Determining Use or Cost per Day in EnergyCAP PowerViews and Reports

Determining Use/Day and/or Cost/Day would be easy in a perfect world where every utility received one and only one utility bill for each commodity in each accounting period. In real life, the situation is complicated by various approaches to accounting, handling of rebills, refunds, corrected bills, etc.

Over time, EnergyCAP has devised and recommended various best practices to minimize the difficulties, which center around how the software counts (or doesn't count) the days associated with each bill received for display and reporting purposes.

This topic explains the current logic as it will be ported over from EnergyCAP 3.6 to our next generation HTML-based software.

NOTE: References to BAM = Bill Account Meter table in the EnergyCAP database

General Principles for Counting the Days – Meter Bill Lists

Affects:

- Use/Day column
- Cost/Day column

Overlapping bills and multi-account meters will NOT inflate the number of days since you are always dealing with only ONE bill

Front-end - bill list column displays what is given to it.

Back-end – the list API calculates "days" as BAM.endDate - BAM.beginDate, usePerDay as nativeUse / days, costPerDay as directCost / days

General Principles for Counting the Days – Benchmarking Building Groups

Affects:

- Building Groups > Cost > Cost/Day
- Building Groups > Use > Energy use/Day

Overlapping bills and multi-account meters will inflate the number of days, resulting in smaller benchmarking numbers. Also, incomplete data may cause numbers to shift.

Front-end - displays what is given to it; the per-day benchmark value is obtained directly from the API

Back-end - Place group digest relies on data contained in BAM, with day values. Unlike the meter digest, this appears to use a simple SUM(days) to get the number of days in a period. The "n/day" values are then calculated by calculating value/days (e.g. totalCost / daySum). The max of the day counts is returned in the data transfer object (DTO); should be 365, but may hide meters which have less than 365 days, making it hard to reconcile by hand using just the data in the DTO.

General Principles for Counting the Days – Benchmarking Meter Groups

Affects:

- Meter Groups > Cost > Cost/Day
- Meter Groups > Use > Average Use/Day

Overlapping bills and multi-account meters will NOT inflate the number of days, resulting in larger and "more correct" benchmarking values. However, incomplete data may deflate numbers.

Front-end - displays what is given to it; the per-day benchmark value is obtained directly from the API

Back-end - Meter group digest relies on data contained in BAM, with day values. The "days" calculation is MAX(endDate) - MIN(beginDate). The "n/day" values are then calculated by calculating value/days (e.g. totalCost / daySum).

General Principles for Counting the Days – Up/Down Arrow Charts in PowerViews

Affects:

- Meter > Actual Data > Summary > Daily Average Cost
- Meter > Actual Data > Summary > Daily Average Use

- Meter > Normalized Data > Summary > Daily Average Use
- Place > Actual Data > Summary > Daily Average Cost
- Place > Actual Data > Commodity > Daily Average Cost
- Place > Actual Data > Commodity > Daily Average Use
- Place > Normalized Data > Summary > Daily Average Use

Overlapping bills and multi-account meters will NOT inflate the number of days, since it's a static 365. However, incomplete data may deflate numbers.

Front-end - All up/down charts work the same way; chartHelper.getArrowChartData() - divide this year/last year totals by 365 days ("annualized")

Back-end - Place digest relies on data contained in staging tables (e.g. BillSummaryPlaceCommodityRollupCP) which DO NOT contain "per day" values... all the staging table views calculate "simple" per day values on the front-end.

General Principles for Counting the Days – Trend Charts in PowerViews

Affects:

- Meter > Actual Data > Trends > Use/Cost Per Day Trend
- Meter > Calendarized Data > Trends > Use/Cost Per Day Trend

Overlapping bills and multi-account meters will NOT inflate the number of days, resulting in "more correct" trend values. Incomplete data will cause missing points on the chart.

Front-end - chartMeterUseCostPerDayTrend.execute() contains the logic. The cost, use and number of days from the API as calculated in the back-end are divided out (e.g. totalCost / days = cost/day)

Back-end - Meter digest relies on data contained in BillAccountMeter which DOES contain "per day" values. SQL for the "days" value is calculated by taking the MAX(BAM.endDate) - MIN(BAM.startDate).

General Principles for Counting the Days – REPORTS

The logic for counting days in the Reports module in EnergyCAP is more straightforward, and relies on two days-related calculations:

1. The difference between the MIN(beginDate) and the MAX(endDate)
2. SUM(days).

MIN/MAX looks at the spectrum of bills included in the report and simply counts the days between the earliest billing period BEGIN DATE and the latest billing period END DATE, not counting the END DATE.

This method works well unless there is a gap in the billing data. In that case, the MIN/MAX value would be larger than the period(s) represented by the included bills. In this (unlikely) scenario, the sum of the days for the included bills will be used, but only if that value is smaller than the MIN/MAX value.