WS-Calendar Platform Independent Model (PIM) Version 1.0

Committee Specification 01

09 October 2014

Specification URIs

This version:

http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/cs01/ws-calendar-pim-v1.0-cs01.pdf (Authoritative)
http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/cs01/ws-calendar-pim-v1.0-cs01.html
http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/cs01/ws-calendar-pim-v1.0-cs01.doc

Previous version:

http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/csprd02/ws-calendar-pim-v1.0-csprd02.pdf (Authoritative)
http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/csprd02/ws-calendar-pim-v1.0-csprd02.html
http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/csprd02/ws-calendar-pim-v1.0-csprd02.doc

Latest version:

http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/ws-calendar-pim-v1.0.pdf (Authoritative)
http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/ws-calendar-pim-v1.0.html
http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/ws-calendar-pim-v1.0.doc

Technical Committee:

OASIS Web Services Calendar (WS-Calendar) TC

Chair:

Toby Considine (toby.considine@unc.edu), University of North Carolina at Chapel Hill

Editors:

William Cox (wtcox@coxsoftwarearchitects.com), Individual
Toby Considine (toby.considine@unc.edu), University of North Carolina at Chapel Hill

Additional artifacts:

This prose specification is one component of a Work Product, which also includes:

- XMI (UML in XML) documents representing the UML model described in the specification. XML is authoritative; EAP file is informative: http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/cs01/xmi/

Related work:

This specification is related to:

- *WS-Calendar Version 1.0*. Edited by Toby Considine and Mike Douglass. Latest version. http://docs.oasis-open.org/ws-calendar/ws-calendar/v1.0/ws-calendar-1.0-spec.html

Abstract:

The Platform Independent Model is an abstract model that defines conformance and improves interoperation of calendar and schedule models with each other and with WS-Calendar and Xcal, which are in turn based on IETF RFCs.

This is a Platform Independent Model under the Object Management Group's Model-Driven Architecture. The Platform Dependent Model to which this specification relates is the full model for WS-Calendar as expressed in XML (xCal).

The focus of this Platform Independent Model is on describing and passing schedule and interval information with information attachments.

Status:

This document was last revised or approved by the OASIS Web Services Calendar (WS-Calendar) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-calendar#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/ws-brsp/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (https://www.oasis-open.org/committees/ws-brsp/ipr.php).

Citation format:

When referencing this specification the following citation format should be used:

[WS-Calendar-PIM-v1.0]

WS-Calendar Platform Independent Model (PIM) Version 1.0. Edited by William Cox and Toby Considine. 09 October 2014. OASIS Committee Specification 01. http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/cs01/ws-calendar-pim-v1.0-cs01.html. Latest version: http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/ws-calendar-pim-v1.0.html.

Notices

Table of Contents

Table of Figures 6
List of Tables 6
1 Introduction 7
1.1 Terminology 7
1.2 Normative References 7
1.3 Non-Normative References 7
1.4 Namespace 8
1.5 Naming Conventions 8
1.6 Editing Conventions 9
2 Architectural Context [Non-Normative] 10
2.1 Architectural Basis for the PIM 10
2.2 Standards for Representation of Time 10
2.3 Service-Oriented Architecture and the PIM 10
2.4 Model Driven Architecture 10
2.5 The PIM and the WS-Calendar PSM 11
2.6 Expression of the PIM UML Model 11
2.7 Structure of the PIM Model and Specification 12
3 WS-Calendar PIM Terminology and Semantics 13
3.1 Time Intervals and Collections of Time-Related Intervals 13
4 The Platform-Independent Model 17
4.1 Overview of the PIM 18
4.1.1 Model Diagram 18
4.1.2 Discussion 18
4.2 Classes for Date and Time, Duration, and Tolerance 19
4.2.1 Model Diagram 19
4.2.2 Discussion 20
4.2.3 Relationship to other PIM Components 20
4.3 The Interval Class 21
4.3.1 Model Diagram 21
4.3.2 Discussion 21
4.3.3 Relationship to other PIM Components 21
4.4 Payload Attachment to an Interval 22
4.4.1 Model Diagram 22
4.4.2 Discussion 22
4.4.3 Relationship to other PIM Components 22
4.5 The Gluon Class 22
4.5.1 Model Diagram 23
4.5.2 Discussion 23
4.5.3 Relationship to other PIM Components 23
4.6 Relationships among Gluons and Intervals 24
4.6.1 Model Diagram 25
4.6.2 Discussion 26
4.6.3 Relationship to other PIM Components 26

4.7 The Availability Package 26
4.7.1 Model Diagram 27
4.7.2 Discussion 27
4.7.3 Relationship to other PIM Components 27
5 Rules for WS-Calendar PIM and Referencing Specifications 28
5.1 Inheritance in WS-Calendar PIM 28
5.2 Covarying Elements 28
5.3 Specific Attribute Inheritance 28
6 Conformance 30
6.1 Conformance for Specifications Claiming Conformance to WS-Calendar PIM 30
6.2 General Conformance Issues (Non-Normative) 30
6.3 Conformance of Intervals 30
6.3.1 Intervals and Gluons 30
6.3.2 Other Attributes 30
6.4 Conformance of Bound Intervals and Sequences 31
6.5 Security Considerations (Non-Normative) 31
7 Examples using the PIM (Non-Normative) 32
7.1 Related Intervals 32
7.2 A Meeting Schedule 32
Appendix A. Acknowledgments 35
Appendix B. Revision History 36
Appendix C. PIM to WS-Calendar PSM Transformation 38
C.1 General Transformations 38
C.2 Specific Transformations 38
C.2.1 Transformation for DateTime and Duration Types 38
C.2.2 Transformation for Tolerance Type 40
C.2.3 Transformation for Interval and Gluon Types 41
C.2.4 Transformation for Relationships 43
C.2.5 Transformation for Vavailability and FreeBusy 44
Appendix D. PIM to IEC TC57 CIM Intervals and Sequences (Non-Normative Example) 46

Table of Figures

Figure 4-1 The Complete WS-Calendar PIM UML Model. Abstract classes have violet background. 18
Figure 4-2 DateTimeType, DurationType, and ToleranceType 19
Figure 4-3 IntervalType 21
Figure 4-4 Attaching a Payload to an Interval 22
Figure 4-5 Gluons, Intervals, and Relationship Links 23
Figure 4-6 Temporal Relationships 25
Figure 4-7 Temporal Relationship--startToFinish Negative 0.5 Gap 25
Figure 4-8 RelationLinkType and Relationship Types 25
Figure 4-9 Vavailability and Availability Recurrence Rules 27
Figure 7-1 PIM Expression of WS-Calendar Examples 3-05 32
Figure 7-2 Simple Meeting Schedule 34
Figure 7-3 PIM Source Classes for DateTimeType and Duration Types 39
Figure 7-4 WS-Calendar Target Classes 39
Figure 7-5 PIM Source Class for ToleranceType 40
Figure 7-6 WS-Calendar Target Classes for Tolerance Type 40
Figure 7-7 PIM IntervalType and GluonType 41
Figure 7-8 WS-Calendar Target IntervalType and GluonType 42
Figure 7-9 PIM RelationLinkType, LinkType, RelationshipType, and TemporalRelationshipType 43
Figure 7-10 PIM Vavailability Package Classes 45
Figure 7-11 Vavailability Package from iCalendar-availability-extension 45

List of Tables

Table 3-1: Semantics: Foundational Elements 13
Table 3-2: Semantics: Relations, Limits, and Constraints 14
Table 3-3: Semantics: Inheritance 14
Table 3-4: Semantics: Describing Intervals 15
Table 7-1 PIM to PSM Mapping for DateTimeType and Duration Types 39
Table 7-2 PIM to PSM Mapping for ToleranceType 40
Table 7-3 PIM to PSM Mapping for IntervalType and GluonType 42
Table 7-4 PIM to PSM Mapping for Attributes of PIM RelationshipType and TemporalRelationshipType 44
Table 7-5 PIM to PSM Mapping for Enumeration Members 44

1 Introduction

All text is normative unless otherwise labeled. Notes and examples are non-normative; see Section 1.6 Editing Conventions.

1.1 Terminology

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

1.2 Normative References

[ISO8601] ISO (International Organization for Standardization). Data elements and interchange formats -- Information interchange -- Representation of dates and times, Edition 3, 3 December 2004, (ISO 8601:2004)

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, Uniform Resource Identifier (URI): Generic Syntax, STD 66, RFC 3986, January 2005. http://www.ietf.org/rfc/rfc3986.txt

[RFC2119] Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt, BCP 14, RFC 2119, March 1997.

[RFC5545] Desruisseaux, B. Internet Calendaring and Scheduling Core Object Specification (iCalendar), http://www.ietf.org/rfc/rfc5545.txt, RFC 5545, September 2009

[UML] OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.4.1, Object Management Group. http://www.omg.org/spec/UML/2.4.1/Infrastructure and OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, http://www.omg.org/spec/UML/2.4.1/Superstructure, Object Management Group, August 2011

[xCal] Daboo, C., Douglass, M., and S. Lees, xCal: The XML format for iCalendar, http://tools.ietf.org/html/rfc6321, IETF RFC 6321, August 2011.

[XMI] MOF 2.0/XMI Mapping Specification, v2.1, September 2005, Object Management Group, http://www.omg.org/spec/XMI/2.1/ [1]

1.3 Non-Normative References

[BPEL] Web Services Business Process Execution Language Version 2.0, 11 April 2007, OASIS Standard. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[BPMN] Business Process Model and Notation (BPMN) Version 2.0, Object Management Group, Version 2.0, http://www.omg.org/spec/BPMN/2.0/, January 2011

[EnergyInterop-v1.0] Energy Interoperation Version 1.0. Edited by Toby Considine. 11 June 2014. OASIS Standard. http://docs.oasis-open.org/energyinterop/ei/v1.0/os/energyinterop-v1.0-os.html. Latest version: http://docs.oasis-open.org/energyinterop/ei/v1.0/energyinterop-v1.0.html. PDF is authoritative.

[Enterprise Architect] Sparx Enterprise Architect 10.0, used to produce **[UML]** 2.4.1 diagrams, EAP and **[XMI]** version 2.1 files, http://sparxsystems.com/.

[IANA] The Internet Assigned Numbers Authority, http://www.iana.org.

[1] The UML tools used by the TC support version 2.1, which is not the most recent as of this date.

[IEC CIM] IEC 61968/61970, International Electrotechnical Commission, collection of specifications, various dates, http://www.iec.ch [2]

[MDA-Overview] The Architecture of Choice for a Changing World, Object Management Group, http://www.omg.org/mda/

[MDA] OMG Model Driven Architecture Specifications, Object Management Group, http://www.omg.org/mda/specs.htm

[PIM Examples] Examples for WS-Calendar Platform-Independent Model (PIM) Version 1.0, OASIS Committee Technical Note, in progress.

[Relationships] M. Douglass, Support for Icalendar Relationships, http://tools.ietf.org/html/draft-douglass-ical-relations-02, IETF Internet Draft Version 02, January 7, 2014

[SOA-RAF] Reference Architecture Foundation for Service Oriented Architecture Version 1.0**,** 04 December 2013. OASIS Committee Specification. http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html PDF is authoritative.

[SOA-RM] OASIS Reference Model for Service Oriented Architecture 1.0,October 2006. OASIS Standard. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html

[Vavailability] C. Daboo, M. Douglass, Calendar Availability, http://tools.ietf.org/search/draft-daboo-calendar-availability-05, IETF Internet Draft Version 05, January 30, 2014

WS-Calendar] WS-Calendar Version 1.0. Edited by Toby Considine and Mike Douglass. 30 July 2011, OASIS Committee Specification. http://docs.oasis-open.org/ws-calendar/ws-calendar-spec/v1.0/cs01/ws-calendar-spec-v1.0-cs01.html (PDF is authoritative)

[XMLSchema] W3C XML Schema Definition Language (XSD) 1.1, World Wide Web Consortium, Part 1: Structures, S. Gao, C. M. Sperberg-McQueen, H. S. Thompson, N. Mendelsohn, D. Beech, M. Maloney, Editors, W3C Recommendation, 5 April 2012, http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/. Latest version available at http://www.w3.org/TR/xmlschema11-1/. Part 2: Datatypes, D. Peterson, S. Gao, A. Malhotra, C. M. Sperberg-McQueen, H. S. Thompson, P. Biron, Editors. W3C Recommendation, 5 April 2012, http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/. Latest version available at http://www.w3.org/TR/xmlschema11-2/

1.4 Namespace

There are no XML namespaces defined in this specification.

1.5 Naming Conventions

This specification follows a set of naming conventions for artifacts defined by the specification, as follows:

For the names of attributes in UML classes the names follow the lower camelCase convention, with all names starting with a lower case letter. For example, an attribute name might be

```
temporalRelationship
```

The names of UML classes follow the upper CamelCase convention with all names starting with an Upper case letter followed by "Type".

```
TemporalRelationshipType
```

The UML Primitive Type *String* **[UML, *Infrastructure*]**[3] is used in this specification.

---

[2] In this specification, the relevant parts are *IEC 61968-9, Edition 2.0, October, 2013, http://webstore.iec.ch/webstore/webstore.nsf/ArtNum_PK/48719?OpenDocument* *and IEC 61970-301, Edition 5.0, December 2013*, http://webstore.iec.ch/webstore/webstore.nsf/ArtNum_PK/49080?OpenDocument

1.6 Editing Conventions

For readability, UML attribute names in tables appear as separate words. The actual names are lowerCamelCase, as specified above, and do not contain spaces.

Attribute and type names are usually in an *italic* face.

All items in the tables not marked as "optional" are mandatory.

Information in the "Specification" column of tables is normative. Information appearing in the "Note" column is non-normative.

Text indicated as "Note" are non-normative.

All sections explicitly described as examples are non-normative.

All examples with gray highlight are non-normative.

All Appendices are non-normative.

[3] See http://www.omg.org/spec/UML/20110701/PrimitiveTypes.xmi

2 Architectural Context [Non-Normative]

In this section we discuss the context in which this specification was developed, its purpose, and selected applications.

2.1 Architectural Basis for the PIM

The PIM is defined as a more abstract model for describing and communicating schedules as defined in **[WS-Calendar]**, **[EMIX]**, **[EnergyInterop-v1.0]**, **[OBIX]**, and **[SPC201]**, among many others. This expression uses typical ways of expressing schedule, linked lists, directed graphs, and is consistent with algorithms for graph, list, and schedule management.

In summary, there are several anticipated architectural benefits of the PIM:

1. Expression of schedules in a common manner showing temporal structures and taking advantage of differing views of a single schedule
2. Relocatable subroutines that may be used dynamically at run time
3. Automatable transformations between the abstract and concrete schedules in the PIM and WS-Calendar respectively
4. Broader use of scheduling concepts in other domains and PSMs allowing automatable transformations across other domains

Schedule and values attached to time intervals in schedule are fundamental to planning and carrying out operations is most domains. The WS-Calendar PIM provides a common model for expressing and managing such schedules.

2.2 Standards for Representation of Time

We rely on **[ISO8601]** for description of date, time, and duration. Many of the concepts in that standard are well known to users of iCalendar **[RFC5545]** and XML Schema **[XMLSchema]**, both of which share similar but slightly different subsets of the expressive power of **[ISO8601]**. For example, we define a conformed string for an attribute called *ISO8601Duration* which differs in detail from the perhaps more familiar XML Schema and iCalendar.

PSMs may restrict or profile time expressions in the PIM. For example, many industrial control systems define time intervals with start and end time, which is a conformant 8601 definition. For purposes of relocatable schedules, as used in e.g. **[EMIX]** and **[EnergyInterop-v1.0]** this PIM uses start time and duration only, another conformant 8601 definition.

2.3 Service-Oriented Architecture and the PIM

WS-Calendar PIM is an information model that may be used to define service request and response message payloads. For that purpose it assumes a background of definitions and of roles, names, and interaction patterns. Non-normative examples may use terminology defined in the OASIS Standard *Reference Model for Service Oriented Architecture* **[SOA-RM]**.

Service-Oriented Architecture comprises not only the services and interaction patterns, but also the information models that support those services and make the actions meaningful. The WS-Calendar PIM is such an information model for expressing schedule and time related information in a consistent manner and to permit easy transformation or adaptation into IETF iCalendar related specifications and among Platform-Specific Models based on this PIM.

2.4 Model Driven Architecture

The Object Management Group's Model Driven Architecture **[MDA-Overview][MDA]** provides a framework to describe relationships between Unified Modeling Language **[UML]** models.

An instance of MDA has two classes of models:

- A single *Platform-Independent Model*, abbreviated *PIM* (pronounced as though *spelled pim*)
- One or more *Platform-Specific Models*, abbreviated *PSM* (pronounced as though spelled *pism*)

The PIM typically captures the more abstract relationships, clarifying the architecture. Each PSM is bound to a particular *platform*.

The art of establishing an MDA includes defining platforms and a PIM and PSMs, to solve interesting important and useful problems. Artifacts expressed in different PSMs may more readily be exchanged and understood with reference to the related PIM, making interoperation simpler and semantics more free from irrelevant detail.

2.5 The PIM and the WS-Calendar PSM

In this specification we define a PIM or Platform-Independent Model with respect to which the **[WS-Calendar]** specification may be treated as a PSM or Platform-Specific Model; the platform may be considered to be iCalendar **[RFC5545]**, **[xCal]**, and **[Vavailability]**.

We use "the PIM" to mean "the WS-Calendar PIM" in this specification.

[iCalendar] uses a set of definitions and a platform, developed over many years and much use, to express relationships, times, events, and availability. The expression is very simple, but in the aggregate relatively complex and less suitable to UML expression—the several key types (components) have sets of values, types, and parameters associated with them in a relatively flat hierarchy.

This PIM addresses the key **[WS-Calendar]** abstractions in a manner that allows for a better understanding of the nature and information model for those abstractions. Our purpose is to create a more abstract model of the key concepts in WS-Calendar for easier use in application development, standardization, and interoperation. As such, this PIM does not normatively reference any PSM, including but not limited to **[WS-Calendar]**.

The MDA presumes transformations from UML models to UML models. The UML model for **[WS-Calendar]** is structured very differently from that of the PIM. We describe the transformation in detail in non-normative Appendix C.

This specification does not rely on any specific MDA tooling or environments to be useful.

2.6 Expression of the PIM UML Model

The PIM is a **[UML]** model. We represent the PIM as a normative **[XMI]** serialization of the PIM UML model. The model itself is described using **[Enterprise Architect]**; an Enterprise Architect Project file is part of this work product but is non-normative. Many modeling tools use XMI serialization for model exchange.

The terminology for attributes of an object, and how to describe an object or type differs between **[XMLSchema]** and **[UML]**. Attributes of a class in UML that is expressed in standards mappings to XML Schema are called either attributes (expressed in *name=value* format in XML) or elements. Since this specification is based on UML, we use the term *attribute* throughout.[4]

The PIM model is constrained, and by applying semantic rules the model allows succinctly described relocatable graphs of Intervals

For example, an instance of *IntervalType* (see Figure 4-3 IntervalType) might have only *duration*; the PIM, however, describes *duration* as optional (cardinality 0..1). Rules in this specification show how a specific representation is to be interpreted, typically by inheriting values from elsewhere. Conceptually, the actual values depend on the context and applied rules.

An Interval notionally has a start time, but that also is optional in the PIM. Finally, an Interval does not have an end time (expressed in Figure 4-3 as dtEnd of cardinality 0. We keep the dtEnd attribute for ease

[4] There are UML stereotypes to express the nature of an XML Schema export, indicating whether a UML attribute should be represented as an XSDattribute or XSDelement.

of use in PSMs and for intermediate stages of mapping into the canonical start and duration model, as well as mapping into and from models that define intervals with all three of start, end, and duration.

These characteristics are as defined in **[WS-Calendar]** and describe an abstract Interval with at most a start time and duration. This is in contrast to some historical models that require each interval to contain a start and end time, or occasionally start, end, and duration. The added flexibility of relocatable sets or schedules comprised of Intervals and Gluons makes the expression of such a relocatable schedule easy and reusable, thus permitting a powerful abstraction to be applied to all sorts of scheduling expressions. In addition the mapping capability to and from the PIM allows interoperation with systems with less conveniently relocatable intervals.

2.7 Structure of the PIM Model and Specification

The PIM consists of a small number of key classes with a sub-package for the Availability **[Vavailability]** abstractions.[5] We have not otherwise subdivided the core model, but expect that conforming specifications and implementations may claim conformance to sub-parts of the PIM, e.g. to only the Interval.

We encourage use of the entire PIM, but understand that some aspects of the abstract model may be more complex than needed to address specific problems. We consider such profiles of the PIM to themselves be Platform-Specific Models.

We generally take the names for abstractions in the PIM from the names in **[WS-Calendar]** to simplify implementations and mappings.

Many values in the XML Serialization **[xCAL]** of iCalendar are conformed strings, that is, strings that meet specific defined patterns. We require similar standardized formats for conformed strings, and record the type in the PIM using the UML primitive type *String*. This allows easy transformation between this PIM and the PSMs. We include references to **[ISO8601]** and other specifications in the comments in the model.

[5] Note: The Vavailability definition is in process in the IETF.

3 WS-Calendar PIM Terminology and Semantics

WS-Calendar PIM semantics are defined in this section. The terminology aligns closely with that is **[WS-Calendar]**.

Note: This specification and **[WS-Calendar]** share the same semantics and terminology, which allows easier exchange of information across execution environments as well as consistency across Platform Specific Models related to this specification.

The normative definitions of terms are included here in Section 3.

3.1 Time Intervals and Collections of Time-Related Intervals

We begin with specialized terminology for the segments of time, and for groups of related segments of time. These terms are defined in Table 3-1 through Table 3-4 below.

Table 3-1: Semantics: Foundational Elements

Time Segment	Definition
Duration	Duration is the length of a time interval. In the PIM the value set from **[ISO8601]** is used; informally there are several additional representations for duration in the PIM compared to either **[xCal]** or **[XMLSchema]** but all those representations are included. See Section 4.2.
Interval	An Interval has as attributes a single Duration derived from **[ISO8601]**). An Interval may be part of a Sequence. An entire Sequence can be scheduled by scheduling a single Interval in that sequence. For this reason, Intervals are defined through Duration rather than through dtStart or dtEnd.
Sequence	A Sequence is a set of Intervals with defined temporal relationships. Sequences may have gaps between Intervals, or may be in parallel or overlapping. A Sequence is re-locatable, i.e., it does not have a specific date and time at which it starts or finishes. A Sequence may consist of a single Interval. A Sequence may optionally include a Lineage. A Sequence CAN be scheduled or applied multiple times through repeated reference by different Gluons that give specific start time to the Sequence.
Partition	A Partition is a set of consecutive Intervals without gaps or overlap among them. The Partition includes the trivial case of a single Interval. Partitions MAY be used to define a single service or value set that varies over time (a time series). Examples include energy prices over time and energy usage over time.
Gluon	A Gluon influences the serialization of Intervals in a Sequence, though inheritance and through schedule setting. The Gluon is similar to the Interval, but has no effect beyond that of a reference until the Gluon is applied to a referenced Interval or Sequence.
Artifact	An Artifact is the information attached to, and presumably that occurs during or is relevant to the associated Interval. The Artifact is a placeholder. The contents of the Artifact are not specified here; rather the Artifact is an abstract type **[UML]** that provides an extension base. Artifacts MAY inherit elements as do Intervals within a Sequence. A Conforming specification MUST describe where and why its inheritance rules differ from those in this specification.

The PIM works with groups of Intervals that have relationships between them. These relations constrain the final description for a schedule or a schedule-based service. Relationships can control the ordering of

Intervals in a Sequence. They can describe when a service can be, or is prevented from, being invoked. They establish the parameters for how information will be shared between elements using Inheritance.

The terminology for these relationships is defined in Table 3-2.

Table 3-2: Semantics: Relations, Limits, and Constraints

Term	Definition
Link	The Link is used by one PIM object to reference another. A link can reference either an internal object, within the same calendar, or an external object in a remote system.
Relationship	Relationships link between Components for Binding. ICalendar defines several relationships, but PIM uses only the CHILD relationship, and that only to bind Gluons to each other and to Intervals.
Temporal Relationship	Temporal Relationships extend the **[RFC5545]** Relationships to define how Intervals become a Sequence by creating an order between Intervals. The Predecessor Interval includes a Temporal Relation, which references the Successor Interval. When the start time and Duration of one Interval is known, the start time of the others can be computed through applying Temporal Relations.
Availability	Availability expresses the range of times in which an Interval or Sequence can be Scheduled. Availability often overlays or is overlaid by Busy. Availability can be Inherited.
Busy	Busy expresses the range of times in which an Interval or Sequence cannot be Scheduled. Busy often overlays Availability. Busy can be Inherited.
Child, Children	The CHILD relationship type (*RelationshipType*) defines a logical link (via URI or UID) from parent object to a child object. A Child object is the target of one or more CHILD relationships and may have one to many Parent objects.
Parent [Gluon]	A Gluon (in a Sequence) that includes a CHILD relationship parameter type (*RelationshipType*) defines a logical link (via URI or UID) from parent object to a child object. A Parent Component contains one or more CHILD Relationships.

WS-Calendar describes how to modify and complete the specification of Sequences. WS-Calendar calls this process Inheritance and specifies a number of rules that govern inheritance. Table 3-3 defines the terms used to describe inheritance, with rewording to address this PIM.

Table 3-3: Semantics: Inheritance

Term	Definition
Lineage	The ordered set of Parents that results in a given inheritance or execution context for a Sequence.
Inheritance	Parents bequeath information to Children that inherit them. If a child does not already possess that information, then it accepts the inheritance. WS-Calendar specifies rules whereby information specified in one informational object is considered present in another that is itself lacking expression of that information. This information is termed the Inheritance of that object.
Bequeath	A Parent Bequeaths attributes (Inheritance) to its Children.
Inherit	A Child Inherits attributes (Inheritance) from its Parent.

Term	Definition
Covarying Attributes	Some attributes are inherited as a group. If any member of that group is expressed in a Child, all members of that group are deemed expressed in that Child, albeit some may be default values. These characteristics are called covarying or covariant. A parent bequeaths covarying characteristics as a group and a child accepts or refuses them as a group.
Decouplable Attributes	Antonym for Covarying Attributes. Decouplable Attributes can be inherited separately.

As Intervals are processed, as Intervals are assembled, and as inheritance is processed, the information conveyed about each element changes. When WS-Calendar is used to describe a business process or service, it may pass through several stages in which the information is not yet complete or actionable, but is still a conforming expression of time and Sequence. Table 3-4 defines the terms used when discussing the processing or processability of Intervals and Sequences.

During the life cycle of communications concerning Intervals, different information may be available or required. For service performance, Start Duration and the Attachment Payload must be complete. These may not be available or required during service advertisement or other pre-execution processes. Table 3-4 defines the language used to discuss how the information in an Interval is completed.

Table 3-4: Semantics: Describing Intervals

Term	Definition
Designated Interval	An Interval that is referenced by a Gluon is the Designated Interval for a Series. An Interval can be Designated and still not Anchored.
Anchored	An Interval is Anchored when it includes a Start or End, either directly or through Binding. A Sequence is Anchored when its Designated Interval is Anchored.
Unanchored	An Interval is Unanchored when it includes neither a Start nor an End, either internally, or through Binding. A Sequence is Unanchored if its Designated Interval Unanchored. *Note: a Sequence that is re-used may be Unanchored in one context even while it is Anchored in another.*
Binding	Binding is the application of information to an Interval or Gluon, information derived through Inheritance or through Temporal Assignment.
Bound Attribute	A Bound Attribute refers to an Attribute and its Value after Binding, e.g., a Bound Duration.
Bound Interval	A Bound Interval refers to an Interval and the values of its Elements after Binding.
Bound Sequence	A Bound Sequence refers to a Sequence and the values of its Intervals after Binding.
Partially Bound	Partially Bound refers to an Interval or a Sequence which is not yet complete following Binding, i.e., the processes cannot yet be executed.
Fully Bound	Fully Bound refers to an Interval or Sequence that is complete after Binding, i.e., the process can be unambiguously executed when Anchored.
Unbound	An Unbound Interval or Sequence is not complete, and must receive inheritance to be fully specified. A Sequence or Partition is Unbound if it contains at least one Interval that is Unbound.

Term	Definition
Constrained	An Interval is Constrained if it is not Anchored and it is bound to one or more Availability or Free/Busy elements
Temporal Assignment	Temporal Assignment determines the start times of Intervals in a Sequence through processing of their Durations and Temporal Relations.
Scheduled	A Sequence or Partition is Scheduled when it is Anchored, Fully Bound, and the schedule is ready to be used.
Unscheduled	An Interval is Unscheduled if it is not Anchored, nor is any Interval in its Sequence Anchored. A Sequence or Partition is Unscheduled if none of its Intervals, when Fully Bound, is Scheduled.
Predecessor Interval	A Predecessor Interval includes a Temporal Relation that references a Successor Interval.
Successor Interval	A Successor Interval is one referred to by a Temporal Relationship in a Predecessor Interval.
Antecedent Interval(s)	Antecedents are an Interval or set of Intervals that precede a given Interval within the same Sequence
Earliest Interval	The set of Intervals at the earliest time in a given Sequence
Composed Interval	A Composed Interval is the virtual Interval specified by applying inheritance through the entire lineage and into the Sequence in accord with the inheritance rules. A Composed Interval may be Bound, Partially Bound, or Unbound.
Composed Sequence	A Composed Sequence is the virtual Sequence specified by applying inheritance through the entire lineage and into the Sequence in accord with the inheritance rules. A Composed Sequence may be Bound, Partially Bound, or Unbound.
Comparable Sequences	Two Sequences are Comparable if and only if the Composed version of each defines the same schedule.

4 The Platform-Independent Model

In this section we define the PIM.

Each subsection has an introduction, a diagram, and discussion of the relationship of the components to the rest of the PIM.

This Platform-Independent Model (PIM) **[MDA]** describes an abstraction from which the Platform-Specific Model (PSM) of **[WS-Calendar]** and other models can be derived. The intent is twofold:

(1) To define an abstraction for calendar and schedule more in the style of web services descriptions, which may be used directly, and
(2) To define the PIM as a model allowing easy transformation or adaptation between systems using the family of WS-Calendar specifications (such as [WS-Calendar], [xCal], [iCalendar]) as well as those addressing concepts of time intervals and Sequences (such as **[IEC CIM]**, **[EnergyInterop-v1.0]**, and **[EMIX]**.

The following subsections each contain a description of the relevant portions of the model, addressing in turn

- Section 4.1 Overview of the PIM
- Section 4.2 Classes for Date and Time, Duration, and Tolerance
- Section 4.3 The Interval
- Section 4.4 Payload Attachment to an Interval
- Section 4.5 The Gluon
- Section 4.6 Relationships among Gluons and Intervals
- Section 4.7 The Availability Package

4.1 Overview of the PIM

4.1.1 Model Diagram

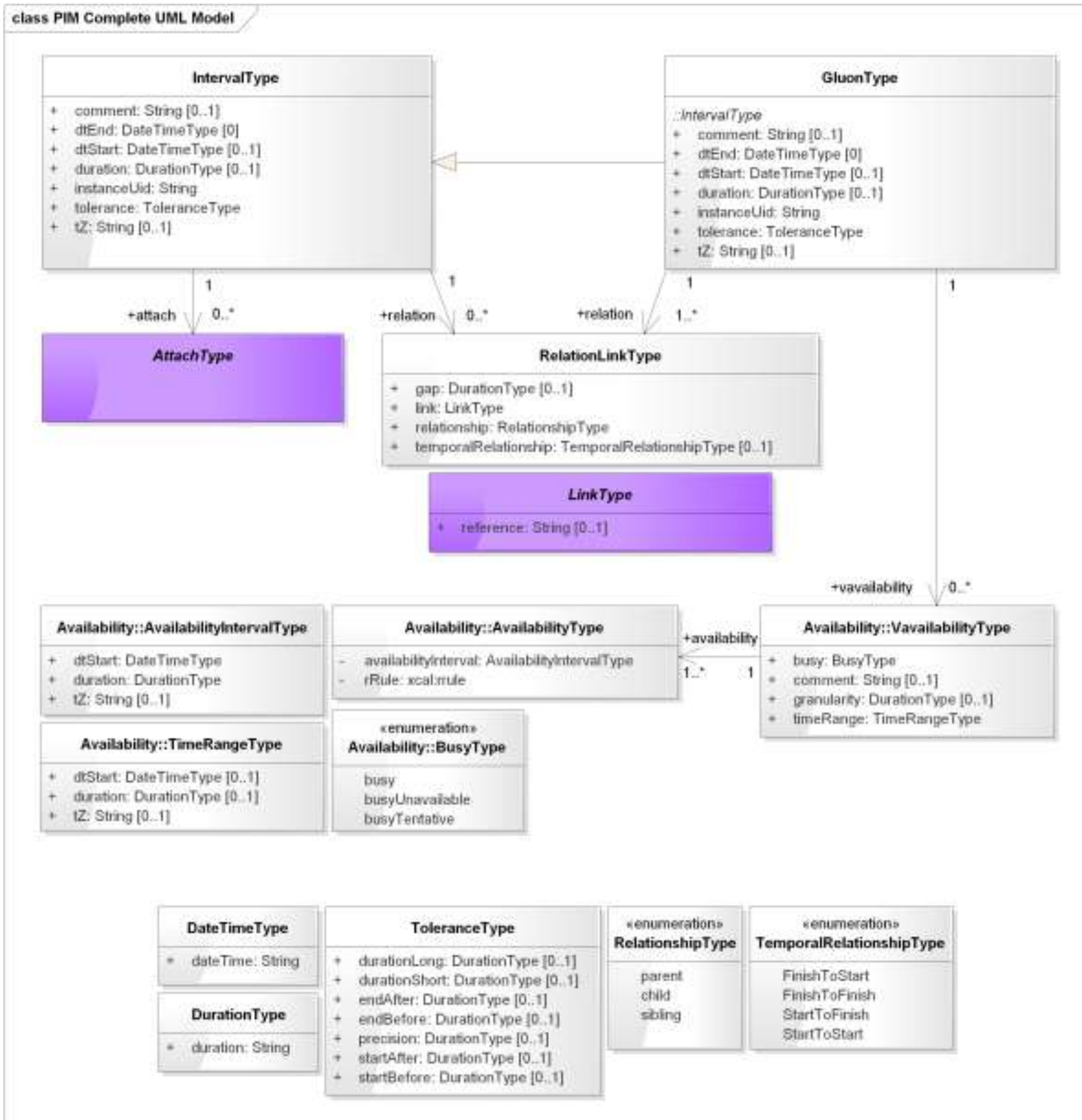

Figure 4-1 The Complete WS-Calendar PIM UML Model. Abstract classes have violet background.

4.1.2 Discussion

Primitive types in the PIM express fundamental information related to date, time, and duration, and follow **[RFC5545] [ISO8601]** and are a superset of those expressed in **[iCalendar]**. Most are conformed versions of the **[UML]** primitive type *String*.

Associations in the PIM are directional, but profiles and PSMs derived or derivable from the PIM MAY have non-directional associations, or vary the direction of associations to fit their particular platform(s) and purposes.

Note: non-directional associations present a barrier to serializability; we expect that PSMs typically would use directional associations unless their purpose is to derive further PSMs.

The cardinality for all attributes and associations' cardinality is specified in the PIM. Profiles and PSMs with respect to the PIM MAY have different cardinality.

Attachments are made via the abstract class *AttachType* as described in Section 4.4.

We have used the **[RFC5545]** and **[ISO8601]** attribute, type, parameter, and value names wherever possible for ease of mapping to that terminology.

Per **[ISO8601]** a fully bound Interval can be described by any two of

- *dtStart*—the date & time for the start of the Interval
- *dtEnd*—the date & time for the end of the Interval
- *duration*—the duration of the interval

In the PIM UML model, the three key values for an interval, only two of which are required in fully bound Intervals, are each optional. This permits a conforming PSM to have zero or more of the three key values. The PIM generally requires that at most dtStart and duration are used to allow relocatable schedules.

The Rules in Section 5 describe how information for a bound interval is determined. *GluonType* is a subclass of *IntervalType* but has a more restrictive cardinality for *dtEnd* and for *relation*.

Availability is a separate package in the model; classes from that package have names starting *Availability::* in the diagrams.

4.2 Classes for Date and Time, Duration, and Tolerance

In this section we introduce key concepts and expressions for time including

- *DateTime*
- *DurationType*
- *ToleranceType*

Relationships are described in Section 4.5.

No timing of events, whether descriptive or prescriptive, can be perfectly accurate within the limits of measurement of real systems. Tolerance is an optional attribute that applies to the duration, allowing full flexibility in the description of permissible or expected variation in duration.

The containing Interval might start early or late, end early or late, or have a duration that may be short or long with respect to the nominal value. The *precision* in *ToleranceType* is a *DurationType* that expresses the precision for tolerances.[6]

4.2.1 Model Diagram

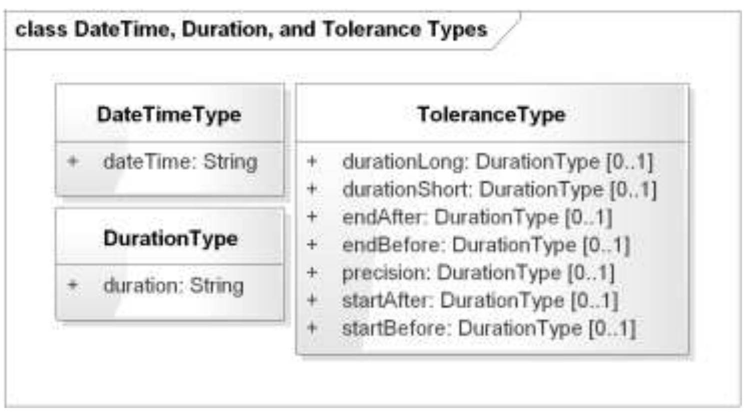

Figure 4-2 DateTimeType, DurationType, and ToleranceType

[6] This differs from *granularity* in *Availability::VavailabilityType*, which describes the availability interval length.

All DateTime and duration values are expressed as conformed strings, that is, the type is *String* and the content of the string determines respectively the date, time, and the duration.

The values of the following SHALL be expressed as conformed strings as described in the normative reference **[RFC5545]**; the optional sign for *DurationValueType* MUST be available in PIM conformed strings for *DurationType*, and the allowable patterns excluding sign MUST conform to **[ISO8601]** section

- DateTime (Section 3.3.5, Date-Time)
- DurationValueType (Section 3.3.6, Duration)

The class *ToleranceType* is comprised of a set of optional attributes of *DurationType*. Tolerances can be expressed in any combination.

The String values for any attribute of *ToleranceType* SHALL be non-negative or a minus sign SHALL be ignored.

A PSM SHALL state rules for non-negative *ToleranceType* attributes in their conformance statement.

PSMs SHOULD specify that the cardinality of *tolerance* MUST be zero if *tolerance* is empty.

Note: The complexity of rules addressing the relationships of tolerances in start, end, and duration will likely lead to implementation-specific rules limiting the concurrent uses of tolerance attributes.

It is RECOMMENDED that a PSM include consistency requirements and limitations on the attributes of ToleranceType that might be used. It is RECOMMENDED that profiled sets of tolerances be specified by a PSM. PSMs MUST document in their conformance statement any consistency requirements, limitations on, and profiled sets of tolerances.

For example, *startAfter* = PT5M and *startBefore* = PT10M indicates that the associated action or the interval to which *AttachType* applies may start in the range from ten minutes before the indicated *dtStart* to five minutes after the indicated *dtStart*.

Tolerances can allow (e.g.) randomization of intervals to ensure that certain activities do not occur “simultaneously.” Continuing the example, additional deployment semantics for randomization might apply to that 15-minute interval.

PSMs MAY include assumptions or explicit statement of e.g. probability density functions or other indications of expected behavior. Such assumptions and/or explicit statements SHALL be included in the conformance statement.

Tolerances also express information about schedules that enables the application of optimization techniques both across and within schedules.

4.2.2 Discussion

These concepts are based on **[ISO8601]** and are as expressed in **[iCalendar]** as conformed strings. It is important to note that *DurationType* is identical to *neither* the XML Schema Specification **[XMLSchema]** *Duration*[7] nor the **[xCal]** and **[iCalendar]** specification for duration.

PSMs MAY express DurationType differently; if so the differences MUST be described in their conformance statement.

4.2.3 Relationship to other PIM Components

These concepts are pervasive in the WS-Calendar PIM. The fundamental understanding of time and duration must be consistent and identical to that in **[iCalendar]** for clean interoperation and transformation. Documentation of any differences in expression MUST be included in the conformance

---

[7] While **[iCalendar]**, **[WS-Calendar],** and this PIM conform to **[ISO8601]**, only this PIM requires all standard notations from 8601 as well as sign for duration. Likewise **[XMLSchema]** does not include the full specification in **[ISO8601]**. There are duration strings included in **[XMLSchema]** that are not in **[iCalendar]** and *vice versa*.

statement for any PSM claiming conformance to this PIM. Moreover, a mapping MUST be provided both directions between the types defined here and those in a PSM claiming conformance.

4.3 The Interval Class

The Interval is fundamental—a bound interval starts at a particular time, runs for a specific duration, and ends at a particular time. This is reflected in Figure 4-3.

But there are many possible standards-based expressions of a time interval, and significant differences in relocatability of schedules including Intervals depending on choices made in representation.

We describe the PIM representation in this section.

4.3.1 Model Diagram

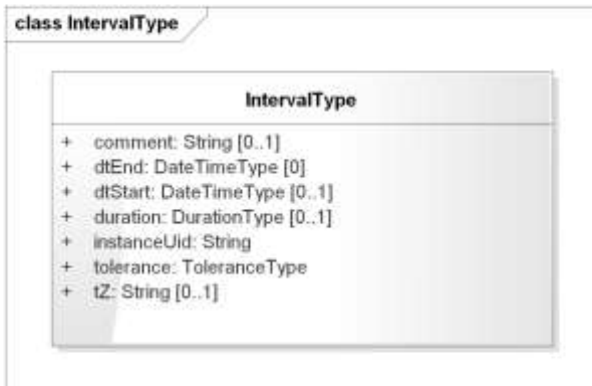

Figure 4-3 IntervalType

4.3.2 Discussion

Class IntervalType is the model for a time interval; while logically any two or three of the set {*dtStart*, *dtEnd*, and *duration*} can express an interval, there are significant advantages to adopting a single canonical form, particularly one where the semantics are cleanly expressed. Intervals may be, and are, expressed many ways; the PIM requires a specific expression that includes start time and duration but not end time.[8]

Individual PSMs may use different expressions, but SHOULD recognize in their design that relocation and scheduling of sets of intervals is a very common operation; as we will show later, an entire schedule of Intervals in this WS-Calendar PIM can be scheduled with a single operation, whereas in other representations each dtStart and dtEnd might have to be modified when scheduling.

PSMs SHALL describe their requirements and restrictions on Interval descriptions in their conformance statements.

4.3.3 Relationship to other PIM Components

The information in the *IntervalType* class is fundamental to expression of time interval. **[ISO8601]**.

To maintain temporal structure while allowing correlated values, payload values are attached to an Interval (or its subclass *GluonType*), as described in the next section.

---

[8] See **[ISO8601]** section 4.4.

4.4 Payload Attachment to an Interval

A payload, which may be comprised of multiple subparts within a single class, or a reference, is attached to an Interval. This differs from other approaches that have been taken, such as

(a) A class containing a value as well as a description of a relevant Interval (e.g. a measurement that applies to an included Interval)

(b) Associating a particular measurement to an interval (the association is the wrong direction)

The association is directional, and must be present for use of an Interval object in a concrete way.

4.4.1 Model Diagram

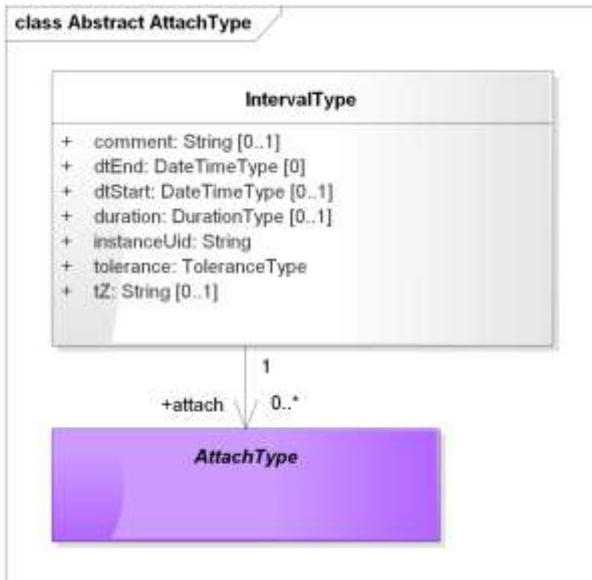

Figure 4-4 Attaching a Payload to an Interval

4.4.2 Discussion

[WS-Calendar] (line 219) requires that the Attachment Payload and Start Duration must be complete for service performance. In contrast, the PIM defines the cardinality of *attach* to be 0..* to allow for abstract schedules, including those to which payloads are bound before use. This mirrors the manner in which attribute values are inherited by Intervals during Binding.

A PSM claiming conformance to this PIM SHALL document in its conformance statement any changes in the definition of *AttachType* and/or the cardinality of associations used for payload attachment.

4.4.3 Relationship to other PIM Components

The *IntervalType* is fundamental; application information is attached to objects of class *IntervalType* by a clear, directional association. This makes the temporal structure of schedules independent of associated information, and of the nature of the associated information by judicious definition of concrete (non-abstract) attachment types.

4.5 The Gluon Class

A Gluon may be thought of as a reference to a Sequence (a set of temporally-related intervals), with the same attributes as an Interval for simplicity of inheritance.

A sequence MAY be referenced by zero or more gluons; the view of a sequence and the values as applied by the Rules in Section 5 are determined by attribute values in the referencing Gluon and values that may be inherited from the referencing gluon such as start time and duration.

More formally, a *Gluon* references schedules comprised of temporally related Intervals and Gluons, while providing that logical information such as the duration of Interval objects may be stated explicitly or be determined by inheritance from the respective Lineages.

The structure defined enables the creation of directed graphs of Interval objects with reuse of components. Those sub graphs may therefore act as reusable sub-schedules, or considered as sub-routines. See Section 7 Examples using the PIM (Non-Normative).

The Gluon acts as a reference into a graph of time-related Intervals or Gluons, allowing differing schedule views depending on the referenced Interval. For example, a room schedule that includes room preparation, meetings, and room cleanup could have a gluon pointing to the preparation Interval for those interested in the preparation starting point and associated actions, and another Gluon pointing to the start of the meetings.

GluonType is a subclass of *IntervalType* with the added requirement that at least one *RelationLinkType* is associated with a *Gluon*; *IntervalType* has zero or more associated *RelationLinkType*.

4.5.1 Model Diagram

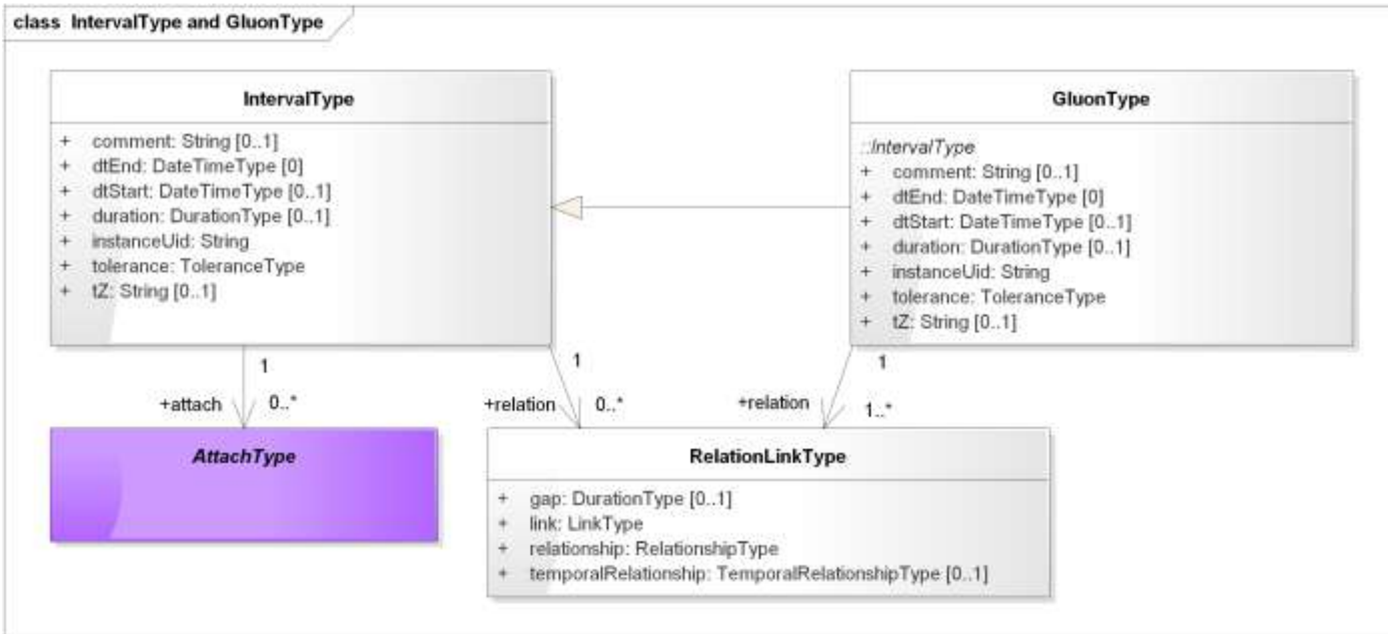

Figure 4-5 Gluons, Intervals, and Relationship Links

Note in Figure 4-5 that the minimum number of relations for a Gluon is 1; Intervals need have no relationships. Only Gluons may have an associated Vavailability.

4.5.2 Discussion

Gluons are Intervals with at least one relation required. One could think of the Gluon as an optional container for values to “fill in” Interval attributes dynamically and depending on the relationships among the instances.

Note: This technique is used in **[EMIX]** and **[EnergyInterop-v1.0]** to build energy schedules with varying values but consistent lengths.

4.5.3 Relationship to other PIM Components

Gluons contain values that may be inherited or overridden in its children in accordance with Section 5.

4.6 Relationships among Gluons and Intervals

Relationships between objects of *IntervalType* are accomplished with *RelationLinkType*. It contains an abstract class *LinkType* that is a String.

The Temporal Relationship and gap together determine the relationship of the referencing Interval and referenced Interval instances.

Note: In **[WS-Calendar]**, **[RFC5545]**, and **[xCal]** the LinkType is a UID, a URI **[RFC3986]**, or a reference string. This supports both distributed schedules and local identifiers that need not be fully qualified as would be a UID or a URI. In the PIM, we use a string, without defining the precise type or uses of that reference—that is left to the PSMs.

The gap SHALL be defined by class *DurationType*, which has a conformed string to the pattern of duration **[ISO8601]** extended by **[RFC5545]** to add a "+" or "-" sign.[9] For example, a gap of P-1H with Temporal Relationship *startToStart* means that the referenced Interval starts one hour before the referencing Interval.

The absence of a sign in the *duration* String SHALL specify a positive value, that is, be treated as if a "+" sign was present.

The absence of a gap attribute in a PIM object SHALL specify a gap of zero duration. An explicit gap of zero duration may be expressed as e.g. P0H.

The *TemporalRelationshipType* enumeration describes the relationship with respect to the referencing and referenced Interval:

- *finishToStart* (the conventional, the referenced interval is after the Finish of the referencing Interval, with an optional gap)
- *finishToFinish* (the end of the referencing Interval aligns with the end of the referenced Interval, with an optional gap)
- *startToFinish* (the start of the referencing Interval aligns with the end of the referenced Interval, with an optional gap)
- *startToStart* (the start of the referencing Interval aligns with the start of the referenced Interval, with an optional gap.

RelationshipType SHALL indicate that the linked Interval is a *child* of the linking object.[10]

If Relationship Types beyond *child* are available in a PSM, that PSM SHALL describe any values other than *child* including syntax and semantics in its conformance statement.

Note that the short forms for the temporal relationships listed in **[WS-Calendar]** are not used in the PIM.

In Figure 4-6 we show two intervals with each of the temporal relationships. Figure 4-7 shows a gap of negative 0.5.

---

[9] **[ISO8601]** duration is unsigned but otherwise more expressive than that in **[RFC5545]** or in **[XMLSchema]**.

[10] **[WS-Calendar]** and **[xCal]**, as do many IETF RFCs, also include in the relationship enumeration an extension point (x-name) and an IANA-registered xCal token (iana-token) **[IANA]**. These are not part of the PIM. **[Relationships]**, an Internet Draft, adds an additional relationship type.

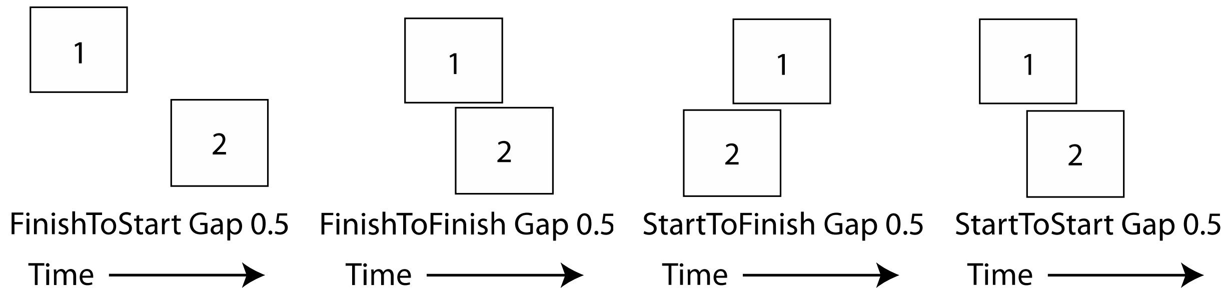

Figure 4-6 Temporal Relationships

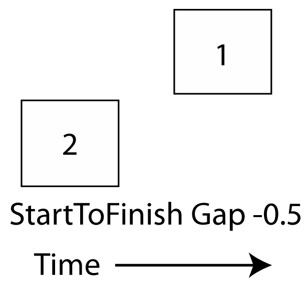

Figure 4-7 Temporal Relationship--startToFinish Negative 0.5 Gap

4.6.1 Model Diagram

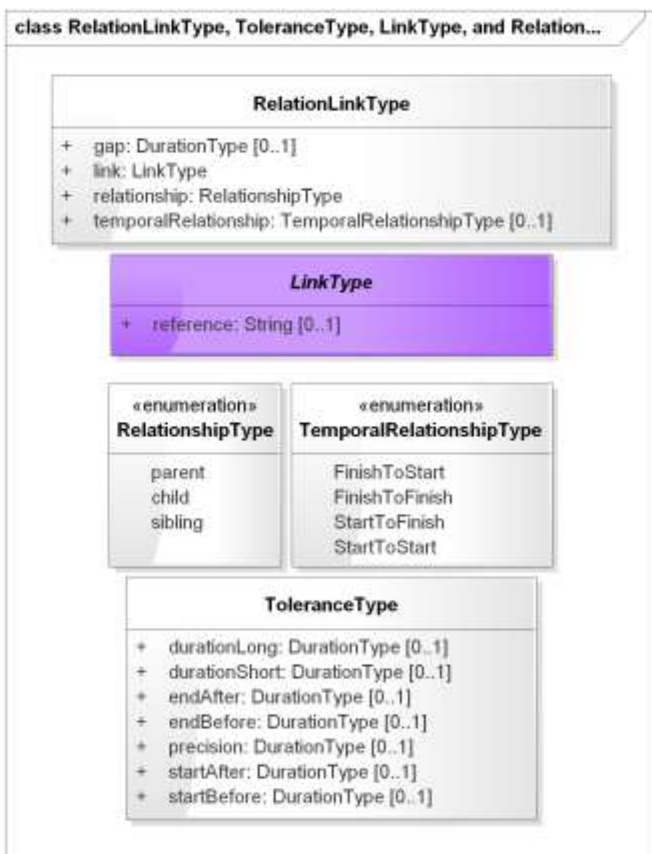

Figure 4-8 RelationLinkType and Relationship Types

4.6.2 Discussion

The PIM supports a complete set of the common relationships between time intervals, as used by and expressed in facility, energy, and other schedules, project management tools, and business process definitions extending e.g. **[BPEL]** and **[BPMN]**.

The relationships are expressed using the (unsigned) temporal relationship, the (signed) temporal gap between intervals, and the *RelationshipType* between Gluons and Intervals.

A gap SHALL be treated as a signed Duration.

ToleranceType attributes SHALL be treated as an unsigned Duration. If a minus sign is present in a Duration expressing a Tolerance it SHALL be ignored.

The PIM SHALL permit only CHILD as *RelationshipType*. Note that other values in the *RelationshipType* enumeration match those in **[RFC6321]**.

Complex structures can be built from primitive relationships, used in data structures, or passed in service invocations, and interpreted unambiguously.

Note: In contrast with the WS-Calendar PSM, *LinkType* contains only a string. The broader range of links in the WS-Calendar PSM includes a UID, a URI, or other kind of reference (implementation-defined). Since the abstract link is conceptually a pointer in the PIM, we define a single kind of reference there. It is maintained as a class to allow a diversity of PSM definitions including but not limited to **[WS-Calendar]**.

A PSM claiming conformance to the PIM SHALL document how it defines, manages, and maintains links. The conformance statement for a PSM SHALL describe uniqueness of references in that PSM.

4.6.3 Relationship to other PIM Components

The PIM allows the common and complete set of temporal relationships between time intervals to be expressed with optional offsets (the optional *Gap*), while abstracting the details of the relationship into the *RelationLinkType* class.

The abstraction maps cleanly to (e.g.) project management schedules and business process descriptions.

4.7 The Availability Package

Availability is a means for describing when an actor can be available, or its complement, not available.

This version of the WS-Calendar PIM includes the necessary classes to express Availability as in **[Vavailability]** which is an Internet Draft as of this date.

Note: Historically, the FreeBusy values conveyed information that is more correctly conveyed by Vavailability. FreeBusy requested all information from a calendar; Vavailability conveys information for a specific purpose known to the responded.

The class *VavailabilityType* includes a partially specified interval in which all blocks of *granularity* size are busy per the *busy* attribute - BUSY, BUY-UNAVAILABLE, BUSY-TENTATIVE.

The entire *timeRange* is *busy* for the purposes of a specific use.

The class *TimeRangeType* MAY have a start time (optional), and if a start time is present MAY contain a duration (optional). It can accordingly apply to

(1) All time (no dtStart, no duration)
(2) A half-infinite interval (dtStart, no duration)
(3) A bound interval (dtStart, duration)

The *granularity* MAY be present and describes the size of the time blocks used for expressing Availability—for example, for one-hour blocks, *granularity* would be the string P1H.

Against this backdrop, the associated *AvailabilityType* objects indicate available times with an *AvailabilityIntervalType* having *dtStart*, *duration*, and optional *tZ*.

Note: *AvailabilityIntervalType* describes a fully bound interval, while *TimeRangeType* describes an interval that may be partially bound or not bound

4.7.1 Model Diagram

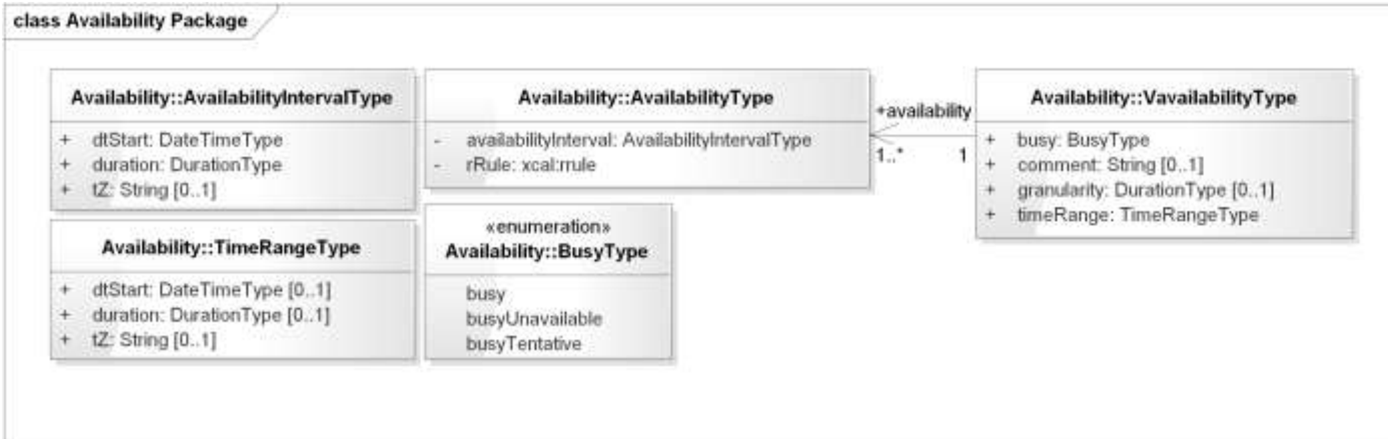

Figure 4-9 Vavailability and Availability Recurrence Rules

4.7.2 Discussion

The *rrule* is an xCal recurrence rule as defined in section 8.6.5.3 of **[RFC6321]**. The recurrence might be (e.g.) Yearly. In its current form, the expression is in iCalendar syntax, and may need adaptation to match the abstraction level of this PIM.

4.7.3 Relationship to other PIM Components

The Availability Package uses recurrence relationships from **[Vavailability].** This allows consistent expression to express availability for (e.g.) Demand Response events in **[EnergyInterop-v1.0]**. Not used by other parts of the PIM.

5 Rules for WS-Calendar PIM and Referencing Specifications

There are five kinds of conformance that must be addressed for WS-Calendar PIM and for specifications that claim conformance to this PIM.

- Conformance to the ***inheritance rules***, including the direction of inheritance
- ***Specific attributes*** for each type that MUST or MUST NOT be inherited
- ***Conformance rules*** that Referencing Specifications MUST follow
- Description of ***Covarying attributes*** with respect to the Reference Specification
- ***Semantic Conformance*** for the information within the artifacts exchanged

We address each of these in the following sections

5.1 Inheritance in WS-Calendar PIM

In this section we define rules that define inheritance including direction.

I1: Proximity Rule Within a given lineage, inheritance is evaluated though each Parent to the Child before what the Child bequeaths is evaluated.

I2: Direction Rule Intervals MAY inherit attributes from the nearest gluon subject to the Proximity Rule and Override Rule, provided those attributes are defined as Inheritable.

I3: Override Rule If and only if there is no value for a given attribute of a Gluon or Interval, that Gluon or Interval SHALL inherit the value for that attribute from its nearest Ancestor in conformance to the Proximity Rule.

I4: Comparison Rule Two Sequences are equivalent if a comparison of the respective Intervals succeeds as if each Sequence were fully Bound and redundant Gluons are removed.

I5: Designated Interval Inheritance [To facilitate composition of Sequences] the Designated Interval in the ultimate Ancestor of a Gluon is the Designated Interval of the composed Sequence. Special conformance rules for Designated Intervals apply only to the Interval linked from the Designator Gluon.

I6: Start Time Inheritance When a start time is specified through inheritance, that start time is inherited only by the Designated Interval; the start time of all other Intervals are computed through the durations and temporal; relationships within the Sequence. The Designated Interval is the Interval whose parent is at the end of the lineage.

5.2 Covarying Elements

Some attributes of PIM objects may be **covarying**, meaning that they change together. Such elements are treated as a single element for inheritance: they are either inherited together or the child keeps its current values intact.

Note: This becomes important if one or more of a covarying set have default values.

If any covarying attributes are present, then inheritance SHOULD deem they are all present, and SHOULD assign those without specific definition appropriate default values.

A PSM SHALL describe definition and treatment of covarying elements in its conformance statement.

5.3 Specific Attribute Inheritance

In PIM classes the following attributes MUST be inherited in conformance to the Rules (same for Gluons and Intervals):

- dtStart
- dtEnd
- Duration
- Designated Interval (Gluon, special upward inheritance rule)
- Tolerance

The following attributes MUST NOT be inherited

- instanceUid (Gluons and Intervals)
- Temporal Relationships (between Intervals)
- Relationship Links

6 Conformance

This section specifies conformance related to the information model contained in this specification.

6.1 Conformance for Specifications Claiming Conformance to WS-Calendar PIM

Specifications that claim conformance to the WS-Calendar PIM SHALL specify inheritance rules for use within their specification.

These rules SHALL NOT modify the Proximity, Direction, or Override Rules. If the specification includes covarying attributes, those attributes and their default values SHALL be clearly designated in the specification and in the PSM conformance statement.

6.2 General Conformance Issues (Non-Normative)

Standards that claim conformance to this specification may need to restrict the variability inherent in the expressions of Date and Time to improve interoperation within their own interactions. Aspects of Date and Time that may reward attention and conformance statements include:

- **Precision** – Does the conforming specification express time in Hours or in milliseconds? Five-minute intervals? A PSM claiming conformance to this PIM SHOULD select a consistent precision.
- **Time Zones and UTC** – Business interactions have a “natural” choice of local, time zone, or UTC based expression of time. Intents may be local, as they tie to the business processes that drive them. Tenders may be Time zone based, as they are driven by the local business process, but may require future action across changes in time and in time zone. Transaction recording may demand UTC, for complete unambiguity. The specification cannot require one or another, but particular business processes may require appropriate conformance statements. A PSM claiming conformance to this PIM SHALL detail Time Zone treatment as well as assumptions and implicit values.
- **Business Purpose** – The PIM does not distinguish between different uses of objects that may have different purposes. For example, a general indication of capability and/or timeliness may be appropriate for a market tender, and an unanchored Sequence may be appropriate. In the same specification, performance execution could require merely that the Gluon Anchor the Interval. If the distinction between Unanchored and Anchored Interval is necessary for a particular use, the PSM claiming conformance SHALL indicate the proper form for each of its uses.

6.3 Conformance of Intervals

6.3.1 Intervals and Gluons

Intervals SHALL have *duration* AND optionally *dtStart*. If a non-compliant Interval is received in a service operation or by reference with a value for *dtEnd*, then *dtEnd* SHALL be ignored.

Within a Sequence, at most one Interval MAY have a *dtStart* or a *dtEnd*.

Specifications that claim conformance SHALL define the business meaning of zero duration Intervals or prohibit zero duration intervals, and include that definition or prohibition in their conformance statement.

6.3.2 Other Attributes

A Gluon MAY have a *dtStart* value.

6.4 Conformance of Bound Intervals and Sequences

Actionable services require Bound Intervals as part of a Bound Sequence. Services may include Intervals that are not bound for informational or negotiation purposes. Some of these are modeled and described as constraints in the UML models that have been produced separately.

- Intervals SHALL have values assigned for dtStart and duration, either explicitly or through inheritance
- Intervals SHALL have no value assigned for dtEnd
- Within a Sequence at most the Designated Interval may have dtStart and duration with a value specified or inherited.
- If Sequences are composed to create other Sequences, then the Designated Intervals within the composing Sequence are ignored.
- Any specification claiming conformance to the WS-Calendar PIM MUST satisfy all of the following conditions:
  - Follow the same style of inheritance (per the Rules)
  - Specify attribute inheritability in the specification claiming conformance
  - Specify whether certain sets of elements must be inherited as a group or specify that all elements can be inherited or not on an individual basis

6.5 Security Considerations (Non-Normative)

The WS-Calendar PIM describes an informational model. Specifications claiming conformance with the WS-Calendar PIM are likely to use the schedule and interval information as but a small part of their overall communications.

Specifications involving communication and messages that claim conformance to this specification should select the communication and select from well-known methods to secure that communication appropriate to the information exchanged, while paying heed to the costs of both communication failure and of inappropriate disclosure. To the extent that iCalendar schedule servers are used, the capabilities of security of those systems should be considered as well. Those concerns are out of scope for this specification.

7 Examples using the PIM (Non-Normative)

We include several examples drawn from a variety of sources. These examples were created to illustrate facility scheduling, energy scheduling, and related topics.

The dashed lines in the Object Diagrams are not UML, but are a graphical depiction of the links, with the head of the arrow indicating the referenced (linked) Interval and the tail indicating the referencing (linking)

A separate Committee Technical Note **[PIM Examples]** is in progress with examples including ones drawn from those in **[WS-Calendar]** and other specifications.

7.1 Related Intervals

This example is based on Example 3-05, line 483 in **[WS-Calendar]**.

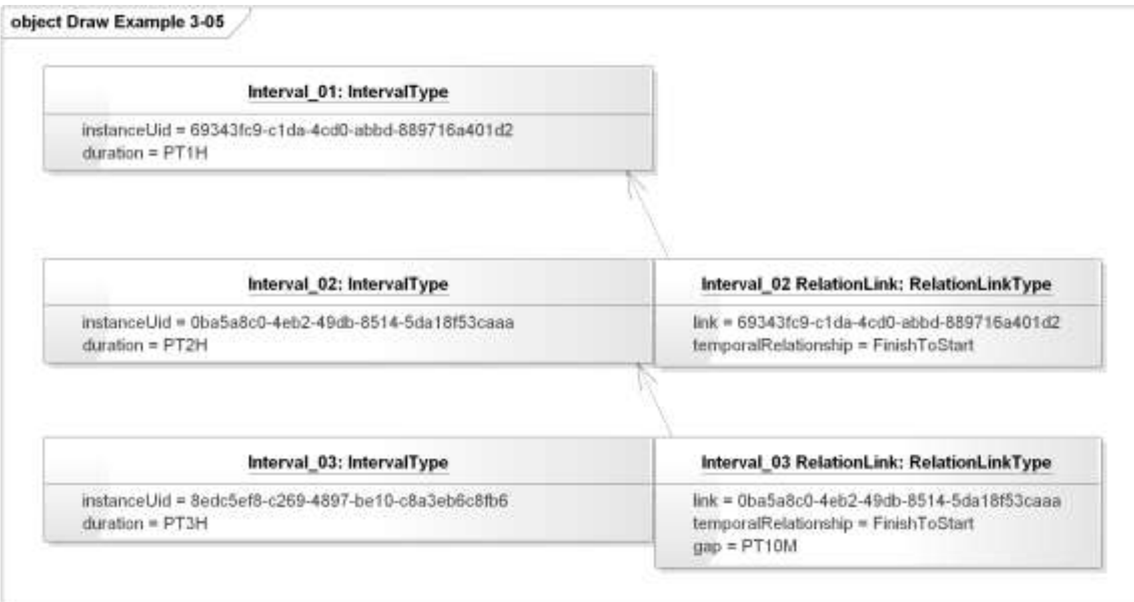

Figure 7-1 PIM Expression of WS-Calendar Examples 3-05

In this diagram, a Gluon could refer to the Sequence with a reference to Interval_03.

7.2 A Meeting Schedule

Consider a meeting scheduled for a specific time – say 2pm and lasting two hours.

The meeting itself can be represented (and scheduled with attendees) as a single interval with duration 2 hours.

To carry out the meeting, there are other activities both before and after, and possibly during, the meeting time. See Figure 7-2 Simple Meeting Schedule below.

First, the room needs to be set up for the meeting. The Heating, Ventilating, and Air Conditioning system (HVAC) may need to pre-cool the room for the scheduled number of attendees. And the room needs to be cleaned up before setup for the next meeting.

Each of these activities can be scheduled separately, and done by different actors. But they need to be completed to set up and restore the room.

Also consider a pre-meeting of the leaders in the room, starting 30 minutes before the main meeting, and lasting 20 minutes so the leaders can meet and greet attendees.

The gluons on the right are references into the sequence of intervals; the respective sequences are *child* to the respective Gluons.

(1) The start of the HVAC pre-cooling is given to the HVAC control system
(2) The start of the main meeting gluon is given to the meeting attendees

Additional gluons could be given to (e.g.) the room set-up team, pointing to the Prepare Room interval, and to the Pre-Meeting interval for the meeting leaders.

Additional elaboration might include the pre-purchase of energy for the pre-cooling (or committing in an energy schedule, which the HVAC control system uses to balance energy use through the day to avoid demand charges.

Finally, the actions are all based on where you reference the schedule—working back from the start time (inherited from the start of main meeting gluon) the pre-meeting is 30 minutes earlier, and the setup is 2 hours and 30 minutes earlier.

The HVAC schedule gluon might be all that the control system needs, combined with the knowledge from the schedule that the meeting is over in 2 hours 30 minutes after the 30-minute pre-cool period, and that cleanup takes another 30 minutes.

We have not tried to show all possible schedules and variations – perhaps the setup takes longer but is finished earlier, using an *endBefore* tolerance (and a zero *endAfter* tolerance).

Note that this schedule may be used for any meeting – the start time can be placed in a gluon that references the Meeting interval. Likewise, the length could also be inherited from that same gluon. The structure of the schedule would be determined by facility policy (e.g. “you must allow two hours for setup”), and the schedule itself is relocatable and reusable.

The figure is informal, and does not reflect all the details of relationships (the arrows indicate the relationships which are not otherwise shown with relations and IDs).

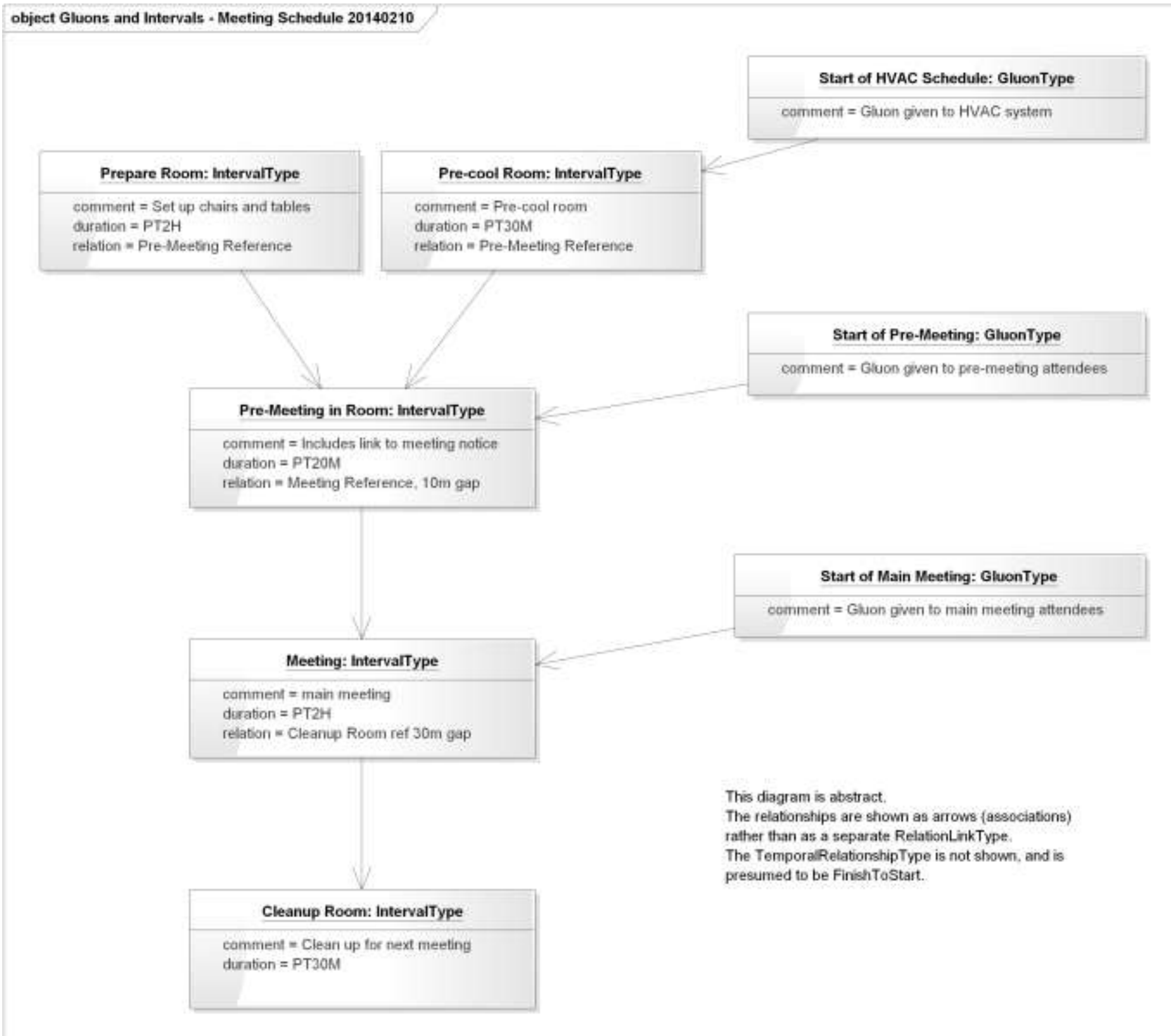

Figure 7-2 Simple Meeting Schedule

Appendix A. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Bruce Bartell	Southern California Edison
Chris Bogen	US Department of Defense (DoD)
Edward Cazalet	Individual
Toby Considine	University of North Carolina at Chapel Hill
Robin Cover	OASIS
William Cox	Individual
Sharon Dinges	Trane
Michael Douglass	Rensselaer Polytechnic Institute
Craig Gemmill	Tridium, Inc.
Dave Hardin	EnerNOC
Gale Horst	Electric Power Research Institute (EPRI)
Gershon Janssen	Individual
Ed Koch	Akuacom Inc.
Benoit Lepeuple	LonMark International
Carl Mattocks	Individual
Robert Old	Siemens AG
Joshua Phillips	ISO/RTO Council (IRC)
Jeremy Roberts	LonMark International
David Thewlis	CalConnect

Appendix B. Revision History

Revision	Date	Editor	Changes Made
01	November 15 2012	William Cox	Initial Draft based on contributed models
02	December 20 2012	William Cox	First draft conformance section. Added explanatory text in individual model sections. GluonType is now a subclass of IntervalType, rather than GluonType having an association to IntervalType.
03	January 31, 2012	William Cox	Completed most sections; indicated questions for the TC as “EDITOR’S NOTE”s. Model is the same as for WD02. WD03 contains a quotation with modifications from the WS-Calendar conformance sections.
04	April 10, 2013	William Cox	Update with responses to questions from WD03; minor changes to the model and many clarifications based on meeting discussions. Included differences between the normative semantics and conformance sections and WS-Calendar 1.0 as non-normative Appendices.
05	April 24, 2013	William Cox	Addressed remaining Editor’s Notes from previous Working Drafts. Changed cardinality for attachment from [1..1] to [0..1] in parallel with unbound attributes expressed in UML. Prepared text for public review.
06	16 January 2014	William Cox	Simplification of relations and LinkType. Addition of instance (object) diagrams to express examples. Includes PIM to WS-Calendar-as-PSM mapping.
07	17 January 2014	William Cox	Addresses comments from TC review of WD06. Eliminated unused DurationParameterEnum, corrected gap to DurationStringType (with no tolerance values), eliminated iana-token and x-name relationship types. Identified but did not correct the application of tolerance to dtStart, dtEnd, and duration. Clarified intended sources of examples. Eliminated unused classes and objects in the model.
08	13 March 2014	William Cox	Simplifies the DurationType, moves tolerance to IntervalType instead of the former DurationValueType. Completed PIM-PSM mapping, updated references, other editorial and technical clarity change. Updated diagrams to express updated model.

Revision	Date	Editor	Changes Made
09	21 April 2014	William Cox	First inclusion of mapping descriptions. Clarified DateTimeType and DurationType relationship to ISO 8601. Many minor edits; minor model changes.
10	08 May 2014	William Cox	Edits throughout based on meeting discussion. lowerCamelCase for ToleranceType, textual changes, and updated diagrams.
11	31 July 2014	William Cox	Address comments from second Public Review. Normative reference to and comparisons to WS-Calendar have largely been removed. Much text has been moved to non-normative sections or appendices. Diagrams and the model were updated.
12	03 August 2014	William Cox	Completed addressing comments from second Public review. Significant modifications to Availability, and simplification of DurationType. Deleted FreeBusy. Detailed corrections to attribute names to align with model. Model updated to reflect corrections, and all figures for PIM UML were updated.
13	14 August 2014	William Cox	Address one comment from TC members, changed all attributes including those in enumerations to lowerCamelCase (ToleranceType, BusyType, RelationshipType). Minor editorial corrections. Use of italic is more consistent except in Appendix C.

Appendix C. PIM to WS-Calendar PSM Transformation

MDA instances include a Platform-Independent Model (PIM), defined in this specification, and a transformation to one or more Platform-Dependent Model (PSM). In this section we briefly describe the mapping from this PIM to **[WS-Calendar]** (considered as a PSM).

Largely the same data types and conformed strings for instance values are used in the PIM, to ensure that the transformation is straightforward.

Diagrams with golden class backgrounds are from **[WS-Calendar]**; diagrams with light class backgrounds are from this PIM.

A UML model for WS-Calendar is of a different style from this PIM. WS-Calendar expresses the information for Intervals, Gluons, and other classes in terms of collections of Parameters, Properties, and Value Types, held in those collections with others that may not reflect the abstractions of WS-Calendar.

C.1 General Transformations

On inspection the transformations between the PIM model and the **[XMLSchema]** for **[WS-Calendar]** are generally clear. The classes in the PIM are similar or identical to those in [WS-Calendar] including attribute/element names, but are arranged as simple classes rather than collections of properties within a potentially larger set of properties.

C.2 Specific Transformations

In the following subsections we describe transformations from the PIM to the WS-Calendar PSM.

In WS-Calendar an Interval or Gluon is a Vcalendar component, expressed as a subclass of ICalendar::VcalendarContainedComponentType.

That class informally contains sets of Properties, Values, and Parameters, based on the widely used iCalendar definition. The PIM does not distinguish between parameters, values, and properties and the differing types.

In the subsections below we describe the transformations for

- DateTime and Duration Types, the fundamental types for talking about time and schedule
- ToleranceType
- Intervals and Gluons
- Relationships
- Vavailability

C.2.1 Transformation for DateTime and Duration Types

DateTimeType and DurationType use **[ISO8601]** conformed strings. In transforming objects of these PIM classes the values must be expressible in the target PSM. The following two figures show selected WS-Calendar classes and the PIM classes DateTimeType, DurationType, and ToleranceType.

There are different conformed strings for DurationType and DateTimeType. The PIM uses **[ISO8601]** duration and date time semantics; these are isolated in the PIM classes DateTimeType and DurationType to facilitate mapping to classes in PSMs including those based on **[WS-Calendar]** and **[XMLSchema]**.

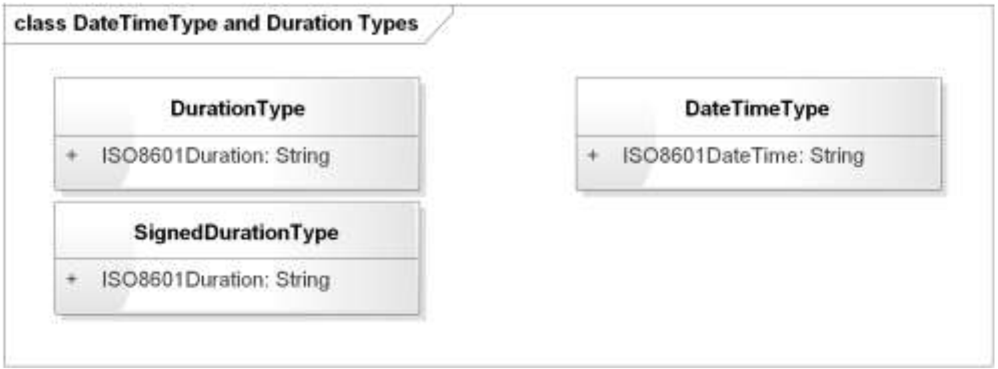

Figure 7-3 PIM Source Classes for DateTimeType and Duration Types

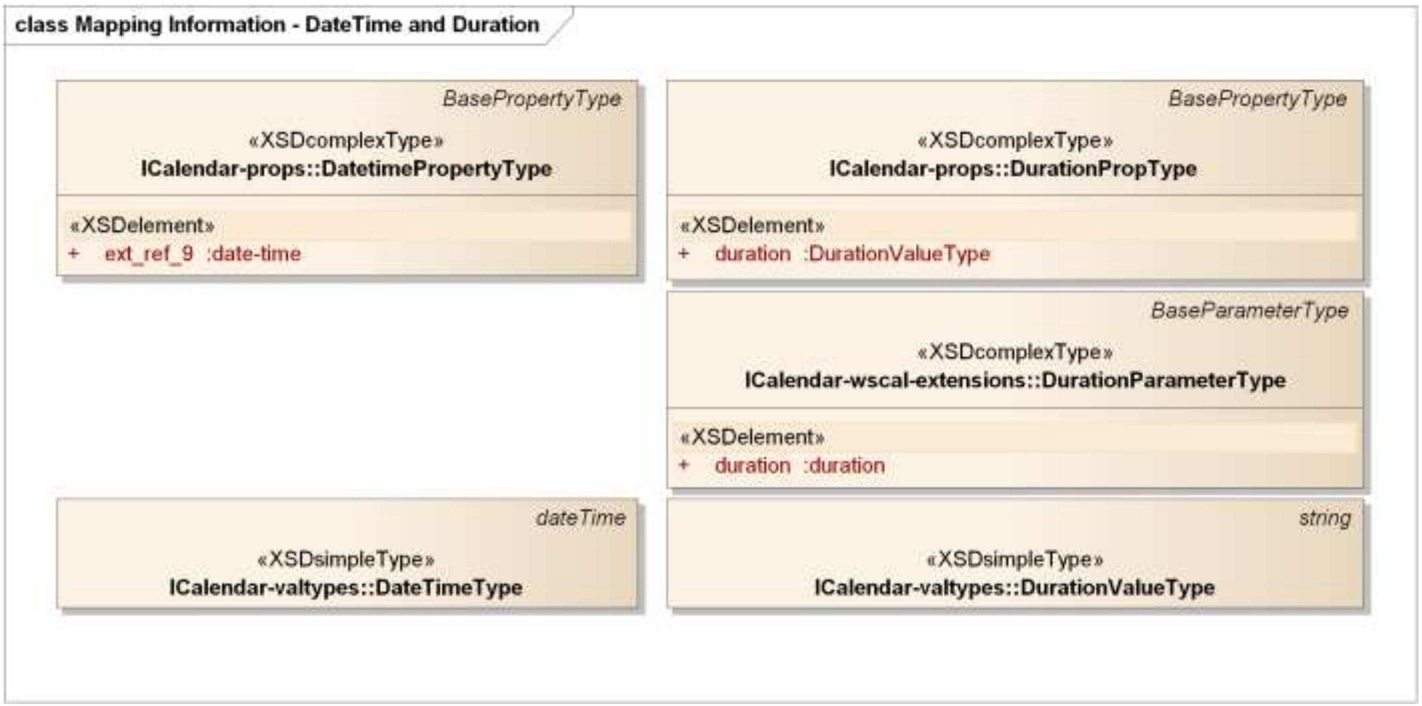

Figure 7-4 WS-Calendar Target Classes

Table 7-1 PIM to PSM Mapping for DateTimeType and Duration Types

PIM Class Name	WS-Calendar Class Name	Notes
DateTimeType	ICalendar-valtypes::DateTimeType	Restrictions on ISO8601 strings when mapped to RFC5545 strings
DurationType	ICalendar-valtypes::DurationValueType	Restrictions on ISO8601 strings when mapped to RFC5545 strings; Gap is signed as in **[WS-Calendar]**. DurationType must be non-negative for ToleranceType.

C.2.2 Transformation for Tolerance Type

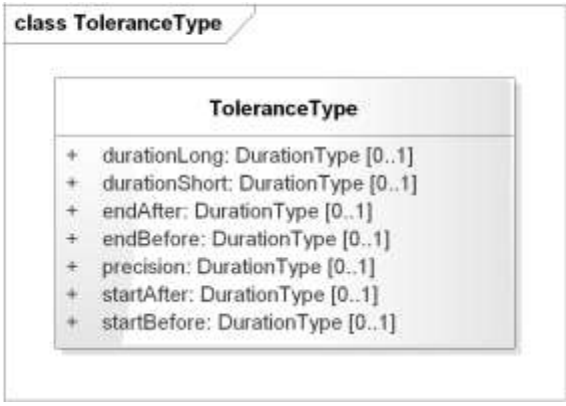

Figure 7-5 PIM Source Class for ToleranceType

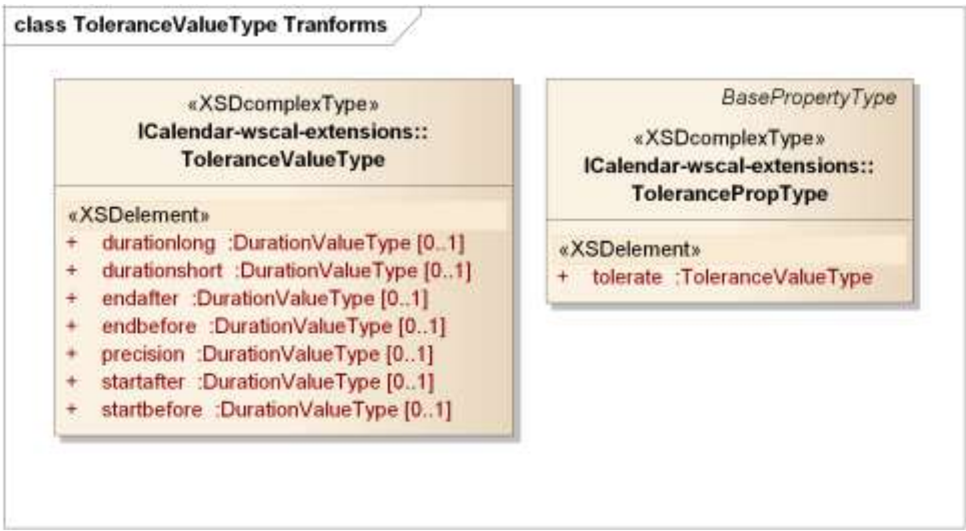

Figure 7-6 WS-Calendar Target Classes for Tolerance Type

The PIM ToleranceType is identical with minor differences in attribute names and types to the WS-Calendar class with the same function, as shown in Figure 7-5 and Figure 7-6 above.

The differences are

- The PIM uses *DurationType* rather than the WS-Calendar *DurationValueType*
- The PIM uses *ToleranceType* rather than the WS-Calendar *ToleranceValueType*
- The PIM attribute names are in lowerCamelCase rather than lower case.

Table 7-2 PIM to PSM Mapping for ToleranceType

PIM Class Name	WS-Calendar Class Name	Notes
ToleranceType	ICalendar-wscal-extensions:: ToleranceValueType	Attributes map respectively to attributes of the same name with lowerCamelCase in PIM *ToleranceType.* Types map per Section C.2.1.

C.2.3 Transformation for Interval and Gluon Types

We treat the Gluon and Interval together; GluonType is a subclass of IntervalType, and extends IntervalType as shown in Figure 7-7:

- Changing the cardinality of the attribute *relation* to require one or more *relation*s
- Optionally including *Vavailability*

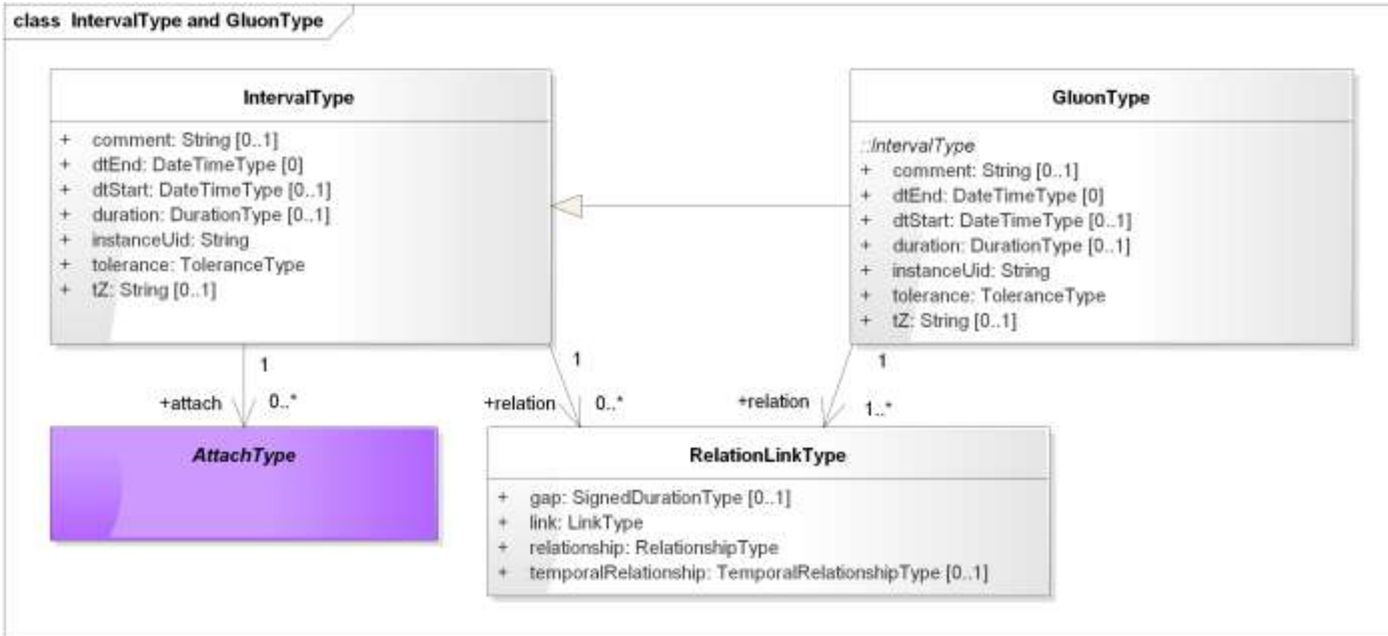

Figure 7-7 PIM IntervalType and GluonType

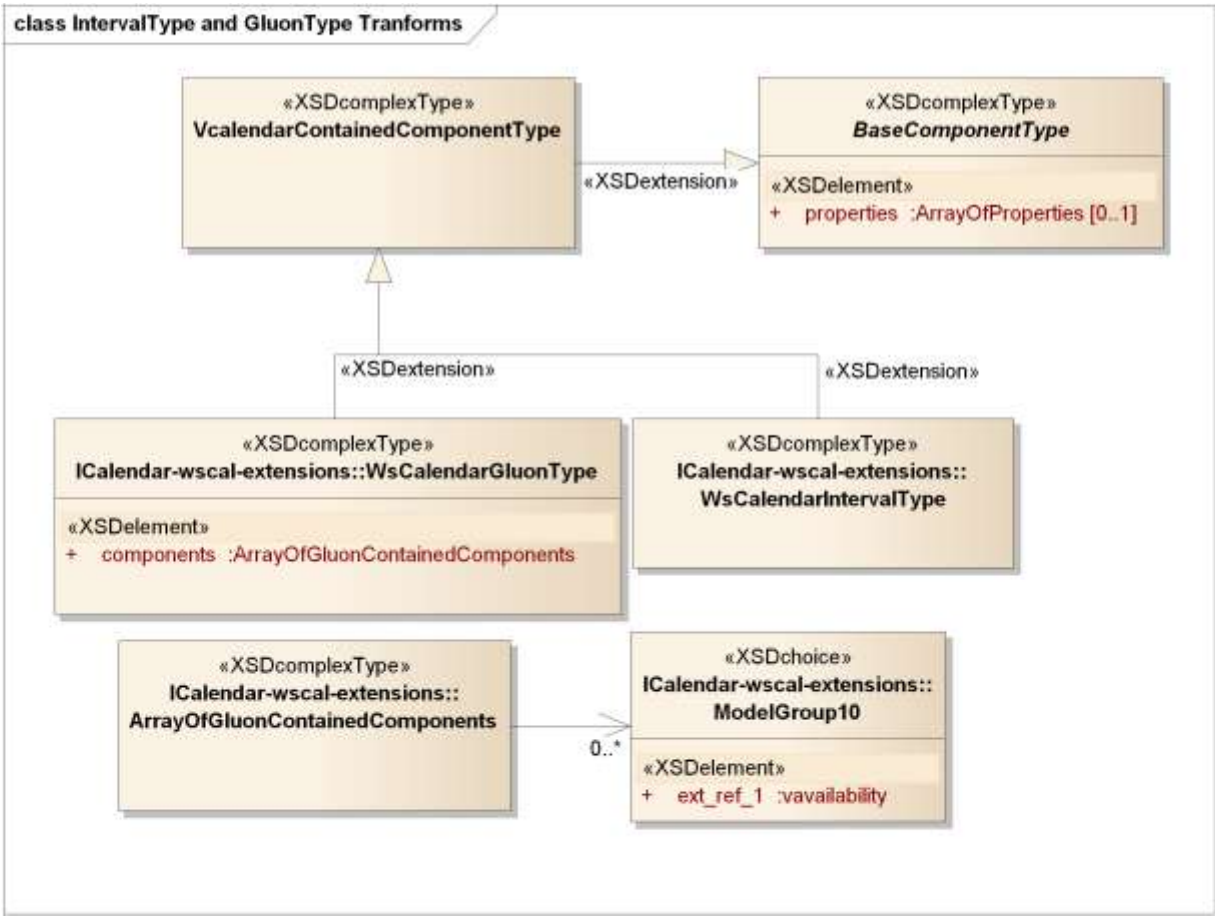

Figure 7-8 WS-Calendar Target IntervalType and GluonType

A WS-Calendar Interval (and its subclass Gluon) is a Vcalendar object, with a set of properties, values, and parameters optionally included. Among those are the attributes of the PIM IntervalType, essentially the same set of attributes of GluonType, and the additional VavailabilityType in GluonType.

Properties with the same semantics and value types exist in WS-Calendar as well as the PIM; the name and type transformations are described in the following table. RelationLinkType is addressed in the next section.

Table 7-3 PIM to PSM Mapping for IntervalType and GluonType

PIM Attribute and Type	WS-Calendar Target Type	Notes
comment: string	ICalendar-Props::CommentPropType	Target takes a text value.
dtEnd: DateTimeType	ICalendar-Props::DtendPropType	Constrained string per **[RFC5545]**
dtStart: DateTimeType	ICalendar-Props::DtstartPropType	Constrained string per **[RFC5545]**
duration: DurationType	ICalendar-wscal-extensions::DurationPropType	Constrained string per **[RFC5545]**
instanceUid: string	ICalendar-Props::UidPropType	

PIM Attribute and Type	WS-Calendar Target Type	Notes
tolerance: ToleranceType	ICalendar-wscal-extensions::ToleranceValueType	Attribute of TolerancePropType is *tolerate*
tZ: string	ICalendar-Props::TzidPropType	Constrained string per **[RFC5545]**
Relation: RelationLinkType	ICalendar-link-extension::LinkPropType	Target has possible UID, URI, Reference attributes

C.2.4 Transformation for Relationships

In this section we detail transformations for *RelationLinkType* and for its attributes: *link*, *relationship*, and *temporalRelationship*.

Both **[WS-Calendar]** and the current draft extending iCalendar **[Relationships]** have a single *ReltypeParamType* which combines relationships (e.g. CHILD) and temporal relationships (e.g. FinishToStart) in one.

In the PIM we maintain separate attributes of RelationLinkType for those two classes of relationship, and separate enumerations, *RelationshipType* and *TemporalRelationshipType*, rather than multiple parameter values. This mirrors current programming practices favoring explicit unitary value enumerations rather than logically combining a set of values. Moreover, the use of text *reltypes* adds brackets around every string no matter how short. The implicit repetition of a parameter, each with its attendant brackets, may not be a correct interpretation.

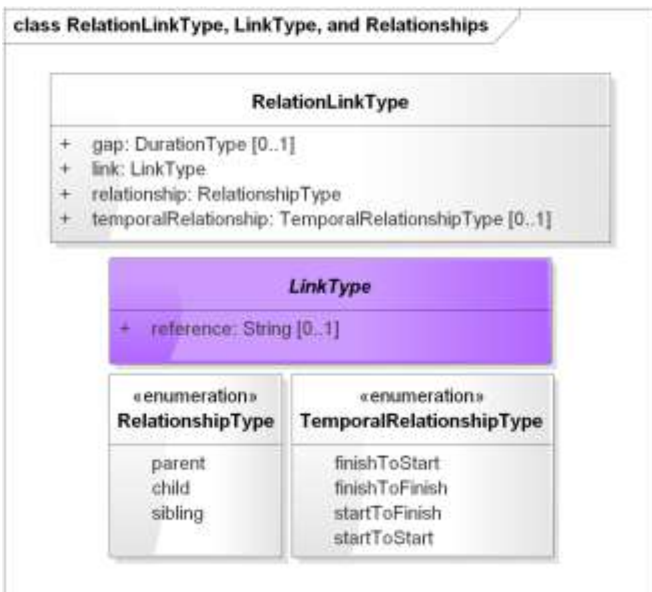

Figure 7-9 PIM RelationLinkType, LinkType, RelationshipType, and TemporalRelationshipType

The following table describes the transformation in detail for the classes and enumerations in Figure 7-9. The *related-to* property in WS-Calendar may include a *reltype* parameter.

The "short form" in Temporal Relationships Table 3-2, line 423 in **[WS-Calendar]** is not used in the PIM; transformation should be to the "long form" in **[WS-Calendar]**.

The values for RelationshipType and TemporalRelationshipType map to the same names in WS-Calendar, excepting only that TemporalRelationshipType in WS-Calendar is all lower case rather than lowerCamelCase.

Table 7-4 PIM to PSM Mapping for Attributes of PIM RelationshipType and TemporalRelationshipType

PIM Enumeration	WS-Calendar Target Type	Notes
RelationshipTypes:: PARENT, CHILD, SIBLING	ICalendar-props::RelatedToPropType	PIM uses only CHILD
TemporalRelationshipType:: FinishToStart, FinishToFinish, StartToFinish, StartToStart	ICalendar-props::RelatedToPropType	

Table 7-5 PIM to PSM Mapping for Enumeration Members

PIM Attribute and Type	WS-Calendar Target Type	Notes
gap: DurationType	ICalendar-Params::DurationParameterType	Duration is an **[ISO8601]** conformed string that maps to a constrained string per **[RFC5545]** with optional sign.
Link: LinkType	ICalendar-props::RelatedToPropType	All of the RelatedToPropType extended choices are strings (*uri*, *uid*, and *text*). PIM *LinkType* is an abstract type with a *String* attribute.
Relationship: RelationshipType	iCalendar-params:: ReltypeParamType – iCalendar-props::related-to: RelatedToPropType	In the same set of RelatedToPropType as temporal relationships
temporalRelationship: TemporalRelationshipType	iCalendar-params:: ReltypeParamType – iCalendar-props::related-to: RelatedToPropType	In the same set of RelatedToPropType as relationships

C.2.5 Transformation for Vavailability and FreeBusy

Vavailability is described in a separate package in the PIM. A future Working Draft will disconnect Vavailability from use of **[xCal]** types, so that the mapping is clearer and is disconnected from changes in **[Vavailability]** as it completes the IETF process.

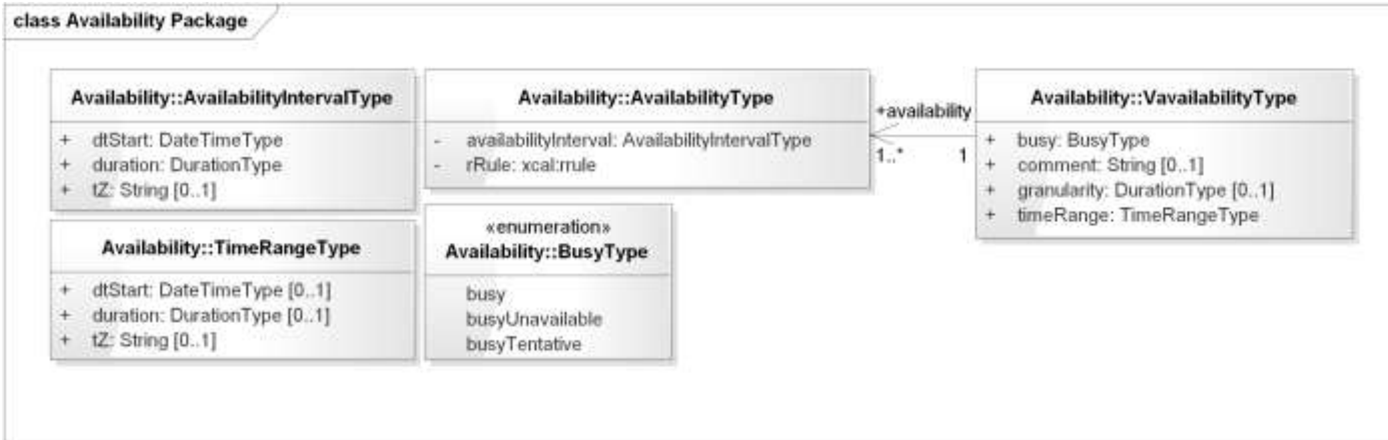

Figure 7-10 PIM Vavailability Package Classes

The Vavailability package in iCalendar-availability-extension.xsd is as follows:

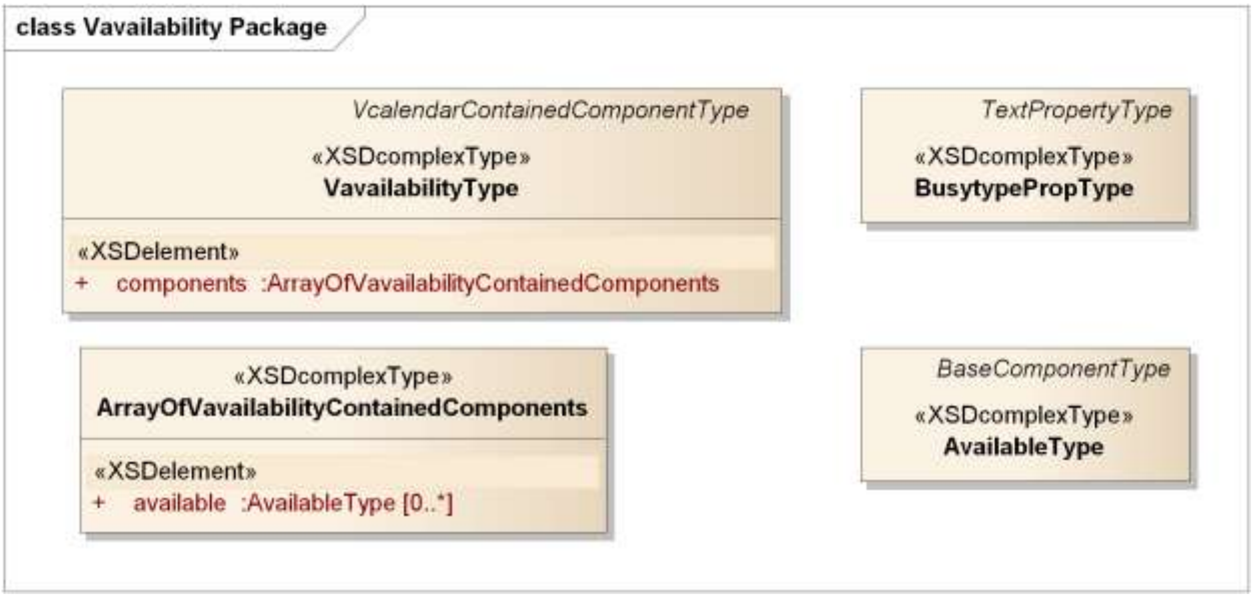

Figure 7-11 Vavailability Package from iCalendar-availability-extension

The VavailabilityType has zero or more AvailabilityType objects inside. The *rrule*s matched a previous draft of **[Vavailability]** and is expected to work with the final standard version. The *AvailabilityIntervalType* is implicit in the way components are defined in **[RFC5545]** and **[RFC6321]**.

Appendix D. PIM to IEC TC57 CIM Intervals and Sequences (Non-Normative Example)

The IEC TC57 Common Information Model **[IEC CIM]** uses time intervals in a variety of ways. We describe straightforward transformations in both directions between

- A fully bound PIM interval (which uses *dtStart* and duration and time zone) with an *Attach*
- To a CIM interval (which uses *dtStart* and *dtEnd* in UTC).

First we must understand that a time interval *per se* does not existing in **[IEC CIM]**. Instead, explicit *dtStart* and *dtEnd* attributes are included, often as a timestamp value. In some part of the CIM model the start and end are implicit. In short in the CIM model is a great variety of expression for time intervals, and all are expressed by including attributes in a class, not something that has a separate definition.

The mapping is from the appropriate calculated or explicit values in an object to an Interval, setting the *dtStart*, *tZ*, and *duration* in the PIM *IntervalType*.

The CIM also assumes UTC, which must be an explicit time zone in iCalendar.

(1) The CIM class *dtStart* maps to *dtStart* and *tZ* with value UTC in the PIM Interval
(2) The CIM class *dtEnd* is used with CIM *dtStart* to compute the PIM *duration*
(3) The CIM class is mapped to an *AttachType* created in a PIM or PSM model

The other direction is also straightforward:

1. Determine the CIM class as target from the concrete *AttachType*
2. The PIM *dtStart* and *tZ* is mapped to the appropriate UTC time, and placed in the CIM class *dtStart*
3. The PIM duration is added to the CIM *dtStart* and placed in CIM *dtEnd*

Because of the interval attributes inserted in each data item (or implicitly present), and because CIM intervals include *dtStart* and *dtEnd*, sequences of CIM intervals are not relocatable in the same way as PIM and **[WS-Calendar]** Intervals.

To relocate, a Sequence of CIM intervals must each be modified with the new *dtStart* and *dtEnd*.

In the PIM changing *dtStart* in the Designated Interval or in a referencing Gluon relocates a sequence.