use one of the calendars defined in this document and should not need many of the concepts defined here.
Note: See Clause 8.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#TimeInfrastructureVocabulary

### Duration Values Vocabulary

General Concept: terminological dictionary
Language: English
Description: Duration values are amounts of time stated in terms of one or more time units.
Note: See Clause 9.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#DurationValuesVocabulary

### Calendars Vocabulary

General Concept: terminological dictionary
Language: English
Description: Calendars use time scales to impose structure on time.
Note: See Clause 10.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#CalendarsVocabulary

## Gregorian Calendar Vocabulary

General Concept: terminological dictionary
Language: English
Description: The Gregorian Calendar is the standard calendar, used worldwide.
Note: See Clause 11.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#GregorianCalendarVocabulary

## ISO Week Calendar Vocabulary

General Concept: terminological dictionary
Language: English
Description: Defines the standard calendar based on 7-day weeks.
Note: See Clause 12.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#WeekCalendarVocabulary

## Time of Day Vocabulary

General Concept: terminological dictionary
Language: English
Description: Defines the time scales, time points, and time coordinates that comprise the calendar day.
Note: See Clause 13.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#TimeOfDayVocabulary

## Internet Time Vocabulary

General Concept: terminological dictionary
Language: English
Description: Internet Time is the calendar of the Network Time Protocol (NTP), published by the Internet Engineering Task Force (IETF).
Note: See Clause 14.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#InternetTimeVocabulary

## Indexical Time Vocabulary

General Concept: terminological dictionary
Language: English
Description: Indexical terms for time periods that are in common business use.
Note: See Clause 15.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#IndexicalTimeVocabulary

## Situations Vocabulary

General Concept: terminological dictionary
Language: English
Description: A vocabulary that relates situations to time intervals and durations.
Note: See Clause 16.
Namespace URI: http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#SituationsVocabulary

## Schedules Vocabulary

General Concept: terminological dictionary
Language: English
Description: Schedules relate repeating situations to time.

| | |
|---|---|
| Note: | See Clause 17. |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#SchedulesVocabulary |

### Sequences Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Language: | English |
| Description: | Model of ordered collections of things in which the things are ordered by assigning numbers (indices) to them within the collection, as distinct from any particular properties of the things themselves. |
| Note: | See Annex D.2. |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#SequencesVocabulary |

### Quantities Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Language: | English |
| Description: | A minimal set of the concepts defined in VIM. |
| Note: | See Annex D.3. |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#QuantitiesVocabulary |

### Mereology Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Language: | English |
| Description: | Concepts about the relationship of wholes and parts. |
| Note: | See Annex D.4. |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#MereologyVocabulary |

### SBVR-DTV Vocabulary

| | |
|---|---|
| General Concept: | terminological dictionary |
| Language: | English |
| Description: | Selected concepts adopted from the SBVR Meaning and Representation Vocabulary or the SBVR Vocabulary for Describing Business Vocabularies. |
| Namespace URI: | http://www.omg.org/spec/DTV/20160301/dtv-sbvr.xml#SBVR-DTVVocabulary |

Various vocabularies, standards, and other publications that are referenced in the SBVR aspects of this specification are formally named as SBVR "individual constants" here.

## F.2 External Vocabularies and Namespaces

### BIPM

| | |
|---|---|
| General Concept: | vocabulary |
| Definition: | The standard of the Bureau International des Poids et Mesures (BIP<), named: *The International System of Units*, 8th edition, 2006 |

### IEC 60050-111

| | |
|---|---|
| General Concept: | vocabulary |
| Definition: | The standard of the International Electrotechnical Committee, International Electrotechnical Vocabulary, number-60050 Chapter 111, named: *Physics and Chemistry*, Edition 2.0, 1996-07 |

## ISO 18026

General Concept: vocabulary
Definition: The standard of the International Standards Organization (ISO), number 18026, named: *Information technology - Spatial Reference Model (SRM)*, 2009

## ISO 80000-3

General Concept: vocabulary
Definition: The standard of the International Standards Organization (ISO), number ISO 80000-3, named: *Quantities and Units -- Part 3: Space and time*, 2006

## ISO 8601

General Concept: vocabulary
Definition: The standard of the International Standards Organization (ISO), number 8601, named: Data elements and interchange formats – Information interchange – Representation of Dates and Times, Third edition, December 1, 2004

## NODE

Definition: The publication named: *New Oxford Dictionary of English*

## NTP

General Concept: vocabulary
Definition: The standard of the Internet Engineering Task Force, RFC 5905, named: *Network Time Protocol Version 4: Protocol and Algorithms Specification*

## SBVR Vocabulary

General Concept: vocabulary
Definition: the vocabulary for terminological dictionaries/ontologies and rulebooks version 1.0 as specified in [OMG formal/08-01-02] available at http://www.omg.org/spec/SBVR/1.0/
Note: This vocabulary is a combination of the following: Meaning and Representation Vocabulary, Logical Formulation of Semantics Vocabulary, Vocabulary for Describing Business Vocabularies, and Vocabulary for Describing Business Rules
Note: The specific concepts from the SBVR Vocabulary that are used by the Date-Time Vocabulary are inventoried in the SBVR-DTV Vocabulary.
Namespace URI: http://www.omg.org/spec/SBVR/20070901/SBVR.xml

## SI

General Concept: vocabulary
Definition: The standard of the International Standards Organization (ISO), number ISO 18026, named: Information technology - Spatial Reference Model (SRM), 2009

## VIM

General Concept: vocabulary
Definition: The standard of the International Standards Organization/International Electrotechnical Commission (ISO/IEC), number JCGM 200: 2008, named: International Vocabulary for Metrology - Basic and General Concepts and Associated Terms (VIM), 3rd edition

# Annex G - UML Profile for the SBVR Elements used in the Date-Time Vocabulary

(normative)

## G.1 General

This annex specifies the stereotypes that are used to mark up UML model elements in the DTV specification.

A general UML Profile for SBVR concepts has not been developed by the OMG. It is expected that such a profile will be developed in the future. At such time, this Annex and the corresponding UML stereotypes in the DTV UML model will be superseded.

The UML metaclass Class is depicted in the diagram because it plays roles in stereotyped relationships. The UML metaclasses Association and Dependency are not depicted. They serve only as the UML base elements for some of the defined stereotypes.

## G.2 Concept types

The SBVR term concept type refers to a concept whose instances are concepts. Two stereotypes are introduced to support this notion.

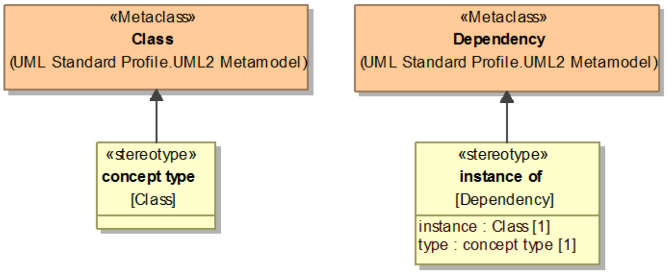


**Figure G.1 - Concept types**

### G.2.1 Stereotype «concept type»

The stereotype «`concept type`» characterizes a UML Class as an SBVR concept type. In UML terms, it is a classifier whose instances are classes.

### G.2.2 Stereotype «instance of»

The stereotype «`instance of`» characterizes a UML Dependency as representing the relationship between a UML Class (representing an SBVR concept) and a concept type that *corresponds to* it. That is, the Dependency can be read "Class X is an instance of concept type Y."

The relationship of the «`instance of`» Dependency to the (client) Class that is the instance is represented in the «`instance of`» element by the Tag "instance".

The relationship of the «`instance of`» Dependency to the (supplier) Class that is the concept type is represented in the «`instance of`» element with the Tag "type".

## G.3 Categorization

The SBVR term categorization type refers to a concept type whose instances are subtypes of a common base concept. A categorization scheme for the common base concept is a specific set of subtypes that are mutually exclusive. Three stereotypes are introduced to support this notion.

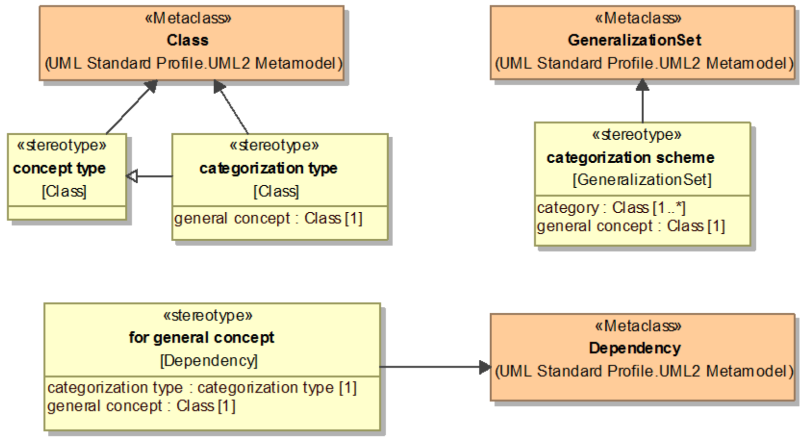


**Figure G.2 - Categorization**

### G.3.1 Stereotype «categorization type»

The stereotype «`categorization type`» characterizes a UML Class as an SBVR categorization type.

A categorization type is similar to a UML Powertype. The instances of a Powertype are *all* the subclasses of a given Class. The instances of a categorization type are *all* the categories (specializations) of a given general concept, which is represented in UML by a Class.

The relationship of the categorization type to the Class that is the general concept that the categorization type is for is represented in the «`categorization type`» element by the Tag "general concept".

By comparison, the extension of categorization type is a particular set of subclasses of a given Class that are mutually exclusive. Only in some cases is the extension of a UML Powertype a set of subclasses that are mutually exclusive, partly because the Powertype necessarily includes all of the subclasses of the categorized Class.

Each categorization type has a «for general concept» Dependency on a 'base class' that is the "common base concept" of the instances.

### G.3.2 Stereotype «for general concept»

The stereotype «for general concept» characterizes a UML Dependency as representing the relationship between a categorization type and the general concept that it categorizes. The Dependency is the diagram element that shows the relationship. The Dependency can be read "Categorization type X is for general concept Y."

The relationship of the «for general concept» Dependency to the (client) categorization type is represented in the «for general concept» element by the Tag "categorization type".

The relationship of the «for general concept» Dependency to the (supplier) Class that is the general concept is represented in the «for general concept» element with the Tag "general concept".

### G.3.3 Stereotype «categorization scheme»

The SBVR term categorization scheme refers to a specific set of categories of a common general concept that are mutually exclusive. The stereotype «categorization scheme» characterizes a UML GeneralizationSet as a categorization scheme.

The relationship of the categorization scheme to the Class that is the general concept that the categorization scheme *is for* is represented in the «categorization scheme» element by the Tag "general concept".

The relationship of the categorization scheme to the Classes that are the mutually exclusive categories that the categorization scheme *includes* is represented in the «categorization scheme» element by the Tag "category".

## G.4 Verb Concepts

The SBVR term verb concept refers to a concept whose instances are states, activities or events. A verb concept is said to have verb concept roles that characterize the participation of individual objects in those states, activities or events.

Verb concepts that involve only one or two participant objects can be represented in UML using Attributes and binary Associations. In a binary Association, the multiplicity on an Association End represents the number of instances of the verb concept that each instance of the other role can participate in, i.e., the number of times an instance of that class can play that role.

In theory, a verb concept involving more than two roles can be represented in UML by an N-ary Association. Support for N-ary associations in UML v2.4 tools is highly variable. For this reason, this specification represents a verb concept with 3 or more participating verb concept roles as a Class with a «verb concept» stereotype. Three stereotypes are introduced to support this approach.

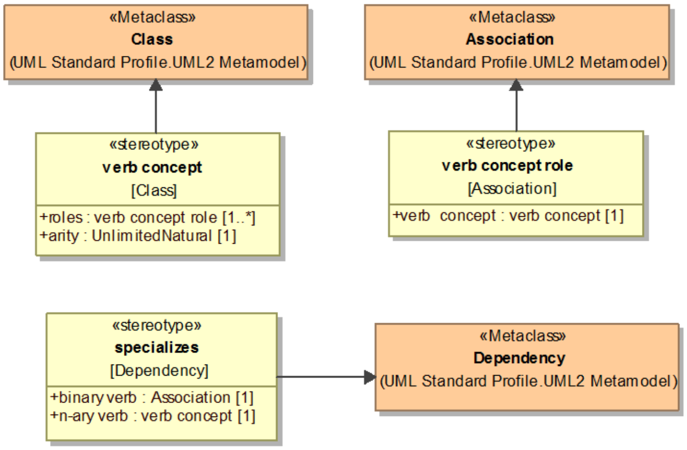


**Figure G.3 - Verb Concept stereotypes**

### G.4.1 Stereotype «verb concept»

The stereotype «verb concept» characterizes a UML Class as an SBVR verb concept. In UML terms, it is a classifier whose instances are states.

Each «verb concept» Class has one «verb concept role» Association for each verb concept role in the SBVR verb concept that it represents.

The set of «verb concept role» Associations for the verb concept are represented in the «verb concept» element by the Tag "roles".

The number of verb concept roles for the verb concept is represented in the «verb concept» element by the Tag "arity".

### G.4.2 Stereotype «verb concept role»

The stereotype «verb concept role» characterizes a UML Association as representing one verb concept role in an SBVR verb concept that is represented by a «verb concept» Class.

Each «verb concept role» Association represents exactly one verb concept role in exactly one SBVR verb concept. Each link that instantiates that Association can be read: In the state (object) X that is the instance of the verb concept Class, the role Y is played by Z, where Y is the association end name on the Association, and Z is the object in the range Class.

One end of the «verb concept role» Association is the «verb concept» Class that represents the verb concept. The other end of the Association is the UML Class that represents the range of the verb concept role. The name of that association end is the placeholder for the verb concept role in the verb concept form.

In a «verb concept role» Association only the association end that refers to the range of the role is navigable, and it always has multiplicity one, because each verb concept role is played exactly once in any one instance of the verb concept.

The relationship of the «verb concept role» Association to the «verb concept» Class is represented in the «verb concept role» element by the Tag "verb concept".

### G.4.3 Stereotype «specializes»

The stereotype «specializes» characterizes a UML Dependency as representing an instance of SBVR concept *specializes* concept, where the narrower concept is a binary verb concept that is represented by a UML Association, and the more general concept is a verb concept with more than two verb concept roles that is represented by a «verb concept» Class. That is, the Dependency can be read "binary verb concept X specializes verb concept Y."

The relationship of the «specializes» Dependency to the (client) binary verb concept is represented in the «specializes» element by the Tag "binary verb".

The relationship of the «specializes» Dependency to the (supplier) «verb concept» Class that is the more general verb concept is represented in the «specializes» element with the Tag "n-ary verb".

**Note**: A binary verb concept can specialize an n-ary verb concept by supplying in its definition a specific thing to play one of the verb concept roles in the n-ary verb concept. In practice, it also constrains the ranges of other verb concept roles in the n-ary verb concept.

# Index of Date Time Designations

(informative)

Commented [EB1]: The text below is unchanged from v1.2. The actual entries must be regenerated by Framemaker (no MS Word feature)

## A



## B



## C

## D

## E



## F

## G

## J, K



## L

## M

## N

**O**

## P

## Q



## R

## S

## T

## U

## V



## W, X



## Y



## Z