current millis

Standard Specification

Client Centric Time

This document specifies a time-keeping strategy designed for 3rd party reference and compliance.
It defines the client-side as the border between universal time-keeping and local time-keeping.
To achieve compliance, the following conventions must be kept between architectural modules.

Network communication relies exclusively on universal time

Machine clock provides current time as a UNIX timestamp

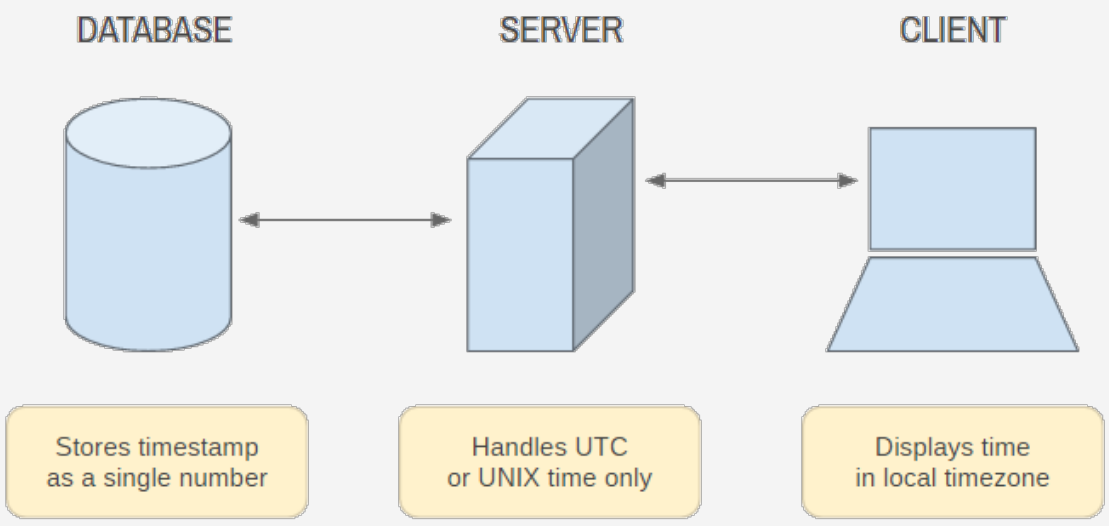
Time is stored in the database as a single number

Server uses UTC for human readable times in administration

Conversion to & from local time is a client-side responsibility

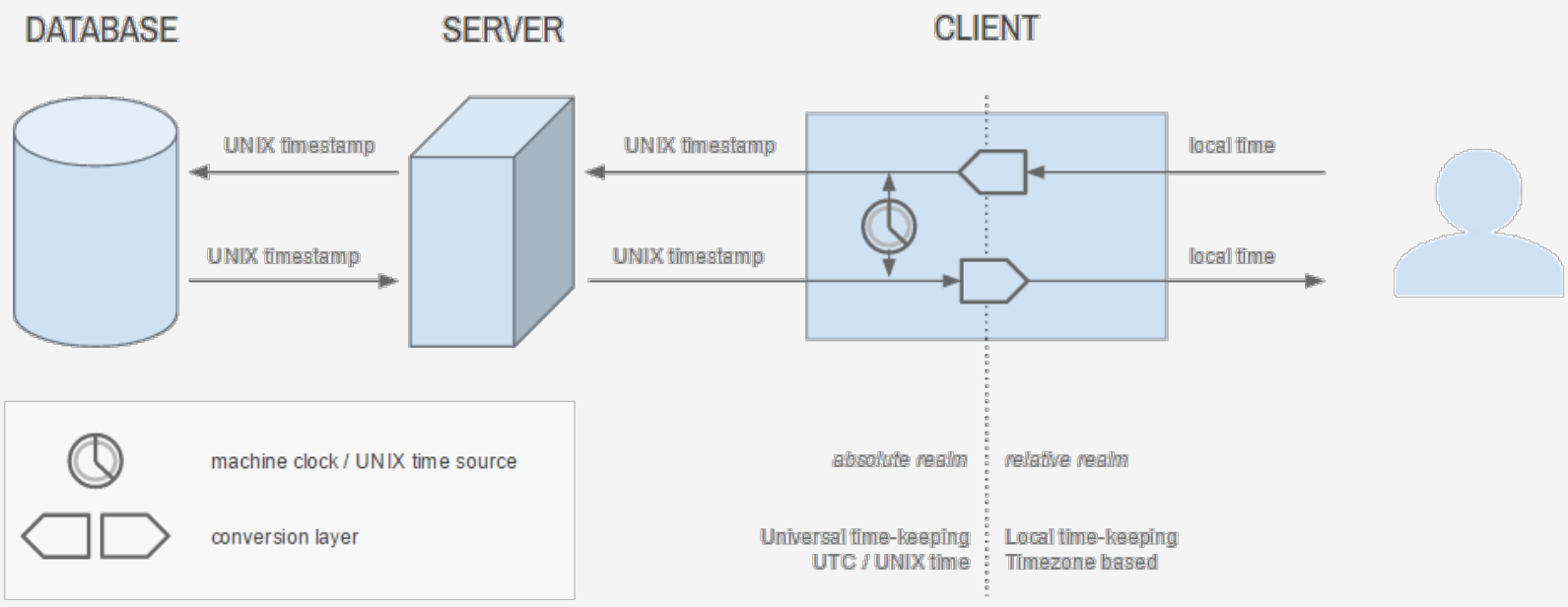
Basic Principles

The purpose of this specification is to maximize scalability, portability, compatibility between modules, independence of arbitrary concepts and the ability to interchange components. The basic principles that arise are centered around key actors in the time flow:



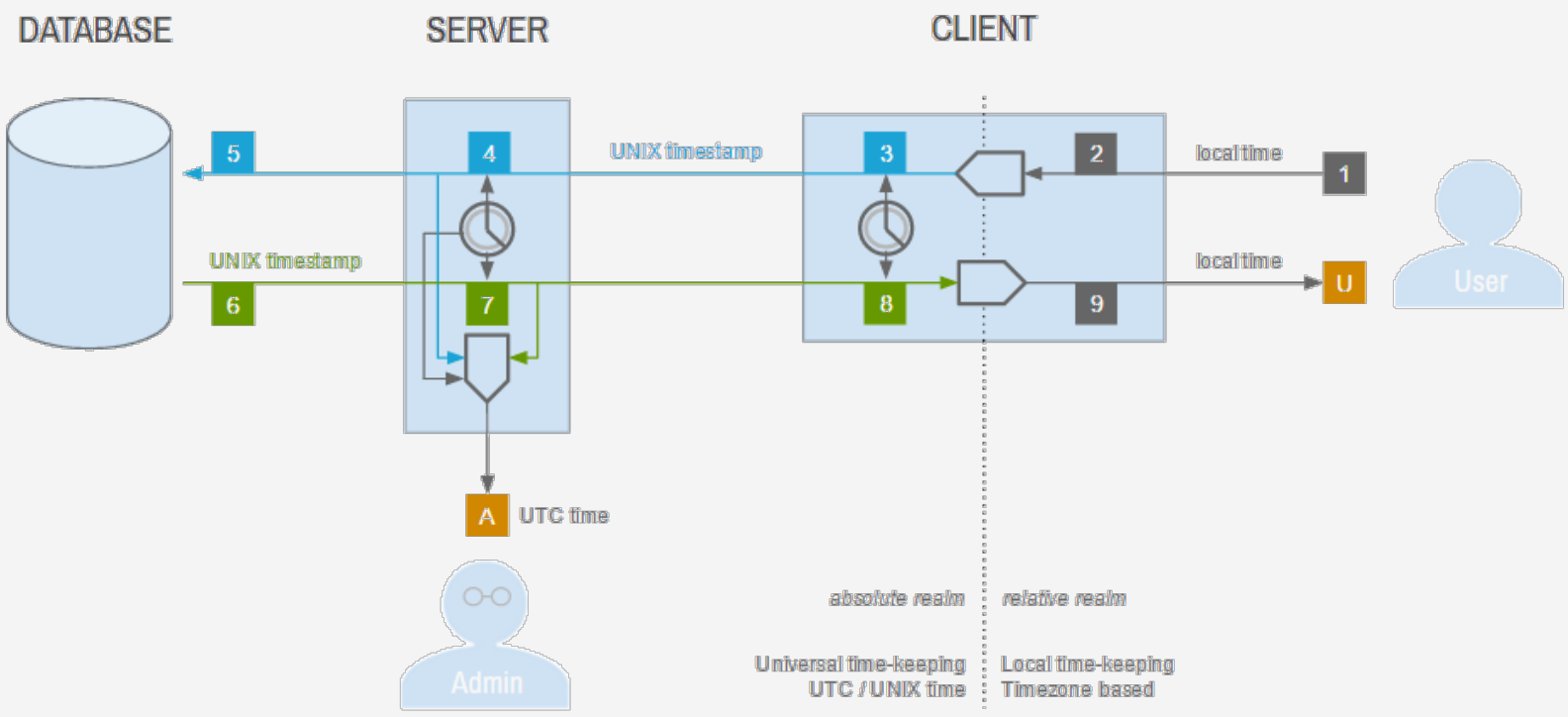
Client Perspective

The role of universal time-keeping is maximized while local time is exclusively kept by client machines. All architectural entities convene to communicate UNIX timestamps, but, in addition, the client-side has the ability to communicate local times to end-users.



Flow & Conventions

The complete perspective also takes into account server administration. It is essentially treated as a regular application client which resides in a universal timezone.



1. Network communication relies exclusively on universal time

1 2 5 6

1.1. Local time is converted to universal time before network communication.

1.2. The universal standard of choice is the UNIX timestamp in milliseconds, measuring time as the number of milliseconds that passed since the UNIX epoch (January 1, 1970, 0:00)

1.3. All operations relative to local time / timezone are finished on the client side before the conversion layer - such operations are optional; thin clients have no need for serious local processing. However, sometimes they are unavoidable - e.g. productivity apps containing features such as alarms or reminders will have local time as an input.

1.4. While p1.1 is explicitly enforced on the client side because of user input, it must be implicit when all architectural components interact: network communication exclusively relies on universal time.

2. Machine clock provides current time as a UNIX timestamp

3 4 7 8

2.1. Current time, whether it is the current client time or the current server time, will be obtained via specific programming language methods that return the current UNIX timestamp in milliseconds.

2.2. There are different perspectives on recording current time. For example, in a client-server flow, current time can be recorded either when the event is generated on the client side or when the server receives it; in addition, servers can be event generators as well. Usage of either client clocks or server clock (or both) is valid and can be done using the same methods described by p1.1.

2.3. Reading values from client-side clocks implies a level of coordination and consistency among user times and across multiple sessions. This implies a minimal synchronization protocol either internal (custom-built) or external to the application (e.g. NTP). Custom protocols might be necessary for an intranet, but synchronization offered by operating systems out of the box to update machine time periodically should be sufficient for internet-based applications.

3. Database stores & retrieves time as a UNIX timestamp

5 6

3.1. Times are stored (and retrieved) by the database as a single, simple number: the UNIX timestamp in milliseconds, avoiding proprietary formats or particularities.

3.2. If client-side timezones are relevant beyond originating user contexts (e.g. software that schedules meetings or conference calls across countries, making a single user timezone relevant to other users as well) the UTC offset at the place and time of the event should also be stored as a single, simple number of minutes. For example, Egypt's UTC+2:00 timezone will be stored as 120.

3.3. Times are stored according to the Standard Specification for Persistence of Time.

4. Server uses UTC for human readable times in administration

A

4.1. Human readable times are often needed server-side, for administration / debugging purposes (timestamps for log entries, analytics, etc). They will use UTC, avoiding the geographical timezone of the server machine itself.

5. Conversion to & from local time is a client-side responsibility

1 2 9 U

5.1. In case local time comes as user / client input, it is converted to a UNIX timestamp before being sent to the server (see p1.1)
5.2. Before displaying times in the UI, universal times (UNIX timestamps) are converted to the local timezone of choice.
5.3. There is only one conversion layer - it is implemented exclusively on the client side. Time, once converted, stays converted until it reaches its final destination - either the database as a UNIX timestamp or the end-user as local time.

Terminology

- In this document the word *Time* refers to the *Date* / *Time* combination, a unique and completely defined moment
- *Absolute realm* / *Relative realm* are concepts that visually separate areas of the architecture that keep universal / local time

version 1.05 - info@currentmillis.com