



Working with Time Zones

W3C Working Group Note 13 October 2005

This version:

<http://www.w3.org/TR/2005/NOTE-timezone-20051013/>

Latest version:

<http://www.w3.org/TR/timezone>

Editors:

Addison Phillips, (Invited Expert)
Felix Sasaki, W3C
Mark Davis, IBM
Martin Dürst, Aoyama University

This document is also available in these non-normative formats: [XML](#).

[Copyright](#) © 2005 [W3C](#)® ([MIT](#), [ERCIM](#), [Keio](#)), All Rights Reserved. W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract


This document discusses some of the problems encountered when working with the date, time, and dateTime values from [XML Schema] when those value include (or omit) time zone offsets. Many W3C technologies rely on date and time types. Examples include the [XPathFO] specification, since it is the basis for XQuery and XSLT processing of date/time values, but the concepts affect any date / time processing.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the [W3C technical reports index](#) at <http://www.w3.org/TR/>.

This document discusses the topic of date, time, and dateTime values from [XML Schema] with and without time zone offsets. Examples are given mainly relying on [XML Schema] and [XPathFO], since these are the basis for [XQuery] and [XSLT 2.0] processing of date/time values.

This document is a W3C Working Group Note. It has been produced by the i18n Core Working Group, which is part of the Internationalization Activity.



This document has been produced as a result of discussions between the i18n Core Working Group and the two Working Groups XQuery Working Group and XSL Working Group. These Working Groups will refer to this document from [XPathFO]. The i18n Core Working Group will use material from this document for FAQs or tutorials on the topic. Please send your comments on this document to the public mailing list www-i18n-comments@w3.org (archive).

Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

Table of Contents

1 Working With Time Zones

1.1 Background

1.2 Identifying Time Zones and Zone Offsets

1.3 Incremental versus Field-Based Time

1.4 Guidelines

1.4.1 Working with Field-Based Dates and Times based on XML Schema

1.4.2 Working with Date and Time Values that Require a Time Zone (and not a zone offset)

1.4.3 Comparing Times

1.5 Recommendations for XQuery / XSLT


Appendix

A References (Non-Normative)

1 Working With Time Zones

Time-related data is a common requirement for many applications. [XML Schema] provides a variety of data types for dates and times, such as `date`, `time`, and `dateTime`. These data types follow internationally friendly formats defined by [ISO 8601] and can be used to address a variety of differing date or time applications.

The `date`, `time`, and `dateTime` types can either include or omit the time zone offset. The presence (or absence) of the offset means that the data value must be handled



differently for certain kinds of operations. In addition, the particular application and source of the date and time values affects how dates and times with different time zones or zone offsets should be handled, as well as how to handle values that lack any time zone or zone offset indication.

Note: Users and implementers of languages or specifications which handle time-related data should take the following recommendations into account even if time-zone-sensitive data is rarely used. Sooner or later some data will be affected by the issues described. Some examples of these include XQuery, XPath, and XSLT.

1.1 Background

There are three main applications of date, time, or dateTime data types in applications.

Incremental or Computer Time Most programming languages and development environments provide data types for handling time which are based on a numeric value: units of some specific length measured from a specific point in time (called the epoch). For example, the Java type `java.util.Date` is a long (integer) value for the number of milliseconds since 00:00 (midnight) on January 1, 1970 in UTC (Universal Coordinated Time, sometimes also called GMT). Other systems use other units and epochs. Date and time values based on a construct of this type (which we'll call computer time) are time-zone-independent, since at any given moment it is the same time in UTC everywhere on Earth: the values can be transformed for display for any particular time zone offset, but the value itself is not tied to a specific location. Values of this type are commonly used in applications as "time stamps", showing when an event occurred. Some applications for these include:

- labeling log file entries with a timestamp
- recording actual process start or stop times
- measuring the duration of an event
- comparing two time values

Time Zone Independent Field-Based Time The human representation of computer times is more complicated, and represents time using various separate field values, such as hour, minute, month, or year. One application for this type of representation is for values that are time zone independent, representing a logical event divorced from a particular location on the Earth. For example, various kinds of "anniversary date" such as a person's birthdate or an employee hire date would normally fall into this category, partly because time is not expressed, and partly because the actual time of the start and end of the day for a given geographic location may not be considered important. Some other examples of this application of dates and times include:

last day of the quarter

holiday schedules

end date of a promotional offer

legally mandated start or end times

Time Zone Dependent Field-Based Time In other cases, field-based dates and times are supposed to represent values linked to a particular location or time zone. For example, if you tell someone that you will make a telephone call to them at 14:00 from Paris, if that person is in London they'll expect the phone to ring at 13:00. As with incremental time, the event happens in the same instant around the globe and meaning of the value depends on the offset from UTC. Some other examples of this application of dates and times include:

- purchase order date
- tracking information for a package

1.2 Identifying Time Zones and Zone Offsets

[XML Schema] follows the [ISO 8601] standard for its lexical representation. Date and time values in ISO 8601 are field-based using the definitions above and can indicate (or omit) the zone offset from UTC. A zone offset is not the same thing as a time zone, and the difference can be important. XML Schema only supports zone offset, but, confusingly, calls it time zone, see for example section 3.2.8.1, lexical representation in [XML Schema].

Although ISO 8601 is expressed in terms of the Gregorian calendar, it can be used to represent values in any calendar system. The presentation of date and time values to end users using different calendar and timekeeping systems is separate from the lexical representation.

What is a "zone offset"? A zone offset is the difference in hours and minutes between a particular time zone and UTC. In ISO 8601, the particular zone offset can be indicated in a date or time value. The zone offset can be Z for UTC or it can be a value "+" or "-" from UTC. For example, the value 08:00-08:00 represents 8:00 AM in a time zone 8 hours behind UTC, which is the equivalent of 16:00Z (8:00 plus eight hours). The value 08:00+08:00 represents the opposite increment, or midnight (08:00 minus eight hours).

What is a "time zone"? A time zone is an identifier for a specific location or region which translates into a combination of rules for calculating the UTC offset. For example, when a website maintaining a group calendar in the United States schedules a recurring meeting for 08:00 Pacific Time, it is referring to what is sometimes known as wall time (so called because that is the time shown "on the clock (or calendar) on the wall"). This is not equivalent to either 08:00-08:00 or 08:00-07:00, because Pacific Time does not have a fixed offset from UTC; instead, the offset changes during the course of the year. As mentioned before, XML Schema only supports zone offsets, and it does not make the terminological distinction between zone offset and time zone. So a wall time expressed as an XML Schema time value, must choose which zone offset to use. This may have the unintended effect of causing a scheduled event to shift by an hour (or more) when wall time changes to or from Daylight/Summer time.

To complicate matters, the rules for computing when daylight savings takes effect may be somewhat complex and may change from year to year or from location to location. In the United States, the state of Indiana, for example, does not follow daylight savings time, but this will change in April 2006. See: <http://www.mccsc.edu/time.html> for further information. The Northern and Southern hemispheres perform Daylight/Summer Time

adjustments during opposing times during the year (corresponding to seasonal differences in the two hemispheres).

To capture these situations, a calendar system must use an ID for the time zone. The most definitive reference for dealing with wall time is the TZ database (also known as the "Olson time zone database" [\[tzinfo\]](#)), which is used by systems such as various commercial UNIX operating systems, Linux, Java, CLDR, ICU, and many other systems and libraries. In the TZ database, "Pacific Time" is denoted with the ID America/Los_Angeles. The TZ database also supplies aliases among different IDs; for example, Asia/Ulan Bator is equivalent to Asia/Ulaanbaatar. From these alias relations, a canonical identifier can be derived. The Common Locale Data Repository [CLDR] can be used to provide a localized form for the IDs: see Appendix J in [UAX 35].

1.3 Incremental versus Field-Based Time



Incremental time and field-based time differ in the way certain operations work. For example, incremental times can be directly compared—their integer values determine which is earlier or later—while field based times must be normalized and their individual fields compared. Field based times can have certain kinds of logical operations performed on them (for example, rolling the date forward or back), while incremental time requires a logical transformation. For example, to set the date 2005-08-30 forward by one day, an implementation can add 'one unit' to the "day" field and adjust the month and year as appropriate. In incremental time, a similar operation might be performed by incrementing the value by 24 hours * 60 minutes * 60 seconds * 1000 milliseconds, which is one logical day, but there may be errors when a particular day has more or fewer seconds in it (such as occur during daylight savings transitions).

The SQL data types date, time, and timestamp are field based time values which are intended to be zone offset independent. The data type timestamp with time zone is the zone offset-dependent equivalent of timestamp in SQL. Programming languages, by contrast, tend to use incremental time and convert to and from a localized textual representation on demand. Databases may use incremental time or either zone offset-dependent or independent field-based structures internally. For example, an Oracle 8 database treats a timestamp field as though it is in the local time of the database instance.

As a result, users may not be clear on the differences between these types or may create a mixture of different representations. For example, a Java programmer using JDBC will retrieve incremental times (java.util.Date objects) from a database, even though the actual field in the database is a (field-based) timestamp value.

In XML Schema, as with SQL, dates and times are always expressed using field-based time. The date or time may express the zone offset from UTC (for example using a format such as 08:00:00+01:00). UTC is indicated by the letter Z (for example 08:00:00Z). Or, the zone offset may be omitted completely.

Properly speaking, an XML Schema date or time value with a zone offset is field-based/zone offset dependent and one without is field-based/zone offset independent.



If the two types are mixed, then the interpretation of the zone offset is not adequately specified in [XML Schema]. In [XPathFO], the interpretation is implementation-defined and is based on an implicit zone offset. This is usually either UTC or local time. The presence or absence of the zone offset in the XML Schema representation may not be indicative of the original data's intention because of the confusion described above. Proper comparisons or processing rely on normalizing all date and time values into zone offset-independent (or zone offset-dependent) forms and never mixing the two in a particular operation.

1.4 Guidelines


This section describes different guidelines that can be applied to various time and date comparisons.

1.4.1 Working with Field-Based Dates and Times based on XML Schema

Field-based time and date values require the user to determine whether to use a fixed zone offset, a time zone, or nothing. While XML Schema times are field-based in terms of the lexical representation, the underlying data may use incremental time, as may the implementation processing the values. Each specific case requires specific handling.

- If all of the data values are used to represent incremental time, then the user should always use a specific zone offset (and UTC is strongly recommended as this offset, since most incremental time systems are based on it) and should always specify that zone offset. Values that do not specify a zone offset should be treated as if they use the same offset. If UTC is used, this produces the least amount of modification in the data.
- If all of the data values are used to represent time zone independent values (such as a list of employee's birth dates), then the zone offset should always be omitted. Any values that have a zone offset should probably ignore the zone offset (actually stripping it off, if possible), since zone changes are probably an artifact of other processing. If a zone offset must absolutely be applied to the data, then UTC should be used.
- If all of the data values are used represent time zone dependent values, then the zone offset must always be supplied. Great care should be used to ensure that the correct offset is used and not just the current zone offset. For example, if a system in the U.S. Pacific time zone (America/Los_Angeles) generates a dateTime value 2005-02-11T11:23:04-07:00 on 2005-08-16, it may be an error (since the offset from UTC during August in that time zone is UTC-7, but the zone offset in February is UTC-8).
- If there are time values (with no date portion) with a fixed UTC offset, then the zone offset should always be indicated if and only if the time value really is fixed. That is, this would not apply to a meeting scheduled in Pacific Time, but would apply to a meeting that is always UTC-08:00 (and thus at 7:00 in the morning in Pacific time during parts of the year).

1.4.2 Working with Date and Time Values that Require a Time Zone (and not a zone offset)



Documents or systems can also choose to accompany a time value with the appropriate time zone identifier or TZID using a complex type. This is very important with recurring times, such as calendar meeting times. If a regular meeting is at "08:00 Pacific Time", it is insufficient to store and interchange just a zone offset.

Unfortunately, XML Schema date and time types do not provide for Olson IDs, so most time operations cannot use TZIDs directly. Time zone identification in the date and time types relies entirely on time zone offset from UTC. It is up to the document designer to keep the TZID in a separate data field from the time value.

There are different ways to compare two <datetime, TZID> pairs. If both the date and time are fixed (2004-09-31T01:30), then this can be done by computing the offsets on that date and at those times, using the TZ database. This order then reflects whether one datetime is (absolutely) before another.

If the dates are not fixed (such as <T01:30, TZID> Notice that the date value is omitted) then in some sense, neither is 'before' the other, since each refers to a repeating, interleaved set of points in time. The simplest comparison mechanism where the dates may not be fully specified is simply to put both in canonical form, then order them first by time then by TZID (alphabetical, caseless order). The Olson database does not maintain a fixed canonical form; however, CLDR does provide such a form (see: [CLDR]).

(It is also possible to have a looser comparison, whereby <time0, TZID0> is compared to <time1, TZID1> over some interval of time: if one consistently has a smaller offset during that period, it is considered to be less than the other value. However, there are cases where this mechanism results in a partial ordering.)

1.4.3 Comparing Times

Conversion between or operations on data sets that mix values with and without zone offsets present certain problems.

Example 1: Values with and without zone offsets

```
<aDateTime>2005-06-07T13:14:27Z</aDateTime>  <!-- with a zone offset -->  
<bDateTime>2005-06-07T11:00:00</bDateTime>  <!-- without -->
```

If one wishes to write a comparison between the value of <aDateTime> and <bDateTime>, then the two values must be reconciled to use the same reference point. <aDateTime> uses UTC and can easily be converted to computer time or shifted to another zone offset. <bDateTime> contains no indication of the zone offset. It may be UTC or any other value (currently up to 14 hours different in either direction from UTC).

It is good practice to use an explicit zone offset wherever possible. If one is not available, best practice is to use UTC as the implicit zone offset for conversions of this nature. This is because the values are exactly centered in the range of possibilities and because representation internally (as computer time) is usually based on UTC. Since a single reference point has been used it may be possible to unwind the change later even if erroneous conversion takes place. When working with multiple documents from various sources, the "implicit" offset of the document may vary widely from that of the



implementation doing the processing. If UTC is widely used, the chances of error are reduced.

Content and query authors are warned that comparing or processing dateTimes with and without time offsets may produce odd results and such processing should be avoided whenever possible. Generating content that omits zone offset information (where it exists) is a recipe for errors later. Of course, data such as the SQL types cited earlier which is meant to represent wall time should continue to omit the zone offset. Query writers can check for the presence (or absence) of zone offset and should do so to modify dates and times explicitly (instead of allowing implicit conversion) whenever possible.

1.5 Recommendations for XQuery / XSLT

Users of XQuery 1.0 and XSLT 2.0 and other standards should take the following recommendations into account even if time-zone-sensitive data is rarely used. Sooner or later some data will be affected by the issues described:

1. If possible, make sure that data always contains an explicit zone offset.
2. Do not apply operations based on date or time types (such as indexing) to collections of data in which some data items may have zone offset information and other data items may not have zone offset information.
3. If you have data that includes implicit and fixed explicit zone offsets, before applying any date- or time-sensitive operations adjust the zone offset of the implicit data to UTC with the functions for zone offset adjustment, cf. sec. 10.7 in [XPathFO](#).
4. If you have data that contains both implicit and fixed explicit time zones and you do not want to adjust the data subset which already has a zone offset, make sure that you recognize this data subset, for example via the component extraction functions, cf. sec. 10.5 in [XPathFO](#).

A References (Non-Normative)

CLDR

[Common Locale Data Repository](#). Unicode Consortium.

ISO 8601

ISO 8601:2004. [Data elements and interchange formats - Information interchange - Representation of dates and times](#). ISO, 2004.

NOTE-datetime

Misha Wolf, Charles Wicksteed. [Date and Time Formats](#). W3C Note 15 September 1997. Available at <http://www.w3.org/TR/1998/NOTE-datetime-19980827>. The latest version of [Note-datetime](#) can be found at <http://www.w3.org/TR/NOTE-datetime>.

RFC 3339

G. Klyne, C. Newman. [Date and Time on the Internet: Timestamps](#), IETF Standard, July 2002.

tzinfo

[Time Zone Information Database](#). Available at <http://www.twinsun.com/tz/tz-link.htm>.

UAX 35

Mark Davis, [Locale Data Markup Language \(LDML\)](#), Unicode Technical Standard #35. Available at <http://unicode.org/reports/tr35/tr35-5.html>. The latest version of [LDML](#) is available at <http://unicode.org/reports/tr35/>.

XML Schema

Paul V. Biron, Ashok Malhotra, editors. [XML Schema Part 2: Datatypes Second Edition](#). W3C Recommendation 28 October 2004. Available at <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>. The latest version of [XML Schema](#) is available at <http://www.w3.org/TR/xmlschema-2/>.

XPath 2.0

Anders Berglund, Mary F. Fernández, Scott Boag, et. al., editors. [XPath 2.0](#). W3C Working Draft 15 September 2005. Available at <http://www.w3.org/TR/2005/WD-xpath20-20050915/>. The latest version of [XPath 2.0](#) is available at <http://www.w3.org/TR/xpath20/>.

XPathFO

Ashok Malhotra, Jim Melton, Norman Walsh, editors. [XQuery 1.0 and XPath 2.0 Functions and Operators](#). W3C Working Draft 15 September 2005. Available at <http://www.w3.org/TR/2005/WD-xpath-functions-20050915/>. The latest version of [XPathFO](#) is available at <http://www.w3.org/TR/xpath-functions/>.

XQuery

Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, editors. [XQuery 1.0: An XML Query Language](#). Available at <http://www.w3.org/TR/2005/WD-xquery-20050915/>. W3C Working Draft 15 September 2005. The latest version of [XQuery](#) is available at <http://www.w3.org/TR/xquery/>.

XSLT 2.0

Michael Kay, editor. [XSL Transformations \(XSLT\) Version 2.0](#). W3C Working Draft 15 September 2005. Available at <http://www.w3.org/TR/2005/WD-xslt20-20050915/>. The latest version of [XSLT 2.0](#) is available at <http://www.w3.org/TR/xslt20/>.