WS-Calendar Version 1.0

Committee Specification Draft 02

28 January 2011

Specification URIs:
This Version:
http://docs.oasis-open.org/ws-calendar/ws-calendar-spec/v1.0/csd02/ws-calendar-spec-v1.0-csd02.pdf (Authoritative)
http://docs.oasis-open.org/ws-calendar/ws-calendar-spec/v1.0/csd02/ws-calendar-spec-v1.0-csd02.html
http://docs.oasis-open.org/ws-calendar/ws-calendar-spec/v1.0/csd02/ws-calendar-spec-v1.0-csd02.doc

Previous Version:
http://docs.oasis-open.org/ws-calendar/ws-calendar/v1.0/CD01/ws-calendar-1.0-spec-cd-01.pdf (Authoritative)
http://docs.oasis-open.org/ws-calendar/ws-calendar/v1.0/CD01/ws-calendar-1.0-spec-cd-01.html
http://docs.oasis-open.org/ws-calendar/ws-calendar/v1.0/CD01/ws-calendar-1.0-spec-cd-01.doc

Latest Version:
http://docs.oasis-open.org/ws-calendar/ws-calendar/v1.0/ws-calendar-1.0-spec.pdf (Authoritative)
http://docs.oasis-open.org/ws-calendar/ws-calendar/v1.0/ws-calendar-1.0-spec.html
http://docs.oasis-open.org/ws-calendar/ws-calendar/v1.0/ws-calendar-1.0-spec.doc

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

Chair:
Toby Considine

Editor(s):
Toby Considine
Mike Douglass

Related work:
XML Schemas for WS-Calendar Version 1.0
This specification is related to:

- IETF RFC5545, ICalendar
- IETF RFC5546, ICalendar Transport
- IETF RFC2447, ICalendar Message Based Interoperability
- IETF XCAL specification in progress
- IETF / CalConnect Calendar Resource Schema specification in progress

Declared XML Namespace(s):
http://docs.oasis-open.org/ns/ws-calendar
http://docs.oasis-open.org/ns/ws-calendar/timestamp

Abstract:

WS-Calendar describes a limited set of message components and interactions providing a common basis for specifying schedules and intervals to coordinate activities between services. The specification includes service definitions consistent with the OASIS SOA Reference Model and XML vocabularies for the interoperable and standard exchange of:

- Schedules, including sequences of schedules
- Intervals, including sequences of Intervals

These message components describe schedules and Intervals future, present, or past (historical). The definition of the services performed to meet a schedule or Interval depends on the market context in which that service exists. It is not in scope for this TC to define those markets or services.

Status:

This version of the specification has known deficiencies. The examples have not been re-written since the schema changes. The Graphics, too, are out of date. The entire sections on SOAP-based calendar services remain to be written. The TC feels that the prose and the schemas are essentially complete.

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 http://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 (http://www.oasis-open.org/committees/ws-calendar/ipr.php).

Citation Format:

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

ws-calendar-spec

WS-Calendar Version 1.0. 28 January 2011. OASIS Committee Specification Draft.

http://docs.oasis-open.org/ws-calendar/ws-calendar-spec/v1.0/csd02/ws-calendar-spec-v1.0-csd02.pdf

Notices

Table of Contents

Tables

Index of Tables

Index of Examples

1 Introduction

This version of the specification has known deficiencies. The examples have not been re-written since the schema changes. The Graphics, too, are out of date. The entire sections on SOAP-based calendar services remain to be written. The TC feels that the prose and the schemas are essentially complete.

One of the most fundamental components of negotiating services is agreeing when something should occur, and in auditing when they did occur. Short running services traditionally have been handled as if they were instantaneous, and have handled scheduling through just-in-time requests. Longer running processes, including physical processes, may require significant lead times. When multiple long-running services participate in the same business process, it may be more important to negotiate a common completion time than a common start time. Pre-existing approaches that rely on direct control of such services by a central system increases integration costs and reduce interoperability as they require the controlling agent to know and manage multiple lead times.

Not all services are requested one time as needed. Processes may have multiple and periodic occurrences. An agent may need to request identical processes on multiple schedules. An agent may request services to coincide with or to avoid human interactions. Service performance be required on the first Tuesday of every month, or in weeks in which there is no payroll, to coordinate with existing business processes. Service performance requirements may vary by local time zone. A common schedule communication must support diverse requirements.

Physical processes are already being coordinated by web services. Building systems and industrial processes are operated using **[oBIX]**, BACnet/WS, LON-WS, OPC XML, and a number of proprietary specifications including TAC-WS, Johnson Controls EnNet™, and MODBUS.NET. In particular, if building systems coordinate with the schedules of the building's occupants, they can reduce energy use while improving performance.

An increasing number of specifications envision synchronization of processes through mechanisms including broadcast scheduling. Efforts to build an intelligent power grid (or smart grid) rely on coordinating processes in homes, offices, and industry with projected and actual power availability; mechanisms proposed include communicating different prices at different times. Several active OASIS Technical Committees require a common means to specify schedule and interval: Energy Interoperation **[EITC]** and Energy Market Information Exchange **[EMIX]**. Emergency management coordinators wish to inform geographic regions of future events, such as a projected tornado touchdown, using **[EDXL]**. The open Building Information Exchange specification **[oBIX]** lacks a common schedule communications for interaction with enterprise activities. These and other efforts would benefit from a common cross-domain, cross specification standard for communicating schedule and interval.

For human interactions and human scheduling, the well-known iCalendar format is used to address these problems. Prior to WS-Calendar, there has been no comparable standard for web services. As an increasing number of physical processes become managed by web services, the lack of a similar standard for scheduling and coordination of services becomes critical.

The intent of the WS-Calendar technical committee was to adapt the existing specifications for calendaring and apply them to develop a standard for how schedule and event information is passed between and within services. The standard adopts the semantics and vocabulary of iCalendar for application to the completion of web service contracts. WS Calendar builds on work done and ongoing in The Calendaring and Scheduling Consortium (CalConnect), which works to increase interoperation between calendaring systems.

Everything with the exception of all examples, all appendices, and the introduction is normative unless otherwise specifically noted.

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

Calendar Resource Schema	C. Joy, C. Daboo, M Douglas, *Schema for representing resources for calendaring and scheduling services*, http://tools.ietf.org/html/draft-cal-resource-schema-00, (Internet-Draft), April 2010.
CalDAV	C. Daboo, B. Desruisseaux, L. Dusseault, Calendaring Extensions to WebDAV (CalDAV), http://www.ietf.org/RFC/RFC4791.txt, IETF RFC4791, March 1997.
FreeBusy Read URL	E York. Freebusy read URL, http://www.calconnect.org/pubdocs/CD0903%20Freebusy%20Read%20URL%20V1.0.pdf
RFC2119	S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/RFC/RFC2119.txt, IETF RFC2119, March 1997.
RFC2447	F. Dawson, S. Mansour, S. Silverberg, iCalendar Message-Based Interoperability Protocol (iMIP), http://www.ietf.org/RFC/RFC2247.txt, IETF RFC2447, December 2009.
RFC2616	R Fielding, et al. et al, Hypertext Transfer Protocol -- HTTP/1.1 http://tools.ietf.org/html/RFC2616, IETF RFC2616, November 1998
RFC3339	G Klyne, C Newman, Date and Time on the Internet: Timestamps http://tools.ietf.org/html/rfc3339
RFC4791	Daboo, et al. Calendaring Extensions to WebDAV (CalDAV). http://www.ietf.org/rfc/rfc4791.txt. IETF RFC 2119, March 2007
RFC4918	L. Dusseault, HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV). http://tools.ietf.org/html/rfc4918
RFC5545	B. Desruisseaux Internet Calendaring and Scheduling Core Object Specification (iCalendar), http://www.ietf.org/rfc/rfc5545.txt, IETF RFC5545, September 2009.
RFC5546	C. Daboo iCalendar Transport-Independent Interoperability Protocol (iTIP), http://www.ietf.org/rfc/rfc5546.txt, IETF RFC5546, December 2009.
RFC5598	M. Nottingham, Web Linking, http://www.ietf.org/rfc/rfc5598.txt, IETF RFC5598, October 2010.
SOA-RM	OASIS Standard, Reference Model for Service Oriented Architecture 1.0, October 2006. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf
SOAP	M Gudgin, M Hadley, N Mendelsohn, J Moreau, H Nielsen, A Karmarkar, SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation http://www.w3.org/TR/soap12-part1/, April 2007
Vavailability	C. Daboo, B. Desruisseaux, Calendar Availability, http://tools.ietf.org/html/draft-daboo-calendar-availability-01, IETF draft, November 2008
WSDL	E Christensen, F Curbera, G Meredith, S Weerawarana, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, W3C Note March 2001
draft xCal	C. Daboo, M Douglas, S Lees xCal: The XML format for iCalendar, http://tools.ietf.org/html/draft-daboo-et-al-icalendar-in-xml-07, Internet-Draft, October 2010.
XPATH	A Berglund, S Boag, D Chamberlin, MF Fernández, M Kay, J Robie, J Siméon XML Path Language (XPath) 2.0, http://www.w3.org/TR/xpath20/ January 2007.
XLINK	S DeRose, E Maler, D Orchard, N Walsh XML Linking Language (XLink) Version 1.1., http://www.w3.org/TR/xlink11/ May 2010.

XPOINTER S DeRose, E Maler, R Daniel Jr. XPointer xpointer Scheme, http://www.w3.org/TR/xptr-xpointer/ December 2002.

XML SCHEMA PV Biron, A Malhotra, XML Schema Part 2: Datatypes Second Edition, http://www.w3.org/TR/xmlschema-2/ October 2004.

XRD OASIS XRI Committee Draft 01, Extensible Resource Descriptor (XRD) Version 1.0, http://docs.oasis-open.org/xri/xrd/v1.0/cd01/xrd-1.0-cd01.pdf October 2009.

1.3 Non-Normative References

NIST Framework and Roadmap for Smart Grid Interoperability Standards, Office of the National Coordinator for Smart Grid Interoperability, Release 1.0, NIST Special Publication 1108, http://www.nist.gov/public_affairs/releases/upload/smartgrid_interoperability_final.pdf.

NAESB Smart Grid Requirements (awaiting publication) (draft contributed) http://lists.oasis-open.org/archives/ws-calendar-comment/201005/doc00000.doc, May 2010

REST T Fielding, Architectural Styles and the Design of Network-based Software Architectures, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

TZDB P Eggert, A.D. Olson, "Sources for Time Zone and Daylight Saving Time Data", http://www.twinsun.com/tz/tz-link.htm

Time Zone Recommendations, CalConnect, CalConnect EDST (Extended Daylight Savings Time) Reflections and Recommendations, Version: 1.1, http://www.calconnect.org/pubdocs/CD0707%20CalConnect%20EDST%20Reflections%20and%20Recommendations%20V1.1.pdf October 2010

Time Zone Service, M Douglas, C Daboo, Timezone Service Protocol, Draft RFC,IETF, http://datatracker.ietf.org/doc/draft-douglass-timezone-service/

1.4 1.4 Namespace

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

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

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

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

Table 1-1: Namespaces Used in this Specification

Prefix	Namespace
Xs	http://www.w3.org/2001/XMLSchema
Xcal	http://docs.oasis-open.org/ns/ws-calendar/i

The normative schemas for EMIX can be found linked from the namespace document that is located at the namespace URI specified above.

Table 1-2: Schemas and Extensions Used in this Specification

Schema	Description
iCalendar.xsd	Base Schema expressing core iCalendar information

iCalendar-params.xsd	Parameters used in iCalendar objects
iCalendar-props.xsd	Properties of iCalendar objects
iCalendar-valtypes.xsd	Values used by iCalendar
iCalendar-link-extension.xsd	Link extensions based on **[web linking]** to define relationships between components.
iCalendar-wscal-extensions.xsd	Extensions to iCalendar to support service functionality
iCalendar-bw-extensions.xsd	Extensions to support integration with Bedeworks server.
iCalendar-ms-extensions.xsd	Extensions to support integration with MS Exchange Server

Reviewers can find the schemas at http://docs.oasis-open.org/ws-calendar/ws-calendar-spec/v1.0/csd02/xsd/.

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 lower camelCase convention, with all names starting with a lower case letter. For example,

```
<element name="componentType" type="energyinterop:ComponentType"/>
```

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

```
<complexType name="type-componentService">
```

For the names of intents, the names follow the lower camelCase 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.

An example of an intent that is an acronym is the "SOAP" intent.

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.

1.7 Architectural References

WS-Calendar assumes incorporation into services. Accordingly it assumes a certain amount of definitions 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]***.

1.8 Terminology

Certain terms appear throughout this document, some with extensive definitions. The table provides summary definitions for the convenience of the reader and reviewer. When full definitions of the terms below appear in later sections of this document, with the exception of in the appendices, then that later definition is normative.

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 1-3, below.

Table 1-3: Terminology – Foundational Elements

Time Segment	Definition
Duration	Well-known element from iCalendar and **[XCAL]**, Duration is the length of an event scheduled using iCalendar or any of its derivatives. The **[XCAL]** duration is a data type using the string representation defined in the iCalendar duration. The Duration is the sole descriptive element of the VTODO object that is mandatory in the Interval.
Interval	The Interval is a single duration derived from the common calendar components as defined in iCalendar (**[RFC5545]**) and refined in **[XCAL]**. In Calendar systems, it is processed as a vtodo, but the constraints and conformance are different.
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.
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 which varies over time. Examples include energy prices over time and energy usage over time.
Gluon	A gluon is 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 thing that occurs during an Interval. WS-Calendar extends the **[XCAL]** attach object to contain this 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.

WS-Calendar works with groups of Intervals that have relationships between them. These relations constrain the final instantiation of a schedule-based service. Relations can control the ordering of Intervals in a Sequence. They can describe when a service can be, or is prevented from, being invoked. They establish the parameters for how information will be shared between elements using Inheritance. The terminology for these relationships is defined in Table 1-4.

Table 1-4: Terminology – Relations, Limits, and Constraints

Term	Definition
Link	The Link is used by one WS-Calendar 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 are incorporated into links and define how Intervals are connected for Binding. ICalendar defines several relationships, but WS-Calendar uses only the Parent relationship, and that only to bind Gluons to each other and to Intervals.

Term	Definition
Temporal Relationship	Temporal Relationships are incorporated into Links and 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 is often used to overlay or be overlaid by Busy. a Availability can be Inherited
Busy	Busy expresses the range of times in which an Interval or Sequence cannot be Scheduled. Busy is often used to overlay or be overlaid by Availability. Busy can be Inherited
Child, Children	The PARENT relationship type (rel_type) 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 PARENT links and may have zero to many Parent objects.
Parent [Gluon]	A Gluon that (in a Sequence) that includes a PARENT relationship toobject. type (rel_type) defines a logical link (via URI or UID) from parent object to a child object. A Parent object is the contains one or more PARENT links
Lineage	The ordered set of Parents that results in a given inheritance or execution context for a Sequence.

WS-Calendar describes how to modify and complete the specification of Sequences. WS-Calendar calls this process Inheritance and specifies a number of rules that govern inheritance. Table 1-5 defines the terms used to describe inheritance.

Table 1-5: Terminology – Sequence State and Completeness

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 1-6 defines the terms used when discussing the processing or processability of Intervals and Sequences.

Table 1-6: Terminology – Describing Intervals

Term	Definition
Anchored	An Interval is Anchored [in time] if it is Bound to a full date and time. A Sequence or Partition is Anchored if it contains an Anchored Interval, and when Fully Bound, the specific date, time, and duration of all Intervals can be determined unambiguously. Specific performance of a Service Contract always occurs in an Anchored Sequence.,
Partially Anchored	An Interval is Partially Anchored if EITHER its Date OR its Time is Bound. A Sequence or Partition is Partially Anchored if its Designated Interval is Partially Anchored.
Unanchored	An Interval is Unanchored if NEITHER its Begin Date nor its Begin Time are known.
Bound	As in mathematical logic where a metasyntactic variable is called "bound", an Interval, Sequence, or Partition is said to be Bound when the values necessary to execute it (as a service) are completely filled in.
Partially Bound	A Partially Bound Interval is one that is still not Bound after receiving its Inheritance. A Sequences or Partitions is Partially Bound if it contains at least one Interval that is Partially Bound.
Unbound	An Unbound Interval or Sequence is not itself complete, but must still receive inheritance to be fully specified. A Sequences or Partitions is Unbound if it contains at least one Interval that is Unbound.
Fully Bound	A synonym for Bound
Constrained	An Interval is Constrained if it is not Anchored and it is bound to one or more Availability or Free/Busy elements
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 its neither its begin date and time nor its end date and time have been set. A Sequence or Partition is Unscheduled if none of its Intervals, after when Fully Bound, is Scheduled.
Designated Interval	In a Sequence the Designated Interval is either (a) (if there are no Gluons related to the Sequence) one of the Earliest Interval(s), or (b) (if there is at least one Gluon related to the Sequence) the single Interval referenced by a Gluon as Parent.
Predecessor Interval	A Predecessor Interval includes a Temporal Relation which references a Successor Interval.
Successor Interval	A Successor Interval is one referred to by a Temporal Relationship in a Predecessor Interval.
Antecedent Interval(s)	An Interval or set of Intervals that precede a given Interval within the same Sequence

Term	Definition
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 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 or Unbound.
Comparable Sequences	Two Sequences are Comparable if and only if there exists a Composed version of each that defines the same schedule.

2 Overview of WS-Calendar

A calendar communication without a real world effect[1] is of little interest. That real world effect is the result of a service execution context within a policy context. Practitioners can use WS-Calendar to add communication of schedule and Interval to the execution context of a service. Use of WS-Calendar will align the performance expectations between execution contexts in different domains. The Technical Committee intends for other specifications and standards to incorporate WS-Calendar, bringing a common scheduling context to diverse interactions in different domains

2.1 Approach taken by the WS-Calendar Technical Committee

The Technical Committee (TC) based its work upon the iCalendar specification as updated in 2009 (IETF **[RFC5545]** and its the XML serialization **[XCAL]**, currently (2010-07) on a standards track in the IETF. Members of the Calendaring and Scheduling Consortium (CalConnect.org) developed both updates to IETF specifications and provided advice to this TC. This work provides the vocabulary for use in this specification.

This committee developed the normative schema (XSD) for iCalendar. This schema, including the schema extensions necessary for the services defined herein, is part of the WS-Calendar specification.

The committee solicited requirements from a range of interests, notably the NIST Smart Grid Roadmap and the requirements of the Smart Grid Interoperability Panel (SGIP) as developed by the North American Energy Standards Board (NAESB). Others submitting requirements included members of the oBIX technical committee and representative of the FIX Protocol Association. These requirements are reflected in the semantic elements described in Chapters 3 and 4.

In a parallel effort, the CalConnect TC-XML committee developed a number of schedule and calendar-related services. CalConnect drew on its experience in interoperability between enterprise calendaring systems as well as interactions with web-based calendars and personal digital assistants (PDAs). These services were developed as RESTfull (using **[REST]**) services by CalConnect and contributed to the WS-Calendar TC. CalConnect also developed and contributed **[SOAP]** and **[WSDL]** definitions to this TC.

2.2 Scheduling Service Performance

Time semantics are critical to WS-Calendar. Services requested differently can have different effects on performance even though they appear to request the same time interval. This is inherent in the concept of a service-oriented architecture.

As defined in the OASIS Reference Model for Service Oriented Architecture 1.0[2], service requests access the capability of a remote system.

[1] This paragraph includes a number of terms of art used in service oriented architecture (SOA). In all cases, the terms are as defined in the *Reference Model for Service Oriented Architecture,* found in the normative references.

[2] See normative references in section 1.2

The purpose of using a capability is to realize one or more real world effects. At its core, an interaction is “an act” as opposed to “an object” and the result of an interaction is an effect (or a set/series of effects). This effect may be the return of information or the change in the state of entities (known or unknown) that are involved in the interaction.

We are careful to distinguish between public actions and private actions; private actions are inherently unknowable by other parties. On the other hand, public actions result in changes to the state that is shared between at least those involved in the current execution context and possibly shared by others. Real world effects are, then, couched in terms of changes to this shared state

A request for remote service performance is a request for specific real world effects. For WS-Calendar, these effects are expected to occur during a given period. Consider two service providers that offer the same service. One must start planning an hour or more in advance. The second may be able to achieve the service in five minutes. The service start time is the time when that service becomes fully available; that is the time specified in service interactions. Because this service start time and service period are all that matters, the same service can be offered by different providers using quite different technologies.

The complement of this is the scheduled end time. The party offering the service may need to ramp down long running processes. Using for example energy demand response, if a system contracts to end energy use by 3:00, it assumes the onus of turning everything off before 3:00.

Duration is how long a behavior is continued. If a service contracts to provide shed load for an hour, it is not necessary for it to stop shedding load 65 minutes later (which may be the end of the work day). It must, however, shed the agreed upon load during all of the 60 minutes.

In this way, the service scheduled to shed load from 4:00 ending at 5:00 may be quite different than the one scheduled to shed load for an hour beginning at 4:00.

2.2.1 Which Time? UTC vs. Local Time

When 2 or more parties attempt to agree on a time, e.g., for a meeting, or when to provide a service, they agree to start at a particular instant of time UTC. They agree on that instant in time by converting from local time, e.g., they want a meeting to start at 13:00 Eastern, 18:00 UK. Our lives and the use of services are bound by local time not by UTC. To humans local time is the invariant and UTC is mapped on to it. If a government modifies the rules we adjust the mappings and we shift the UTC time. We still want to meet at 13:00 local or have the heating start at 07:00.

As long as the rules never change this causes no confusion—but they do. Recent experience has included considerable efforts when the rules for the start of Daylight Savings Time (DST) have changed. If all information is in UTC, and no record of the events basis in the local time and time zone remains, there is no way to re-compute existing contracts. We don't know if that UTC was calculated based on an old or new rule.

A triplet of Local time + timezoneid + (UTC or offset) always allows you to determine if the time is valid. If a recalculation of UTC for that local time + tzid results in a different value from that stored then presumably the DST rules have changed since the data was stored. If you can detect that the scheduled time is no longer valid you can take corrective action.

For simplicity, all examples and discussion in this document are based on Greenwich Mean Time also known as Coordinated Universal Time (UTC). The Technical Committee makes no representation as whether UTC or local time are more appropriate for a given interaction. Because WS-Calendar is based on **[iCalendar]**, business practices built upon WS-Calendar can support either.

Practitioners should consult **[Time Service Recommendations]** and **[Time Zone Service]** in the non-normative references.

2.3 Overview of This Document

The specification consists of a standard schema and semantics for schedule and interval information. Often the most important service schedule communications involve series of related services over time, which WS-Calendar defines as a Sequence. These semantic elements are defined and discussed in Section 3.

Section 4 introduces notions of performance, i.e. what does it mean to be “on time”. This section also describes the different ways to associate a service request with each Interval in a Sequence.

Managing information exchanges about a Sequence of events can easily become cumbersome, or prone to error. WS-Calendar defines the Calendar Gluon, a mechanism for making assertions about all or most of the Intervals in a Sequence. Intervals can inherit from a Calendar Gluon, or they can override locally assertions inherited from the Calendar Gluon. Section 5 discusses inheritance and parsimony of communication and introduces contract scheduling.

In Sections 8-16, this document describes **[REST]**-based, (RESTfull) web services for interacting with remote calendars. These interactions are derived from the well-known interactions defined in **[CalDAV]**, although they do not specify any interaction with **[CalDAV]** servers. This specification defines services for calendar inquiries, event scheduling, event updating, and event cancelation.

In Sections n-n, this document describes **[SOAP]**-based interactions for Calendar services. As with REST, the specification defines services for calendar inquiries, event scheduling, event updating, and event cancelation using the iCalendar schema.

With incompatible communications defined (REST, SOAP), the specification is not prescriptive of the communications used. The practitioner must decide whether to use either of these communication protocols, or whether WS-Calendar artifacts are better used when embedded within other messages. These decisions, along with decisions about the specific security needed by the communication must be based upon the specific application and message content.

3 Intervals, Temporal Relations, and Sequences

WS-Calendar Elements are semantic elements derived from the [XCAL] specification. These elements are smaller than a full schedule interaction, and describe the intervals, durations, and time-related events that are relevant to service interactions. The elements are used to build a precise vocabulary of time, duration, Sequence, and schedule.

WS-Calendar elements adapt the iCalendar objects to make interaction requirements explicit. For example, in human schedule interactions, different organizations have their own expectations. Meetings may start on the hour or within 5 minutes of the hour. As agents scheduled in those organizations, people learn the expected precision. In WS-Calendar, that precision must be explicit to prevent interoperation problems. WS-Calendar defines a performance element to elaborate the simple specification of **[XCAL]** to make explicit the performance expectations within a scheduled event.

WS-Calendar defines common semantics for recording and exchanging event information.

3.1 Core Semantics derived from [XCAL]

The iCalendar data format **[RFC5545]** is a widely deployed interchange format for calendaring and scheduling data. The **[XCAL]** specification (in process) standardizes the XML representation of iCalendar information. WS-Calendar relies on **[XCAL]** standards and data representation to develop its semantic components.

3.1.1 Time

Time is an ISO 8601 compliant time string with the optional accompaniment of a duration interval to define times of less than 1 second. Examples of date and time representations the from the ISO 8601 standard include:

```
Year:
  YYYY (eg 1997)
Year and month:
  YYYY-MM (eg 1997-07)
Complete date:
  YYYY-MM-DD (eg 1997-07-16)
Complete date plus hours and minutes:
  YYYY-MM-DDThh:mmTZD (eg 1997-07-16T19:20+01:00)
Complete date plus hours, minutes and seconds:
  YYYY-MM-DDThh:mm:ssTZD (eg 1997-07-16T19:20:30+01:00)
Complete date plus hours, minutes, seconds and a decimal fraction of a second
  YYYY-MM-DDThh:mm:ss.sTZD (eg 1997-07-16T19:20:30.45+01:00)
```

Normative information on **[ISO 8601]** is found in section 1.2.

3.1.2 The iCalendar Components (VComponents)

iCalendar and **[XCAL]** have a number of long defined component objects that comprise the payload inside of an iCalendar message. These include the VTODO, the VALARM, the VEVENT. (The “v” that begins each element name is there for historic purposes.) The definitions and use of each of the vObjects can be found in **[RFC5545]**.

The vObjects share the same parameters and properties. The distinctions between these informational objects is in which are permitted, and which are required. Because of its flexibility, the VTODO object is the basis for WS-Calendar objects for service performance. Because WS-Calendar services support all traditional iCalendar-based interactions (CalDAV, et al.), all VComponents SHALL be supported.

The Interval and Gluon are new vObjects, and each is derived from vtodo.

3.2 Intervals

Time Segments, i.e., increments of continuous passage of time, are a critical component of service alignment using WS-Calendar. There are many overloaded uses of terms about time, and within a particular time segment, there may be many of them. Within this document, we use the term Time Segments to encompass all the items defined in *Table 1-3: Terminology – Foundational Elements.*

The building block for the WS-Calendar information model is the Interval. The Interval is a time segment whose length is specified by the Duration. The Duration is represented by a string as defined in the iCalendar specification **[RFC5545]**. The Committee listened to arguments that we should redefine the use and meaning of Duration. Whatever their merit, the iCalendar Duration has a pre-existing meaning of the length of time of scheduled within an event.

An Interval is a unit of service delivery, bindable to time. An Unscheduled Interval is not linked to a specific date and time. A Scheduled Interval has a known start date and time. Intervals can legally complain all elements of the VTODO as defined in **[RFC5545]**. For convenience, the elements essential to Intervals are listed in Table 3-1.

Nothing in this section supersedes **[RFC5545].** Implementers SHALL refer to those respective specifications **[RFC5545]** and the **[XCAL]** specifications for the normative description of each element.

Table 3-1: Properties of Intervals

Elements	Use	Use in WS-Calendar
Dtstamp	xcal:dtstamp Mandatory	Identifies when Interval object was created
Uid	Mandatory	Used to enable unambiguous referencing by other components
Duration	xcal:Duration Optional	Identifies length of time for Interval
DtStart	Xcal:dateTime Optional	Scheduled start date and time for Interval
attach	Mandatory	In [xCal], any attachment. In WS-Calendar, the Attach contains the informational payload used by incorporating specifications. Defined in section 4.
link	As defined in **[RFC5998]** Optional	Links contain the temporal relations between Intervals that create Sequences. Section 3.3. describes Temporal Relations and their use.

An Interval specifies how long an activity lasts. An Unanchored Interval is not linked to a specific date and time. The example below shows the components section of a WS-Calendar message containing a single Interval

Example 3-2: An Unanchored Interval

```
<components>
<vtodo>
   <properties>
          <x-wscalendartype>Interval</x-wscalendartype>
          <uid>
                 <text>00959BC664CA650E933C892C@example.com</text>
          </uid>
          <description>
                 <text>Sample Contract</text>
          </description>
          <duration>
                 <duration>
                        <duration>T10H</duration>
```

```
                </duration>
            </duration>
        </properties>
    </vtodo>
<components>
```

Note that no start time is specified, and no relationship. Relationships are not needed until an Interval is incorporated into a Sequence.

3.3 Connecting the Intervals

Many iCalendar communications involve more than one Interval. Classic iCalendar **[RFC5545]** defines relationships internally. WS-Calendar instead uses the Web Link **[RFC5998]**, both for the traditional Relationships (parent, child, sibling) and for the Temporal Relationships. Links include a reference, a relation, and optional business parameters.

Temporal Relationships, new in WS-Calendar, use Web Linking **[RFC5998]** in an Interval (the Predecessor) to reference another Interval (the Successor). Temporal Relationships optionally include a Gap that specifies any lag between Predecessor and Successor.

Table 3-3: Temporal Relationships

Temporal Relationship	Short Form	Definition	Example
finishToStart	FS	As soon as the predecessor Interval finishes, the successor Interval starts.	When sanding is complete, painting begins.
finishToFinish	FF	The successor Interval continues as long as the predecessor Interval.	The concession stand stops serving 20 minutes after the end of the game.
startToFinish	SF	The start of the predecessor controls the finish of the successor.	The start of Attendee Check-in controls the end of the Interval "Set up registration booth."
startToStart	SS	The Predecessor Interval triggers the start of the second task. The Gap indicates the lag time	20 minutes after the caterer begins work, the dining lines are open.
Gap		Duration indicating the time between the predecessor and the successor. Optional, where missing, Gap is treated as a zero duration	Gap may be positive or negative.

While simple relationships may be ordered based on which task occurs first (finishToStart), if a later Interval is controlling, other choices may make more useful. For example, if ramp-up time must be completed before run-time, and run-time start is indicated in a contract, it may be useful to specify that the Ramp Interval (Successor) must complete before (startToFinish) the Designated Interval’s (Predecessor) scheduled start time. Referencing specifications should consider conformance around Temporal Relationships.

The relationship below indicates that this Interval is to start ten minutes following the finish of the Interval specified.

Example 3—1: Temporal Relationship

```
<x-wscalendarrelation>
<temporalrelationshiptype>finishtostart</temporalrelationshiptype>
<gap>
  <duration>
```

```
            <xcal:duration>T00:10</xcal:duration>
    </duration>
</gap>
<relatedto>
    <uid>00959BC664CA650E933C892C@example.com</uid>
</relatedto>
</x-wscalendarrelation>
```

If there is no temporal separation between Intervals, the gap element is optional. The following examples are equivalent expressions to express a relationship wherein both Intervals must start at the same moment.

Example 3—2: Temporal Relationship with Gap

```
<x-wscalendarrelation>
<temporalrelationshiptype>starttostart</temporalrelationshiptype>
<gap>
    <duration>
            <xcal:duration>T00:10</xcal:duration>
    </duration>
</gap>
<relatedto>
    <uid>00959BC664CA650E933C892C@example.com</uid>
</relatedto>
</x-wscalendarrelation>
```

Leaving out the optional Gap element, we have:

Example 3—3: Temporal Relationship without Gap

```
<x-wscalendarrelation>
<temporalrelationshiptype>starttostart</temporalrelationshiptype>
<relatedto>
    <uid>00959BC664CA650E933C892C@example.com</uid>
</relatedto>
</x-wscalendarrelation>
```

The two expressions of a Temporal Relationship above are equivalent.

Intervals with Temporal Relationships enable the message to express complex temporal relations to form a Sequence, as well as express the simple consecutive Intervals named a Partition

A Sequence describes a coherent set of Intervals that can be assembled from a collection of Intervals. As the rules for parsing XML do not mandate preservation of order within a sub-set, we cannot assume that order is preserved when parsing a set of Components. For Sequences in WS-Calendar, then, mere order is not enough—a Sequences is a collection of Intervals each of which Interval either refers to or ir referred by at least one Interval. Using the references, expressed as Temporal Relations, WS-Calendar describes a single coherent Sequence that is assembled from a set of Intervals in a collection.

3.4 Sequences: Combining Intervals

A Sequences is a collection of Intervals with a coherent set of Temporal Relationships. Temporal Relationships are transitive, so that if Interval A is related to Interval B, and Interval B is related to Interval C, then Interval A is related to Interval C. Sequences can also include Gluons *(see section 5.1, References and Inheritance.,* but for this section, we will discuss Sequences only as a set of Intervals.

Table 3-4: Introducing the Sequence

```
<components>
<vtodo>
    <properties>
            <x-wscalendartype>Interval</x-wscalendartype>
            <xcal:uid>
```

```
                <xcal:text>10959BC664CA650E933C892C@example.com</xcal:text>
            </xcal:uid>
            <xcal:description>
                <xcal:text>First Interval in Sequence</xcal:text>
            </xcal:description>
            <xcal:duration>
                <xcal:duration>T1H</xcal:duration>
            </xcal:duration>
    </properties>
</vtodo>
<vtodo>
    <properties>
            <x-wscalendartype>Interval</x-wscalendartype>
            <xcal:uid>
                <xcal:text>20959BC664CA650E933C892C@example.com</xcal:text>
            </xcal:uid>
            <xcal:description>
                <xcal:text>Second Interval in Sequence</xcal:text>
            </xcal:description>
            <xcal:summary>
                <xcal:text>Note the Temporal Relation to the First
                    Interval</xcal:text>
            </xcal:summary>
            <xcal:duration>
                <xcal:duration>T15M</xcal:duration>
            </xcal:duration>
            <x-wscalendarrelation>

    <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
    <relatedto>
                    <uid>10959BC664CA650E933C892C@example.com</uid>
                </relatedto>
            </x-wscalendarrelation>
    </properties>
</vtodo>
<vtodo>
    <properties>
            <x-wscalendartype>Interval</x-wscalendartype>
            <xcal:uid>
                <xcal:text>30959BC664CA650E933C892C@example.com</xcal:text>
            </xcal:uid>
            <xcal:description>
                <xcal:text>Third Interval in Sequence</xcal:text>
            </xcal:description>
            <xcal:duration>
                <xcal:duration>T30M</xcal:duration>
            </xcal:duration>
            <x-wscalendarrelation>

    <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
    <relatedto>
                    <uid>20959BC664CA650E933C892C@example.com</uid>
                </relatedto>
                <gap>
                    <xcal:duration>
                        <xcal:duration>T10M</xcal:duration>
                    </xcal:duration>
                </gap>
            </x-wscalendarrelation>
    </properties>
</vtodo>
```

```
</components>
```

In this example, the Intervals are one hour, 15 minutes, and 30 minutes long. There is a ten minute period between the second and third periods.

3.4.1 Anchoring a Sequence

A Sequence becomes an Anchored Sequence whenever a single Interval within the Sequence is Anchored. An Interval is Anchored when it has a specific starting date and time (dtstart).

Example 3—4: A Scheduled Sequence

```
<components>
<vtodo>
   <properties>
          <x-wscalendartype>Interval</x-wscalendartype>
          <xcal:uid>
                 <xcal:text>10959BC664CA650E933C892C@example.com</xcal:text>
          </xcal:uid>
          <xcal:description>
                 <xcal:text>First Interval in Sequence</xcal:text>
          </xcal:description>
          <dtstart>2010-09-11T13:00</dtstart>
          <xcal:duration>
                 <xcal:duration>T1H</xcal:duration>
          </xcal:duration>
   </properties>
</vtodo>
<vtodo>
   <properties>
          <x-wscalendartype>Interval</x-wscalendartype>
          <xcal:uid>
                 <xcal:text>20959BC664CA650E933C892C@example.com</xcal:text>
          </xcal:uid>
          <xcal:description>
                 <xcal:text>Second Interval in Sequence</xcal:text>
          </xcal:description>
          <xcal:duration>
                 <xcal:duration>T15M</xcal:duration>
          </xcal:duration>
          <x-wscalendarrelation>

   <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
   <relatedto>
                        <uid>10959BC664CA650E933C892C@example.com</uid>
                 </relatedto>
          </x-wscalendarrelation>
   </properties>
</vtodo>
</components>
```

Note that the entire Sequence is Anchored when a single Interval within the Sequence is Anchored.

3.5 State Changes

A common service interaction is to request that, at a certain time, a discrete state change will occur. It could be that the price will rise. It could be that a report will be run. Such a communication has no logical Duration. WS-Calendar communicates state changes through use of an Interval with the Duration explicitly set to zero time. Because the Duration is explicit, it will not be over-ridden through inheritance.

Specifications that incorporate WS-Calendar SHALL defined the business meaning of zero duration Intervals.

4 Service Characteristics: Payloads & Performance

While iCalendar expresses time and intervals, WS-Calendar associates those intervals with specific services and service performance characteristics. In iCalendar components, the ATTACH component is used to include information outside the scope of traditional Calendar services. WS-Calendar extends the ATTACH element to support payloads developed in other specifications. WS-Calendar also defines a new class of parameters for iCalendar components that specify the temporal performance requirements of the service.

4.1 Attachment and the Artifact

The WS-Calendar ATTACH component supports including or referencing an informational payload. This payload is something that is or can be provided over an Interval, or whose service qualities vary over several intervals. As the Technical Committee cannot know all the specifications that may incorporate WS-Calendar, this specification cannot discuss the contents of this payload. WS-Calendar does expect, however, that these payloads will respect and extend the inheritance and conformance rules herein specified.

The payload may be in-line, i.e., contained within the ATTACH element, object, or it may be found in another section of the same XML object, sharing the same message as WS-Calendar element, or it may be discovered by external reference. ATTACH, then, are used to request “perform as described here”, or “perform as described below”, or “perform as described elsewhere.”

The ATTACH element in WS-Calendar has three elements as below.

Table 4-1: Elements of a WS-Calendar Attachment

Attachment Element	Use	Discussion
artifact	any in-line XML (xs:any) *Optional.* An attachment must have at least one artifact or reference	Unevaluated container for payload describing service.
reference	[XPOINTER] *Optional* An attachment must have at least one of artifact or reference	Points to external XML, or XML located elsewhere in document

When a WS-Calendar reference uses an external reference to specify a service, that reference is an object of the type [XPOINTER] (see section 1.2)..[XPOINTER] is a general purpose URI and XML traversal standard. This [XPOINTER] object is in the named data element “Reference.”

Example 4—1: Use of an Attachment with inline XML artifact

```
<vtodo>
   <properties>
          <x-wscalendartype>Interval</x-wscalendartype>
          <xcal:uid>
                 <xcal:text>00959BC664CA650E933C892C@example.com</xcal:text>
          </xcal:uid>
          <xcal:description>
          <xcal:text>Sample Contract</xcal:text></xcal:description>
          <attach>
                 <artifact>
                        <emix-wip>
                               <price>8.45</price>
                               <quantity>8.45</quantity>
                        </emix-wip>
                 <artifact/>
```

```
        </attach>
   </properties>
</vtodo>
```

The Artifact is of type xs:any, allowing compliant XML from any namespace to be submitted as a payload. Per the conformance rules, the payload should be Fully Bound before evaluation.

Example 4—2: Use of an Attachment with external reference

```
<vtodo>
   <properties>
        <x-wscalendartype>Interval</x-wscalendartype>
        <xcal:uid>
               <xcal:text>00959BC664CA650E933C892C@example.com</xcal:text>
        </xcal:uid>
        <xcal:description>
        <xcal:text>Sample Contract</xcal:text></xcal:description>
        <attach>
               <reference>http://scheduled.ws-calendar-
                      service.com/contract1<reference>
        </attach>
   </properties>
</vtodo>
```

4.2 Specifying Timely Performance

WS-Calendar elements adapt the iCalendar objects to make interaction requirements explicit. For example, in human schedule interactions, different organizations have their own expectations. Meetings may start on the hour or within 5 minutes of the hour. As agents scheduled in those organizations, people learn the expected precision. In WS-Calendar, that precision must be explicit to prevent interoperation problems.

Service coordination between systems requires precise communication about expectation for the timeliness of performance. WS-Calendar defines performance parameters that can be added to any iCalendar component to make explicit the performance expectations within a scheduled event. Performance expectations can be set for each Interval or for an entire Sequence.

The Performance component refines the meaning of time-related service communication. All elements of the Performance object use the Duration element as defined in [RFC5545].

Table 4-2: Performance Characteristics

Performance Characteristic	Definition	Discussion
startBeforeTolerance	A Duration enumerating how far before the requested start time the requested service may commence.	Indicates if a service that begins at 1:57 is compliant with a request to start at 2:00
startAfterTolerance	A Duration enumerating how far after the requested start time the requested service may commence.	Indicates if a service that begins at 2:01 is compliant with a request to start at 2:00
endBeforeTolerance	A Duration enumerating how far before scheduled end time may end.	Indicates if a service that ends at 1:57 is compliant with a request to end at 2:00

Performance Characteristic	Definition	Discussion
endAfterTolerance	A Duration enumerating how far after the scheduled end time the requested service may commence.	Indicates if a service that ends at 2:01 is compliant with a request to end at 2:00
durationLongTolerance	A Duration indicating by how much the performance Duration may exceed the Duration specified in the Interval . It may be 0.	Used when run time is more important than start and stop time. DurationLongTolerance SHALL NOT be used when Start and End Tolerances are both specified.
durationShortTolerance	A Duration indicating by how much the performance Duration may fall short of Duration specified in the Interval . It may be 0.	Used when run time is more important than start and stop time. DurationShortTolerance SHALL NOT be used when Start and End Tolerances are both specified.
granularity	A Duration enumerating the smallest unit of time measured or tracked	Whatever the time tolerance above, there is some minimum time that is considered insignificant. A Granularity of 1 second defines the tracking and reporting requirements for a service.

Performance is part of the core WS-Calendar service definition. Similar prodUTCs or services, identical except for different Performance characteristics may appear in different markets. Performance characteristics influence the price offered and the service selected.

Note that Performance object does not indicate time, but only Duration. A performance object associated with an unscheduled Interval does not change when that Interval is scheduled.

The Performance object is an optional component of each WS-Calendar attachment.

Example 4—3: Performance Component

```
<attach>
  <uri>http://scheduled.ws-calendar-service.com/contract1</uri>
  <performance>
        <properties>
              <startbeforetolerance>
                    <duration>
                          <duration>T10M</duration>
                    </duration>
              </startbeforetolerance>
              <startaftertolerance>
                    <duration>
                          <duration>T0M</duration>
                    </duration>
              </startaftertolerance>
              <durationlongtolerance>
                    <duration>
                          <duration>T0M</duration>
                    </duration>
              </durationlongtolerance>
              <durationshorttolerance>
                    <duration>
                          <duration>T0M</duration>
                    </duration>
              </durationshorttolerance>
        </properties>
```

```
   </performance>
</attach>
```

In the example, the service can start as much as 10 minutes earlier than the scheduled time, and must start no later than the scheduled time. Whenever the service starts, the service must execute for exactly the Duration indicated.

Generally, the implementer should refrain from expressing unnecessary or redundant performance characteristics.

4.3 Expressing Service and Performance

Services, references and performance each appear in the examples below

Example 4—4: Interval with inline XML artifact and optional specified Performance

```
<vtodo>
   <properties>
         <x-wscalendartype>Interval</x-wscalendartype>
         <xcal:uid>
               <xcal:text>00959BC664CA650E933C892C@example.com</xcal:text>
         </xcal:uid>
         <xcal:description>
         <xcal:text>Sample Contract</xcal:text></xcal:description>
         <attach>
               <artifact>
                     <emix-wip>
                           <price>8.45</price>
                           <quantity>8.45</quantity>
                     </emix-wip>
               <artifact/>
               <performance>
                     <properties>
                           <startbeforetolerance>
                                 <duration>
                                       <duration>T10M</duration>
                                 </duration>
                           </startbeforetolerance>
                           <startaftertolerance>
                                 <duration>
                                       <duration>T0M</duration>
                                 </duration>
                           </startaftertolerance>
                     </properties>
               </performance>
         </attach>
   </properties>
</vtodo>
```

Example 4—5: Interval with external reference and optional specified performance

```
<vtodo>
   <properties>
         <x-wscalendartype>Interval</x-wscalendartype>
         <xcal:uid>
               <xcal:text>00959BC664CA650E933C892C@example.com</xcal:text>
         </xcal:uid>
         <xcal:description>
         <xcal:text>Sample Contract</xcal:text></xcal:description>
         <attach>
               <reference>http://scheduled.ws-calendar-
                     service.com/contract1<reference>
```

```
                <performance>
                      <properties>
                            <startbeforetolerance>
                                  <duration>
                                        <duration>T10M</duration>
                                  </duration>
                            </startbeforetolerance>
                            <startaftertolerance>
                                  <duration>
                                        <duration>T0M</duration>
                                  </duration>
                            </startaftertolerance>
                      </properties>
                </performance>
          </attach>
  </properties>
</vtodo>
```

5 Using Sequences: referencing, modifying, and remote access

Sequences can define specific progressions of performance or state within a wide range of services and specifications. They become more useful as they can be re-used or modified. A Sequence that is not fully specified can be adapted and re-used without re-statement. An abstract Sequence can become a service through iterative referencing.

As a Sequence is reified through reference, WS-Calendar specifies how additional information is applied or not applied to each Interval through a chain of references. We refer to this process as inheritance. Derivative specification can take advantage of inheritance by defining specific rules that conform to the WS-Calendar inheritance pattern.

This section describes how to create references to Sequences, including remote references, the rules that allow schedule-related information to become more complete through those references, and how to specify conforming rules in derivative specifications.

5.1 References and Inheritance.

Sequences are composed of Intervals for which a set of temporal relations have been defined. **[RFC5545]** also defines the “parent”, “child” and “sibling” relationships, in which one component references another by UID. In WS-Calendar, we reference a Sequence by creating a relationship with any single Interval in the Sequence. We refer to the Interval within a Sequence that has this relationship as the Designated Interval.

Wherever there is “missing” information in the Designated Interval, it can be inherited is inherited from the referring component; we use the “parent” relationship to reference the designated Interval. These references may be local or remote. Some, but not all, of the information can be inherited by the other Intervals in the Sequence. Adding additional references can further specify information in the Sequence through inheritance; these additional references created by specifying an additional component that has a parent relation to the previous referring component. In this way, we can create a grand-parent and a great grand-parent.

Each parent bequeaths information to its child. A child inherits this information in accord with the inheritance rules. If the child is itself a parent, it bequeaths its information, the bound result of its internal information and its inheritance, to its child. Information to complete the specification of a Sequence flows in this way from parent to child, from the outer reference to the inner Sequence.

Inheritance by the designated Interval is governed by slightly different inheritance rules than the other Intervals in the Sequence. In particular, only the designated Interval can inherit the start date and time from its parent. The starting date and times if other Intervals in a Sequence are computed using the temporal relationships within the Sequence. Other information can be inherited by all Intervals in a Sequence. Full inheritance rules are specified at [reference].

The referring components are named Gluons. In physics, gluons are particles that affect the exchanges of force between quarks, but are not themselves quarks. By analogy, WS-Calendar Gluons affect the referencing and binding of Intervals in a Sequence, but are not themselves Intervals or part of Sequences. Because Intervals can inherit almost any property from a Gluon, Gluons must contain most of the same information elements as Intervals. Because Intervals can contain information payloads for specifications that use WS-Calendar, and these payloads can inherit information from gluons in the same way Intervals do, Gluons must be able to contain information payloads from those specifications as well. Gluons are described in the next section.

5.1.1 Introducing the Gluon

WS-Calendar Gluons are used to referencing and bind the Intervals in a Sequence, but are not themselves Intervals or part of Sequences. Gluons must contain most of the same information elements as Intervals, because Intervals can inherit almost any property from a Gluon. When Intervals are used in other specifications, they contain payloads for that are not defined in WS-Calendar. Gluons can also hold the same payloads, and conforming specifications MUST define inheritance rules that govern inheritance within these payloads. Conformance rules, including those for inheritance conformance, are discussed in section 17 *Conformance and Rules for WS-Calendar and Referencing Specifications.*

The WS-Calendar Gluon is in essence an the Interval component profiled down to minimal elements for which inheritance rules defined, and able to carry a conforming informational payload. (See Appendix *Overview of WS-Calendar, its Antecedents and its Use*) Calendar Gluons use iCalendar relations to apply service information to Sequences.

Table 5-1: Calendar Gluon elements in WS-Calendar

Calendar Gluon Element	Use	Discussion
dtStamp	[XCAL]:dtstamp *Mandatory*	Time and date that Calendar Gluon object was created
Uid	*Mandatory*	Used to enable unambiguous referencing of each VTODO object
Summary	Text *Optional*	Text describing the Calendar Gluon
child	As defined in **[RFC5998]** Parent	A Calendar Gluon must have a link to at least one child.
dtStart	dateTime Start time for the related Interval of the Sequence. *Optional*	An Calendar Gluon may either have a dtStart or a dtEnd, but may not have both.
dtEnd	DateTime. Scheduled completion time for the related Interval of the Sequence. *Optional*	An Calendar Gluon may either have a dtStart or a dtEnd, but may not have both.
duration	Duration *Optional*	If specified, a Duration is inherited by all Intervals in the referred-to Sequence,
attach	WSCalendar:Attachment Optional	The Attach contains the informational payload used by incorporating specifications. Defined in section 4..
Availability	Vavailability, Optional	Provides information as to when information the service can or cannot be offered.

Because the properties of the Calendar Gluon are bequeathed to the child Sequence, they can stand for the elements in any Interval in the Sequence, as defined in the Conformance Section. An inherited element can even serve as a substitute for an Interval mandatory element. For example, Duration is mandatory for all Intervals. Intervals are able to inherit Duration from a parent. A single Duration in the Parent can be inherited by each Interval in a Sequence. In this way, a Sequence in which every Interval does not have a Duration, could be made complete through inheritance. If one of those Intervals does include a Duration, the Bound Duration would be its own, rather than that it inherited from a Parent of the Sequence.

5.1.2 Availability

One use for gluons is to expose a Sequence for remote invocation. The service offered may be sometimes unavailable. WS-Calendar incorporates the iCalendar extension **[Vavailability].**

It is likely that the service requester is aware only of when he wants the service and for how long. These are properties of the Designated Interval. Availability will be interpreted as a filter on the Designated Interval, but on no others.

5.1.3 Calendar Gluons and Sequences

Calendar Gluons express common service requirements for an entire Sequence. If a Gluon is parent to an Interval in a Sequence, then the Gluon's Attachment expresses service attributes inheritable by all Intervals in the Sequence.

In this example, the Sequence in the previous example is expressed using a Calendar Gluon.

Example 5—1: Sequence with Performance defined in the Calendar Gluon

```
<components>
<x-calendargluon>
   <properties>
         <x-wscalendartype>CalendarGluon</x-wscalendartype>
         <xcal:uid>
                <xcal:text>00959BC664CA650E933C892C@example.com</xcal:text>
         </xcal:uid>
         <xcal:description>
                <xcal:text> Calendar Gluon with Sequence </xcal:text>
         </xcal:description>
         <xcal:comment>
                <xcal:text> creates common performance expectations (+/- 1
second)
                      for the entire Sequence. Also sets common duration (15
                      minutes) for all members of the Sequence, No Interval
may end
                      after its scheduled end-time </xcal:text>
         </xcal:comment>
         <xcal:duration>
                <xcal:duration>T15M</xcal:duration>
         </xcal:duration>
         <attach>
                <performance>
                      <properties>
                             <endbeforetolerance>
                                    <duration>
                                          <duration>T1S</duration>
                                    </duration>
                             </endbeforetolerance>
                             <endaftertolerance>
                                    <duration>
                                          <duration>T1S</duration>
                                    </duration>
                             </endtaftertolerance>
                      </properties>
                </performance>
        </attach>
        <xcal:related-to>
                <xcal:parameters>
                                <xcal:reltype>PARENT</xcal:reltype>
                </xcal:parameters>
                <xcal:text>10959BC664CA650E933C892C@example.com </xcal:text>
        </xcal:related-to>
   </properties>
</x-calendargluon>
<vtodo>
   <properties>
        <x-wscalendartype>Interval</x-wscalendartype>
        <xcal:uid>
                <xcal:text>10959BC664CA650E933C892C@example.com</xcal:text>
        </xcal:uid>
        <xcal:description>
                <xcal:text>First Interval in Sequence</xcal:text>
        </xcal:description>
        <xcal:summary>
                <xcal:text>Inherits all performance from Gluon as well
                        As the duration</xcal:text>
        </xcal:summary>
        <attach>
                <artifact>
                        <emix-wip>
                                <price>8.45</price>
                                <quantity>4200</quantity>
                        </emix-wip>
                <artifact/>
        </attach>
   </properties>
</vtodo>
<vtodo>
   <properties>
        <x-wscalendartype>Interval</x-wscalendartype>
        <xcal:uid>
                <xcal:text>20959BC664CA650E933C892C@example.com</xcal:text>
        </xcal:uid>
        <xcal:description>
                <xcal:text>Second Interval in Sequence</xcal:text>
        </xcal:description>
        <xcal:summary>
                <xcal:text>Inherits all performance from Gluon, follows
                        Finish of Interval 1, inherits duration</xcal:text>
        </xcal:summary>
        <attach>
                <artifact>
                        <emix-wip>
                                <price>8.49</price>
                                <quantity>4500</quantity>
                        </emix-wip>
                <artifact/>
        </attach>
        <x-wscalendarrelation>

   <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
   <relatedto>
                        <uid>10959BC664CA650E933C892C@example.com</uid>
                </relatedto>
                <gap>
                        <xcal:duration>
```

```
                                <xcal:duration>T0M</xcal:duration>
                        </xcal:duration>
                </gap>
        </x-wscalendarrelation>
   </properties>
</vtodo>
<vtodo>
   <properties>
        <x-wscalendartype>Interval</x-wscalendartype>
        <xcal:uid>
                <xcal:text>30959BC664CA650E933C892C@example.com</xcal:text>
        </xcal:uid>
        <xcal:description>
                <xcal:text>Third Interval in Sequence</xcal:text>
        </xcal:description>
        <xcal:summary>
                <xcal:text>Inherits all performance from Gluon, follows
                        Finish of Interval 2, overrides duration</xcal:text>
        </xcal:summary>
        <attach>
                <artifact>
                        <emix-wip>
                                <price>7.45</price>
                                <quantity>6000</quantity>
                        </emix-wip>
                <artifact/>
        </attach>
        <xcal:duration>
                <xcal:duration>T30M</xcal:duration>
        </xcal:duration>
        <x-wscalendarrelation>

   <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
   <relatedto>
                        <uid>20959BC664CA650E933C892C@example.com</uid>
                </relatedto>
                <gap>
                        <xcal:duration>
                                <xcal:duration>T5M</xcal:duration>
                        </xcal:duration>
                </gap>
        </x-wscalendarrelation>
   </properties>
</vtodo>
</components>
```

Note that the performance expectations, identical for each Interval, have moved into the Calendar Gluon. Not also that while the duration for all Intervals in the partition is set in the Calendar Gluon, Interval 3 overrides that with a half hour duration assigned locally. This Calendar Gluon happens to be related to the first Interval in the Sequence; there are specific use cases (discussed below) which require it to be linked to other Intervals.

5.2 Inheritance rules for Calendar Gluons

In general, the rules that anything specified in the Parent Calendar Gluon applies to each Child. The Parent of an Interval in a Sequence is parent to all Intervals in the Sequence. As a Sequence creates single temporal relationship, assigning a start time (dtstart) to any Interval allows computation of the starting time for any of them.

Table 5-2 Gluon Inheritance rules

Attribute	Inheritance Rules
General	A Interval or Calendar Gluon inherits its attributes through it's the parent. Local specification of an attributes overrides any inheritance.
Duration	Follows general rules
Temporal Relation	Relationship Type and Gap only are inherited. Either may be overridden locally. To specify no gap when a parent specifies a gap, an explicit zero duration gap must be specified. Related-to is not inherited.
Performance	Performance is either inherited intact or overridden completely. There are no rules for recombining partial Performance objects through inheritance.
Artifacts	Artifacts hold payload from other specifications. Elements within Artifacts are inherited in accord with the rules in those specifications, which must be consistent the inheritance rules in WS-Calendar. Artifacts are evaluated for completeness and conformance only after processing inheritance.
Schedules	Schedules, i.e., the start date and time, are inherited only by the designated Interval. The start date and times of other Intervals are computed by reference to the designated Interval. Between the Gluon bequeathing a schedule and the Designated Interval, an intervening Gluon may set availability. It is up to the application or to the specification incorporating WS-Calendar to assert whether an Interval that is outside the availability is conforming or not.
Availability	Availability communicates restrictions on when a service is offered. Service availability is interpreted for the Designated Interval only. If there are two availability objects, they are evaluated for the union of the two availabilities. For example, if I am available all week from 2:00 to 6:00 in one, and available all day Tuesday in the other, then after inheritance, there remains only 2:00 to 6:00 on Tuesday,

5.3 Optimizing the expression of a Partition

Partitions are Sequences composed of consecutive Intervals. A Partition can be further optimized by bringing the relationship into the Gluon. Notice that while the type of the relationship is defined in the Calendar Gluon, the Temporal Relation for each Interval must still be expressed within the Interval.

Example 5—2: Partition with Duration and Performance defined in the Calendar Gluon

```
<components>
<x-calendargluon>
   <properties>
          <x-wscalendartype>CalendarGluon</x-wscalendartype>
          <xcal:uid>
                 <xcal:text>00959BC664CA650E933C892C@example.com</xcal:text>
          </xcal:uid>
          <xcal:description>
                 <xcal:text>Calendar Gluon for energy markets with consecutive
                        Identical Intervals</xcal:text>
          </xcal:description>
          <xcal:comment>
                 <xcal:text> creates common performance expectations that the
                        granularity for measuring all Intervals is (+/- 1
second)
                        Also sets common duration (15 minutes)for all members
of
                        the Sequence). Each Interval in the partitions begins
                        immediately after its predecessor finishes.
</xcal:text>
          </xcal:comment>
          <xcal:duration>
                 <xcal:duration>T15M</xcal:duration>
          </xcal:duration>
          <attach>
                 <artifact>
                        <emix-wip>PRODUTC SPECIFICATION UNDEFINED</emix-wip>
                 <artifact/>
                 <performance>
                        <properties>
                               <granularity>
                                      <duration>
                                             <duration>T1S</duration>
                                      </duration>
                               </granularity>
                        </properties>
                 </performance>
          </attach>
          <x-wscalendarrelation>

   <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
                 <gap>
                        <xcal:duration>
                               <xcal:duration>T0S</xcal:duration>
                        </xcal:duration>
                 </gap>
          </x-wscalendarrelation>
          <xcal:related-to>
                 <xcal:parameters>
                               <xcal:reltype>PARENT</xcal:reltype>
                 </xcal:parameters>
                 <xcal:text>10959BC664CA650E933C892C@example.com </xcal:text>
          </xcal:related-to>
   </properties>
</x-calendargluon>
<vtodo>
   <properties>
          <x-wscalendartype>Interval</x-wscalendartype>
          <xcal:uid>
                 <xcal:text>10959BC664CA650E933C892C@example.com</xcal:text>
          </xcal:uid>
          <xcal:description>
                 <xcal:text>First Interval in Sequence</xcal:text>
          </xcal:description>
          <xcal:summary>
                 <xcal:text>Inherits all performance from Gluon as well
                        As the duration and the Temporal Relation</xcal:text>
          </xcal:summary>
          <attach>
                 <artifact>
                        <emix-wip>
                               <price>8.45</price>
                               <quantity>4200</quantity>
                        </emix-wip>
                 <artifact/>
          </attach>
   </properties>
</vtodo>
<vtodo>
   <properties>
```

```
          <x-wscalendartype>Interval</x-wscalendartype>
          <xcal:uid>
                <xcal:text>20959BC664CA650E933C892C@example.com</xcal:text>
          </xcal:uid>
          <xcal:description>
                <xcal:text>Second Interval in Sequence</xcal:text>
          </xcal:description>
          <attach>
                <artifact>
                      <emix-wip>
                            <price>8.49</price>
                            <quantity>4500</quantity>
                      </emix-wip>
                <artifact/>
          </attach>
          <x-wscalendarrelation>
                <relatedto>
                      <uid>10959BC664CA650E933C892C@example.com</uid>
                </relatedto>
          </x-wscalendarrelation>
   </properties>
</vtodo>
<vtodo>
   <properties>
          <x-wscalendartype>Interval</x-wscalendartype>
          <xcal:uid>
                <xcal:text>30959BC664CA650E933C892C@example.com</xcal:text>
          </xcal:uid>
          <xcal:description>
                <xcal:text>Third Interval in Sequence</xcal:text>
          </xcal:description>
          <xcal:summary>
                <xcal:text>Inherits all performance from Gluon, follows
                      Finish of Interval 2, overrides duration</xcal:text>
          </xcal:summary>
          <attach>
                <artifact>
                      <emix-wip>
                            <price>7.45</price>
                            <quantity>6000</quantity>
                      </emix-wip>
                <artifact/>
          </attach>
          <x-wscalendarrelation>
                <relatedto>
                      <uid>20959BC664CA650E933C892C@example.com</uid>
                </relatedto>
          </x-wscalendarrelation>
   </properties>
</vtodo>
<components>
```

This Partition shows a school schedule in which classes start one hour apart. Each class is for 50 minutes, and there is a 10 minute gap between each as students move between classes. Classes may not begin before the schedule, but they may start up to five minutes late.

Stripped of all annotations, this can be expressed as follows:

Example 5—3: Partition without annotations

```
<components>
<x-calendargluon>
   <properties>
			<x-wscalendartype>CalendarGluon</x-wscalendartype>
			<xcal:uid>
				<xcal:text>00959BC664CA650E933C892C@example.com</xcal:text>
			</xcal:uid>
			<xcal:duration>
				<xcal:duration>T50M</xcal:duration>
			</xcal:duration>
			<attach>
				<artifact>
					<classroom>demonstration specification</classroom>
				<artifact/>
			</attach>
			<x-wscalendarrelation>

   <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
				<gap>
					<xcal:duration>
						<xcal:duration>T10M</xcal:duration>
					</xcal:duration>
				</gap>
			</x-wscalendarrelation>
			<xcal:related-to>
				<xcal:parameters>
						<xcal:reltype>PARENT</xcal:reltype>
				</xcal:parameters>
				<xcal:text>10959BC664CA650E933C892C@example.com </xcal:text>
			</xcal:related-to>
   </properties>
</x-calendargluon>
<vtodo>
   <properties>
			<x-wscalendartype>Interval</x-wscalendartype>
			<xcal:uid>
				<xcal:text>10959BC664CA650E933C892C@example.com</xcal:text>
			</xcal:uid>
			<attach>
				<artifact>
					<classroom><students>48</students></classroom>
				<artifact/>
			</attach>
   </properties>
</vtodo>
<vtodo>
   <properties>
			<x-wscalendartype>Interval</x-wscalendartype>
			<xcal:uid>
				<xcal:text>20959BC664CA650E933C892C@example.com</xcal:text>
			</xcal:uid>
			<attach>
				<artifact>
					<classroom><students>65</students></classroom>
				<artifact/>
			</attach>
			<x-wscalendarrelation>
				<relatedto>
					<uid>10959BC664CA650E933C892C@example.com</uid>
				</relatedto>
			</x-wscalendarrelation>
   </properties>
</vtodo>
<vtodo>
   <properties>
```

```
            <x-wscalendartype>Interval</x-wscalendartype>
            <xcal:uid>
                  <xcal:text>30959BC664CA650E933C892C@example.com</xcal:text>
            </xcal:uid>
            <attach>
                  <artifact>
                        <classroom><students>34</students></classroom>
                  <artifact/>
            </attach>
            <x-wscalendarrelation>
                  <relatedto>
                        <uid>20959BC664CA650E933C892C@example.com</uid>
                  </relatedto>
            </x-wscalendarrelation>
      </properties>
   </vtodo>
   <components>
```

A Sequence can also be scheduled in the Calendar Gluon.

Example 5—4: A Scheduled Sequence showing Temporal Relationship Inheritance

```
<components>
<x-calendargluon>
   <properties>
         <x-wscalendartype>CalendarGluon</x-wscalendartype>
         <xcal:uid>
               <xcal:text>00959BC664CA650E933C892C@example.com</xcal:text>
         </xcal:uid>
               <dtstart>2010-09-11 T00:15</dtstart>
         <attach>
               <artifact>
                     <classroom>demonstration specification</classroom>
               <artifact/>
         </attach>
         <xcal:related-to>
               <xcal:parameters>
                           <xcal:reltype>PARENT</xcal:reltype>
               </xcal:parameters>
               <xcal:text>10959BC664CA650E933C892C@example.com </xcal:text>
         </xcal:related-to>
   </properties>
</x-calendargluon>
<vtodo>
   <properties>
         <x-wscalendartype>Interval</x-wscalendartype>
         <xcal:uid>
               <xcal:text>10959BC664CA650E933C892C@example.com</xcal:text>
         </xcal:uid>
         <xcal:description>
               <xcal:text>First Interval in Sequence</xcal:text>
         </xcal:description>
         <dtstart>2010-09-11T13:00</dtstart>
         <xcal:duration>
               <xcal:duration>T1H</xcal:duration>
         </xcal:duration>
   </properties>
</vtodo>
<vtodo>
   <properties>
         <x-wscalendartype>Interval</x-wscalendartype>
         <xcal:uid>
```

```
                    <xcal:text>20959BC664CA650E933C892C@example.com</xcal:text>
            </xcal:uid>
            <xcal:description>
                    <xcal:text>Second Interval in Sequence</xcal:text>
            </xcal:description>
            <xcal:duration>
                    <xcal:duration>T15M</xcal:duration>
            </xcal:duration>
            <x-wscalendarrelation>

   <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
   <relatedto>
                            <uid>10959BC664CA650E933C892C@example.com</uid>
                    </relatedto>
            </x-wscalendarrelation>
   </properties>
</vtodo>
</components>
```

5.4 Mixed Inheritance of Start Time

A Sequence has not been scheduled until it has both a start time and a start date. Start time and date SHALL be expressed together when all components are in a single communication. Time and Date MAY be separated when the full Sequence and schedule are created by reference.

To illustrate this, here is the classroom scheduling Partition from Example 5—3, updated to include each day's school opening.

Example 5—5: Partition with Duration and Performance defined in the Calendar Gluon

```
<components>
<x-calendargluon>
   <properties>
            <x-wscalendartype>CalendarGluon</x-wscalendartype>
            <xcal:uid>
                    <xcal:text>
                            90959BC664CA650E933C892C@invokingexample.com
                    </xcal:text>
            </xcal:uid>
            <dtstart>2010-09-13T09:00</dtstart>
            <xcal:related-to>
                    <xcal:parameters>
                            <xcal:reltype>PARENT</xcal:reltype>
                    </xcal:parameters>
                    <xcal:uri>http://scheduled.ws-calendar-
service.com/classSchedule
                            </xcal:uri>
            </xcal:related-to>
   </properties>
</x-calendargluon>
</components>
```

Here, an external Calendar Gluon (above) makes reference to a published classroom schedule service:

```
<x-calendargluon>
   <properties>
            <x-wscalendartype>CalendarGluon</x-wscalendartype>
            <xcal:uid>
                    <xcal:text>http://scheduled.ws-calendar-
service.com/classSchedule
                            </xcal:text>
            </xcal:uid>
```

```
         <xcal:description>
                <xcal:text>MWF Classroom Schedule
                      Identical Intervals</xcal:text>
         </xcal:description>
         <xcal:comment>
                <xcal:text>Publishes a common classroom schedule for Monday,
                      Wednesday, Friday for a semester at a school. Note that
each
                      day starts at 9:00</xcal:text>
         </xcal:comment>
         <dtstart>T09:00</dtstart>
         <xcal:duration>
                <xcal:duration>T50M</xcal:duration>
         </xcal:duration>
         <attach>
                <xcal:uri></xcal:uri>
         </attach>
         <x-wscalendarrelation>

   <temporalrelationshiptype>finishtostart</temporalrelationshiptype>
                <gap>
                      <xcal:duration>
                            <xcal:duration>T10M</xcal:duration>
                      </xcal:duration>
                </gap>
         </x-wscalendarrelation>
         <xcal:related-to>
                <xcal:parameters>
                      <xcal:reltype>PARENT</xcal:reltype>
                </xcal:parameters>
                <xcal:text>10959BC664CA650E933C892C@example.com </xcal:text>
         </xcal:related-to>
   </properties>
</x-calendargluon>
<vtodo>
   <properties>
         <x-wscalendartype>Interval</x-wscalendartype>
         <xcal:uid>
                <xcal:text>10959BC664CA650E933C892C@example.com</xcal:text>
         </xcal:uid>
         <attach>
                <artifact>
                      <classroom><students>48</students></classroom>
                <artifact/>
         </attach>
   </properties>
</vtodo>
<vtodo>
   <properties>
         <x-wscalendartype>Interval</x-wscalendartype>
         <xcal:uid>
                <xcal:text>20959BC664CA650E933C892C@example.com</xcal:text>
         </xcal:uid>
         <attach>
                <artifact>
                      <classroom><students>65</students></classroom>
                <artifact/>
         </attach>
         <x-wscalendarrelation>
                <relatedto>
                      <uid>10959BC664CA650E933C892C@example.com</uid>
                </relatedto>
```

```
        </x-wscalendarrelation>
  </properties>
</vtodo>
<vtodo>
  <properties>
        <x-wscalendartype>Interval</x-wscalendartype>
        <xcal:uid>
              <xcal:text>30959BC664CA650E933C892C@example.com</xcal:text>
        </xcal:uid>
        <attach>
              <artifact>
                    <classroom><students>34</students></classroom>
              <artifact/>
        </attach>
        <x-wscalendarrelation>
              <relatedto>
                    <uid>20959BC664CA650E933C892C@example.com</uid>
              </relatedto>
        </x-wscalendarrelation>
  </properties>
</vtodo>
<components>

```

In the example above, a general purpose classroom calendar has been created and advertised with an URL. The class day always starts at 9:00. The referring Calendar Gluon scheduled a particular instance of this Sequence for Monday, September 13.

This double inheritance, in which a Sequence inherits from a Calendar Gluon which inherits from a Calendar Gluon is a useful pattern for scheduling an advertised service.

5.5 Other Scheduling Scenarios

Sometimes, the invoker of a service is interested only in single Interval of the Sequence, but the entire Sequence is required. In the example below, the second Interval is advertised, i.e., the Calendar Gluon points to the second Interval. The first Interval might be a required ramp-period, during which the underlying process is “warming up”, and which may bring some lesser service to market during that ramp time. The ramp-down time at the end is similarly fixed. The entire Service offering is represented by the exposed (it has a public URI) Calendar Gluon.

Example 5—6: Standard Sequence with Ramp-Up and Ramp Down

```
<components>
<x-calendargluon>
  <properties>
        <x-wscalendartype>CalendarGluon</x-wscalendartype>
        <xcal:uid>
              <xcal:text>http://scheduled.ws-calendar-
service.com/runcontract
              </xcal:text>
        </xcal:uid>
        <xcal:description>
              <xcal:text>Advertisement of schedule with ramp up and ramp
down
                    services.</xcal:text>
        </xcal:description>
        <xcal:comment>
              <xcal:text>Invokes second of three Intervals in the Sequence
              </xcal:text>
        </xcal:comment>
        <attach>
```

```
                <xcal:uri><emix-wip[3]></emix-wip></xcal:uri>
            </attach>
            <xcal:related-to>
                <xcal:parameters>
                    <xcal:reltype>PARENT</xcal:reltype>
                </xcal:parameters>
                <xcal:text>20959BC664CA650E933C892C@example.com </xcal:text>
            </xcal:related-to>
    </properties>
            <xcal:duration>
                <xcal:duration>T6H</xcal:duration>
            </xcal:duration>
</x-calendargluon>
<vtodo>
    <properties>
            <x-wscalendartype>Interval</x-wscalendartype>
            <xcal:uid>
                <xcal:text>10959BC664CA650E933C892C@example.com</xcal:text>
            </xcal:uid>
            <xcal:description>
                <xcal:text>Ramp-Up Interval</xcal:text>
            </xcal:description>
            <xcal:summary>
                <xcal:text>Required as part of operations. Slowly increasing,
yet
                    fixed over-all, energy produced.
                </xcal:text>
            </xcal:summary>
            <xcal:duration>
                <xcal:duration>T45M</xcal:duration>
            </xcal:duration>
            <attach>
                <artifact>
                    <emix-wip>describes ramp-up</emix-wip>
                </artifact>
            </attach>
    </properties>
</vtodo>
<vtodo>
    <properties>
            <x-wscalendartype>Interval</x-wscalendartype>
            <xcal:uid>
                <xcal:text>20959BC664CA650E933C892C@example.com</xcal:text>
            </xcal:uid>
            <xcal:description>
                <xcal:text>Run Interval</xcal:text>
            </xcal:description>
            <xcal:summary>
```

[3] There is no EMIX-WIP specification. EMIX-WIP represents a generic energy market artifact, perhaps indicative of future specifications.

```
                <xcal:text>Inherits all performance from Gluon, follows
                        Finish of Interval 2, overrides duration</xcal:text>
        </xcal:summary>
        <attach>
                <artifact>
                        <emix-wip >ProdUTC Definition</emix-wip>
                </artifact>
        </attach>
        <x-wscalendarrelation>
                <relatedto>
                        <uid>10959BC664CA650E933C892C@example.com</uid>
                </relatedto>
        </x-wscalendarrelation>
  </properties>
</vtodo>
<vtodo>
  <properties>
        <x-wscalendartype>Interval</x-wscalendartype>
        <xcal:uid>
                <xcal:text>30959BC664CA650E933C892C@example.com</xcal:text>
        </xcal:uid>
        <xcal:description>
                <xcal:text>Ramp-down Interval</xcal:text>
        </xcal:description>
        <xcal:summary>
                <xcal:text>Required as part of operations. Fixed time and
fixed,
                        while diminishing, energy produced.
                </xcal:text>
        </xcal:summary>
        <xcal:duration>
                <xcal:duration>T30M</xcal:duration>
        </xcal:duration>
        <attach>
                <artifact>
                        <emix-wip>describes ramp-up</emix-wip>
                </artifact>
        </attach>
        <x-wscalendarrelation>
                <relatedto>
                        <uid>20959BC664CA650E933C892C@example.com</uid>
                </relatedto>
        </x-wscalendarrelation>
  </properties>
</vtodo>
```

When the service is scheduled, the time and duration are specified. The duration only applies to the Second Interval as all others have their duration explicitly specified.

```
<components>
<x-calendargluon>
  <properties>
        <x-wscalendartype>CalendarGluon</x-wscalendartype>
        <xcal:uid>
                <xcal:text>http://scheduled.ws-calendar-service.com/scheduleB
                        </xcal:text>
        </xcal:uid>
        <xcal:description>
                <xcal:text>Advertisement of schedule with ramp up and ramp
down
                        services.</xcal:text>
        </xcal:description>
```

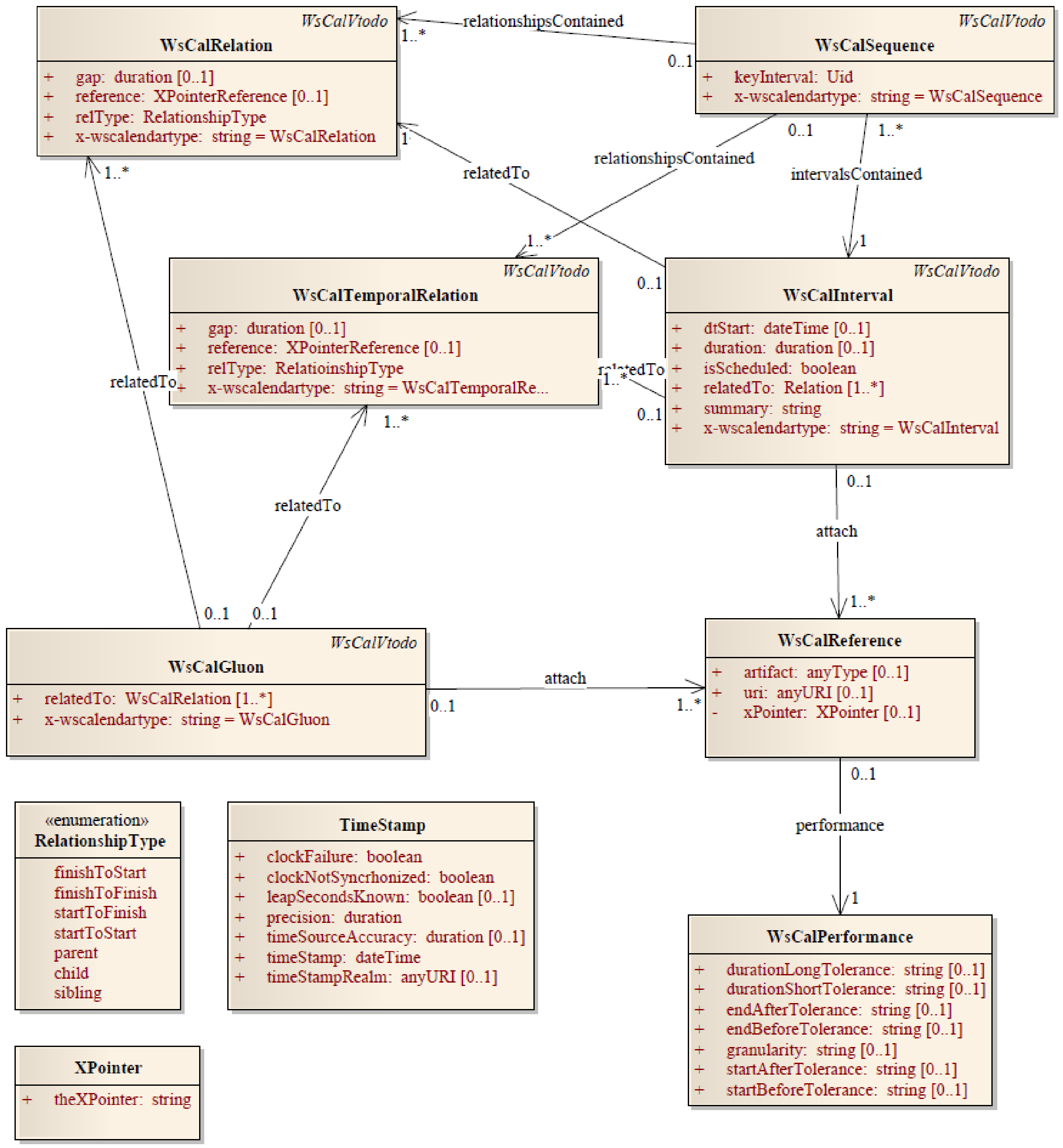
```
         <xcal:comment>
                <xcal:text>Invokes second of three Intervals in the Sequence
                </xcal:text>
         </xcal:comment>
         <attach>
                <artifact>
                       <emix-wip[4]>
                              <execute-price>15000</execute-price>
                       </emix-wip>
                </artifact>
         </attach>
         <xcal:related-to>
                <xcal:parameters>
                              <xcal:reltype>PARENT</xcal:reltype>
                </xcal:parameters>
                <xcal:text>http://scheduled.ws-calendar-
service.com/runcontract
                </xcal:text>
         </xcal:related-to>
         <xcal:duration>
                <xcal:duration>T6H</xcal:duration>
         </xcal:duration>
   </properties>
</x-calendargluon>
</components>
```

In this case, the specific Interval is scheduled and a run time of 6 hours is specified for a price of $15,000.

[4] There is no EMIX-WIP specification. EMIX-WIP represents a generic energy market artifact, perhaps evocative of future specifications.

6 WS-Calendar Models

6.1 Abstract model for WS-Calendar Objects



WsCalVtodo
WsCalRelation
+ gap: duration [0..1]
+ reference: XPointerReference [0..1]
+ relType: RelationshipType
+ x-wscalendartype: string = WsCalRelation
relationshipsContained
1..*
0..1
WsCalVtodo
WsCalSequence
+ keyInterval: Uid
+ x-wscalendartype: string = WsCalSequence
1
relatedTo
relationshipsContained
0..1
1..*
intervalsContained
1..*
1
1..*
WsCalVtodo
WsCalTemporalRelation
+ gap: duration [0..1]
+ reference: XPointerReference [0..1]
+ relType: RelatioinshipType
+ x-wscalendartype: string = WsCalTemporalRe...
0..1
WsCalVtodo
WsCalInterval
+ dtStart: dateTime [0..1]
+ duration: duration [0..1]
+ isScheduled: boolean
+ relatedTo: Relation [1..*]
+ summary: string
+ x-wscalendartype: string = WsCalInterval
relatedTo
relatedTo
1..*
0..1
1..*
relatedTo
0..1
attach
1..*
0..1
0..1
WsCalVtodo
WsCalGluon
+ relatedTo: WsCalRelation [1..*]
+ x-wscalendartype: string = WsCalGluon
attach
0..1
1..*
WsCalReference
+ artifact: anyType [0..1]
+ uri: anyURI [0..1]
- xPointer: XPointer [0..1]
0..1
performance
1
«enumeration»
RelationshipType
finishToStart
finishToFinish
startToFinish
startToStart
parent
child
sibling
TimeStamp
+ clockFailure: boolean
+ clockNotSyncrhonized: boolean
+ leapSecondsKnown: boolean [0..1]
+ precision: duration
+ timeSourceAccuracy: duration [0..1]
+ timeStamp: dateTime
+ timeStampRealm: anyURI [0..1]
WsCalPerformance
+ durationLongTolerance: string [0..1]
+ durationShortTolerance: string [0..1]
+ endAfterTolerance: string [0..1]
+ endBeforeTolerance: string [0..1]
+ granularity: string [0..1]
+ startAfterTolerance: string [0..1]
+ startBeforeTolerance: string [0..1]
XPointer
+ theXPointer: string

Figure 1: Abstract UML Model

6.2 Implementation Model for WS-Calendar

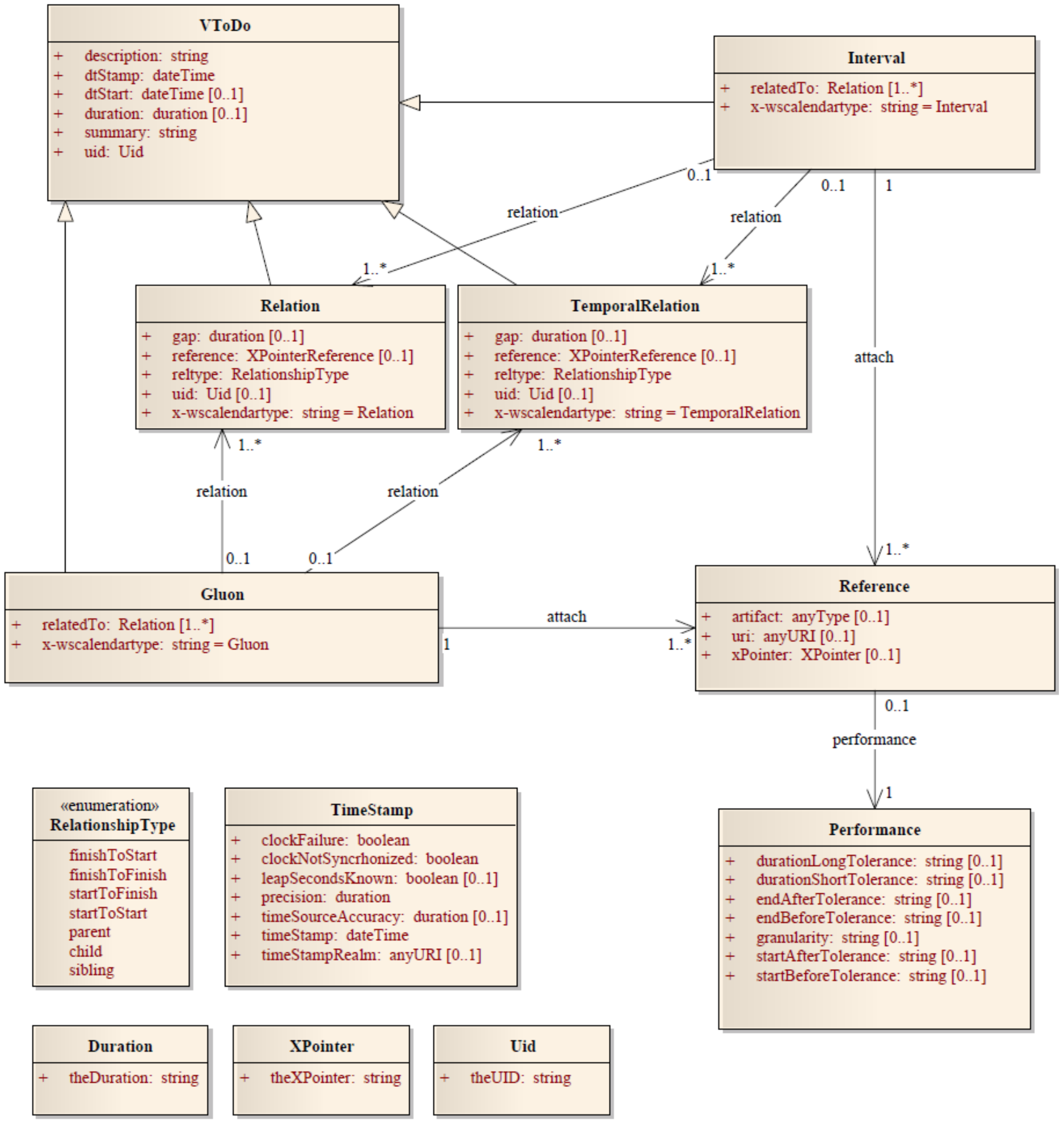

VToDo
+ description: string
+ dtStamp: dateTime
+ dtStart: dateTime [0..1]
+ duration: duration [0..1]
+ summary: string
+ uid: Uid
Interval
+ relatedTo: Relation [1..*]
+ x-wscalendartype: string = Interval
Relation
+ gap: duration [0..1]
+ reference: XPointerReference [0..1]
+ reltype: RelationshipType
+ uid: Uid [0..1]
+ x-wscalendartype: string = Relation
TemporalRelation
+ gap: duration [0..1]
+ reference: XPointerReference [0..1]
+ reltype: RelationshipType
+ uid: Uid [0..1]
+ x-wscalendartype: string = TemporalRelation
Gluon
+ relatedTo: Relation [1..*]
+ x-wscalendartype: string = Gluon
Reference
+ artifact: anyType [0..1]
+ uri: anyURI [0..1]
+ xPointer: XPointer [0..1]
Performance
+ durationLongTolerance: string [0..1]
+ durationShortTolerance: string [0..1]
+ endAfterTolerance: string [0..1]
+ endBeforeTolerance: string [0..1]
+ granularity: string [0..1]
+ startAfterTolerance: string [0..1]
+ startBeforeTolerance: string [0..1]
«enumeration»
RelationshipType
finishToStart
finishToFinish
startToFinish
startToStart
parent
child
sibling
TimeStamp
+ clockFailure: boolean
+ clockNotSyncrhonized: boolean
+ leapSecondsKnown: boolean [0..1]
+ precision: duration
+ timeSourceAccuracy: duration [0..1]
+ timeStamp: dateTime
+ timeStampRealm: anyURI [0..1]
Duration
+ theDuration: string
XPointer
+ theXPointer: string
Uid
+ theUID: string
relation
0..1
1..*
relation
0..1
1..*
1
attach
1..*
relation
1..*
0..1
relation
1..*
0..1
attach
1
1..*
0..1
performance
1

Figure 2: Implementation Model for WS-Calendar

7 Time Stamps

Time stamps are used everywhere in inter-domain service performance analysis and have particular use in smart grids to support event forensics. Time stamps are often assembled and collated from events across multiple time zones and from multiple systems.

Different systems may track time and therefore record events with different levels of Tolerance. It is not unusual for a time stamp from a domain with a low Tolerance to appear to have occurred after events from a domain with high-Tolerance time-stamps that it caused. A fully qualified time-stamp includes the granularity measure.

Table 7-1: Aspects of Time Stamps

Time Stamp Element	Definition (Normative)	Note (Non-Normative)
timestamp	WS-Calendar:time A fully qualified date and time of event. Mandatory.	May include two objects as defined above.
precision	A Duration defining the accuracy of the TimeStamp value. Mandatory.	Identifies whether one hour Interval is indeed one hour or plus or minus some number of milliseconds, seconds and minutes.
timeStampRealm	Of type Uri, shall identify the system where the TimeStamp value originated. The value of this element shall be set by: • The component at the realm border in a particular inter-domain interaction or, • By any component able to accurately set it within a system or sub-system. In the latter case, nothing prevents the component at the realm border to overwrite it without any notice. Optional.	A set of points originating from the same realm are reasonably synchronized. Within a realm, one can assume that time-stamped objects sorted by time are in the order of their occurrence. Between realms, this assumption is rebuttable. A system border is crossed in an interaction when the 2 communication partners are not synchronized based on the same time source. See the example below for more information.
leapSecondsKnown	Xs:bool If True, shall indicate that the TimeStamp value takes into account all leap seconds occurred. Otherwise False. Optional.	Indicates that the time source of the sending device support leap seconds adjustments.

Time Stamp Element	Definition (Normative)	Note (Non-Normative)
clockFailure	xs:bool If True, shall indicate a failure on the time source preventing the TimeStamp value issuer from setting accurate timestamps. Otherwise False. Mandatory.	Indicates that the time source of the sending device is unreliable. The timestamp should be ignored.
clockNotSynchronized	xs:bool If True, shall indicate the time source of the TimeStamp value issuer is not synchronized correctly, putting in doubt the accuracy of the timestamp. Mandatory.	Indicates that the time source of the sending device is not synchronized with the external UTC time source.
timeSourceAccuracy	A Duration defining the accuracy of the time source used in the TimeStampRealm system. Optional.	Represents the time accuracy class of the time source of the sending device relative to the external UTC time source.

7.1.1 Time Stamp Realm Discussion

Within a single system, or synchronized system of systems, one can sort the temporal order of event by sorting them by TimeStamp. Determining the order of events is the first step of event forensics. This assumption does not apply when events are gathered across systems.

Different systems may not have synchronized time, or may synchronize time against different sources. This means different system clocks may drift apart. It may be that a later timestamp from one system occurred before an earlier timestamp in another. As this drift is unknown, it cannot be automatically corrected for without additional information.

The TimeStampRealm element identifies which system created an event time-stamp. The TimeStampRealm identifies a source system in inter-domain interactions (a system of systems). For example: http://SystemA.com and http://SystemB.com identify 2 systems. This example assumes SystemA and SystemB do not have a common time source.

The TimeStampRealm can also be used to identify sub-systems in intra-domain interactions (sub-systems of a system). For example: http://SystemA.com/SubSystem1 and http://SystemA.com/SubSystem2 identify 2 subsystems of the same higher level system. In case the upper level SystemA does not have a global time source for synchronizing all of its sub-system, it can be useful to identify sub-systems in such a way.

8 Calendar Services

The Service interactions are built upon and make the same assumptions about strUTCure as the CalDAV protocol defined in **[RFC4791]** and related specifications. It does NOT require nor assume the WebDAV nor CalDAV protocol but does make use of some of the same elements and strUTCures in the CalDAV XML namespace.

Calendar resources, for example events and tasks are stored as named resources (files) inside special collections (folders) known as "**Calendar Collections**".

These services can be looked upon as a layer built on top of CalDAV and defines the basic operations which allow creation, retrieval, update and deletion. In addition, query, and free-busy operations are defined to allow efficient, partial retrieval of calendar data.

These services assume a degree of conformity with CalDAV is established such that services built in that manner do not have a significant mismatch. It is assumed that some WS-Calendar services will be built without any CalDAV support.

8.1 Overview of the protocol

The protocol is an HTTP based RESTfull protocol using a limited set of methods. Each request may be followed by a response containing status information.

The following methods are specified in the protocol description, PUT, POST, GET, DELETE. To avoid various issues with certain methods being blocked clients may use the X-HTTP-Method-Override: header to specify the intended operation. Servers SHOULD behave as if the named method was used.

```
POST /user/fred/calendar/ HTTP/1.1
...
X-HTTP-Method-Override: PUT
Properties
```

A service or resource will have a number of properties which describe the current state of that service or resource. These properties are accessed through a GET on the target resource or service with an ACCEPT header specifying application/xrd+xml. See Section 8.1.3.6

The following operations are defined by this specification:

- Retrieval and update of service and resource properties
- Creation of a calendar object
- Retrieval of a calendar object
- Update of a calendar object
- Deletion of a calendar object
- Query
- Free-busy query

8.1.1 Calendar Object Resources

The same restrictions apply to Calendar Object Resources as specified in CalDAV **[RFC4791]** section 4.2. An additional constraint for CalWS is that no timezone specifications are transferred.

8.1.2 Timezone information

It is assumed that the client and server each have access to a full set of up to date timezone information. Timezones will be referenced by a timezone identifier from the full set of Olson data together with a set of well-known aliases defined **[TZDB]**. CalWS services may advertise themselves as timezone servers through the server properties object.

8.1.3 Issues not addressed by this specification.

A number of issues are not addressed by this version of the specification, either because they should be addressed elsewhere or will be addressed at some later date.

8.1.3.1 Access Control

It is assumed that the targeted server will set an appropriate level of access based on authentication. This specification will not attempt to address the issues of sharing or Access Control Lists (ACLs).

8.1.3.2 Provisioning

The protocol will not provide any explicit provisioning operations. If it is possible to authenticate or address a principal's calendar resources then they MUST be automatically created if necessary or appropriate

8.1.3.3 Copy/Move

These operations are not yet defined for this version of the CalWS protocol. Both operations raise a number of issues. In particular implementing a move operation through a series of retrievals, insertions and deletions may cause undesirable side-effects. Both these operations will be defined in a later version of this specification.

8.1.3.4 Creating Collections

We will not address the issue of creating collections within the address space. The initial set is created by provisioning.

8.1.3.5 Retrieving collections

This operation is currently undefined. A GET on a collection may fail or return a complete calendar object representing the collection.

8.1.3.6 Setting service and resource properties.

These operations are not defined in this version of the specification. In the future it will be possible to define or set the properties for the service or resources within the service.

8.1.4 CalWS Glossary

8.1.4.1 Hrefs

An href is a URI reference to a resource, for example

```
"http://example.org/user/fred/calendar/event1.ics".
```

The URL above reflects a possible strUTCure for a calendar server. All URLs should be absolute or path-absolute following the rules defined in RFC4918 Section 8.3.

8.1.4.2 Calendar Object Resource

A calendar object resource is an event, meeting or a task. Attachments are resources but NOT calendar object resources. An event or task with overrides is a single calendar resource entity.

8.1.4.3 Calendar Collection

A folder only allowed to contain calendar object resources.

8.1.4.4 Scheduling Calendar Collection

A folder only allowed to contain calendar resources which is also used for scheduling operations. Scheduling events placed in such a collection will trigger implicit scheduling activity on the server.

8.1.4.5 Principal Home

The collection under which all the resources for a given principal are stored. For example, for principal "fred" the principal home might be "/user/fred/"

8.2 Error conditions

Each operation on the calendar system has a number of pre-conditions and post-conditions that apply.

A "precondition" for a method describes the state of the server that must be true for that method to be performed. A "post-condition" of a method describes the state of the server that must be true after that method has been completed. Any violation of these conditions will result in an error response in the form of a CalWS XML error element containing the violated condition and an optional description. \

Each method specification defines the preconditions that must be satisfied before the method can succeed. A number of post-conditions are generally specified which define the state that must exist after the execution of the operation. Preconditions and post-conditions are defined as error elements in the CalWS XML namespace.

8.2.1 Example: error with CalDAV error condition

```
<?xml version="1.0" encoding="utf-8"
      xmlns:CW="Error! Reference source not found.""
      xmlns:C="urn:ietf:params:xml:ns:caldav" ?>
<CW:error>
  <C:supported-filter>
    <C:prop-filter name="X-ABC-GUID"/>
  </C:supported-filter>
  <CW:description>Unknown property </CW:description>
</CW:error>
```

9 Properties and link relations

9.1 Property and relation-type URIs

In the XRD entity returned properties and related services and entities are defined by absolute URIs which correspond to the extended relation type defined in **[web linking]** Section 4.2. These URIs do NOT correspond to any real entity on the server and clients should not attempt to retrieve any data at that target.

Certain of these property URIs correspond to CalDAV preconditions. Each URL is prefixed by the CalWS relations and properties namespace http://docs.oasis-open.org/ns/wscal/calws. Those properties which correspond to CalDAV properties have the additional path element "caldav/", for example

```
http://docs.oasis-open.org/ns/wscal/calws/caldav/supported-calendar-data
```

corresponds to

```
CalDAV:supported-calendar-data
```

In addition to those CalDAV properties, the CalWS specification defines a number of other properties and link relations with the URI prefix of http://docs.oasis-open.org/ns/wscal/calws**.**

9.2 supported-features property.

http://docs.oasis-open.org/ns/wscal/calws/supported-features

This property defines the features supported by the target. All resources contained and managed by the service should return this property. The value is a comma separated list containing one or more of the following

- calendar-access - the service supports all MUST requirements in this specification

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/supported-features"
          >calendar-access</Property>
```

9.3 max-attendees-per-instance

http://docs.oasis-open.org/ns/wscal/calws/max-attendees-per-instance

Defines the maximum number of attendees allowed per event or task.

9.4 max-date-time

http://docs.oasis-open.org/ns/wscal/calws/max-date-time

Defines the maximum date/time allowed on an event or task

9.5 max-instances

http://docs.oasis-open.org/ns/wscal/calws/max-instances

Defines the maximum number of instances allowed per event or task

9.6 max-resource-size

http://docs.oasis-open.org/ns/wscal/calws/max-resource-size

Provides a numeric value indicating the maximum size of a resource in octets that the server is willing to accept when a calendar object resource is stored in a calendar collection.

9.7 min-date-time

http://docs.oasis-open.org/ns/wscal/calws/min-date-time

Provides a DATE-TIME value indicating the earliest date and time (in UTC) that the server is willing to accept for any DATE or DATE-TIME value in a calendar object resource stored in a calendar collection.

9.8 description

http://docs.oasis-open.org/ns/wscal/calws/description

Provides some descriptive text for the targeted collection.

9.9 timezone-service relation.

http://docs.oasis-open.org/ns/wscal/calws/timezone-service

The location of a timezone service used to retrieve timezone information and specifications. This may be an absolute URL referencing some other service or a relative URL if the current server also provides a timezone service.

```
<Link rel="http://docs.oasis-open.org/ns/wscal/calws/calws/timezone-service"
           href="http://example.com/tz" />
```

9.10 principal-home relation.

http://docs.oasis-open.org/ns/wscal/calws/principal-home

Provides the URL to the user home for the currently authenticated principal.

```
<Link rel="http://docs.oasis-open.org/ns/wscal/calws/principal-home"
           href="http://example.com/user/fred" />
```

9.11 current-principal-freebusy relation.

http://docs.oasis-open.org/ns/wscal/calws/current-principal-freebusy

Provides the URL to use as a target for freebusy requests for the current authenticated principal.

```
<Link rel="http://docs.oasis-open.org/ns/wscal/calws/current-principal-freebusy"
           href="http://example.com/freebusy/user/fred" />
```

9.12 principal-freebusy relation.

http://docs.oasis-open.org/ns/wscal/calws/principal-freebusy

Provides the URL to use as a target for freebusy requests for a different principal.

```
<Link rel="http://docs.oasis-open.org/ns/wscal/calws/principal-freebusy"
           href="http://example.com/freebusy" />
```

9.13 child-collection relation.

http://docs.oasis-open.org/ns/wscal/calws/child-collection

Provides information about a child collections for the target. The href attribute gives the URI of the collection. The element should only have CalWS child elements giving the type of the collection, that is the CalWS:collection link property and the CalWS-calendar-collection link property. This allows clients to determine the strUTCure of a hierarchical system by targeting each of the child collections in turn.

The xrd:title child element of the link element provides a description for the child-collection.

```
<Link rel="http://http://docs.oasis-open.org/ns/wscal/calws/child-collection"
      href="http://example.com/calws/user/fred/calendar">
  <Title xml:lang="en">Calendar</Title>
  <Property type="http://docs.oasis-open.org/ns/wscal/calws/collection"
            xsi:nil="true" />
  <Property type="http://docs.oasis-open.org/ns/wscal/calws/calendar-
collection"
            xsi:nil="true" />
</Link>
```

9.14 created link property

http://docs.oasis-open.org/ns/wscal/calws/created

Appears within a link relation describing collections or entities. The value is a date-time as defined in **RFC3339** Section 5.6

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/created"
          >1985-04-12T23:20:50.52Z</Property>
```

9.15 last-modified property

http://docs.oasis-open.org/ns/wscal/calws/last-modified

Appears within an xrd object describing collections or entities. The value is the same format as would appear in the Last-Modified header and is defined in **[RFC2616],** Section 3.3.1

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/last-modified"
          >Mon, 12 Jan 1998 09:25:56 GMT</Property>
```

9.16 displayname property

http://docs.oasis-open.org/ns/wscal/calws/displayname

Appears within an xrd object describing collections or entities. The value is a localized name for the entity or collection.

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/displayname"
          >My Calendar</Property>
```

9.17 timezone property

http://docs.oasis-open.org/ns/wscal/calws/timezone

Appears within an xrd object describing collections. The value is a text timezone identifier.

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/timezone"
          >America/New_York</Property>
```

9.18 owner property

http://docs.oasis-open.org/ns/wscal/calws/owner

Appears within an xrd object describing collections or entities. The value is a server specific uri.

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/owner"
          >/principals/users/mike</Property>
```

9.19 collection link property

http://docs.oasis-open.org/ns/wscal/calws/collection

Appears within a link relation describing collections or entities. The property takes no value and indicates that this child element is a collection.

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/collection"
          xsi:nil="true" />
```

9.20 calendar-collection link property

http://docs.oasis-open.org/ns/wscal/calws/calendar-collection

Appears within a link relation describing collections or entities. The property takes no value and indicates that this child element is a calendar collection.

```
<Property type="http://docs.oasis-open.org/ns/wscal/calws/calendar-collection"
           xsi:nil="true" />
```

9.21 CalWS:privilege-set XML element

http://docs.oasis-open.org/ns/wscal/calws:privilege-set

Appears within a link relation describing collections or entities and specifies the set of privileges allowed to the current authenticated principal for that collection or entity.

```
<!ELEMENT calws:privilege-set (calws:privilege*)>
<!ELEMENT calws:privilege ANY>
```

Each privilege element defines a privilege or access right. The following set is currently defined

- CalWS: Read - current principal has read access
- CalWS: Write - current principal has write access

```
<calWS:privilege-set>
  <calWS:privilege><calWS:read></calWS:privilege>
  <calWS:privilege><calWS:write></calWS:privilege>
</calWS:privilege-set>
```

10 Retrieving Collection and Service Properties

Properties, related services and locations are obtained from the service or from service resources in the form of an XRD document as defined by [XRD-1.0].

Given the URL of a CalWS service a client retrieves the service XRD document through a GET on the service URL with an ACCEPT header specifying application/xrd+xml.

Retrieving resource properties is identical to obtaining service properties, that is, execute a GET on the target URL with an ACCEPT header specifying application/xrd+xml.

The service properties define the global limits and defaults. Any properties defined on collections within the service hierarchy override those service defaults. The service may choose to prevent such overriding of defaults and limits when appropriate.

10.1 Request parameters

- None

10.2 Responses:

- 200: OK
- 403: Forbidden
- 404: Not found

10.3 Example - retrieving server properties:

```
>>Request

GET / HTTP/1.1
Host: example.com
ACCEPT:application/xrd+xml

>>Response
<XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <Expires>1970-01-01T00:00:00Z</Expires>
   <Subject>http://example.com/calws</Subject>
   <Property type="http://docs.oasis-open.org/ns/wscal/calws/created"
         >1970-01-01</Property>

   <Link rel="http://docs.oasis-open.org/ns/wscal/calws/timezone-service"
         href="http://example.com/tz" />

   <calWS:privilege-set>
   <calWS:privilege><calWS:read></calWS:privilege>
   </calWS:privilege-set>

   <Link rel="http://docs.oasis-open.org/ns/wscal/calws/principal-home"
         type="collection"
         href="http://example.com/calws/user/fred">
   <Title xml:lang="en">Fred's calendar home</Title>
   </Link>

   <Link rel="http://docs.oasis-open.org/ns/wscal/calws/child-collection"
         type="calendar,scheduling"
         href="http://example.com/calws/user/fred/calendar">
   <Title xml:lang="en">Calendar</Title>
   </Link>

   <Property type="http://docs.oasis-open.org/ns/wscal/calws/max-instances"
          >1000</Property>

   <Property type="http://docs.oasis-open.org/ns/wscal/calws/max-attendees-
per-instance"
          >100</Property>

</XRD>
```

11 Creating Calendar Object Resources

Creating calendar object resources is carried out by a POST on the parent collection. The body of the request will contain the resource being created. The request parameter "action=create" indicates this POST is a create. The location header of the response gives the URL of the newly created object.

11.1 Request parameters

- action=create

11.2 Responses:

- 201: created
- 403: Forbidden - no access

11.3 Preconditions for Calendar Object Creation

- **CalWS:target-exists**: The target of a PUT must exist. Use POST to create entities and PUT to update them.
- **CalWS:not-calendar-data:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST be a supported media type (i.e., iCalendar) for calendar object resources;
- **CalWS:invalid-calendar-data:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST be valid data for the media type being specified (i.e., MUST contain valid iCalendar data);
- **CalWS:invalid-calendar-object-resource:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST obey all restrictions specified in Calendar Object Resources (e.g., calendar object resources MUST NOT contain more than one type of calendar component, calendar object resources MUST NOT specify the iCalendar METHOD property, etc.);
- **CalWS:unsupported-calendar-component:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST contain a type of calendar component that is supported in the targeted calendar collection;
- **CalWS:uid-conflict:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST NOT specify an iCalendar UID property value already in use in the targeted calendar collection or overwrite an existing calendar object resource with one that has a different UID property value. Servers SHOULD report the URL of the resource that is already making use of the same UID property value in the CalWS:href element
  <!ELEMENT uid-conflict (CalWS:href)>
- **CalWS:invalid-calendar-collection-location:** In a COPY or MOVE request, when the Request-URI is a calendar collection, the Destination-URI MUST identify a location where a calendar collection can be created;
- **CalWS:exceeds-max-resource-size:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have an octet size less than or equal to the value of the CalDAV:max-resource-size property value on the calendar collection where the resource will be stored;
- **CalWS:before-min-date-time:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have all of its iCalendar DATE or DATE-TIME property values (for each

recurring instance) greater than or equal to the value of the CalDAV:min- date-time property value on the calendar collection where the resource will be stored;

- **CalWS:after-max-date-time:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have all of its iCalendar DATE or DATE-TIME property values (for each recurring instance) less than the value of the CalDAV:max-date-time property value on the calendar collection where the resource will be stored;
- **CalWS:too-many-instances:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST generate a number of recurring instances less than or equal to the value of the CalDAV: max-instances property value on the calendar collection where the resource will be stored;
- **CalWS:too-many-attendees-per-instance:** The resource submitted in the PUT request, or targeted by a COPY or MOVE request, MUST have a number of ATTENDEE properties on any one instance less than or equal to the value of the CalDAV:max-attendees-per-instance property value on the calendar collection where the resource will be stored;

11.4 Example - successful POST:

```
>>Request

POST /user/fred/calendar/?action=create HTTP/1.1
Host: example.com
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: ?

<?xml version="1.0" encoding="utf-8" ?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
  <vcalendar>
  ...
  </vcalendar>
</icalendar>

>>Response

HTTP/1.1 201 Created
Location: http://example.com/user/fred/calendar/event1.ics
```

11.5 Example - unsuccessful POST:

```
>>Request

POST /user/fred/readcalendar/?action=create HTTP/1.1
Host: example.com
Content-Type: text/text; charset="utf-8"
Content-Length: ?

This is not an xml calendar object

>>Response

HTTP/1.1 403 Forbidden
<?xml version="1.0" encoding="utf-8"
      xmlns:D="DAV:"
      xmlns:C="urn:ietf:params:xml:ns:caldav" ?>
<D:error>
      <C:supported-calendar-data/>
      <D:description>Not an icalendar object</D:description>
</D:error>
```

12 Retrieving resources

A simple GET on the href will return a named resource. If that resource is a recurring event or task with overrides, the entire set will be returned. The desired format is specified in the ACCEPT header. The default form is application/xml+calendar

12.1 Request parameters

- none

12.2 Responses:

- 200: OK
- 403: Forbidden - no access
- 406 The requested format specified in the accept header is not supported.

12.3 Example - successful fetch:

```
>>Request

GET /user/fred/calendar/event1.ics HTTP/1.1
Host: example.com

>>Response

HTTP/1.1 200 OK
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: ?

<?xml version="1.0" encoding="utf-8" ?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
  <vcalendar>
   ...
  </vcalendar>
</icalendar>
```

12.4 Example - unsuccessful fetch:

```
>>Request

PUT /user/fred/calendar/noevent1.ics HTTP/1.1
Host: example.com

>>Response

HTTP/1.1 404 Not found
```

13 Updating resources

Resources are updated with the PUT method targeted at the resource href. The body of the request contains a complete new resource which effectively replaces the targeted resource. To allow for optimistic locking of the resource use the if-match header.

When updating a recurring event all overrides and master must be supplied as part of the content.

Preconditions as specified in Section 11.3 are applicable.

13.1 Responses:

- 200: OK
- 304: Not modified - entity was modified by some other request
- 403: Forbidden - no access, does not exist etc. See error response

Example 13—1: Successful Update

```
>>Request

PUT /user/fred/calendar/event1.ics HTTP/1.1
Host: example.com
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: ?

<?xml version="1.0" encoding="utf-8" ?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
  <vcalendar>
     ...
  </vcalendar>
</icalendar>

>>Response

HTTP/1.1 200 OK
```

Example 13—2: Unsuccessful Update

```
>>Request

PUT /user/fred/readcalendar/event1.ics HTTP/1.1
Host: example.com
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: ?

<?xml version="1.0" encoding="utf-8" ?>
<icalendar xmlns="urn:ietf:params:xml:ns:icalendar-2.0">
  <vcalendar>
    ...
  </vcalendar>
</icalendar>

>>Response

HTTP/1.1 403 Forbidden
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx
```

```
<?xml version="1.0" encoding="utf-8"
      xmlns:D="DAV:"
      xmlns:CW=" http://docs.oasis-open.org/ws-calendar/CalWS" ?>
<CW:error>
  <CW:target-exists/>
  <CW:description>Target of update must exist</C:description>
</CW:error>
```

14 Deletion of resources

Delete is defined in **[RFC 2616]** Section 9.7. In addition to conditions defined in that specification, servers must remove any references from the deleted resource to other resources. Resources are deleted with the DELETE method targeted at the resource URL. After a successful completion of a deletion a GET on that URL must result in a 404 - Not Found status.

14.1 Delete for Collections

Delete for collections may or may not be supported by the server. Certain collections are considered undeletable. On a successful deletion of a collection all contained resources to any depth must also be deleted.

14.2 Responses:

- 200: OK
- 403: Forbidden - no access
- 404: Not Found

15 Querying calendar resources

Querying provides a mechanism by which information can be obtained from the service through possibly complex queries. A list of icalendar properties can be specified to limit the amount of information returned to the client. A query takes the parts

- Limitations on the data returned
- Selection of the data
- Optional timezone id for floating time calculations.

The current specification uses CalDAV multiget and calendar-query XML bodies as specified in **[RFC 4791]** with certain limitations and differences.

1. The POST method is used for all requests, the action being identified by the outer element.
2. While CalDAV servers generally only support [RFC 5545] and assume that as the default, the delivery format for CalWS will, by default, be [draft-xcal].
3. The CalDAV query allows the specification of a number of DAV properties. Specification of these properties, with the exception of DAV:getetag, is considered an error in CalWS.
4. The CalDAV:propnames element is invalid

With those differences, the CalDAV specification is the normative reference for this operation.

15.1 Limiting data returned

This is achieved by specifying one of the following

- CalDAV:allprop return all properties (some properties are specified as not being part of the allprop set so are not returned)
- CalDAV:prop An element which contains a list of properties to be returned . May only contain DAV:getetag and CalDAV:calendar-data

Of particular interest, and complexity, is the calendar-data property which can contain a time range to limit the range of recurrences returned and/or a list of calendar properties to return.

15.2 Pre/postconditions for calendar queries

The preconditions as defined in in **[RFC 4791]** Section 7.8 apply here. CalDav errors may be reported by the service when preconditions or postconditions are violated.

15.3 Example: time range limited retrieval

This example shows the time-range limited retrieval from a calendar which results in 2 events, one a recurring event and one a simple non-recurring event.

```
>> Request <<

POST /user/fred/calendar/ HTTP/1.1
Host: calws.example.com
Depth: 1
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<C:calendar-query xmlns:D="DAV:"
              xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:prop>
```

```
    <D:getetag/>
    <C:calendar-data content-type="application/xml+calendar" >
      <C:comp name="VCALENDAR">
        <C:prop name="VERSION"/>
        <C:comp name="VEVENT">
          <C:prop name="SUMMARY"/>
          <C:prop name="UID"/>
          <C:prop name="DTSTART"/>
          <C:prop name="DTEND"/>
          <C:prop name="DURATION"/>
          <C:prop name="RRULE"/>
          <C:prop name="RDATE"/>
          <C:prop name="EXRULE"/>
          <C:prop name="EXDATE"/>
          <C:prop name="RECURRENCE-ID"/>
        </C:comp>
      </C:comp>
    </C:calendar-data>
  </D:prop>
  <C:filter>
    <C:comp-filter name="VCALENDAR">
      <C:comp-filter name="VEVENT">
        <C:time-range start="20060104T000000Z"
                      end="20060105T000000Z"/>
      </C:comp-filter>
    </C:comp-filter>
  </C:filter>
</C:calendar-query>

>> Response <<

HTTP/1.1 207 Multi-Status
Date: Sat, 11 Nov 2006 09:32:12 GMT
Content-Type: application/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:"
               xmlns:C="urn:ietf:params:xml:ns:caldav">
  <D:response>
    <D:href>http://cal.example.com/bernard/work/abcd2.ics</D:href>
    <D:propstat>
      <D:prop>
        <D:getetag>"fffff-abcd2"</D:getetag>
        <C:calendar-data content-type="application/xml+calendar" >
          <xc:icalendar
             xmlns:xc="urn:ietf:params:xml:ns:icalendar-2.0">
   <xc:vcalendar>
     <xc:properties>
      <xc:calscale><text>GREGORIAN</text></xc:calscale>
      <xc:prodid>
       <xc:text>-//Example Inc.//Example Calendar//EN</xc:text>
      </xc:prodid>
      <xc:version><xc:text>2.0</xc:text></xc:version>
     </xc:properties>
     <xc:components>
      <xc:vevent>
       <xc:properties>
        <xc:dtstart>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060102T120000</xc:date-time>
        </xc:dtstart>
        <xc:duration><xc:duration>PT1H</xc:duration></xc:duration>
        <xc:summary>
         <xc:text>Event #2</xc:text>
        </xc:summary>
        <xc:uid>
         <xc:text>00959BC664CA650E933C892C@example.com</xc:text>
        </xc:uid>
        <xc:rrule>
          <xc:recur>
            <xc:freq>DAILY</xc:freq>
            <xc:count>5</xc:count>
          </xc:recur>
        </xc:rrule>
       </xc:properties>
      </xc:vevent>

      <xc:vevent>
       <xc:properties>
        <xc:dtstart>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060104T140000</xc:date-time>
        </xc:dtstart>
        <xc:duration><xc:duration>PT1H</xc:duration></xc:duration>
        <xc:summary>
         <xc:text>Event #2 bis</xc:text>
        </xc:summary>
        <xc:uid>
         <xc:text>00959BC664CA650E933C892C@example.com</xc:text>
        </xc:uid>
        <xc:recurrence-id>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060104T120000</xc:date-time>
        </xc:recurrence-id>
        <xc:rrule>
          <xc:recur>
            <xc:freq>DAILY</xc:freq>
            <xc:count>5</xc:count>
          </xc:recur>
        </xc:rrule>
       </xc:properties>
      </xc:vevent>

      <xc:vevent>
       <xc:properties>
        <xc:dtstart>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060106T140000</xc:date-time>
        </xc:dtstart>
        <xc:duration><xc:duration>PT1H</xc:duration></xc:duration>
        <xc:summary>
         <xc:text>Event #2 bis bis</xc:text>
        </xc:summary>
        <xc:uid>
         <xc:text>00959BC664CA650E933C892C@example.com</xc:text>
        </xc:uid>
        <xc:recurrence-id>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060106T120000</xc:date-time>
        </xc:recurrence-id>
        <xc:rrule>
          <xc:recur>
            <xc:freq>DAILY</xc:freq>
            <xc:count>5</xc:count>
          </xc:recur>
        </xc:rrule>
       </xc:properties>
      </xc:vevent>
     </xc:components>
    </xc:vcalendar>
   </xc:icalendar>
           </C:calendar-data>
         </D:prop>
         <D:status>HTTP/1.1 200 OK</D:status>
       </D:propstat>
     </D:response>
     <D:response>
       <D:href>http://cal.example.com/bernard/work/abcd3.ics</D:href>
       <D:propstat>
         <D:prop>
           <D:getetag>"fffff-abcd3"</D:getetag>
           <C:calendar-data content-type="application/xml+calendar" >
             <xcal:icalendar
                xmlns:xc="urn:ietf:params:xml:ns:icalendar-2.0">
    <xc:vcalendar>
     <xc:properties>
      <xc:calscale><text>GREGORIAN</text></xc:calscale>
      <xc:prodid>
       <xc:text>-//Example Inc.//Example Calendar//EN</xc:text>
      </xc:prodid>
      <xc:version><xc:text>2.0</xc:text></xc:version>
     </xc:properties>
     <xc:components>
      <xc:vevent>
       <xc:properties>
        <xc:dtstart>
          <xc:parameters>
            <xc:tzid>US/Eastern<xc:tzid>
          <xc:parameters>
          <xc:date-time>20060104T100000</xc:date-time>
        </xc:dtstart>
        <xc:duration><xc:duration>PT1H</xc:duration></xc:duration>
        <xc:summary>
         <xc:text>Event #3</xc:text>
        </xc:summary>
        <xc:uid>
         <xc:text>DC6C50A017428C5216A2F1CD@example.com</xc:text>
        </xc:uid>
        <xc:rrule>
          <xc:recur>
            <xc:freq>DAILY</xc:freq>
            <xc:count>5</xc:count>
          </xc:recur>
        </xc:rrule>
       </xc:properties>
      </xc:vevent>
     </xc:components>
    </xc:vcalendar>
   </xc:icalendar>
        </C:calendar-data>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

16 Free-busy queries

Free-busy queries are used to obtain free-busy information for a calendar-collection or principals. The result contains information only for events to which the current principal has sufficient access.

When targeted at a calendar collection the result is based only on the calendaring entities contained in that collection. When targeted at a principal free-busy URL the result will be based on all information which affect the principals free-busy status, for example availability.

The possible targets are:

- A calendar collection URL
- The XRD link with relation CalWS/current-principal-freebusy
- The XRD link with relation CalWS/principal-freebusy with a principal given in the request.

The query follows the specification defined in **[FreeBusy Read URL]** with certain limitations. As an authenticated user to the CalWS service scheduling read-freebusy privileges must have been granted. As an unauthenticated user equivalent access must have been granted to unauthenticated access.

Freebusy information is returned by default as xcalendar vfreebusy components, as defined by **[draft-xcal]**. Such a component is not meant to conform to the requirements of VFREEBUSY components in **[RFC 5546]**. The VFREEBUSY component SHOULD conform to section "4.6.4 Free/Busy Component" of **[RFC 5545]**. A client SHOULD ignore the ORGANIZER field..

Since a Freebusy query can only refer to a single user, a client will already know how to match the result component to a user. A server MUST only return a single vfreebusy component.

16.1 ACCEPT header

The Accept header is used to specify the format for the returned data. In the absence of a header the data should be returned as specified in **[draft-xcal]**, that is, as if the following had been specified

```
ACCEPT: application/xml+calendar
```

16.2 URL Query Parameters

None of these parameters are required except for the conditions noted below. Appropriate defaults will be supplied by the server.

16.2.1 start

Default: The default value is left up to the server. It may be the current day, start of the current month, etc.

Description: Specifies the start date for the Freebusy data. The server is free to ignore this value and return data in any time range. The client must check the data for the returned time range.

Format:A profile of an **[RFC3339]** Date/Time. Fractional time is not supported. The server MUST support the expanded version e.g.

```
2007-01-02T13:00:00-08:00
```

It is up to the server to interpret local date/times.

Example:

```
2007-02-03T15:30:00-0800
2007-12-01T10:15:00Z
```

Notes: Specifying only a start date/time without specifying an end-date/time or period should be interpreted as in **[RFC 5545]**. The effective period should cover the remainder of that day.

Date-only values are disallowed as the server cannot determine the correct start of the day. Only UTC or date/time with offset values are permitted.

16.2.2 end

Default: Same as start

Description: Specifies the end date for the Freebusy data. The server is free to ignore this value.

Format: Same as start

Example: Same as start

16.2.3 period

Default: The default value is left up to the server. The recommended value is "P42D".

Description: Specifies the amount of Freebusy data to return. A client cannot specify both a period and an end date. Period is relative to the start parameter.

Format: A duration as defined in section 4.3.6 of **[RFC 5545]**

Example:

```
P42D
```

16.2.4 account

Default: none

Description: Specifies the principal when the request is targeted at the XRD CalWS/principal-freebusy. Specification of this parameter is an error otherwise.

Format: Server specific

Example:

```
fred
/principals/users/jim
user1@example.com
```

16.3 URL parameters - notes

The server is free to ignore the start, end and period parameters. It is recommended that the server return at least 6 weeks of data from the current day.

A client MUST check the time range in the VFREEBUSY response as a server may return a different time range than the requested range.

16.4 HTTP Operations

The server SHOULD return an Etag response header for a successful GET request targeting a Freebusy read URL. Clients MAY use the Etag response header value to do subsequent "conditional" GET requests that will avoid re-sending the Freebusy data again if it has not changed.

16.5 Response Codes

Below are the typical status codes returned by a GET request targeting a Free-busy URL. Note that other HTTP status codes not listed here might also be returned by a server.

- 200 OK
- 302 Found
- 400 Start parameter could not be understood / End parameter could not be understood / Period parameter could not be understood

- 401 Unauthorized
- 403 Forbidden
- 404 The data for the requested principal is not currently available, but may be available later.
- 406 The requested format in the accept header is not supported.
- 410 The data for the requested principal is no longer available
- 500 General server error

16.6 Examples

The following are examples of URLs used to retrieve Free-busy data for a user:

```
http://www.example.com/freebusy/user1@example.com?
start=2007-09-01T00:00:00-08:00&end=2007-09-31T00:00:00-08:00

http://www.example.com/freebusy/user1@example.com?
start=2007-09-01T00:00:00-08:00&end=2007-09-31T00:00:00-08:00

http://www.example.com/freebusy/user1@example.com

http://www.example.com/freebusy?user=user%201@example.com&
start=2008-01-01T00:00:00Z&end=2008-12-31T00:00:00Z
```

Some Request/Response Examples:

A URL with no query parameters:

```
>> Request <<
GET /freebusy/bernard/ HTTP/1.1
Host: www.example.com

>> Response <<
HTTP/1.1 200 OK
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: xxxx

<xc:icalendar xmlns:xc="urn:ietf:params:xml:ns:icalendar-2.0">
  <xc:vcalendar>
    <xc:properties>
      <xc:calscale><text>GREGORIAN</text></xc:calscale>
      <xc:prodid>
        <xc:text>-//Example Inc.//Example Calendar//EN</xc:text>
      </xc:prodid>
      <xc:version><xc:text>2.0</xc:text></xc:version>
    </xc:properties>
    <xc:components>
      <xc:vfreebusy>
        <xc:properties>
          <xc:uid>
            <xc:text>76ef34-54a3d2@example.com</xc:text>
          </xc:uid>
          <xc:dtstart>
            <xc:date-time>20060101T000000Z</xc:date-time>
          </xc:dtstart>
          <xc:dtend>
            <xc:date-time>20060108T000000Z</xc:date-time>
          </xc:dtend>
          <xc:dtstamp>
            <xc:date-time>20050530T123421Z</xc:date-time>
          </xc:dtstamp>
          <xc:freebusy>
```

```
            <xc:parameters>
              <xc:fbtype>BUSYTENTATIVE<xc:fbtype>
            <xc:parameters>
            <xc:period>20060102T100000Z/20060102T120000Z</xc:period>
          </xc:freebusy>
          <xc:freebusy>
            <xc:period>20060103T100000Z/20060103T120000Z</xc:period>
          </xc:freebusy>
          <xc:freebusy>
            <xc:period>20060104T100000Z/20060104T120000Z</xc:period>
          </xc:freebusy>
          <xc:freebusy>
            <xc:parameters>
              <xc:fbtype>BUSYUNAVAILABLE<xc:fbtype>
            <xc:parameters>
            <xc:period>20060105T100000Z/20060105T120000Z</xc:period>
          </xc:freebusy>
          <xc:freebusy>
            <xc:period>20060106T100000Z/20060106T120000Z</xc:period>
          </xc:freebusy>
      </xc:vfreebusy>
    </xc:components>
  </xc:vcalendar>
<xc:icalendar>
```

A URL with start and end parameters:

```
>> Request <<
GET /freebusy/user1@example.com?start=2007-09-01T00:00:00-08:00&end=2007-09-
31T00:00:00-08:00
HTTP/1.1
Host: www.example.com

>> Response <<
HTTP/1.1 200 OK
Content-Type: application/xml+calendar; charset="utf-8"
Content-Length: xxxx

<xc:icalendar xmlns:xc="urn:ietf:params:xml:ns:icalendar-2.0">
  <xc:vcalendar>
    <xc:properties>
      <xc:calscale><text>GREGORIAN</text></xc:calscale>
      <xc:prodid>
        <xc:text>-//Example Inc.//Example Calendar//EN</xc:text>
      </xc:prodid>
      <xc:version><xc:text>2.0</xc:text></xc:version>
    </xc:properties>
    <xc:components>
      <xc:vfreebusy>
        <xc:properties>
          <xc:uid>
            <xc:text>76ef34-54a3d2@example.com</xc:text>
          </xc:uid>
          <xc:dtstart>
            <xc:date-time>20070901T000000Z</xc:date-time>
          </xc:dtstart>
          <xc:dtend>
            <xc:date-time>20070931T000000Z</xc:date-time>
          </xc:dtend>
          <xc:dtstamp>
            <xc:date-time>20050530T123421Z</xc:date-time>
          </xc:dtstamp>
          <xc:freebusy>
            <xc:period>20070915T230000Z/20070916T010000Z</xc:period>
          </xc:freebusy>
      </xc:vfreebusy>
    </xc:components>
  </xc:vcalendar>
<xc:icalendar>
```

A URL for which the server does not have any data for that user:

```
>> Request <<
GET /freebusy/user1@example.com?start=2012-12-01T00:00:00-08:00&end=2012-12-
31T00:00:00-08:00
HTTP/1.1
Host: www.example.com

>> Response <<
HTTP/1.1 404 No data
```

17 Conformance and Rules for WS-Calendar and Referencing Specifications

17.1 Introduction

This section specifies conformance related to the semantic model, REST Web Services, and to SOAP Web Services While the semantic model applies to all WS-Calendar implementations; the other conformance statements are relevant only to those using those services.

If the implementer is merely using WS-Calendar 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.

17.2 Semantic Conformance Rules for WS-Calendar

There are five kinds of conformance that must be addressed for WS-Calendar and specifications that reference WS-Calendar.

- 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.

17.2.1 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.[5] 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.

17.2.2 Specific Attribute Inheritance in WS-Calendar

In WS-Calendar the following attributes MUST be inherited in conformance to the Rules (same for Gluons and Intervals):

- dtStart
- dtEnd
- duration
- designatedInterval (Gluon, special upward inheritance rule)
- performance
- performanceInterval

In WS-Calendar the following attributes MUST NOT be inherited

- UID (Gluons and Intervals)
- Temporal Relationships (Intervals)

Some elements of WS-Calendar 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.

17.2.3 Conformance of Intervals in WS-Calendar

17.2.3.1 Intervals

WS-Calendar Intervals SHALL have a Duration.

Intervals MAY have a StartTime.

Intervals SHALL NOT include an END time. If a non-compliant Interval is received with an END time, it may be ignored.

[5] We are assuming here that Sequences can be composed to form new Sequences. This needs detailed discussion as the rules for Designated Intervals cannot easily be applied to a Sequence of Sequences.

17.2.3.2 Other Elements

A performance component SHALL not include Start, Stop, and Duration elements. Two out of the three elements is acceptable, but not three.

In Partitions, the Description, Summary and Priority of each Interval SHALL be excluded.

A Calendar Gluon may have either a dtStart or a dtEnd, but may not have both.

17.2.4 Conformance of Bound Intervals and Sequences in WS-Calendar

Actionable services require Bound Intervals as part of a Bound Sequence. Services may Intervals that arenot 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
- Intervals SHALL have no value assigned for dtEnd[6]
- Within a Sequence at most the Designated Interval may have dtStart and duration with a value specified or inherited.[7]
- Any specification claiming conformance to WS-Calendar 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

17.3 Conformance Rules for REST Web Services

Still to come

17.4 Conformance Rules for SOAP Services

Still to Come

[6] While VTODO objects allow for all three of dtStart, dtEnd, and duration, the scheduling use for automation is simpler if only dtStart and duration are used.

[7] Note that composition of Sequences to create other Sequences raises issues both of inheritance direction and the meaning of subSequences. We suggest an approach of ignoring Designated Intervals with respect to the composed Sequence as simpler than having the new subSequences change form and not be reusable.

17.5 Conformance Rules for Specifications incorporating WS-Calendar

Specifications that incorporate WS-Calendar SHALL specify inheritance rules for use within their specification. These rules SHALL NOT violate override the Proximity, Direction, or Override Rules. If the specification includes covariant elements, those elements SHAL be clearly designated in the specification.

A. Acknowledgements

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

Participants:

Bruce Bartell, Southern California Edison
Brad Benson, Trane
Edward Cazalet, Individual
Toby Considine, University of North Carolina at Chapel Hill
William Cox, Individual
Sharon Dinges, Trane
Craig Gemmill, Tridium, Inc.
Girish Ghatikar, Lawrence Berkeley National Laboratory
Gerald Gray, Southern California Edison
Gale Horst, Electric Power Research Institute (EPRI)
Gershon Janssen, Individual
Ed Koch, Akuacom Inc.
Benoit Lepeuple, LonMark International*
Carl Mattocks, CheckMi*
Robert Old, Siemens AG
Alexander Papaspyrou, Technische Universitat Dortmund
Jeremy J. Roberts, LonMark International
David Thewlis, CalConnect

The Calendaring and Scheduling Consortium (CalConnect) TC-XML committee worked closely with WS-Calendar Technical Committee, bridging to developing IETF standards and contributing the Services definitions that make up Services beginning in Section 8, Calendar Services. The Technical Committee gratefully acknowledges their assistance and cooperation as well. Contributors to TC XML include:

Cyrus Daboo, Apple
Mike Douglas, Rensselaer Polytechnic Institute
Steven Lees, Microsoft
Tong Li, IBM

B. An Introduction to Internet Calendaring

The WS-Calendar Technical Committee thanks CalConnect for contributing this overview of iCalendar and its use.

B.1 icalendar

B.1.1 History

The iCalendar specification was first produced by the IETF in 1998 as RFC 2445 [1]. Since then it has become the dominant standard for calendar data interchange on the internet and between devices (desktop computers, mobile phones etc.). The specification was revised in 2009 as RFC 5545 [4].

Alongside iCalendar is the iTIP specification (RFC 2446 [2] and revised as RFC 5546[5]) that defines how iCalendar is used to carry out scheduling operations (for example, how an organizer can invite attendees to a meeting and receive their replies). This forms the basis for email-based scheduling using iMIP (the specification that describes how to use iTIP with email - RFC 2447 [3]).

iCalendar itself is a text-based data format. However, an XML format is also available, providing a one-to-one mapping to the text format (draft [7]).

iCalendar data files typically have a .ics file name extension. Most desktop calendar clients can import or export iCalendar data, or directly access such data over the Internet using a variety of protocols.

B.1.2 Data model

The iCalendar data format has a well defined data model. "iCalendar objects" encompass a set of "iCalendar components" each of which contains a set of "iCalendar properties" and possibly other sub-components. An iCalendar property consists of a name, a set of optional parameters (specified as "key-value" pairs) and a value.

iCalendar components include:

"VEVENT" which represents an event

"VTODO" which represents a task or to-do

"VJOURNAL" which represents a journal entry

"VFREEBUSY" which represents periods of free or busy time information

"VTIMEZONE" which represents a timezone definition (timezone offset and daylight saving rules)

"VALARM" is currently the only defined sub-component and is used to set alarms or reminders on events or tasks.

Properties include:

"DTSTART" which represents a start time for a component

"DTEND" which represents an end time for a component

"SUMMARY" which represents a title or summary for a component

"RRULE" which can specify rules for repeating events or tasks (for example, every day, every week on Tuesdays, etc.)

"ORGANIZER" which represents the calendar user who is organizing an event or assigning a task

"ATTENDEE" which represents calendar users attending an event or assigned a task

In addition to this data model and the pre-defined properties, the specification defines how all those are used together to define the semantics of calendar objects and scheduling. The semantics are basically a set of rules stating how all the components and properties are used together to ensure that all iCalendar prodUTCs can work together to achieve good interoperability. For example, a rule requires that all events must have one and only one "DTSTART" property. The most important part of the iCalendar specification is the semantics of the calendaring model that it represents. The use of text or XML to encode those is secondary.

B.1.3 Scheduling

The iTIP specification defines how iCalendar objects are exchanged in order to accomplish the key task needed to schedule events or tasks. An example of a simple workflow is as follows:

1. To schedule an event, an organizer creates the iCalendar object representing the event and adds calendar users as attendees.
2. The organizer then sends an iTIP "REQUEST" message to all the attendees.
3. Upon receipt of the scheduling message, each attendee can decide whether they want to attend the meeting or not.
4. Each attendee can then respond back to the organizer using an iTIP "REPLY" message indicating their own attendance status.

iTIP supports other types of scheduling messages, for example, to cancel meetings, add new instances to a repeating meeting, etc.

B.1.4 Extensibility

iCalendar was designed to be extensible, allowing for new components, properties and parameters to be defined as needed. A registry exists to maintain the list of standard extensions with references to their definitions to ensure anyone can use them and work well with others.

B.2 Calendar data access and exchange protocols

B.2.1 Internet Calendar Subscriptions

An Internet calendar subscription is simply an iCalendar data file made available on a web server. Users can use this data in two ways:

– The data can be downloaded from the web server and then imported directly into an iCalendar aware client. This solution works well for calendar data that is not likely to change over time (for example the list of national holidays for the next year).

– Calendar clients that support "direct" subscriptions can use the URL to the calendar data on the web server to download the calendar data themselves. Additionally, the clients can check the web server on a regular basis for updates to the calendar data, and then update their own cached copy of it. This allows calendar data that changes over time to be kept synchronized.

B.2.2 CalDAV

CalDAV is a calendar access protocol and is defined in RFC 4791 [6]. The protocol is based on WebDAV which is an extension to HTTP that provides enhanced capabilities for document management on web servers.

CalDAV is used in a variety of different environments, ranging from very large internet service providers, to large and small corporations or institutions, and to small businesses and individuals.

CalDAV clients include desktop applications, mobile devices and browser-based solutions. It can also be used by "applets", for example, a web page panel that displays a user's upcoming events.

One of the key aspects of CalDAV is its data model. Simply put, it defines a "calendar home" for each calendar user, within which any number of "calendars" can be created. Each "calendar" can contain any number of iCalendar objects representing individual events, tasks or journal entries. This data model ensures that clients and servers can interoperate well.

In addition to providing simple operations to read, write and delete calendar data, CalDAV provides a querying mechanism to allow clients to fetch calendar data matching specific criteria. This is commonly used by clients to do "time-range" queries, i.e., find the set of events that occur within a given start/end time period.

CalDAV also supports access control allowing for features such as delegated calendars and calendar sharing.

CalDAV also specifies how scheduling operations can be done using the protocol. Whilst it uses the semantics of the iTIP protocol, it simplifies the process by allowing simple calendar data write operations to trigger the sending of scheduling messages, and it has the server automatically process the receipt of scheduling messages. Scheduling can be done with other users on the CalDAV server or with calendar users on other systems (via some form of "gateway").

B.2.3 ActiveSync/SyncML

ActiveSync and SyncML are technologies that allow multiple devices to synchronize data with a server, with calendar data being one of the classes of data supported. These have typically been used for low-end and high-end mobile devices.

B.2.4 CalWS

CalWS is a web services calendar access API developed by The Calendaring and Scheduling Consortium and the OASIS organization, to be used as part of the Oasis WS-Calendar standard. It provides an API to access and manipulate calendar data stored on a server. It follows a similar data model to CalDAV and has been designed to co-exist with a CalDAV service offering the same data.

B.2.5 iSchedule

iSchedule is a protocol to allow scheduling between users on different calendaring systems and across different internet domains. It transports iTIP scheduling messages using HTTP between servers. Servers use DNS and various security mechanisms to determine the authenticity of messages received.

It has been specifically designed to be independent of any calendar system in use at the endpoints, so that it is compatible with many different systems. This allows organizations with different calendar systems to exchange scheduling messages with each other, and also allows a single organization with multiple calendar systems (for example due to mergers, or different departmental requirements) to exchange scheduling messages between users of each system.

B.3 References

[1] https://datatracker.ietf.org/doc/rfc2445/ : ‘Internet Calendaring and Scheduling Core Object Specification’

[2] https://datatracker.ietf.org/doc/rfc2446/ :‘iCalendar Transport-Independent Interoperability Protocol’

[3] https://datatracker.ietf.org/doc/rfc2447/ : ‘iCalendar Message-Based Interoperability Protocol’

[4] https://datatracker.ietf.org/doc/rfc5545/ :‘Internet Calendaring and Scheduling Core Object Specification’

[5] https://datatracker.ietf.org/doc/rfc5546/ : ‘iCalendar Transport-Independent Interoperability Protocol’

[6] https://datatracker.ietf.org/doc/rfc4791/ : ‘Calendaring Extensions to WebDAV’

[7] https://datatracker.ietf.org/doc/draft-daboo-et-al-icalendar-in-xml/ : ‘xCal: The XML format for iCalendar’

C. Overview of WS-Calendar, its Antecedents and its Use

iCalendar has long been the predominant message format for an Internet user to send meeting requests and tasks to other Internet users by email. The recipient can respond to the sender easily or counter propose another meeting date/time. iCalendar support is built into all major email systems and email clients. While SMTP is the predominant means to transport iCalendar messages, protocols including WebDAV and SyncML are used to transport collections of iCalendar information. No similar standard for service interactions has achieved similar widespread use.

The Calendar and Scheduling Consortium (CalConnect), working within the IETF, updated the iCalendar standard in the summer of 2009 to support extension ([RFC5545]). In 2010, the same group defined [XCAL], a canonical XML serialization for iCalendar, currently (08/21/2008) on the recommended standards track within the IETF. This specification supports extensions, including handling non-standard, i.e., non-iCalendar, data during message storage and retrieval.

WS-Calendar builds on this work, and consists of extensions to the vocabulary of iCalendar, along with standard services to extend calendaring and scheduling into service interactions. iCalendar consists of a number of fields that support the delivery, update, and synchronization of if calendar messages and a list of components. The components can specify defined relationships between each other.

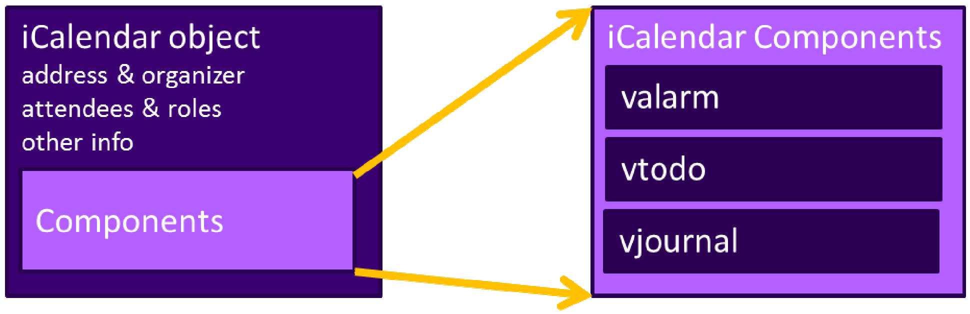


Figure 3: iCalendar overview

WS-Calendar defines the Interval, a profile of the vtodo component requiring only a duration and an artifact to define service delivery and performance. WS-Calendar also defines the CalendarGluon component, a container for holding only a service delivery and performance artifact, to associate with a component or group of components.

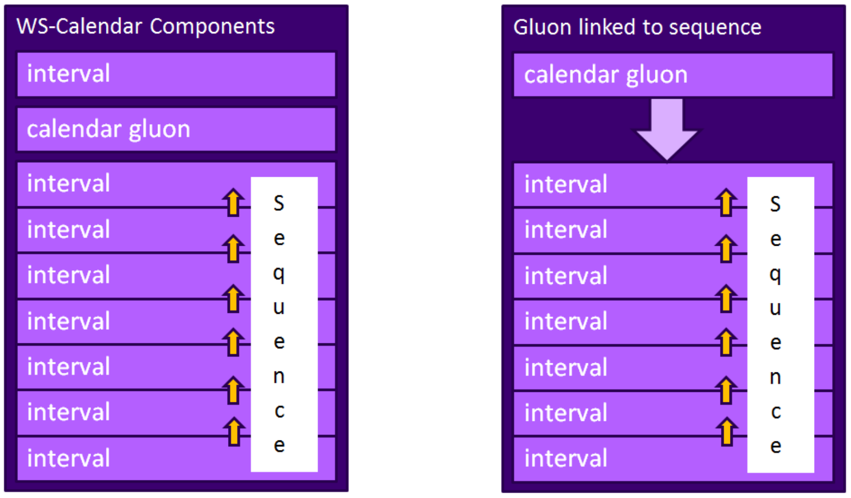


Figure 4: WS-Calendar and EMIX

A set of Intervals that have defined temporal relationships is a Sequence. Temporal relationships express how the occurrence of one Interval is related to another. For example, Interval B may begin 10 minutes after Interval A completes, or Interval D may start 5 minutes after Interval C starts. An Calendar Gluon linked to a Sequence defines service performance for all Intervals in the Sequence. Because each Interval has its own service performance contract, specifications built on WS-Calendar can define rules for inheritance and over-rides with a Sequence.

The Partition is a sub-class of a Sequence in which all Intervals follow consecutively with no lag time. Intervals in a Partition normally have the same Duration, but WS-Calendar does support overriding the duration on an individual basis.

C.1 Scheduling Sequences

A Sequence is a general pattern of behaviors and results that does not require a specific schedule. A publishing service may advertise a Sequence with no schedule, i.e., no specific time for performance. When the Sequence is invoked or contracted, a specific performance time is added. In the original iCalendar components, this would add the starting date and time (dtStart) to the component. In WS-Calendar, we add the starting date and time only to the first Interval of a Sequence; the performance times for all other Intervals in the Sequence are derived from that one start time.

C.1.1 Academic Scheduling example

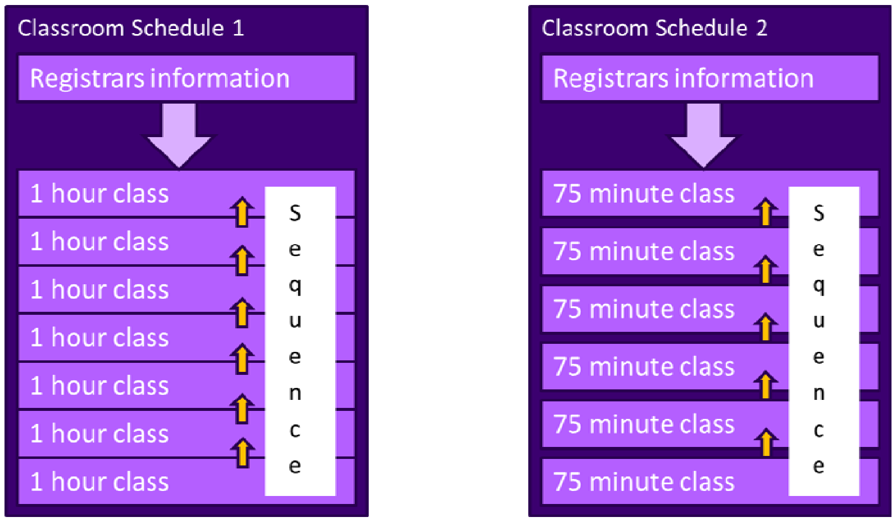


Figure 5: Classroom Scheduling Example

A college campus uses two schedules to schedule its buildings. In Schedule 1, classes start on the hour, and follow one after another; each class starts on the hour. In the second schedule, each class lasts an hour and a quarter, and there is a fifteen minute gap between classes; classes start on the half hour. On many campuses, the Sequence in Schedule 1 may describe classes taught on Monday, Wednesday, and Friday. Schedule 2 may describe classes taught on Tuesday and Thursday.

The registrar's office knows some key facts about each classroom, including whether it hosts a class during a particular period, and the number of students that will be in that class. The college wishes to optimize the provision of building services for each class. Such services may include adequate ventilation and comfortable temperatures to assure alert students. Other services may ensure that the classroom projection systems and A/V support services are warmed up in advance of a class, or powered off when a classroom is vacant.

Although most classes meet over typical schedule for the week (M-W-F or Tu-Th), some classes may not meet on Friday, or may have a tutorial section one day a week. The registrar's system, ever mindful of student privacy, shares only minimal information with the building systems such as how many students will be supported.

The Registrar's system schedule building systems using the Calendar Gluon (registrar's information) and the student counts for each Interval, and schedules the Sequence in classroom schedule 1 three days a week for the next 10 weeks. The Registrar's system also schedules the Sequence in classroom schedule 2 two days a week, also for 10 weeks.

This example demonstrates a system (A) that offers services using either of two Sequences. Another business system (B) with minimal knowledge of how (A) works determines the performance requirements for (A). The business system (B) communicates what these expectations are by scheduling the Sequences offered by (A).

C.1.2 Market Performance schedule

A factory relies on an energy-intensive process which is performs twice a year for eight weeks. The factory has some flexibility about scheduling the process; it can perform the work in either the early morning or the early evening; it avoids the afternoon when energy costs are highest. The factory works up a detailed profile of when it will need energy to support this process.

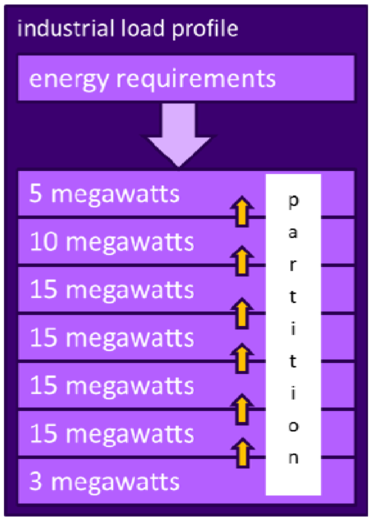


Figure 6: Daily Load Profile for Market Operations Example

Factory management has decided that they want to use only renewable energy prodUTCs for this process. They approach two regional wind farms with the intent of making committed purchases of wind energy. The wind farms consider their proposals taking into account the seasonal weather forecasts they use to project their weather capacity, and considering the costs that may be required to buy additional wind energy on the spot market to make up any shortfalls.

Each energy supplier submits of the same Sequence, a schedule, i.e. a daily starting time, and a price for the season's prodUTCion. After considering the bids, and other internal costs of each proposal, the factory opts to accept a contract for the purchase of a fixed load profile (Partition), using the evening wind generation from one of the suppliers. This contract specifies Schedules of load purchases (starting data and time for the Sequence) for each day.

2838 # D. Revision History

Revision	Date	Editor	Changes Made
1.0 WD 01	2010-03-11	Toby Considine	Initial document, largely derived from Charter
1.0 WD 02	2010-03-30	Toby Considine	Straw-man assertion of elements, components to push conversation
1.0 WD 03	2010-04-27	Toby Considine	Cleaned up Elements, added [XPOINTER] use, xs:duration elements
1.0 WD 04	2010-05-09	Toby Considine	Aligned Chapter 4 with the vAlarm and vToDo objects.
1.0 WD 05	2010-05-18	Toby Considine	Responded to comments, added references, made references to [XCAL] more consistent,
1.0 WD 06	2010-05-10	Toby Considine	Responded to comments from CalConnect, mostly constancy of explanations
1.0 WD 07	2010-07-28	Toby Considine	Incorporated input from informal public review, esp. SGIP PAP04. Firmed up relationships between scheduled objects
1.0 WD 08	2010-08-07	Toby Considine	Aligned with Interval / Partition / Sequence language. Reduced performance characteristics to before / after durations.
1.0 WD 09	2010-08-15	Toby Considine	Formalized Attachment section and rolled Performance into the Attachment. Created RelatedComponent object. Added CalWS Outline to specification. Removed SOOP section
1.0 WD 10	2010-08-28	Toby Considine, Benoit Lepeuple	Updated Time Stamp section Added background Appendices Incorporated Association language to replace RelatedComponent Recast examples to show inheritance, remove inconsistencies
1.0 WD 11	2010-09-11	Toby Considine	Traceability Release in support of a re-shuffling of the document. Sections 3, 4 were re-shuffled to create: 3: Interval / Relationships / Time Stamps 4: Performance / Attachments 5: Associations & Inheritance Also, changed all associations to Gluons. No paragraphs have been changed, just shuffled, changes accepted, to create clean base for editing
1.0 WD 12	2010-09-14	Toby Considine Dave Thewlis	Edits for clarity and flow following changes in WD11, updated examples based upon XSD artifacts. Adding final contribution from

			CalConnect for Services.
1.0 WD 13		Toby Considine	Mechanistic processing of trivial comments for grammar, spelling, etc.
1.0 WD 14	2011-01-17	Toby Considine	Added Conformance rules, redefined inheritance, added terminology section in Section 1, added language on separability of information model, REST, and SOAP sections
1.0 WD 15	2011-01-27	Toby Considine	Pulled more definitions into Terminology Section, re-factored into multiple tables, Added Availability. Have not updated examples.
1.0 WD 15	2011-01-29	Toby Considine	Re-added footers to document (?!?) Added disclaimers on completeness prior to committee spec draft.