WS-Calendar Platform Independent Model (PIM) Version 1.0

Committee Specification Draft 02 / Public Review Draft 02

09 May 2014

Specification URIs

This 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

Previous version:

http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/csprd01/ws-calendar-pim-v1.0-csprd01.pdf (Authoritative)
http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/csprd01/ws-calendar-pim-v1.0-csprd01.html
http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/csprd01/ws-calendar-pim-v1.0-csprd01.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 that 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/csprd02/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.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at https://www.oasis-open.org/committees/ws-calendar/.

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-calendar/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 May 2014. OASIS Committee Specification Draft 02 / Public Review Draft 02. http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/csprd02/ws-calendar-pim-v1.0-csprd02.html. Latest version: http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/ws-calendar-pim-v1.0.html.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see https://www.oasis-open.org/policies-guidelines/trademark for above guidance.

Table of Contents

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 8
1.7 Architectural References and Background 8
2 Architecture 9
2.1 The PIM and the WS-Calendar PSM 9
2.2 Key Abstractions 9
2.3 Expression of the PIM 9
2.4 Structure of the PIM Model and Specification 10
2.5 Expression in UML 10
3 WS-Calendar PIM Terminology and Semantics 11
3.1 Time Intervals and Collections of Time-Related Intervals 11
4 The Platform-Independent Model 15
4.1 Introduction to the PIM 15
4.1.1 Model Diagram of the PIM 16
4.1.2 Discussion 16
4.2 Types for Date and Time, Duration, and Tolerance 17
4.2.1 Model Diagram 18
4.2.2 Discussion 19
4.2.3 Relationship to other PIM Components 19
4.3 Interval 19
4.3.1 Model Diagram 19
4.3.2 Discussion 19
4.3.3 Relationship to other PIM Components 20
4.4 Payload Attachment to an Interval 20
4.4.1 Model Diagram 20
4.4.2 Discussion 20
4.4.3 Relationship to other PIM Components 21
4.5 Gluons 21
4.5.1 Model Diagram 22
4.5.2 Discussion 22
4.5.3 Relationship to other PIM Components 22
4.6 Relationships among Gluons and Intervals 22
4.6.1 Model Diagram 23
4.6.2 Discussion 25
4.6.3 Relationship to other PIM Components 26
4.7 Availability 26
4.7.1 Model Diagram 26

4.7.2 Discussion 26
4.7.3 Relationship to other PIM Components 26
5 Examples using the PIM (Non-Normative) 27
5.1 Related Intervals 27
5.2 A Meeting Schedule 27
6 Architectural Basis for the PIM (Non-Normative) 30
7 PIM to WS-Calendar PSM Transformation 31
7.1 General Transformation 31
7.2 Specific Transformations 31
7.2.1 Transformation for DateTime and Duration Types 31
7.2.2 Transformation for Tolerance Type 33
7.2.3 Transformation for Interval and Gluon Types 34
7.2.4 Transformation for Relationships 36
7.2.5 Transformation for Vavailability 37
8 PIM to IEC TC57 CIM Intervals and Sequences 39
9 Conformance and Rules for WS-Calendar PIM and Referencing Specifications 40
9.1 Relationship to WS-Calendar [Non-Normative] 40
9.2 Conformance Rules for WS-Calendar PIM 40
9.2.1 Inheritance in WS-Calendar 40
9.2.2 Specific Attribute Inheritance 41
9.2.3 General Conformance Issues 41
9.2.4 Covarying Elements 41
9.2.5 Conformance of Intervals 42
9.2.6 Conformance of Bound Intervals and Sequences 42
9.3 Conformance Rules for Specifications Claiming Conformance to WS-Calendar PIM 42
9.4 Security Considerations 42
Appendix A. Acknowledgments 44
Appendix B. Revision History 45
Appendix C. PIM and WS-Calendar Semantics Differences 47
Appendix D. PIM and WS-Calendar Conformance Differences 48

Table of Figures

Figure 4-1 The Complete WS-Calendar PIM UML Model 16
Figure 4-2 DateTimeType, DurationType, and ToleranceValueType 18
Figure 4-3 The PIM IntervalType 19
Figure 4-4 Attaching a Payload to an Interval 20
Figure 4-5 Gluons, Intervals, and Relationship Links 22
Figure 4-6 Temporal Relationships 23
Figure 4-7 Temporal Relationship--StartToFinish Negative 0.5 Gap 23
Figure 4-8 Relation Link Type and Relationship Types 25
Figure 4-9 Vavailability and Availability Recurrence Rules 26
Figure 5-1 PIM Expression of WS-Calendar Examples 3-05 27
Figure 5-2 Simple Meeting Schedule 29
Figure 7-1 WS-Calendar Target Classes 32
Figure 7-2 PIM Source Classes for DateTimeType and DurationType 32
Figure 7-3 PIM Source Class for ToleranceType 33
Figure 7-4 WS-Calendar Target Classes for Tolerance Type 33
Figure 7-5 PIM IntervalType and GluonType 34
Figure 7-6 Target IntervalType and GluonType Transforms 35
Figure 7-7 PIM RelationLinkType, LinkType, RelationshipType, and TemporalRelationshipType 36
Figure 7-8 PIM Vavailability Package Classes 38
Figure 7-9 Vavailability Package from iCalendar-availability-extension 38

List of Tables

Table 3-1: Semantics: Foundational Elements 11
Table 3-2: Semantics: Relations, Limits, and Constraints 12
Table 3-3: Semantics: Inheritance 12
Table 3-4: Semantics: Describing Intervals 13
Table 7-1 PIM to PSM DateTime and Duration Mappings 32
Table 7-2 PIM to PSM Mapping for DateTimeType and DurationType 32
Table 7-3 PIM Classes IntervalType and GluonType Transforms 35
Table 7-4 Attributes of PIM RelationLinkType and Transforms to WS-Calendar 37
Table 7-5 PIM Enumeration Member Transforms to WS-Calendar 37

1 Introduction

All text is normative unless otherwise labeled.

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). *Representations of dates and times, third edition*, December 2004, (ISO 8601:2004)

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

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

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

[WS-Calendar] *WS-Calendar Version 1.0*, 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)

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

[XMI] *XMI Version 2.1*, September 2005, Object Management Group, http://www.omg.org/spec/XMI/2.1/ [1]

1.3 Non-Normative References

[BPEL] **OASIS Standard, *Web Services Business Process Execution Language Version 2.0*,** https://www.oasis-open.org/standards#wsbpelv2.0

[BPMN] **OMG Standards**, http://www.bpmn.org/ .

[EnergyInteroperation] **OASIS Committee Specification**, *OASIS Energy Interoperation 1.0*, 02 December 2013, http://docs.oasis-open.org/energyinterop/ei/v1.0/cs03/energyinterop-v1.0-cs03.html

[EnterpriseArchitect] **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.

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

[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

[1] The version of XMI as of this Working Draft is 2.4.1, but the tools used for UML support version 2.1.

[PIM Examples]	**OASIS Committee Technical Note, *Examples for WS-Calendar Platform-Independent Model (PIM) Version 1.0***, 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]	**OASIS Committee Specification, *Reference Architecture Foundation for Service Oriented Architecture Version 1.0*,** http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.pdf
[SOA-RM]	**SOA-RM OASIS Standard, *OASIS Reference Model for Service Oriented Architecture 1.0*, October 2006 http://docs.oasis-open.org/soa-rm/v1.0/**
[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
[UML]	*Unified Modeling Language*, Object Management Group, http://uml.org/
[XMLSchema]	***XML Schema***, World Wide Web Consortium, http://www.w3.org/standards/xml/schema

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

```
durationShort
```

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

```
ToleranceValueType
```

Some UML predefined types are used, e.g. *string*.

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.

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

Information in the “Specification” column of the tables is normative. Information appearing in the “Note” column is explanatory and non-normative.

All sections explicitly noted as examples are informational and are not to be considered normative.

1.7 Architectural References and Background

WS-Calendar and this WS-Calendar PIM assume incorporation into services. Accordingly it assumes a certain background of definitions and discussion of roles, names, and interaction patterns. This document relies heavily on roles and interactions as 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 Architecture

The Object Management Group's Model Driven Architecture **[MDA-Overview][MDA]** provides a framework to describe relationships between Unified Modeling Language **[UML]** models.1

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.

2.1 The PIM and the WS-Calendar PSM

In this specification we define a PIM or Platform-Independent Model for the **[WS-Calendar]** extensions to IETF **[iCalendar][xCal]** and the relevant types included in **[xCal]**. 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. As such, the expression is very simple, but in the aggregate relatively complex and less suitable to UML expression—the several key types have sets of values and attributes associated with them in a relatively flat hierarchy.

This PIM addresses the key abstractions as defined in **[WS-Calendar]** in a manner that allows for understanding the nature and information model for those abstractions. In effect, we are creating a PIM with respect to which the WS-Calendar specification is a PSM. 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.

The MDA presumes transformations from UML models to UML models. The UML model for **[WS-Calendar]** is structured differently than the PIM. We describe transformations in detail in Section 7.

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

2.2 Key Abstractions

We define the following in order:

- Types for date, time, and duration
- The Interval
- Payload attachment to an Interval
- Relations
- The Gluon
- Tolerance
- Availability

2.3 Expression of the PIM

The PIM is a **[UML]** model. We represent the PIM as a normative **[XMI]** serialization of the model. The model is described using **[Enterprise Architect]**. The Enterprise Architect Project file is part of this work product but is non-normative.

2.4 Structure of the PIM Model and Specification

The PIM consists of a small number of key concepts and constructs as listed in Section 2.2. These are expressed in a largely flat structure, with a sub-package only for the Availability **[Vavailability]** abstractions.[2] We have not otherwise subdivided the core model, but expect (See Section 9) that conforming specifications and implementations MAY claim conformance to sub-parts of the PIM, e.g. to only the Interval.

We encourage consideration and 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. We avoid multiple fully qualified terms of the same end component name.

Many values in the XML Serialization **[xCAL]** of iCalendar are conformed strings, that is, strings with certain defined patterns. We require similar standardized formats for conformed strings, and record the type in this PIM as *string*, again to allow easy transformation between this PIM and the PSMs.

In a future version of this specification, the conformance rules will be included in the UML model; as of this Working Draft they are of type *string* with references to the standards source for the patterns.

2.5 Expression in UML

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 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.[3]

There are constraints and semantic rules and conformance that apply to a UML model defining the WS-Calendar PIM. The UML constraints allow for determining values in fully bound class instances that are less succinct than the standardized expressions.

For example, an instance of Interval (see Figure 4-3 The PIM IntervalType) might have only duration; the PIM, however, lists 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 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] The Vavailability definition is in process in the IETF.

[3] 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.

3 WS-Calendar PIM Terminology and Semantics

WS-Calendar PIM semantics are nearly identical to those defined in Section 1.9 of **[WS-Calendar]**. The minor differences between the referenced section and those in Section **Error! Reference source not found.** below are listed in Appendix C.

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. In addition, the definition of conformed strings for representation of duration and date and time per **[ISO8601]** is identical.

[WS-Calendar] defines XML and XML Schema artifacts; the terminology differs from UML, most obviously in that XML distinguishes between elements and attributes within a type, while UML uniformly uses the term *attributes*.

3.1 Time Intervals and Collections of Time-Related Intervals

Certain terms appear throughout this document and are defined in Table 3-1. Some terms are discussed in greater depth in later sections. In all cases, the normative definition is in this section.

WS-Calendar terminology begins with a 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, and are quoted from **[WS-Calendar]**. The definitions are normative because this is a standalone specification.

Table 3-1: Semantics: Foundational Elements

Time Segment	Definition
Component	In iCalendar, the primary information structure is a Component. Intervals and Gluons are new Components defined in this specification.
Duration	Duration is the length of an event scheduled. The **[xCal]** and **[RFC5545]** duration is a data type using the string representation defined in the iCalendar duration. In the PIM the value set from **[ISO8601]** is used instead; informally there are several additional representations for duration in the PIM but all durations from **[xCal]** are included in the PIM. See Section 4.2.
Interval	The Interval is a single Duration derived from the common calendar Components as defined in iCalendar (**[RFC5545]**). An Interval is 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 even simultaneous activities. A Sequence is re-locatable, i.e., it does not have a specific date and time. A Sequence may consist of a single Interval. A Sequence may optionally include a Lineage. A Sequence can be scheduled multiple times through repeated reference by different Gluons. Intervals are defined through their Duration, and the schedule, dtEnd or dtStart, is applied to the Sequence as a whole.
Partition	A Partition is a set of consecutive Intervals. The Partition includes the trivial case of a single Interval. Partitions are used to define a single service or behavior that varies over time. Examples include energy prices over time and energy usage over time.

Time Segment	Definition
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 service or schedule effects until applied to an Interval or Sequence.
Artifact	An Artifact is the information attached to, and presumably that occurs or is relevant to the time span described by an Interval. WS-Calendar uses the Artifact as a placeholder. The contents of the Artifact are not specified in WS-Calendar; rather the Artifact provides an extension base for the use of WS-Calendar in other specifications. 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.

182 WS-Calendar works with groups of Intervals that have relationships between them. These relations
183 constrain the final instantiation of a schedule-based service. Relations can control the ordering of
184 Intervals in a Sequence. They can describe when a service can be, or is prevented from, being invoked.
185 They establish the parameters for how information will be shared between elements using Inheritance.
186 The terminology for these relationships is defined in Table 3-2.

187 *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.

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

191 *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.
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 Element	A Bound Element refers to an Element 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.

Term	Definition
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 itself complete, but must still receive inheritance to be fully specified. A Sequence or Partition is Unbound if it contains at least one Interval that is Unbound.
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 said to be Scheduled when it is Anchored, Fully Bound, and service performance has been requested.
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.

202

4 The Platform-Independent Model

In this section we introduce the PIM, and treat in turn each component of the PIM. Each subsection has an introduction, a diagram, and discussion that may include the relationship of the respective 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]** 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]**, **[EnergyInteroperation]**, and **[EMIX]**.

4.1 Introduction to the PIM

In this section we present the entire PIM together with architectural discussion. The following sections begin with a description of the full model, and then address

- Types for Date and Time, Duration, and Tolerance
- The Interval
- Payload attachment an Interval
- The Gluon
- Relations
- Tolerance
- Availability

4.1.1 Model Diagram of the PIM

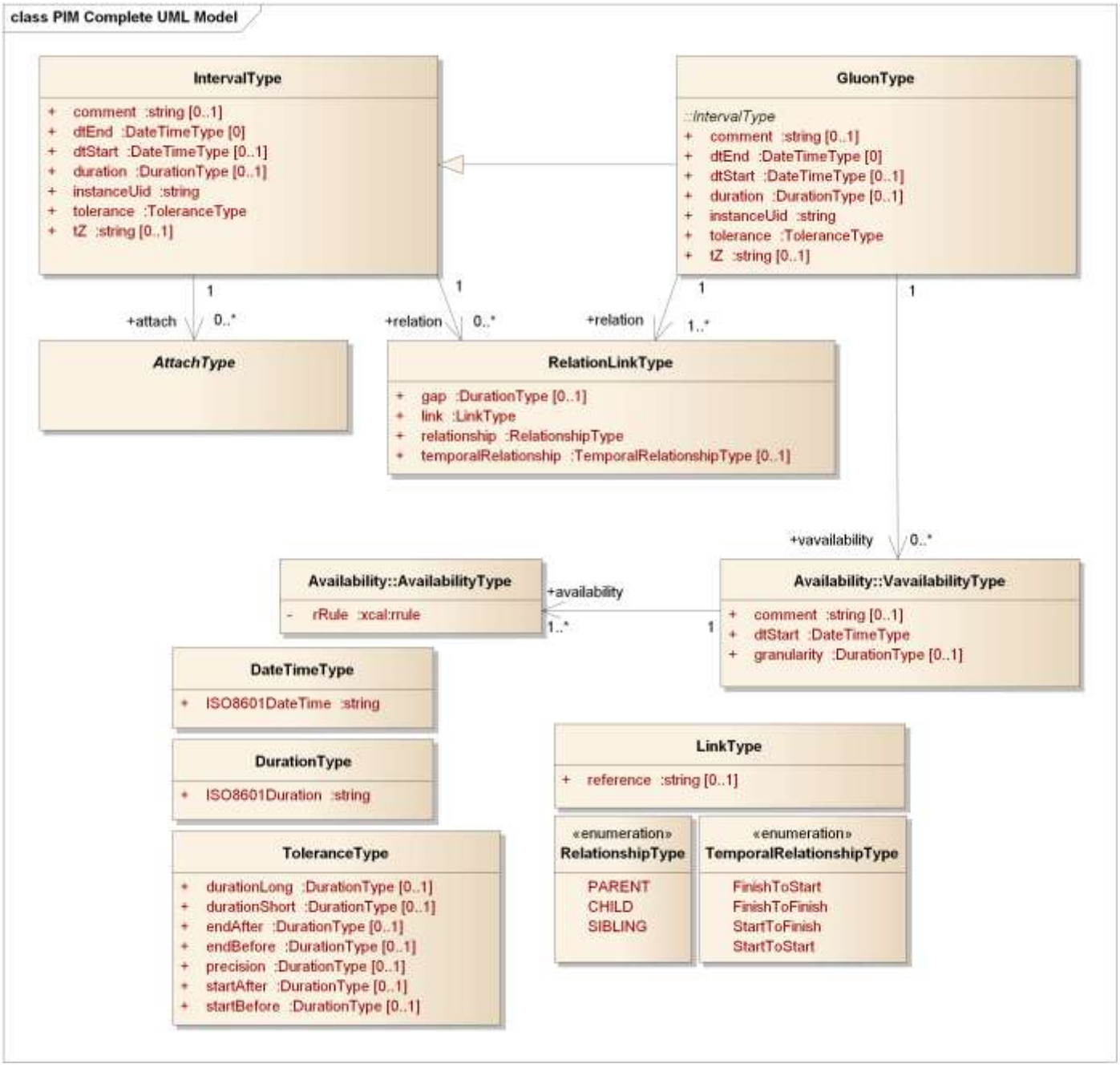


Figure 4-1 The Complete WS-Calendar PIM UML Model

4.1.2 Discussion

Primitive types express fundamental information related to date, time, and duration, and follow **[RFC5545] [ISO8601]** and are a superset of those expressed in **[iCalendar]**.[4]

[4] See discussion in Section 4.2 that includes relationship to **[XMLSchema]** DateTime types.

Associations in the PIM are directional, but profiles and PSMs derived or derivable from the PIM may have non-directional associations, vary the direction of associations to fit their particular platform(s) and purposes.[5]

Attributes and associations' cardinality is defined in the PIM; profiles and PSMs derived or derivable from 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] [ISO8601] [iCalendar]** attribute names wherever possible for ease of mapping to that common terminology. In particular, a fully bound Interval is defined by 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 (expressed as in **[ISO8601]**) for the Interval[6]

For UML model purposes, the three key values for an interval, only two of which are required in fully bound Intervals, are each optional. This permits a conforming instantiation to have zero or more of the three key values; the semantics of Gluons and Intervals in Section **Error! Reference source not found.** describe how information for a bound interval is determined. Note also that *GluonType* while a subclass of *IntervalType* has a more restrictive cardinality for *dtEnd* and for *relation*.

4.2 Types for Date and Time, Duration, and Tolerance

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

- DateTime
- DurationType
- ToleranceValueType

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 may 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* used to describe tolerance is also included

[5] Note that non-directional associations are a barrier to serializability in messages; hence all PSMs typically would use directional associations unless their purpose is to derive further PSMs.

[6] But see discussion below related to **[ISO8601]** strings and those omitted by **[xCal]** and by **[XML Schema]**.

4.2.1 Model Diagram

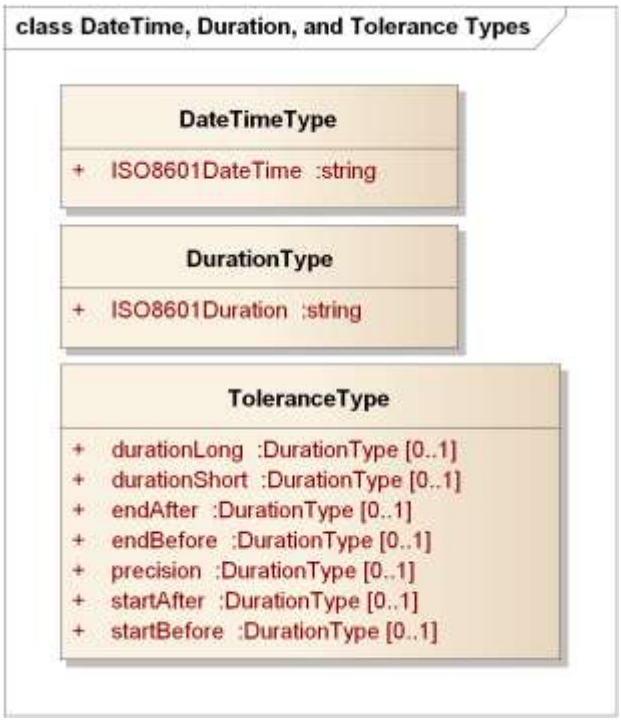


Figure 4-2 DateTimeType, DurationType, and ToleranceValueType

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]:**

- DateTime (See Section 3.3.5, Date-Time, in **[RFC5545]**)
- DurationValueType (See Section 3.3.6, Duration, in **[RFC5545]**)

Both DateTime and DurationValueType SHALL include **[ISO8601]** date & time and duration standard expression conformed strings, thus supporting a union of the **[XMLSchema]** and **[iCalendar]** expressions.

The class

- ToleranceValueType

is comprised of a set of optional attributes of DurationType. Tolerances can be expressed in any combination; however, 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 expected that a PSM MAY include consistency requirements and limitations on the attributes of ToleranceValueType that might be used.

Tolerances can allow (e.g.) randomization of intervals to ensure that certain activities do not occur "simultaneously." For example, a *startBefore* of 5s and *startAfter* of 10s indicates that the actual start time may be in the range from 5s before to 10s after the indicated dtStart. Additional deployment semantics for randomization may use this same expressed range. PSMs MAY include assumptions or explicit statement of e.g. probability density functions or other indications of expected behavior.

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 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 Interval

The Interval is a fundamental notion—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 expressions of a time interval, and significant differences in relocatability of schedules including Intervals.

4.3.1 Model Diagram

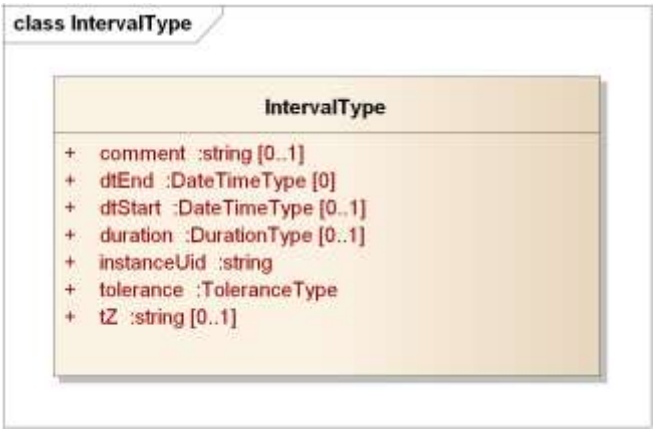


Figure 4-3 The PIM IntervalType

4.3.2 Discussion

The IntervalType class is the fundamental unit for expressing 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]

This follows **[WS-Calendar]**, the canonical PSM.

---

[7] While **[iCalendar]**, **[WS-Calendar],** and this PIM conform to **[ISO8601]**, none of them include all standard notations from 8601.Likewise **[XMLSchema]** does not include the full specification in **[ISO8601]**. In particular, there are duration strings included in **[XMLSchema]** that are not in **[iCalendar]** and *vice versa*.

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

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

See also Section 2.5.

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, as in **[WS-Calendar]** Payload values are attached to an Interval (or its subclass Gluon), as described in the next section.

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 the Interval instance in a concrete way.

4.4.1 Model Diagram

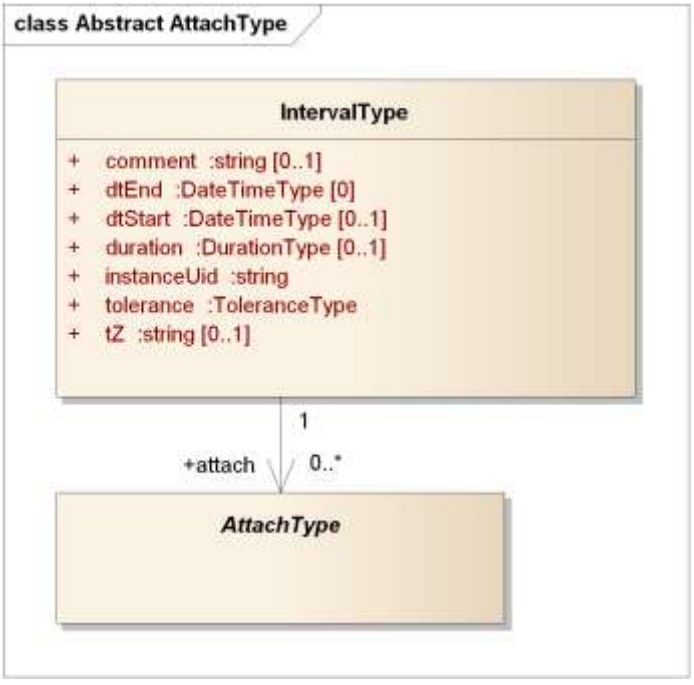


Figure 4-4 Attaching a Payload to an Interval

4.4.2 Discussion

[WS-Calendar] (line 219) requires that the Attachment Payload and Start Duration must be complete for service performance. For consistency with WS- we have defined the cardinality of *attach* to be 0..* to allow for abstract schedules, including those to which payloads are bound before service performance or concrete use.

A PSM claiming conformance to this PIM SHALL document any changes in cardinality or of associations used for payload attachment in its conformance statement.

4.4.3 Relationship to other PIM Components

The IntervalType is fundamental; application information is attached to instances of the IntervalType by a clear, directional association. This maintains the temporal structure of schedules with a variety of attachments of various types.

4.5 Gluons

The Gluon was defined in **[WS-Calendar]**. 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 indicated values are determined by the referencing gluon and values that may be inherited from the referencing gluon such as start time and duration. See Section 9.

More formally, a Gluon references schedules comprised of related Intervals and Gluons, while providing that logical information such as the duration of Interval instances may be determined by inheritance from other Interval instances. The structure permits directed graphs of instances with reuse of components. Those sub graphs may therefore act as reusable sub-schedules, or considered as sub-routines.

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 there be at least one *RelationLink*; *IntervalType* has zero or more associated *RelationLink*s.

These relationships are capable of being used to compose arbitrarily complex graphs of instances of Intervals and Gluons. See the examples in Section 5 and references.

4.5.1 Model Diagram

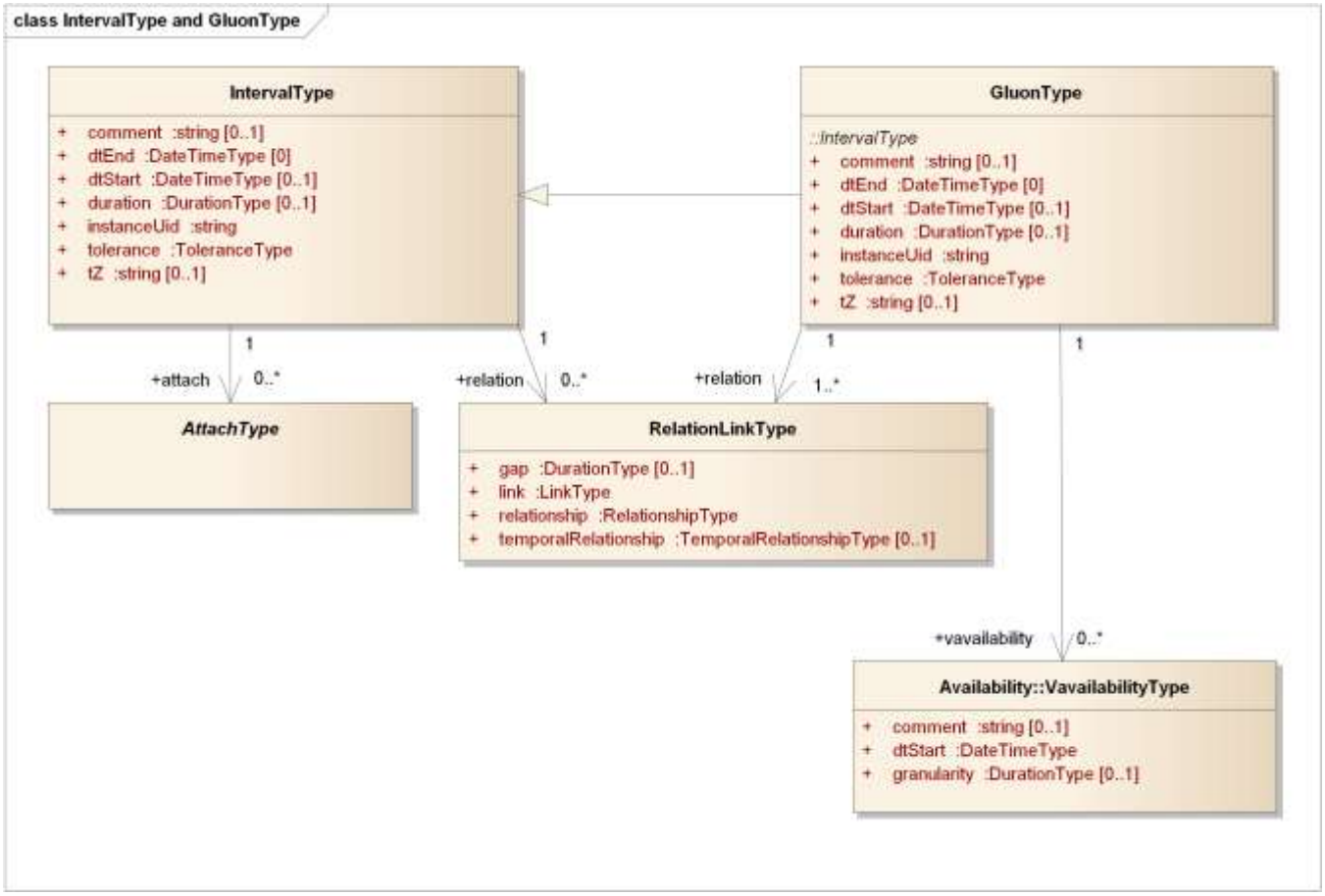


Figure 4-5 Gluons, Intervals, and Relationship Links

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.

This technique is used in **[EMIX]** and **[EnergyInteroperation]** to build energy schedules with varying values but consistent lengths.

4.5.3 Relationship to other PIM Components

See the detailed discussion of Semantics in Section **Error! Reference source not found.**. Gluons contain values that may be inherited or overridden in its children in accordance with Section 9.2.

4.6 Relationships among Gluons and Intervals

Relationships between instances of IntervalType are accomplished with RelationLinks. In **[WS-Calendar]** the LinkType is defined to be a UID (as defined in **[xCal]**), 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 Temporal Relationship and gap together determine the relationship of the referencing Interval and referenced Interval instances.

The gap SHALL be expressed as an ISO8601 duration, taken as a signed offset. 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 gap attribute in a PIM object SHALL be the equivalent of a gap of zero duration.

The TemporalRelationshipType enumeration SHALL express 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 express whether the linked Interval is a CHILD of the linking object.[9]

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.

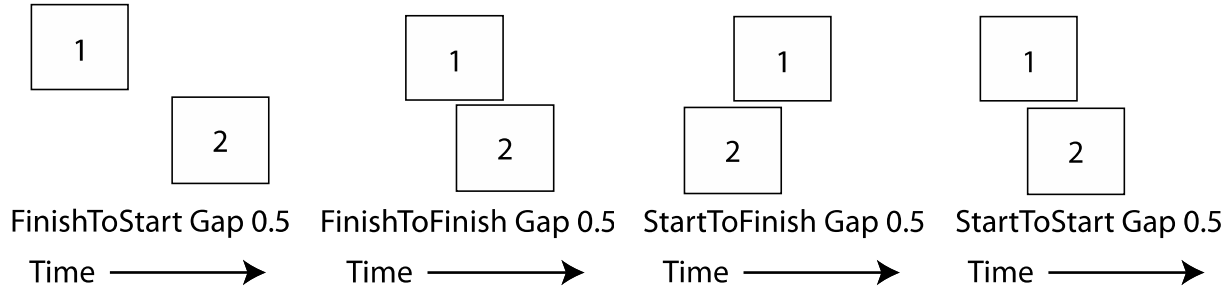


Figure 4-6 Temporal Relationships

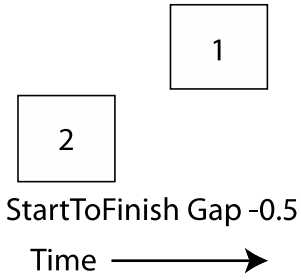
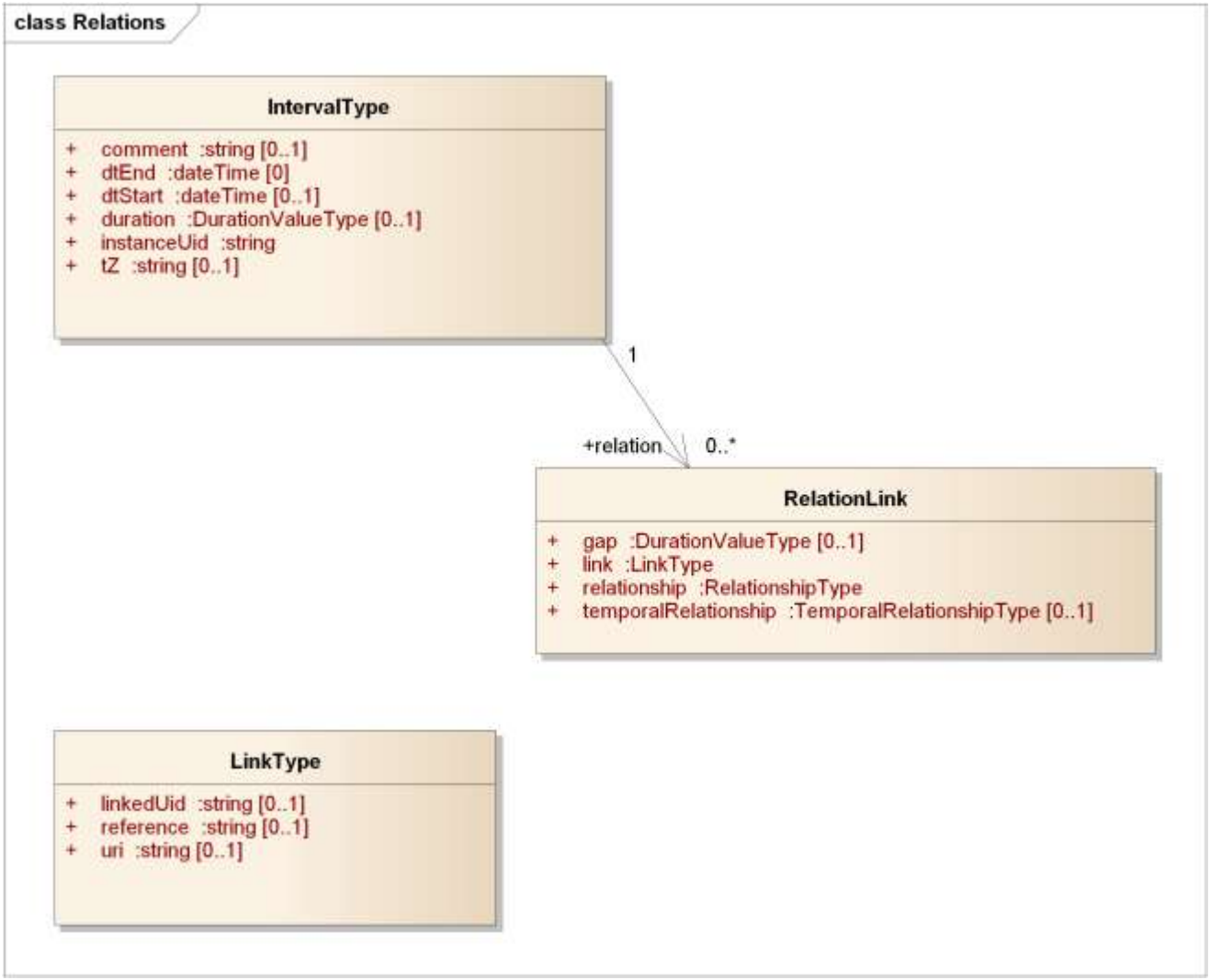


Figure 4-7 Temporal Relationship--StartToFinish Negative 0.5 Gap

4.6.1 Model Diagram

[9] **[WS-Calendar]** and **[xCal]**, as with many other IETF RFCs,, also include as enumeration values 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.



405

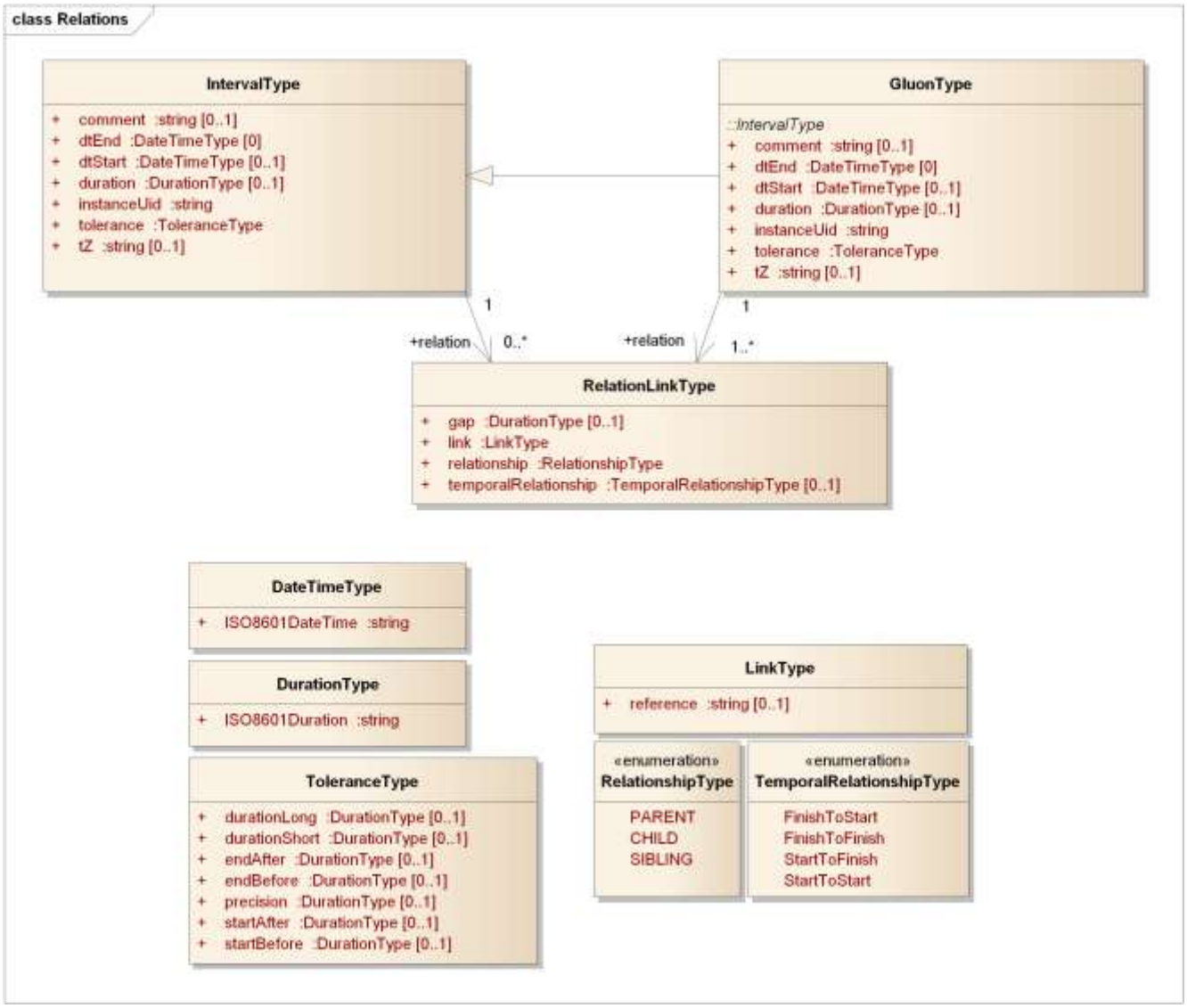


Figure 4-8 Relation Link Type and Relationship Types

4.6.2 Discussion

The PIM and WS-Calendar both support the common relationships between time intervals, as expressed in facility, energy, and other schedules, project management tools, and business process definitions such as **[BPEL]** and **[BPMN]**.

The relationships are expressed using the temporal relationship, the temporal gap between intervals, and the kind of relationship between Gluons and Intervals expressed as Parent, Child, and Sibling.

The set is logically complete, and allows complex structures to be built from primitive relationships, passed in service invocations, and interpreted unambiguously.

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 manages and maintains links. In particular, the conformance statement SHALL include a description of uniqueness of references in that PSM.

4.6.3 Relationship to other PIM Components

Relation 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, excepting dependency-defined gaps.

4.7 Availability

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.

4.7.1 Model Diagram

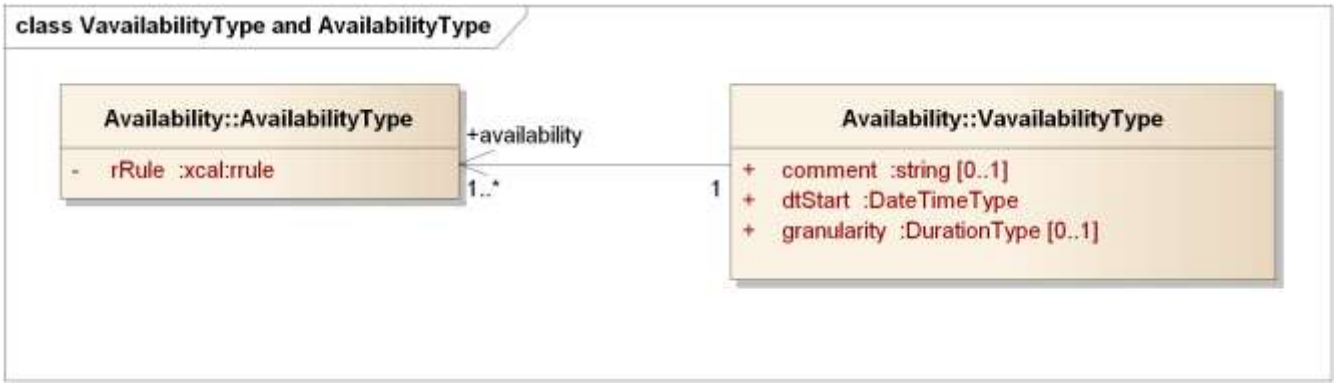


Figure 4-9 Vavailability and Availability Recurrence Rules

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 will need future adaptation to match the abstraction level of this PIM.

4.7.3 Relationship to other PIM Components

Consumes recurrence relationships from **[Vavailability].** Used in consumers of WS-Calendar to express availability for (e.g.) Demand Response events in **[EnergyInteroperation]**. Not used by other parts of the PIM.

5 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.

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

5.1 Related Intervals

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

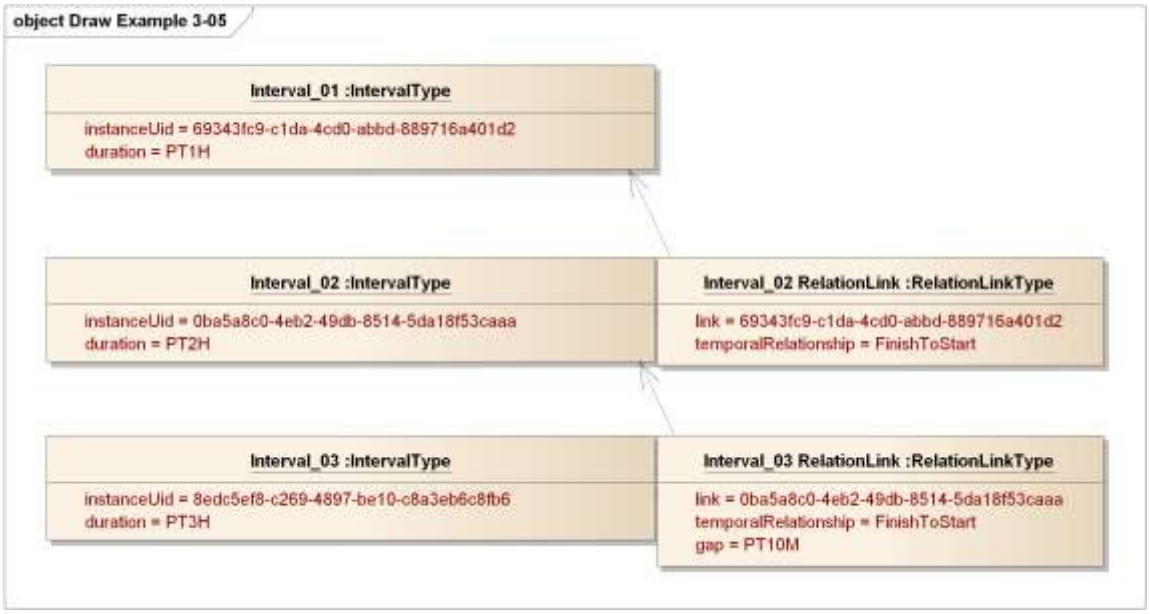


Figure 5-1 PIM Expression of WS-Calendar Examples 3-05

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

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

5.2 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 5-2 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 CHILDren of 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.

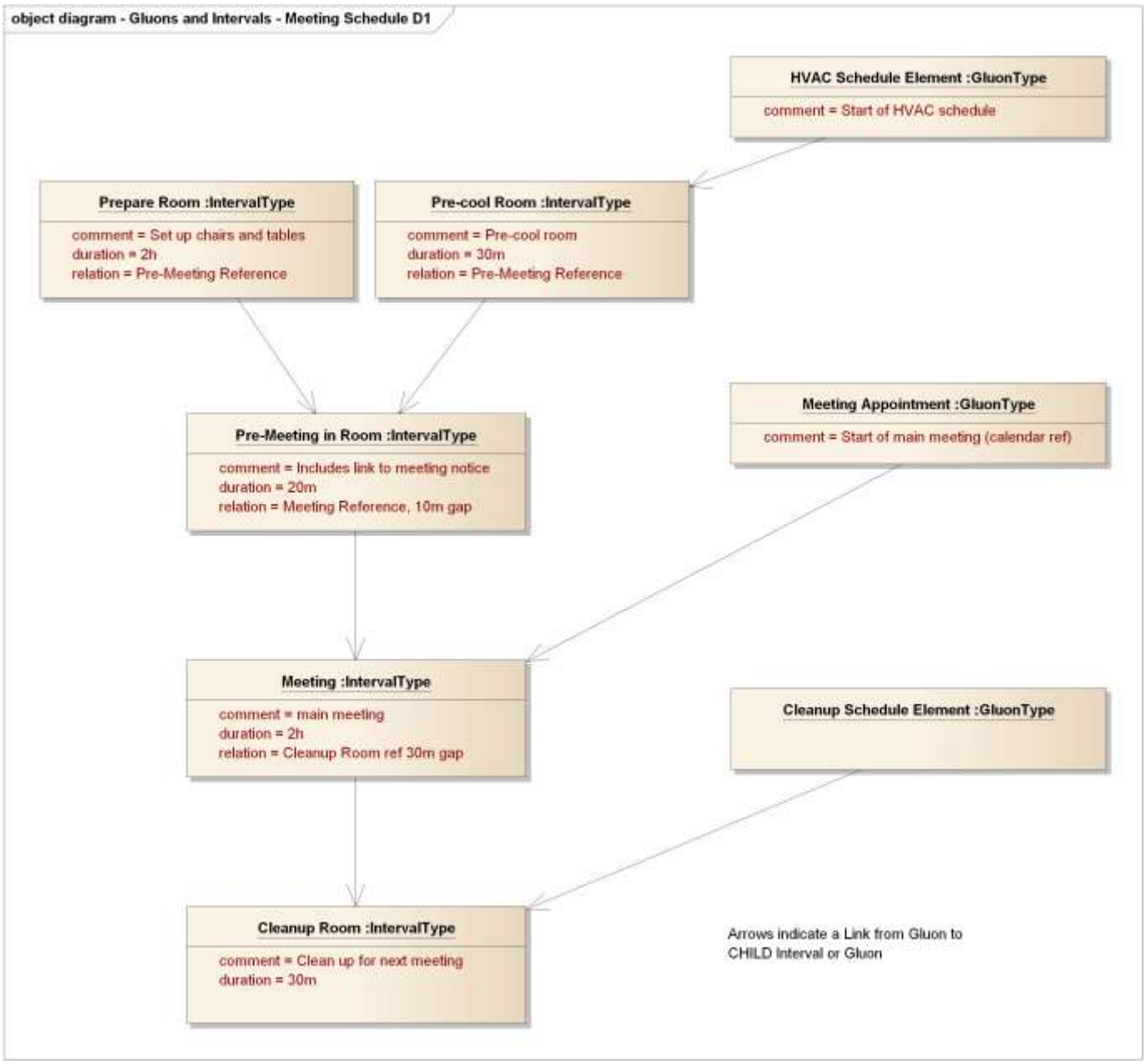
Finally, the actions are all based on where you reference the schedule—working back from the start time (inherited from the start of main meeting gluon) the pre-meeting is 30 minutes earlier, and the setup is 2 hours and 30 minutes earlier.

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).



491

492 *Figure 5-2 Simple Meeting Schedule*

6 Architectural Basis for the PIM (Non-Normative)

The PIM is defined as a more abstract model for describing and communicating schedules as defined in **[WS-Calendar]**, **[EMIX]**, **[EnergyInteroperation]**, **[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, e.g. those in **[Aho]** and **[Knuth]**.

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.

7 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.

The 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.

7.1 General Transformation

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.

7.2 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

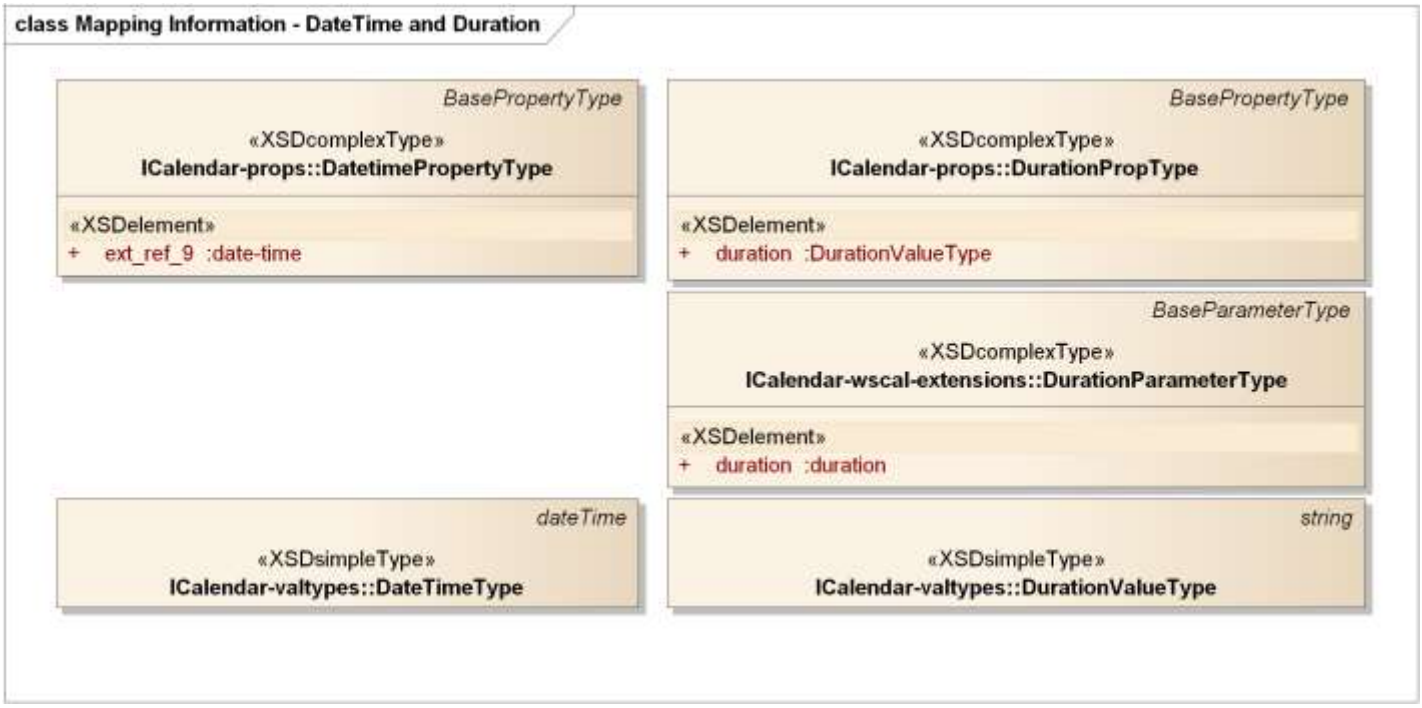
- DateTime and Duration Types, the fundamental types for talking about time and schedule
- ToleranceType
- Intervals and Gluons
- Relationships
- Vavailability

7.2.1 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.

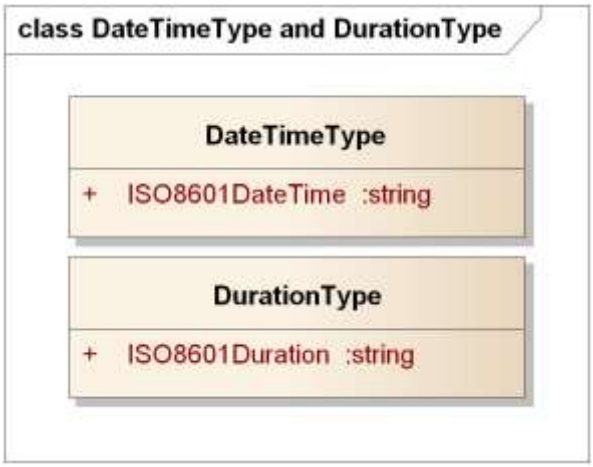
There are different conformed strings for DurationType and DateTimeType. The PIM uses **[ISO8601]** duration and date time semantics; these are isolated in the PIM classes DateTimeType and DurationType to facilitate mapping to classes in PSMs including those based on **[WS-Calendar]** and **[XMLSchema]**.

544 *Table 7-1 PIM to PSM DateTime and Duration Mappings*



545

546 *Figure 7-1 WS-Calendar Target Classes*



547

548 *Figure 7-2 PIM Source Classes for DateTimeType and DurationType*

549 *Table 7-2 PIM to PSM Mapping for DateTimeType and DurationType*

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

7.2.2 Transformation for Tolerance Type

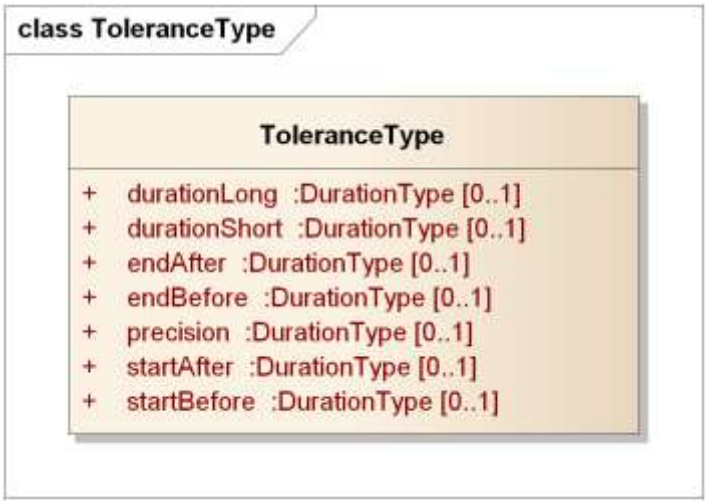


Figure 7-3 PIM Source Class for ToleranceType

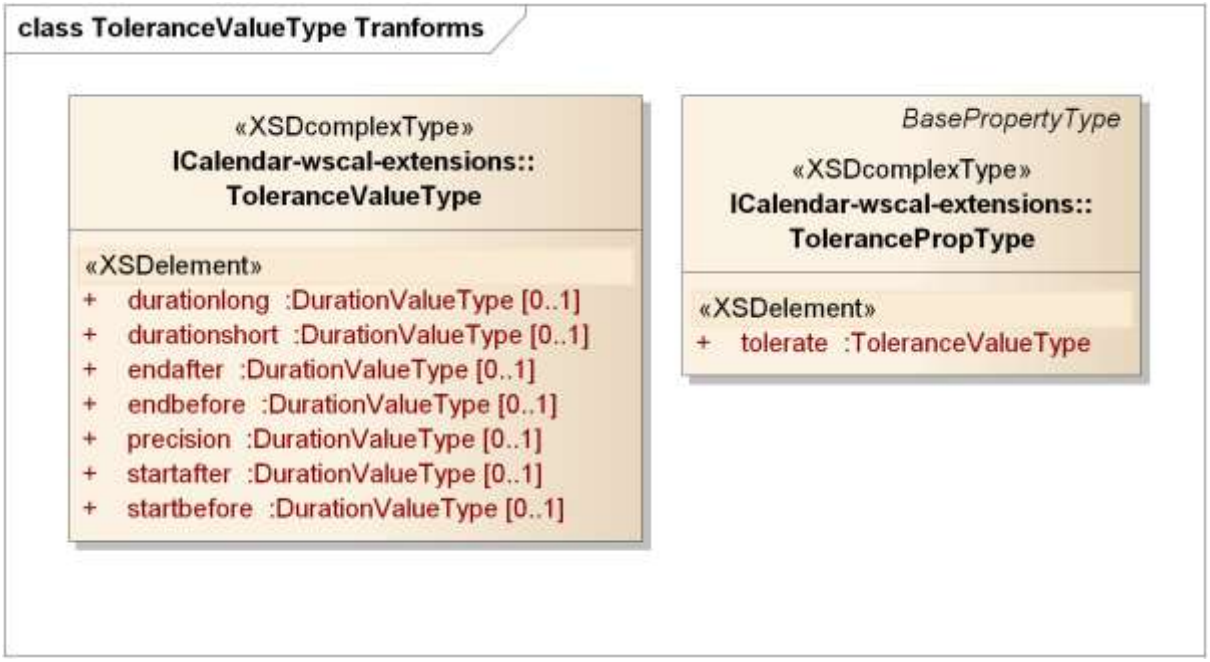


Figure 7-4 WS-Calendar Target Classes for Tolerance Type

The PIM ToleranceType is identical in attribute names and types to the WS-Calendar class with the same function, as shown in Figure 7-3 and Figure 7-4 above. The only differences are that PIM uses *DurationType*, the WS-Calendar mapping is to WS-Calendar *ToleranceValueType*, and the PIM attribute names are lowerCamelCase rather than lower case.

PIM Class Name	WS-Calendar Class Name	Notes
ToleranceType	ICalendar-wscal-extensions:: ToleranceValueType	Attributes map respectively to attributes of the same name; Types map per Section 7.2.1.

7.2.3 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 7-5:

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

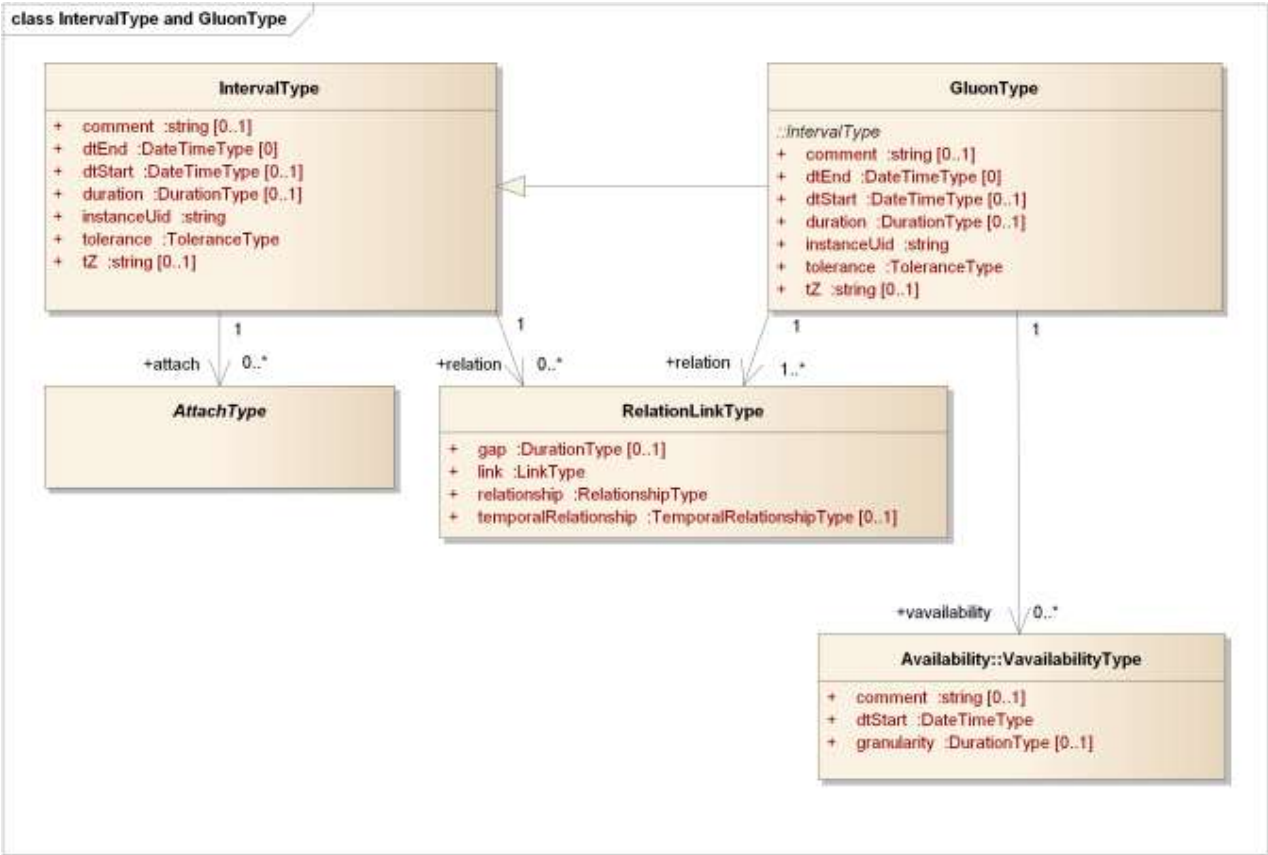


Figure 7-5 PIM IntervalType and GluonType

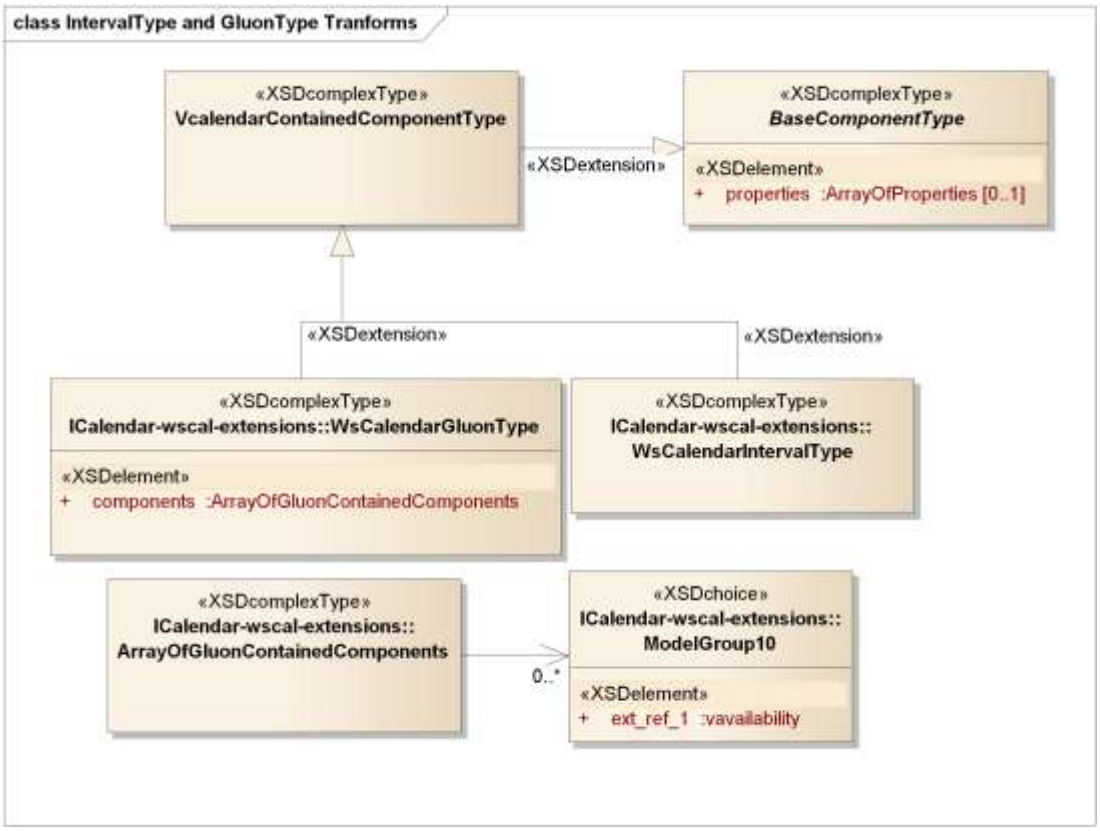


Figure 7-6 Target IntervalType and GluonType Transforms

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 Classes IntervalType and GluonType Transforms

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	
tolerance: ToleranceType	ICalendar-wscal-extensions::ToleranceValueType	Attribute of TolerancePropType is *tolerate*
Tz: string	ICalendar-Props::TzidPropType	Constrained string per **[RFC5545]**

PIM Attribute and Type	WS-Calendar Target Type	Notes
Relation: RelationLinkType	ICalendar-link-extension::LinkPropType	Target has possible UID, URI, Reference attributes

7.2.4 Transformation for Relationships

In this section we detail transformations for RelationLinkType and the types for its attributes: LinkType, RelationshipType, and TemporalRelationshipType.

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.

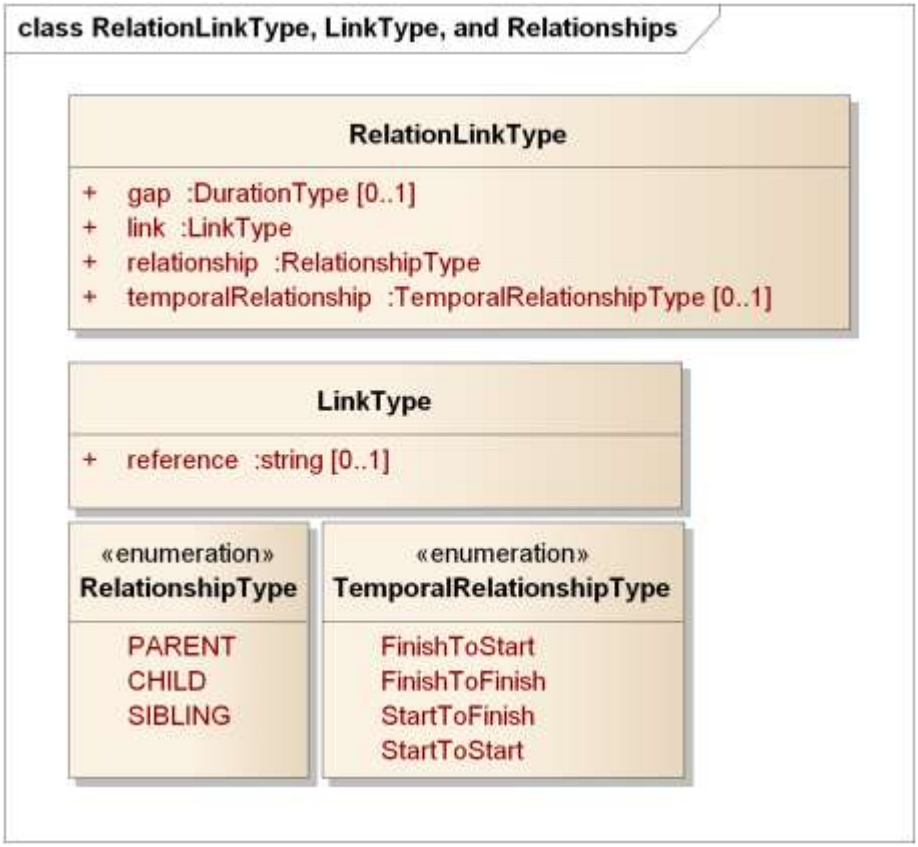


Figure 7-7 PIM RelationLinkType, LinkType, RelationshipType, and TemporalRelationshipType

The following table describes the transformation in detail for the classes and enumerations in Figure 7-7. 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]**.

593 *Table 7-4 Attributes of PIM RelationLinkType and Transforms to WS-Calendar*

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

PIM Attribute and Type	WS-Calendar Target Type	Notes
gap: DurationType	ICalendar-Params::DurationParameterType	Duration is an **[ISO8601]** conformed string which maps to a constrained string per **[RFC5545]**
Link: LinkType	ICalendar-props::RelatedToPropType	All of the RelatedToPropType extended choices are strings (uri, uid, and text). PIM LinkType is a string.
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

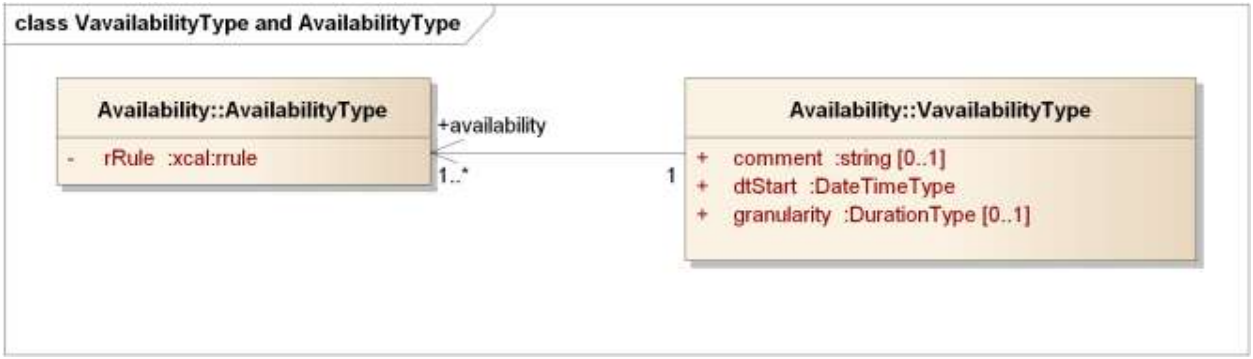
596 lowerCamelCase.

597 *Table 7-5 PIM Enumeration Member Transforms to WS-Calendar*

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

598 ### 7.2.5 Transformation for Vavailability

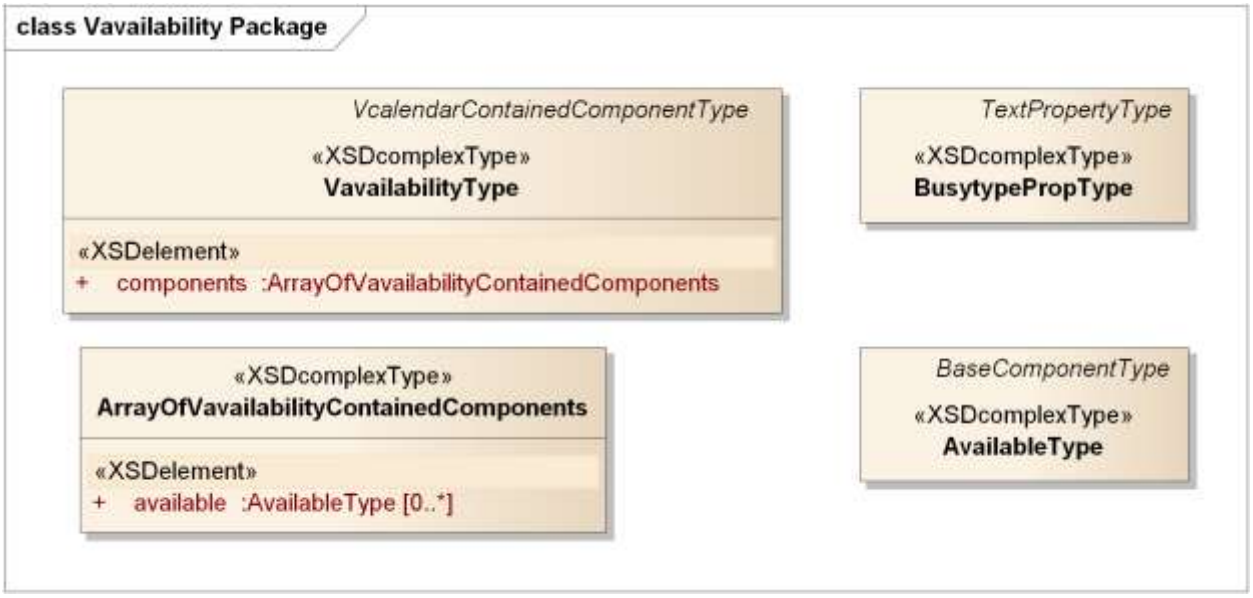
599 Vavailability is described in a separate package in the PIM. A future Working Draft will disconnect
600 Vavailability from use of **[xCal]** types, so that the mapping is clearer and is disconnected from changes in
601 **[Vavailability]** as it completes the IETF process.



602

603 *Figure 7-8 PIM Vavailability Package Classes*

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



605

606 *Figure 7-9 Vavailability Package from iCalendar-availability-extension*

607 The VavailabilityType has zero or more AvailabilityType objects inside. The *rRule*s matched a previous
608 draft of **[Vavailability]** and is expected to work with the final standard version.

8 PIM to IEC TC57 CIM Intervals and Sequences

The IEC TC57 Common Information Model **[IEC CIM]** uses time intervals in a variety of ways. A transformation from a fully bound PIM interval (which uses dtStart and duration and time zone) to a CIM interval (which uses dtStart and dtEnd in UTC) is straightforward:

(1) the CIM dtStart is the UTC equivalent of the (dtStart, time zone) pair in the PIM Interval
(2) the CIM dtEnd is the CIM dtStart plus the PIM Interval duration

In some cases intervals are contained in the related data; the sense of the inclusion needs to be reversed with the class containing the CIM interval being referenced as an attachment by the PIM interval.

Because of the interval information inserted in each data item, and including *dtStart* and *dtEnd*, sequences of CIM intervals are not relocatable in the same way as PIM and **[WS-Calendar]** Intervals. CIM intervals would each be modified with the new dtStart and dtEnd. Changing dtStart in the Designated Interval or in a referencing Gluon relocates a PIM sequence.

9 Conformance and Rules for WS-Calendar PIM and Referencing Specifications

This Conformance section differs in minor detail from that in **[WS-Calendar]**. The conformance behavior is in general identical to that of WS-Calendar; see Appendix D for details. We do not describe changes in that Appendix that involve the use of the name “WS-Calendar PIM” rather than “WS-Calendar.”

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

If the implementer and/or implementation claiming conformance is using WS-Calendar PIM as part of a larger business or service communication, they SHALL follow not only the semantic rules herein, but SHALL also conform to the rules for specifying inheritance in referencing standards.

9.1 Relationship to WS-Calendar [Non-Normative]

This Platform-Independent Model for WS-Calendar shares all of the conformance statements from **[WS-Calendar]** subject to

- Renaming of attributes (e.g., *UID* from **[WS-Calendar]** is named *instanceID*)
- Disambiguation of rules (e.g., “Intervals SHALL have a Duration AND (either a dtStart OR a dtEnd)” in 9.2.5.1)
- Simplification (e.g., the section 9.2.5.2)
- Rewording to define conformance to WS-Calendar PIM rather than **[WS-Calendar]**.

9.2 Conformance Rules for WS-Calendar PIM

There are five kinds of conformance that must be addressed for WS-Calendar and specifications that reference WS-Calendar. This PIM references WS-Calendar and requires the same conformance rules.

- Conformance to the ***inheritance rules*** in WS-Calendar, 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

9.2.1 Inheritance in WS-Calendar

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.

9.2.2 Specific Attribute Inheritance

In WS-Calendar and this PIM 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

In WS-Calendar and this PIM the following attributes MUST NOT be inherited

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

9.2.3 General Conformance Issues

This specification is general purpose. 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. Consider a standard format recommendation.
- **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.
- **Business Purpose** – Because WS-Calendar is general purpose, it does not distinguish between different exchanges 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 the Gluon to Anchor the Interval. If the distinction between Unanchored and Anchored Interval is critical for a set of interactions, the referencing specification SHALL indicate the proper form for a given exchange.

9.2.4 Covarying Elements

Some elements of WS-Calendar and 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. This becomes important if one or more of a covarying set have default values. In that case, if any are present, then inheritance should deem they are all present, albeit some perhaps in their default values.

9.2.5 Conformance of Intervals

9.2.5.1 Intervals

WS-Calendar PIM Intervals SHALL have a Duration.

Intervals MAY have a Start Time.

Intervals SHALL have a Duration AND optionally dtStart. If a non-compliant Interval is received in a service operation with dtEnd, then the dtEnd SHALL be ignored.

Within a Sequence, a maximum of a single Interval MAY have a dtStart or a dtEnd.

9.2.5.2 Other Elements

A Gluon may have a dtStart value.

9.2.6 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

9.3 Conformance Rules 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 covariant elements, those elements SHALL be clearly designated in the specification.

Specifications that normatively reference and claim conformance with the WS-Calendar PIM SHALL define the business meaning of zero duration Intervals.

9.4 Security Considerations

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

745 security of those systems should be considered as well. Those concerns are out of scope for this
746 specification.

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.

754

Appendix C. PIM and WS-Calendar Semantics Differences

The following is a non-normative list of changes required to convert the **[WS-Calendar]** Section 1.9 Semantics section to the Semantics section of the PIM.

We have excluded changes to table numbering, page footers, and purely typographic changes such as deletion of extra spaces.

Line numbers are with respect to **[WS-Calendar]** in PDF form.

Line Number	***Change to [WS-Calendar] to PIM***
200-201	Added references to WS-Calendar and PIM tables
202 Table 1-3, Gluon Entry	Changed "…gluon is influences…" to "…gluon influences…" (typographic)
202 Table 1-3, Artifact Entry	Changed first sentence to "An Artifact is the information attached to, and presumably that occurs or is relevant to the time span described by an Interval."
208, Table 1-4, Busy Entry	Changed "Busy often overlays is overlaid by Availability" to "Busy often overlays Availability."

Appendix D. PIM and WS-Calendar Conformance Differences

The following is a non-normative list of changes required to convert the **[WS-Calendar]** Section 4 Conformance and Rules for WS-Calendar and Referencing Specification to Section 5 of the PIM.

We have excluded changes to table numbering, page footers, and purely typographic changes such as deletion of extra spaces. Text was reworded to refer to the PIM rather than WS-Calendar as needed; such changes are not captured here.

Line numbers are with respect to **[WS-Calendar]** in PDF form.

Line Number	***Change to [WS-Calendar] to PIM***
1450-1453 Introduction	Modified Introduction to apply to the WS-Calendar PIM.
1490	Changed “UID (Gluons and Intervals)” to “instanceUid (Gluons and Intervals)”
1491	Added “Relationship Links” to list.
1522-1523	Changed “Duration AND a dtStart OR a dtEnd” to “Duration AND optionally dtStart.” Changed “received with both a dtStart and a dtEnd then the dtEnd SHALL be ignored” to” “received in a service operation with dtEnd then the dtEnd SHALL be ignored.”
1525-1529	Replaced with “A Gluon may have a dtStart value>” Other conditions are excluded by the UML in PIM.
1550	Change “override” to “modify” […the Proximity, Direction, or Override Rules.]

NOTE: This table was current as of PIM WD05; needs to be updated.