Unix time, timestamps, and timezone information

What is Unix time?

Unix time is simply a numerical value of the number of seconds that have elapsed since January 1, 1970, midnight, UTC. This date is called **Unix epoch**.

Unix epoch is not to be confused with the eGauge or meter epoch, which is configured in *Settings → Date & time when recording started*. The meter epoch represents the time the meter was comissioned and began recording accurate data.

What is UTC?

Coordinated Universal Time (abbreviated UTC) is a standard to coordinate and identify timezone differences. When times are related to UTC, they are written in a format that indicates the offset of the local time to UTC.

For example, U.S. Eastern Standard time is 5 hours behind UTC (-5), and a timestamp that is relative to Eastern Standard time may be written as `2023-12-22 15:30:00-05:00`. This removes any ambiguity or question of what timezone it is for, as well as whether it is Standard or Daylight Savings Time.

Likewise, Japan Standard Time is 9 hours ahead of UTC (+9), and the same moment in the previous example (`2023-12-22 15:30:00-05:00`) is the same time as `2023-12-22 05:30:00+9`. In UTC, the time would be `2023-12-22 20:30:00-00:00`. The equivilent Unix timestamp for this is `1703277000`.

What about timezone information?

As Unix timestamps use UTC, there is no concept of a timezone for a Unix timestamp. When processing Unix timestamps, you determine the human-friendly time of the Unix timestamp (which is in UTC), and then apply the desired timezone's offset.

For example, a Unix timestamp of `1703277000` is Friday, December 22, 2023 8:30:00 PM UTC. If we want to convert that to U.S. Eastern Standard Time, which is 5 hours before UTC, we subtract 5 hours and get Friday, December 22, 2023 3:30:00 PM UTC.

Most common programming libraries contain methods for converting Unix timestamps into human-friendly timestamps with timezone information. For example, Python3 can use the `datetime` and `pytz` to easily convert a Unix timestamp into a human-friendly, localized time. For example, here we are using Python3 to convert the timestamp in our previous example to U.S. Eastern Time (pytz will automatically choose Standard or DST):

```
>>> from datetime import datetime
>>> import pytz
>>>
>>> tz = pytz.timezone('US/Eastern')
>>> ts = int('1703277000')
>>> print(datetime.fromtimestamp(ts,tz).strftime('%Y-%m-%d %H:%M:%S'))
2023-12-22 15:30:00
```

Where are Unix timestamps used?

When interacting and obtaining data from an eGauge API, BACnet, or Modbus service, timestamps and time-ranges are most often reported and requested as Unix Timestamps.

For example, the following JSON was obtained through the WebAPI. There are two Unix timestamps, the first in the main body (`"ts": "1703275428"`) reporting the current time of the meter that the response was generated, and another within the `ranges` section (`"ts": "1703030400"`).

The first timestamp translates to *Friday, December 22, 2023 8:03:48 PM, UTC*. This is the time the response was made. The second timestamp translates to *Wednesday, December 20, 2023 12:00:00 AM, UTC*, which is the timestamp for the first `row`'s values.


```
{
 "ts": "1703275428",
 "registers": [
  {
   "name": "S15 A",
   "type": "I",
   "idx": 4,
   "did": 0
  }
 ],
 "ranges": [
  {
   "ts": "1703030400",
   "rows": [
    [
```

```
        "2078357324"
       ]
      ],
      "delta": 0
    }
  ]
}
```

Please visit kb.egauge.net for the most up-to-date documentation.