**How To Convert Non-UTC Timestamp Into UNIX Epoch Time In Python**

When we ingest API data, the query URI string often takes [Unix epoch time](https://en.wikipedia.org/wiki/Unix_time) (or Unix time) in order to specify the datetime range. The epoch time is the way to represent timestamp as the number of seconds that have elapsed since 1970-01-01 00:00:00 UTC.

When you have an input timestamp string and convert it to the epoch time, it is absolutely critical to understand the time zone of your input string as well as the default server time zone and deal with it correctly. This sounds complicated. But, do not worry. I will show you how to convert a timestamp string into the epoch time in different scenarios.

**Examples**

Let’s take Melbourne/Australia time zone (AEST) for example. 2018-02-15 00:00:00 in AEST is 2018-02-14 13:00:00 in UTC, which is 1518613200. You can check this with this online [epoch time converter](https://www.epochconverter.com/).

So, when the input string is 2018-02-15 00:00:00 in AEST, the epoch output should be 1518613200.

For a starter, you need to check the default time zone on your server. Here are the commands.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

# Find timezone in Windows Server:

systeminfo | findstr /C:"Time Zone"

# Find timezone in Linux Server:

date

[/cc]

**Scenario 1: Server time zone is the same as the input string time zone (AEST)**

This is the easier scenario. You can use mktime() function from the [time](https://docs.python.org/3/library/time.html) module. This takes struct\_time in local time and convert to epoch. You need to format the string into the struct\_time datatype with strptime() first and use it in mktime().

Assuming the server time zone is AEST, let’s convert 2018-02-15 00:00:00 in AEST to epoch. When you use the input string as 2018-02-15, it assumes the time is 00:00:00. So, both functions below work. You will see the output of 1518613200.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

**Scenario 2: Server time zone is UTC and the input string time zone is AEST.**

This scenario is slightly more complex. The mktime() uses the local time from the server. Therefore, you first need to convert AEST into UTC before passing it to mktime().

To convert local time to UTC, you can use the [pytz](https://pypi.python.org/pypi/pytz) module. To begin with, let’s convert local time (2018-02-15 00:00:00) to UTC. You will get the output of 2018-02-14 13:00:00 from the function below.

You first need to define your time zone with pytz.timezone() function. Then, add the time zone to the time\_struct object with the localize() function. This can be converted to UTC with astimezone (pytz.utc).

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

Ok, once you get the hang of converting your local time to UTC, all you need to do is use mktime() to convert it to epoch. Remember mktime() uses the default local time in the server as UTC. Therefore, passing the time\_struct object with UTC time will automatically converts to the right epoch time. In this case, 2018-02-15 00:00:00 in AEST gets converted to 1518613200.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

Now, you do not need to fear time zone!