# XML API

eGauge has a free XML-format API for requesting data

- Interpreting XML data and examples
- XML API
- Timezone and "Z" parameter
- Authentication (HTTP Digest Authentication)

# Interpreting XML data and examples

Contained in this article:



---

# General Information

> XML data is sent with **cumulative** register values. In the case for power registers, the cumulative values are in watt-seconds. To convert watt-seconds to kWh, divide the watt-seconds by 3,600,000. Cumulative values can be imagined as meter readings at that point in time, where consumption readings continuously increase over time. To get the kWh usage between two dates, subtract the more recent value from the older value and divide by 3,600,000.

> EG4xxx meters support TLSv1.2, while EG30xx only supports TLSv1.0

At the bottom of the page is the output of a minute granular export showing the past ten minutes of cumulative data (obtained with the URL http://DEVNAME.egaug.es/cgi-bin/egauge-show?m&n=10). **time_stamp** represents the time of the export (unix timestamp in hex); **time_delta** indicates the time in seconds between exports; **epoch** represents the date and time recording started on that device (unix timestamp in hex).

Each **<cname>** tag contains data on a column header; in this case, that translates to a physical register on the device. **t** indicates the register type (P for power, V for Voltage, etc) and the text inside the tag represents the register name as recorded on the device. Virtual registers may also be shows in the appropriate parameter is passed in the initial request.

Each row (**<r>** tag) contains a series of columns (**<c>** tag) that show the cumulative value of each register. To determine the average value of a register over a given time, simply find the difference

between the two cumulative values and divide by the appropriate time delta. The resulting values are expressed in units based on the register type (see the chart in section 2.2.1 of the XML API document for the unit type).  It should be noted that prior to firmware 3.01 cumulative values do not necessarily count up from zero. On firmware 3.01 and newer passing the option **E** when requesting data returns values relative to device epoch (ie, values start at zero). Using the parameter **epoch** in a data push has the same effect. This assumes that the date and time recording started option is set correctly on the eGauge.

Some examples (based on the sample output below):

Average Grid usage over the most recent minute 55357226851 - 55357243343 = −16492 / 60 = -274.86 Watts (remember, Power registers are bidirectional)

Average Voltage L2 over ten minutes 4511385868513 - 4511319123106 = 66745407 / 600 = 111242.345 mV / 1000 = 111.24 Volts

```
<group serial="0x4e842294">
<data columns="12" time_stamp="0x564cb0e8" time_delta="60" epoch="0x55973268">
<cname t="P">Grid</cname>
<cname t="S">Grid*</cname>
<cname t="V">VL2</cname>
<cname t="V">VL1</cname>
<cname t="F">Frequency</cname>
<r>
<c>55357226851</c>
<c>7375247726</c>
<c>4511385868513</c>
<c>4528987513211</c>
<c>2217532746128</c>
</r>
<r>
<c>55357243343</c>
<c>7375223338</c>
<c>4511378482617</c>
<c>4528980146863</c>
<c>2217529147760</c>
</r>
<r>
<c>55357259861</c>
<c>7375198952</c>
<c>4511371100578</c>
<c>4528972784417</c>
<c>2217525549473</c>
</r>
<r>
<c>55357276431</c>
<c>7375174516</c>
<c>4511363715094</c>
```

<c>4528965418400</c>
<c>2217521950920</c>
</r>
<r>
<c>55357293137</c>
<c>7375149735</c>
<c>4511356276347</c>
<c>4528957999802</c>
<c>2217518352640</c>
</r>
<r>
<c>55357309872</c>
<c>7375124940</c>
<c>4511348838707</c>
<c>4528950580365</c>
<c>2217514754150</c>
</r>
<r>
<c>55357326630</c>
<c>7375100152</c>
<c>4511341406162</c>
<c>4528943162279</c>
<c>2217511155334</c>
</r>
<r>
<c>55357343410</c>
<c>7375075359</c>
<c>4511333976595</c>
<c>4528935743683</c>
<c>2217507556120</c>
</r>
<r>
<c>55357360207</c>
<c>7375050569</c>
<c>4511326550844</c>
<c>4528928327670</c>
<c>2217503956798</c>
</r>
<r>
<c>55357377048</c>
<c>7375025739</c>
<c>4511319123106</c>
<c>4528920909766</c>
<c>2217500357151</c>
</r>
</data>
</group>

# Example: Energy and power for specific dates

You can make CGI calls to http://*DEV-URL*/cgi-bin/egauge-show where *DEV-URL* is the URL of your eGauge (for example, http://egaugehq.d.egauge.net/).

The **T** parameter lets you request data from specific points in time. It expects a comma separated list of Unix time-stamps.

The **E** parameter requests the values be relative to the Date and Time when recording started. This needs to be set correctly in **Settings -> Date & time when recording started**. It effectively makes the reading start at zero when the date and time when recording started is set to, otherwise the raw database value could be arbitrary. This requires firmware v3.02 or greater.

The **a** parameter requests total and virtual registers, such as "Usage" and "Generation". This is optional.

http://egaugehq.d.egauge.net/cgi-bin/egauge-show?a&E&T=1514764800,1483228800 returns data for January 1 2018 00:00:00 UTC, and January 1 2017 00:00:00 UTC respectively, using epoch-relative values and requesting total and virtual registers. The output below has all the other registers except for Usage and Generation removed for readability.

```
<group serial="0x3b2d1cb7">
 <data columns="27" time_stamp="0x5a497a00" time_delta="60" epoch="0x52a0f760">
  <cname t="P">use</cname>
  <cname t="P">gen</cname>
  <r>
   <c>241517238757</c>
   <c>0</c>
  </r>
 </data>
 <data time_stamp="0x58684680" time_delta="900">
 <r>
  <c>171138633823</c>
  <c>0</c>
 </r>
 </data>
</group>
```

Generation is zero because there is none recorded on this device.

Usage for timestamp 0x5a497a00 (Jan 1 2018) is 241517238757 joules (241517238757/3600000 = 67088 kWh).

Usage for timestamp 0x58684680 (Jan 1 2017) is 171138633823 joules (171138633823/3600000 = 47538 kWh).

If you want power instead of energy, subtract the values and divide by the amount of time between them:

67088 kWh - 47538 kWh = 19550 kWh were used between 2017 and 2018. 1 year is 8760 hours, so 19550 kWh / 8760 h = 2.23 kW average over the year. This can be done using any two points in time.

---

Another description example:

# XML API

eGauge Systems offers access to a free, unrestricted API for use in developing applications which fetch data from the eGauge meter. This API covers the same functions used by the default eGauge interface. The eGauge push service functionality is also covered in the API.

eGauge technical support can offer assistance interpreting XML data. Code review and similar support is **not** available.

> EG4xxx meters support TLSv1.2, while EG30xx only supports TLSv1.0


XML API

# Timezone and "Z" parameter

> The "Z" parameter and timezone information is only used in the XML API when exporting in CSV format. This only affects human-friendly Date & Time values in CSV exports. It does not have any affect when returning XML formatted data, nor any affect on any time-related input parameters.

> Beginning in firmware v1.2, omitting the **value** of the "Z" parameter will cause the CSV output to use the locally configured timezone for human-friendly CSV Date & Time values (configured in Settings -> Date & Time). For this to work, the "Z" parameter must be passed, but with an empty value (e.g., `egauge-show?n=60&m&c&Z=` )

When using the XML API to request data in CSV format, the query string parameter "Z" is used to specify a timezone to format the Date & Time column in the CSV output. Omitting this parameter completely will cause the Date & Time column to output Unix Timestamps. Providing an invalid value is undefined, but may cause the Date & Time to output in UTC time in human-friendly format.

The format of this string is described under the environment variable TZ at https://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap08.html. Examples of how to decode and write custom timezones can be found at the end of this page.

## Common Timezones

| | |
|---|---|
| US/Eastern | LST5LDT4,M3.2.0/02:00,M11.1.0/02:00 |
| US/Central | LST6LDT5,M3.2.0/02:00,M11.1.0/02:00 |
| US/Mountain | LST7LDT6,M3.2.0/02:00,M11.1.0/02:00 |
| US/Arizona | LST7 |
| US/Pacific | LST8LDT7,M3.2.0/02:00,M11.1.0/02:00 |
| US/Alaska | LST9LDT8,M3.2.0/02:00,M11.1.0/02:00 |
| US/Hawaii | LST10 |
| US/Baker Island | LST-12 |
| US/Samoa | LST11 |
| Australia/Central | LST-10:30 |

| | |
|---|---|
| Australia/Eastern | LST-10LDT-11,M10.1.0/02:00,M4.1.0/03:00 |
| Australia/Norfolk | LST-12:30 |
| Azores Islands | LST1LDT0,M3.5.6/24:00,M10.5.0/01:00 |
| Brazil | LST3LDT2,M10.3.6/24:00,M2.5.6/24:00 |
| Canada/Atlantic | LST4LDT3,M3.2.0/02:00,M11.1.0/02:00 |
| China/Beijing | LST-8 |
| Europe/Central | LST-1LDT-2,M3.5.0/02:00,M10.5.0/03:00 |
| Europe/Eastern | LST-2LDT-3,M3.5.0/03:00,M10.5.0/04:00 |
| Europe/Western | LST0LDT-1,M3.5.0/01:00,M10.5.0/02:00 |
| India | LST-6:30 |
| Iran | LST-4:30LDT-5:30,M3.3.2/24:00,M9.3.4/24:00 |
| Iraq/Baghdad | LST-3 |
| Kazakhstan/Astana | LST-6 |
| New Zealand | LST-12LDT-13,M9.5.0/02:00,M4.1.0/03:00 |
| Pakistan/Karachi | LST-5 |
| Russia/Moscow | LST-4 |
| Russia/Vladivostok | LST-11 |
| South Sandwich | LST2 |
| Thailand/Bangkok | LST-7 |
| Tokyo | LST-9 |

# Decoding and understanding timezone strings

In the timezone strings, "LST" and "DST" stand for "Local Standard Time" and "Daylight Standard Time", respectively.

For a full description of the timezone string format, see the environment variable TZ at https://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap08.html.

The string is generally divided into 3 sections separated by commas. The first section describes the difference between UTC and local times. The second section describes when daylight time begins, and the third section describes when it ends.

The sections describing when daylight savings starts and ends is in the following format:

`Mm.n.d/t`

The `d`'th day (0 <= d <= 6) of week `n` of month `m` of the year (1 <= n <= 5, 1 <= m <= 12, where week 5 means "the last d day in month m" which may occur in either the fourth or the fifth week). Week 1 is the first week in which the d'th day occurs. Day zero is Sunday. `t` is the 24-hour time in which it occurs. If omitted, it defaults to 2:00 AM.

## US/Eastern

Timezone string: `LST5LDT4,M3.2.0/02:00,M11.1.0/02:00`

Each section, separated by commas, is described as such:

- `LST5LDT4` UTC is 5 hours after local standard time (`LST5`), and UTC is 4 hours after daylight savings time (`LDT4`)
- `M3.2.0/02:00` Daylight Saving Time starts in March (`3`) on the second week (`2`) on Sunday (`0`) at 2:00AM (`2:00`)
- `M11.1.0/02:00` Daylight Savings Time ends in November (`11`) on the first week (`1`) on Sunday (`0`) at 2:00AM (`2:00`)

---

## US/Hawaii

Timezone string: `LST10`

UTC is 10 hours after local Hawaii time. Daylight savings time is not observed, so there is no LDT definition or additional sections.

---

## New Zealand

Timezone string: `LST-12LDT-13,M9.5.0/02:00,M4.1.0/03:00`

Each section, separated by commas, is described as such:

- `LST-12LDT-13` UTC is 12 hours earlier than local standard time (`LST-12`), and UTC is 13 hours earlier than local daylight time (`LDT-13`)

- `M9.5.0/02:00` Daylight Saving Time starts in September (`9`) on the last week (`5`) on Sunday (`0`) at 2:00AM (`2:00`)
- `M4.1.0/03:00` Daylight Savings Time ends in April (`4`) on the first week (`1`) on Sunday (`0`) at 3:00AM (`3:00`)

# Authentication (HTTP Digest Authentication)

If the eGauge meter is set with site-wide password authentication, all resources will require HTTP Digest Authentication. Many programming languages and libraries have support for HTTP Digest Authentication, so well maintained and trusted libraries for HTTP Digest Authentication are preferable to use over custom implementations. In the event there is no existing support for HTTP Digest Authentication, it can be manually implemented. eGauge Systems cannot provide assistance on this implementation.

Wikipedia has information and examples of how to perform Digest Authentication and what header values must be returned. Note, any protected resources under the XML API will use a "quality of protection" (qop) of "auth", which will affect how the header values are calculated using HTTP Digest Authentication.

The basic steps to perform HTTP Digest Authentication are:

1. Make an unauthenticated request to the resource you wish to access.

2. Look at the WWW-Authentication header returned. Store the following values:
   `Digest realm` (realm space of protected resource. Generally this is "eGauge Administration" unless the interface is custom-branded)
   `nonce` (server nonce, changes for each unauthenticated request)
   `qop` (always will be "auth" for the eGauge XML API)

3. Perform the following calculations:
   `HA1 = MD5(username:realm:password)`
   `HA2 = MD5(method:digestURI)` (note the *digestURI* is the path of the resource, not including host or FQDN. E.g., */cgi-bin/egauge-show*)
   `response = MD5(HA1:nonce:nonceCount:cnonce:qop:HA2)` (*nonceCount* may be "1" for the initial request. *cnonce*
   is a nonce generated by the client, such as a 64 bit hexadecimal value)

4. Send a request to the same URI in step 1, and this time include an "Authorization" header with the following parameters and values:
   1. `Digest username` = username for authentication
   2. `realm` = realm (from step 2)
   3. `nonce` = server nonce (from step 2)
   4. `uri` = request uri (used in HA2 in step 3)
   5. `response` = response (generated in step 3)

6. `qop` = "auth"
7. `nc` = nonce count (can simply be "1" for testing, should be incremented for subsequent requests that reuse the same *nonce*)
8. `cnonce` = client nonce generated in step 3

> It is not always necessary to perform step 2 in this process if making repeated requests to the same resource (URI or "digestURI"). A server nonce expire after 10 minutes and attempting to re-use will result in a 401 with WWW-Authenticate header like found in step 1, with a "stale=true" parameter in the header string.