WS-Calendar Minimal PIM-Conformant Schema Version 1.0

Working Draft 02

22 July 2015

Technical Committee:
OASIS Web Services Calendar (WS-Calendar) TC

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

Editor:
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:
• XML schemas: (list file names or directory name)
• Other parts (list titles and/or file names)

Related work:
This specification is related to:

Declared XML namespaces:
• http://docs.oasis-open.org/ws-calendar/ns/**

Abstract:
iCalendar (RFC5545) and its peer specification XCAL (also in WS-Calendar 1.0) is a well-known and long used means to convey schedule-related information. iCalendar makes extensive use of extension and recursion. The WS-Calendar Platform Independent Model (PIM) constrains iCalendar and defines a simpler information model which shares iCalendar semantics and can be used to create as the common basis for any number of Platform Specific Models (PSMs).

Because an information model is abstract, it can apply to many transmission and serialization schemas. The PIM itself does not include a transmission and serialization schemas. Through transitive conformance such PSMs themselves conform to WS-Calendar.

The Minimal PIM-Conformant (MPC) schema defines an XML Schema that conforms just with the PIM. MPC can be used by itself or as a seed-schema for other specifications.

Status:
This Working Draft (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or approved as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document Approval Process begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

Copyright © OASIS Open 2012-2015. All Rights Reserved.
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.
# Table of Contents

1  Introduction .......................................................................................................................... 5
1.1  Terminology ...................................................................................................................... 5
1.2  Normative References ...................................................................................................... 6
1.3  Non-Normative References .............................................................................................. 6
1.4  Namespace .......................................................................................................................... 6
1.5  Naming Conventions ......................................................................................................... 7
1.6  Editing Conventions ......................................................................................................... 7
2  Overview of the Minimal PIM-Conformant Schema ............................................................ 8
2.1  Schedule Semantics from WS-Calendar PIM .................................................................... 8
2.2  Schedules and Inheritance ............................................................................................... 9
2.3  When: Start, End and Duration ....................................................................................... 10
3  Low Level Semantic Elements ........................................................................................... 11
3.1  When: Start, End and Duration ....................................................................................... 11
3.2  Specifying Date and Time ............................................................................................... 11
3.3  The Duration-based Types ............................................................................................. 12
   3.3.1  Tolerance .................................................................................................................. 12
3.4  Instance IDs and Links ..................................................................................................... 13
3.5  Other Simple Types ......................................................................................................... 14
4  Core Components: Intervals, Sequences, and Gluons .......................................................... 15
4.1  The Interval ...................................................................................................................... 15
4.2  Temporal Links and Sequences ........................................................................................ 16
   4.2.1  Temporal Links .......................................................................................................... 16
   4.2.2  Sequences .................................................................................................................. 17
4.3  The Gluon ........................................................................................................................ 18
5  Enhancing Service Advertising and Request: Recurrence and Availability ......................... 20
5.1  Recurrence Rules ............................................................................................................. 20
5.2  Recurrence ....................................................................................................................... 21
5.3  Availability and VAvailability .......................................................................................... 21
   5.3.1  Availability ................................................................................................................ 21
   5.3.2  VAvailability ............................................................................................................. 22
6  Conformance ....................................................................................................................... 24
   6.1  Conformance with the Semantic Models of WS-Calendar-PIM .................................... 24
Appendix A.  Acknowledgments .............................................................................................. 25
Appendix B.  Revision History ................................................................................................. 26
Table of Figures

1. Figure 2-1: Basic Power Object from EMIX ................................................................. 9
2. Figure 2-2: WS-Calendar Partition, a simple sequence of 5 intervals ................................ 9
3. Figure 2-3: Applying Basic Power to a Sequence ......................................................... 9
4. Figure 2-4: Simplifying back to Power in a Single Interval ............................................. 10
5. Figure 3-1: Date-Time-Date type .................................................................................. 11
6. Figure 3-2: Tolerance .................................................................................................. 13
7. Figure 4-1: The Interval ............................................................................................... 16
8. Figure 4-2: The Temporal Link ..................................................................................... 17
9. Figure 4-3: The Gluon ................................................................................................. 18
10. Figure 5-1: The Recurrence Rule .................................................................................. 20
11. Figure 5-2: Recurrence ............................................................................................... 21
12. Figure 5-3: Availability .............................................................................................. 22
13. Figure 5-4: VAValiability Type ..................................................................................... 23

Table of Tables

14. Table 1-1: Namespaces Used in this Specification ..................................................... 6
15. Table 2-1: Core Semantics from WS-Calendar ........................................................... 8
16. Table 2-2: WS-Calendar Semantics: Inheritance (non-normative) .............................. 10
17. Table 3-1: Date Time Types ....................................................................................... 11
18. Table 3-2: Performance-related Duration Types ......................................................... 12
19. Table 3-3: ID-Related Types ...................................................................................... 13
20. Table 4-1: High Level Component Types of WS-Calendar MIN ................................. 15
21. Table 4-2: Temporal Relations ................................................................................... 17
22. Table 5-1: Computable Schedules: Availability and Gluon ......................................... 20
1 Introduction

All text is normative unless otherwise labeled.

iCalendar (RFC5545) and its peer specification XCAL (also in WS-Calendar 1.0) is a well-known and long
used means to convey schedule-related information. iCalendar makes extensive use of extension and
recursion. The WS-Calendar Platform Independent Model (PIM) constrains iCalendar and defines a
simpler information model which shares iCalendar semantics and can be used to create as the common
basis for any number of Platform Specific Models (PSMs).

The iCalendar model is almost infinitely malleable in the number and manner of intervals in time that it
can communicate. Separate intervals exist as separate calendar information objects; a single
communication can include any number of these objects. This model is verbose in that each of these
calendar information objects must include all distinct information.

A key concern for [WS-Calendar] is direct compatibility with xCal, the XML Format for iCalendar defined
in [RFC6321]. While this format is flexible, it can offer too much optionality to be easily analyzed. To this
end, the TC developed a Platform Independent Model [WS-Calendar PIM] which supports all the
functions and messages from WS-Calendar, while restricting extension so that the models can be
analyzed and validated. This approach redefined WS-Calendar as what Model Driven Architecture calls a
Platform Specific Model (PSM) which conforms to [WS-Calendar PIM]

The Platform Independent Model [WS-Calendar PIM] describes how to make use of the general model
and semantics defined in [WS-Calendar] when defining information exchanges subject to specific
constraints. Artifacts that are conformant with [WS-Calendar PIM] can be transformed into a form that is
conformant to [WS-Calendar], even while their expression may not support the general purpose
expression required for [WS-Calendar].

[WS-Calendar PIM] is a general specification and makes no assumptions about how its information
model is used. [WS-Calendar PIM] has specific rules which define Inheritance as a means to reduce the
conveyance of repetitive information. As this specification constrains schedule communications to specific
business interactions, these inheritance rules are extended to embrace rules of interaction and rules of
process that further reduce the information that must be expressed in each interval.

Because an information model is abstract, it can apply to many transmission and serialization schemas.
The PIM itself does not include a transmission and serialization schemas. Through transitive
conformance such PSMs themselves conform to WS-Calendar.

[WS-Calendar PIM] does not define a normative structure for the information conveyed. [WS-Calendar
PIM] is an information model, and information models can be conveyed in a number of ways. High speed
transaction processing requires more predictable means to convey structured information concerning
time-based events, states, and transactions. Even legal and conformant conveyances of calendar
information may fail to meet the requirements for basic interoperability requirements [WSI-Basic].

The document defines a normative structure for conveying time series of information that is conformant
with [WS-Calendar PIM]. It is the intent of the TC meet the requirements of [WSI-Basic].

The Minimal PIM-Conformant (MPC) schema defines an XML Schema that conforms just with the PIM.
MPC can be used by itself or as a seed-schema for other specifications, There is a common need to
communicate information linked to repetitive intervals of time, for history, for telemetry, for projections,
and for bids. Such communications benefit from a common model for conveying these series of
information.

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


1.3 Non-Normative References


1.4 Namespace

The XML namespace [XML-ns] URI that MUST be used by implementations of this specification is:

http://docs.oasis-open.org/ws-calendar/ns/****

Dereferencing the above URI will produce the Resource Directory Description Language [RDDL 2.0] document that describes this namespace.

Table 1 lists the XML namespaces that are used in this specification. The choice of any namespace prefix is arbitrary and not semantically significant.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
</tr>
<tr>
<td>min</td>
<td><a href="http://docs.oasis-open.org/ws-calendar/ns/">http://docs.oasis-open.org/ws-calendar/ns/</a>****</td>
</tr>
</tbody>
</table>

The normative schemas for WS-Calendar MPC can be found linked from the namespace document that is located at the namespace URI specified above.
1.5 Naming Conventions

This specification follows some naming conventions for artifacts defined by the specification, as follows:

For the names of elements and the names of attributes within XSD files, the names follow the lowerCamelCase convention, with all names starting with a lower case letter. For example,

```xml
<element name="componentType" type="ComponentType"/>
```

For the names of types within XSD files, the names follow the UpperCamelCase convention with all names starting with a lower case letter prefixed by “type-“. For example,

```xml
<complexType name="ComponentType">
```

For the names of intents, the names follow the lowerCamelCase convention, with all names starting with a lower case letter, EXCEPT for cases where the intent represents an established acronym, in which case the entire name is in upper case.

1.6 Editing Conventions

For readability, element names in tables appear as separate words. The actual names are lowerCamelCase, as specified above, and as they appear in the XML schemas.

All elements 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.
2 Overview of the Minimal PIM-Conformant Schema

This section provides a non-normative introduction to WS-Calendar and the WS-Calendar PIM. It is provided solely for the convenience of the practitioner. The normative definitions of the [PIM] are found in that specification.

[WS-Calendar] defines how to use the semantics of the enterprise calendar communications within service communications. [WS-Calendar PIM] defines a Platform Independent Model and re-defined [WS-Calendar] as a semantically richer and more variable conformant Platform Specific Model (PSM). Other Platform Specific Models that conform to [WS-Calendar PIM] transitive conform to [WS-Calendar]. In this manner, [WS-Calendar PIM] defines how conformance to [WS-Calendar] is to be achieved on platforms that cannot themselves interact directly with traditional calendar servers.

The Minimal PIM-conformant schema defines a message similar to the simplest [WS-Calendar] messages, with reduced optionality to enhance conformance with [WSI-Basic], and conforming to the [PIM].

2.1 Schedule Semantics from WS-Calendar PIM

Without an understanding of certain terms defined in [WS-Calendar PIM], the reader may have difficulty achieving complete understanding of their use in this standard. The table below provides summary descriptions of certain key terms from that specification. This specification does not redefine these terms; they are listed here solely as a convenience to the reader.

<table>
<thead>
<tr>
<th>WS-Calendar Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artifact</td>
<td>The placeholder in a Component that holds that thing that occurs during an Interval. In this specification, the [PIM] Artifact is named the Payload.</td>
</tr>
<tr>
<td>Availability</td>
<td>Availability provides computable rules for the Intervals during which a particular Gluon will accept a Start. Availability is an anti-fragile pattern informing a service requester when a request will be refused.</td>
</tr>
<tr>
<td>Duration</td>
<td>Duration is the length of time for an event scheduled using iCalendar or any of its derivatives.</td>
</tr>
<tr>
<td>Gluon</td>
<td>A Gluon influences the serialization of Intervals in a Sequence, through 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.</td>
</tr>
<tr>
<td>Interval</td>
<td>The Interval is a single discrete segment, an element of a Sequence, and expressed with a Duration. The Interval is derived from the common calendar Components. An Interval is part of a Sequence.</td>
</tr>
<tr>
<td>Link</td>
<td>A reference to an internal object within the same calendar, or an external object in a remote system. The Link is used by one Component to reference another.</td>
</tr>
<tr>
<td>Partition</td>
<td>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.</td>
</tr>
<tr>
<td>Recurrence</td>
<td>Recurrence computes a set of Starts for a sequence, each of which is treated as if an new Gluon supplied a new Start for the same sequence.</td>
</tr>
<tr>
<td>WS-Calendar Term</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Sequence</strong></td>
<td>A set of Intervals with defined temporal relationships. Sequences may have gaps between Intervals, or even simultaneous activities. A Sequence is relocatable, i.e., it does not have a specific date and time. A Sequence may consist of a single Interval, and can be scheduled by scheduling that single Interval in that Sequence.</td>
</tr>
</tbody>
</table>

Normative descriptions of the terms in the table above are in [WS-Calendar PIM].

In [iCalendar], the primary information structure is a Component. Several RFCs have extended iCalendar by defining new Components using a core extensible set of semantics defined in that specification. Components themselves have Parameters. Availability is one such component defined as an [iCalendar] extension.

Parameters may themselves include Components. The potentially recursive structure of [iCalendar] Components makes that specification immensely expressive. It also makes [iCalendar], and the closely aligned WS-Calendar 1.0, difficult for certain types of machine processing. The [PIM] specification eliminates the general extension mechanisms and recursion of iCalendar by eliminating the generic components and parameters.

In the list below, Interval, Gluon, and Availability reference Components. Duration, Link, Recurrence, and Relationship are Parameters. A Sequence is set of Components, primarily Intervals and Gluons, but is not itself a Type.

### 2.2 Schedules and Inheritance

Nearly every response, every event, every offer, and every interaction can have payloads with values that vary over time, i.e., it a set of intervals can be described using a Sequence of Intervals. Many energy market communications involve information about or a request for power delivered over a single interval of time. Simplicity and parsimony of expression must coexist with complexity and syntactical richness.

Consider a request to reduce power consumption in response to market conditions on a smart grid (Demand Response). The simplest demand response is to reduce power for a set interval.

![Figure 2-1: Basic Power Object from EMIX](image)

At its simplest, though, WS-Calendar expresses repeating intervals of the same duration, one after the other, and something that changes over the course of the schedule.

![Figure 2-2: WS-Calendar Partition, a simple sequence of 5 intervals](image)

The PIM specification defines how to spread an object like the first over the schedule. The information that is true for every interval is expressed once only. The information that changes during each interval, is expressed as part of each interval.*

![Figure 2-3: Applying Basic Power to a Sequence](image)
Many communications communicate requirements for a single interval. When expressing market information about a single interval, the market object (Power) and the single interval collapse to a simple model:

![Figure 2-4: Simplifying back to Power in a Single Interval](image)

[PIM] calls this pattern Inheritance and specifies a number of rules that govern Inheritance. Table 2-2 summarizes those terms defined in WS-Calendar to describe Inheritance that are used in this specification as well. This specification does not redefine these terms; they are listed here solely as a convenience to the reader.

<table>
<thead>
<tr>
<th>Streams Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lineage</td>
<td>The ordered set of Parents that results in a given inheritance or execution context for a Sequence.</td>
</tr>
<tr>
<td>Inherit</td>
<td>A Child Inherits attributes (Inheritance) from its Parent.</td>
</tr>
<tr>
<td>Inheritance</td>
<td>A pattern by which information in Sequence is completed or modified by information from a Gluon. Information specified in one informational object is considered present in another that is itself lacking expression of that information.</td>
</tr>
<tr>
<td>Bequeath</td>
<td>A Parent Bequeaths attributes (Inheritance) to its Children.</td>
</tr>
</tbody>
</table>

Normative descriptions of the terms in the table above are in [PIM].

This specification uses Inheritance as defined in [PIM]. Each Interval in a Sequence contains an information Payload. Incomplete Payloads are each completed through inheriting information from the Gluon. The Gluon provides the information necessary to fully each Interval in the Sequence.

From another perspective, a Gluon acts as a service advertisement point for a potential set of activities in time, and a service request may act as a Gluon to fully bind that schedule that set of activities.

### 2.3 When: Start, End and Duration

Any interval can be fully defined by two out of these three elements: when it begins, how long it lasts, and when it ends. With any two, you can compute the third.

This specification assigns predominance to how long it lasts, the Duration. This approach is commonly used to request human scheduling, i.e., “Find a time when the three of us can meet for an hour.” Activities are then normally scheduled by Start Time, again to reflect human usage: “We will meet for lunch at Noon”.

An application or specification MAY choose to specify the Duration and the End of an event, if this is simpler for its domain. Such a specification MUST make this expectation clear, as allowing a mix of Start and End based requests makes programming and conformance more difficult. For simplicity, in this document, all scheduling is described refining an Interval with a Duration and adding a Start.

A service request MAY specify both. For example, a Sequence may be advertised with no fixed duration, and a service request MAY specify both the Duration and the Start.

The use of the Start and the End without a definition is discouraged because it reduces flexibility while increasing required computation.
3 Low Level Semantic Elements

This section describes the simple types used to construct WS-Calendar MIN messages.

Beyond the conventions of WS-Calendar, the normative specification of Date, Time, and Duration is in [ISO8601], [RFC5545], which was developed prior to XML implements a subset of [ISO8601]. The XML specification also includes only a subset of that specification. The two subsets are not the same. For those elements in this section that are derived from [ISO8601], the standard XML definitions rather than those of [RFC5545] are used.

3.1 When: Start, End and Duration

Any interval can be fully defined by two out of these three elements: when it begins, how long it lasts, and when it ends. With any two, you can compute the third. This specification assigns predominance to how long it lasts, the Duration. This approach is commonly used to request human scheduling, i.e., “Find a time when the three of us can meet for an hour.” Activities are then normally scheduled by Start Time, again to reflect human usage: “We will meet for lunch at Noon.”

An application or specification MAY choose to specify the Duration and the End of an event, if this is simpler for its domain. Such a specification MUST make this expectation clear, as allowing a mix of Start and End based requests makes programming and conformance more difficult. For simplicity, in this document, all scheduling is described refining an Interval with a Duration and adding a Start.

A service request MAY specify both. For example, a Sequence may be advertised with no fixed duration, and a service request MAY specify both the Duration and the Start.

The use of the Start and the End without a definition is discouraged because it reduces flexibility while increasing required computation.

3.2 Specifying Date and Time

To fully qualify the when an activity begins, one must specify both a Date and a Time. A partially qualified starting time might include either a data or a time. Under the principals of inheritance, a Component may convey a time and inherit the start date, or convey a start date and inherit the time.

The DateTimeDate type is used in the following fields throughout the larger components.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtStart</td>
<td>The DateTimeDate that indicates when an interval begins. Should not be in any Interval that has a dtEnd.</td>
</tr>
<tr>
<td>dtEnd</td>
<td>The DateTimeDate that indicates when an interval ends. Should not be in any Interval that has a dtStart.</td>
</tr>
<tr>
<td>rDate</td>
<td>The Recurrence Date. The rDate is added to a set of dtStarts in Recurrence.</td>
</tr>
<tr>
<td>exDate</td>
<td>An exDate is excluded from previously computed set of rDates.</td>
</tr>
</tbody>
</table>
### 3.3 The Duration-based Types

Duration is a complex string as defined in [ISO8601] that specifies the passage of time. This specification uses the subset of the Duration strings in the native XML data type.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>Duration specifies a passage in time. Duration is an essential element for each fully-bound Interval.</td>
</tr>
<tr>
<td>Granularity</td>
<td>Granularity describes the “chunkiness” of time. For example, a Intervals of a Sequence may all have durations that are multiples of 1 hour. Such a Sequence has a Granularity of 1H.</td>
</tr>
<tr>
<td>Precision</td>
<td>Precision defines the accuracy in time required. A service whose outcomes are plus or minus 5 minutes (5m) should not accept a service request with a precision of three seconds (3s).</td>
</tr>
<tr>
<td>startBefore</td>
<td>startBefore indicates whether it is acceptable for an interval to begin before the requested time. The duration specifies the maximum period in advance of the dtStart that is acceptable.</td>
</tr>
<tr>
<td>startAfter</td>
<td>startAfter indicates whether it is acceptable for an interval to begin after the requested time. The duration specifies the maximum period after of the dtStart that is acceptable.</td>
</tr>
<tr>
<td>endBefore</td>
<td>endBefore indicates whether it is acceptable for a [task] to be completed before the requested or computed time. The duration specifies the maximum period in advance of the dtEnd that is acceptable.</td>
</tr>
<tr>
<td>endAfter</td>
<td>endAfter indicates whether it is acceptable for a [task] to be completed after the requested or computed time. The duration specifies the maximum period beyond the dtEnd that is acceptable.</td>
</tr>
<tr>
<td>durationLong</td>
<td>durationLong indicates how much longer than the specified duration of an Interval is acceptable.</td>
</tr>
<tr>
<td>durationShort</td>
<td>durationShort indicates how much less than the specified duration of an Interval is acceptable.</td>
</tr>
</tbody>
</table>

Many of the Duration-based types are used to exchange performance expectations. Performance is outside the scope of this project. A specification that claims conformance and allows the performance types MUST specify how the performance is to be handled. For example, if a service is unable to comply with the performance request, must it reject the request?

### 3.3.1 Tolerance

Tolerance names information about how precise the service must or can be when delivering its service. The Tolerance Type bundles together many of the Duration-derived types into a single object.
If supplied by the service provider, Tolerance is an indication of the service performance advertised. If supplied by a service requestor, Tolerance is the requested performance. Specifications that use Tolerance SHALL specify whether a Service should accept or refuse a request when it is unable to perform at or better than the requested Tolerance. Specifications that incorporate this specification MAY choose to eliminate Tolerance as a message element.

### 3.4 Instance IDs and Links

An Instance UID is a conformed string used to uniquely identify each component and artifact.

A Sequence is defined through Links between Intervals. That Sequence is not fully Bound. A Gluon may point to the Sequence, binding it. A Specification MAY indicate that the InstanceUID of an Interval in the Bound Sequence is identified by Gluon.InstanceID-Interval.InstanceID. IF it is necessary to store and record the Bound Interval, THEN is may be necessary to re-write the Temporal Links to use the new computed UIDs. Alternately, a specification in which Sequences consist of a single Interval, may prefer to never replicate the Bound Sequence.

<table>
<thead>
<tr>
<th>ID Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InstanceUID</td>
<td>The Instance UID identifies each instance of a Component. Instance UIDs may be computed.</td>
</tr>
<tr>
<td>Link</td>
<td>The Link is a foreign InstanceUID, that is a named pointer to another Component.</td>
</tr>
<tr>
<td>Relation</td>
<td>A Relation is a Link that implements the &quot;Child&quot; relation defined in WS-Calendar. The relation is used in a Gluon, or an artifact acting as a Gluon, to point at a child Gluon or Interval.</td>
</tr>
<tr>
<td>Temporal Link</td>
<td>Temporal Links point between Intervals to define a Sequence. A Temporal Link optional includes a Temporal Relation. In the absence of a Temporal Relation, a “FinishToStart” relation is assumed. Temporal Relations are discussed in the section on Sequences.</td>
</tr>
</tbody>
</table>

The InstanceUID MAY be computed based on context, depending on use. Consider the following non-normative example: 

A Sequence is defined through Links between Intervals. That Sequence is not fully Bound. A Gluon may point to the Sequence, binding it. A Specification MAY indicate that the InstanceUID of an Interval in the Bound Sequence is identified by Gluon.InstanceID-Interval.InstanceID. IF it is necessary to store and record the Bound Interval, THEN is may be necessary to re-write the Temporal Links to use the new computed UIDs. Alternately, a specification in which Sequences consist of a single Interval, may prefer to never replicate the Bound Sequence.
A specification claiming conformance MUST specify how these computed InstanceUIDs are computed, or if they are not used at all.

### 3.5 Other Simple Types

Just a few other string-derived types are used in this specification.

<table>
<thead>
<tr>
<th>Simple Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TimeZone</td>
<td>TimeZone is a constrained string indicating the rules for computing time that apply to an exchange of messages. Defining valid Time Zones is outside the scope of this specification.</td>
</tr>
<tr>
<td>Comment</td>
<td>The Comment is free-form text that optionally accompanies many of the component in this specification.</td>
</tr>
</tbody>
</table>
4 Core Components: Intervals, Sequences, and Gluons

The interval is the essential element of this specification, it is that thing which can be bound in time, or scheduled. A Sequence names a set of one or more intervals bound together by temporal relationships. A Gluon appears as a degenerate Interval used to complete information in the Sequences it references.

Table 4-1: High Level Component Types of WS-Calendar MIN

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval</td>
<td>An Interval is the essential element of any service, action, or telemetry. When fully bound it conveys the when and for how long the information in the payload will occur. The expression of even an actionable (“Fully Bound”) Interval MAY be incomplete, as the can be inherited. An Interval that is not Fully Bound describes a service that cannot be provided until the missing information is provided.</td>
</tr>
<tr>
<td>Sequence</td>
<td>A Sequence is a set of temporally related Intervals that can be invoked as a single service. Intervals in a Sequence may occur one after the other or in branches and constraints as defined by the Temporal Relations. A Sequence is Fully Bound when each of its member Intervals is Fully Bound. The simplest Sequence consists of a single Interval.</td>
</tr>
<tr>
<td>Gluon</td>
<td>A Gluon is an Interval that is not itself actionable, while it influences whether Intervals and Sequences are actionable. Gluons are degenerate Intervals that provide missing information to the Intervals in a Sequence while serving as a handle or service entry point to that Sequence. For example, a Gluon MAY be used as a service advertisement to advertise a Sequence. A method that invokes that service acts as a Gluon to the first Gluon, making the referenced Sequence actionable.</td>
</tr>
</tbody>
</table>

A Gluon also supports elements not seen in the Interval, i.e., Availability and Recurrence. For clarity, these are ignored in this section, and discussed in Section 4.

4.1 The Interval

The Interval is the essential element of WS. If something is being done, it is what is being done and for how long. If something is being measured, it is the period of measurement.

The Interval is pictured in Figure 4-1: The Interval:
Note that there is considerable optionality, even for necessary information. Under the model described in the [PIM], information is potentially bequeathed from a lineage of Gluons to an Interval. dtStart, duration, timeZone, tolerance, and even payload MAY be inherited.

The Temporal Link cannot be inherited. Temporal Links are used to defined relations between Intervals in a Sequence.

### 4.2 Temporal Links and Sequences

Temporal Links convey the relationships between Intervals in a Sequence.

Each Interval can be considered as a distinct activity for a period of time. A Sequence is a set of such activities. These activities may follow one after another. There may be mandatory gaps, as in paint drying for at least six hours before the next step. It may be a requirement that two Intervals finish at the same time.

If a Sequence describes a ramp-time of activities prior to the Inherited dtStart, then the ramp activities must complete prior to the start time. Similarly, a system MAY need to ramp down at the end of a requested Duration of activity.

There is a special case of Sequence in which all Intervals proceed linearly without pause, and all Intervals share a common Duration. A Sequence of this Type is referred to as a Partition.

#### 4.2.1 Temporal Links

Temporal Links are so named because they convey how Intervals are related in Time.
As defined in the PIM, there are four types of Temporal Relation. Temporal Relations combine with the Duration to describe the sequence.

Table 4-2: Temporal Relations defines the Temporal Relations. For this table, the Interval that contains the Temporal Link is considered the Originating Interval. The Interval whose InstanceUID is referenced in the Temporal Link is considered the Referenced Interval. Note that any Interval may be Originating in one context, and Referenced in another.

<table>
<thead>
<tr>
<th>Temporal Relation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>finishToStart</td>
<td>The Referenced Interval will start after the Originating Interval ends, separated by a defined Duration. If the Duration is absent, then the Referenced Interval will begin immediately on completion of the Originating Interval.</td>
</tr>
<tr>
<td>finishToFinish</td>
<td>The Referenced Interval finish when the Originating Interval ends, separated by a defined Duration. If the Duration is absent, then the Referenced Interval will finish at the same time as does the Originating Interval. A negative Duration specifies that the Referenced Interval finishes before the Originating Interval does.</td>
</tr>
<tr>
<td>startToFinish</td>
<td>The Referenced Interval will finish a defined Duration start after the Originating Interval starts, separated by a defined Duration. If the Duration is absent, then the Referenced Interval finish just as the Originating Interval starts. A negative Duration specifies that the Referenced Interval will end before the Originating Interval starts.</td>
</tr>
<tr>
<td>startToStart</td>
<td>The Referenced Interval will start after the Originating Interval starts, separated by a defined Duration. If the Duration is absent, then the Referenced Interval will start at the same time that the Originating Interval starts. A negative Duration specifies that the Referenced Interval starts before the Originating Interval.</td>
</tr>
</tbody>
</table>

If a specification that claims conformance this specification permits a missing temporalRelation, then that specification MUST state which Temporal Relation is implied. A conforming specification MAY disallow a missing temporalRelation.

4.2.2 Sequences

Sequences are collections of Intervals connected by Temporal Relationships. There is no Sequence structure per-se. A Sequence is referenced by referencing the InstanceUID of one Interval in the Sequence. That Interval is referred to as the Designated Interval. The Designated Interval has special rules for Inheritance. For example, when a Gluon Bequeaths a dtStart to a Sequence, is it the Designated Interval that starts at that time.

Inheritance within a Sequence is specified in the [PIM].
4.3 The Gluon

The Gluon links a Sequence to a service interaction. The Gluon can be considered a degenerate Interval that cannot itself be executed. It does, however, provide missing information to fully bind each Interval in the Sequence.

Another perspective describes the Gluon as the service entry point for an activity defined by a Sequence. Sequence execution is launched by providing a dtStart through a Gluon. A service request acting as a Gluon bequeaths missing information that is inherited by the entry point Gluon to bind the Sequence.

The Gluon Type is shown in Figure 4-3: The Gluon.
Notice that the Gluon is nearly identical to the Interval. A Requested Start replaces the dtStart. Requested Start is of type Recurrence. Recurrence describes how to compute a collection of dtStarts. Recurrence is discussed in below in Section 5.

The significant changes are as follows:

1) The Gluon has no Temporal Links. It cannot be part of a Sequence, so it maintains not Temporal Relations with other Components.

2) A Gluon must have at least one Relation, and it can have many. The Relation connects a Gluon to a Sequence, to establish Inheritance. A Relation MAY connect a Gluon to another Gluon, establishing a Lineage that eventually binds a Sequence.

3) A Gluon may convey multiple dtStarts. This collection is computed in RequestedStart, which is of type Recurrence. A recurrence is a structure to convey or compute a collection of starting dates and times. These act as if there were multiple Gluons, each conveying a single dtStart.

4) vAvailability. vAvailability is an outward looking element that conveys information about potential schedules for the underlying Sequence.
5 Enhancing Service Advertising and Request: Recurrence and Availability

Up until this section, dates and times were specific. This section introduces Recurrence and types that enable patterns of dates and schedules to be computed. These computable schedules expand the capability of a Gluon.

Table 5-1: Computable Schedules: Availability and Gluon

<table>
<thead>
<tr>
<th>Service Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recurrence</td>
<td>Conceptually, the last thing specified to Fully Bind a Sequence is When, i.e., the start date and time. Recurrence is an information structure that enables the computation of a collection of starting dates and times.</td>
</tr>
<tr>
<td>Recurrence Rule</td>
<td>Recurrence rules define periodic repletion of times and dates. The language of Recurrence is well known as defined in [RFC5545] section 3.3.10.</td>
</tr>
<tr>
<td>VAvailability</td>
<td>VAvailability is an optional component of a Gluon that is not inherited, but rather specifies when the underlying Sequence may be invoked</td>
</tr>
<tr>
<td>Availability</td>
<td>Availability is a component of VAvailability, and presents as a single instance of a Recurrence Rule applied to an Interval with Duration.</td>
</tr>
<tr>
<td>Availability Interval</td>
<td>An Interval (Duration and Start) used as a seed for recurrence within an Availability component</td>
</tr>
<tr>
<td>Time Range</td>
<td>An Interval (Duration and Start) used to bound VAvailability.</td>
</tr>
<tr>
<td>Busy Type</td>
<td>Used to indicate whether the schedule in a vAvailability is Free, Busy (unavailable), or tentatively busy.</td>
</tr>
</tbody>
</table>

There may be good reasons for a specification that claims conformance with this specification to forbid Recurrence. Requiring each service invocation to require its own message that acts as a Gluon MAY simplify the system. A conforming specification MUST state of the use of these components is forbidden.

5.1 Recurrence Rules

Recurrence Rules are used in both Recurrence and in Availability to compute patterns of schedules and dates. Each Rule consists of a Rule Part, which names a type of Rule, and Rule Values, constrained lists which operate within the Rule Part.

Representative recurRuleParts indicate that a Rule is hourly, or at a fixed frequency, or on certain days of the month. Rule Values are constrained depending on the RulePart, to indicate days of the week, every three hours, and so on.

Recurrence Rules are normatively described in [RFC5545] section 3.3.10. Many web-sites and open source libraries discuss these rules; no efforts will be made in this specification to re-state these rules.
5.2 Recurrence

Recurrence is a mechanism to compute a collection of starting date-times. At its simplest, it is a `dtStart`, just as in the Interval. Recurrence Rules then describe how to compute additional starting dates and times using the `dtStart` as a seed. `rDates` add additional starting dates to the collection. `xDates` then block out dates, that is, remove specific date-times from the collection.

The requested `Start` in the Gluon is of type Recurrence.

5.3 Availability and VAvailability

VAvailability is the sum of one or more patterns (Availability) that together express when a Service can be invoked.

As a non-normative illustration, the well-known pattern of “During Business Hours” can be described as the hours from 9:00 AM to 5:00 PM repeated weekly on Monday, Tuesday, Wednesday, Thursday, and Friday. Alternately, it might be the sum of two patterns, 8:00 AM until noon, Monday, Tuesday, Wednesday, Thursday, and Friday and 1:00 until 5:00 on Monday, Tuesday, Wednesday, Thursday, and Friday. An additional pattern of 9:00 AM until 1:00 PM might be added each Saturday. The smaller patterns are named “Availability” and the top-level summary is named VAvailability.

Note that a Gluon may have an array of VAvailability components. These components MAY be both Available and Unavailable in the same set. There are specific rules for overlaying VAvailability components which the practitioner should be aware of. These rules are described in [vAvailability].

5.3.1 Availability

The Availability type uses the same computational rules as Recurrence and applies then to a seed Interval, that is a Duration and `dtStart`. The `DateTime` and the `Duration` are known as the Availability Interval.
Availability applies the Recurrence Rules (RRules) defined in [RFC5545] to the availability interval.

5.3.2 VAvailability

VAvailability represents the sum of a collection of Availability types applied within the bounds of a defined Time Range.
Note that Granularity, when applied to vAvailability has a special meaning. A three hour interval advertised with a granularity of 15 minutes may only be invoked at the 15 minute interval. For example, the interval may be 9:00 until Noon, but the only dtStarts that may be requested are at 9:00, 9:15, 8:30, 9:25 and so on,
6 Conformance

6.1 Conformance with the Semantic Models of WS-Calendar-PIM

This section specifies conformance with the semantic models of [WS-Calendar-PIM]. This specification requires that specifications claiming conformance also conform to the specific conformance requirements of [WS-Calendar-PIM] are described in section 5.3 of that specification, "Conformance Rules for WS-Calendar PIM".
Appendix A. Acknowledgments

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

Participants:
- David Thewlis, CalConnect
- William Cox, Individual
- Gershon Janssen, Individual
- Benoit Lepeuple, LonMark International
- Michael Douglass, Rensselaer Polytechnic Institute
- Toby Considine, University of North Carolina at Chapel Hill
- Chris Bogen, US Department of Defense (DoD)
## Appendix B. Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>WD01</td>
<td>21-Jul-2015</td>
<td>Toby Considine</td>
<td>Initial Draft</td>
</tr>
<tr>
<td>WD02</td>
<td>22-Jul-2015</td>
<td>Toby Considine</td>
<td>Added section on Recurrence and Availability. Added recurrence to Gluons.</td>
</tr>
</tbody>
</table>