

CEVAC Python Package Intellectual Disclosure

Bennett Meares, 22 July 2020

Table of Contents

Introduction.....	2
CEVAC Pipes Paradigm.....	2
Problems and Solutions.....	2
Problem 1: Large Data Volumes.....	2
Solution 1: Partitioning.....	2
Problem 2: Stress on Infrastructure.....	2
Solution 2: Caching System Scheduler.....	2
Problem 3: Inconsistent Data Sources.....	3
Solution 3: CEVAC Interface System.....	3
Problem 4: Raw Data Needs Aggregation.....	3
Solution 4: Custom Pipes.....	3
Problem 5: Administration.....	5
Solution 5a: Actions.....	5
Solution 5b: Administrative Console.....	5
Problem 6: Providing Pipes To Developers.....	6
Solution 6a: Pipe Objects.....	6
Solution 6b: CEVAC REST API.....	6
Problem 7: Live Data.....	7
Solution 7: Live System Scheduler.....	8

Introduction

The CEVAC Python package is the foundational software library behind the CEVAC system, developed by Bennett Meares from May 2019 to July 2020. It orchestrates back-end processes like caching historical and live data and efficiently provides access to CEVAC data sets.

CEVAC Pipes Paradigm

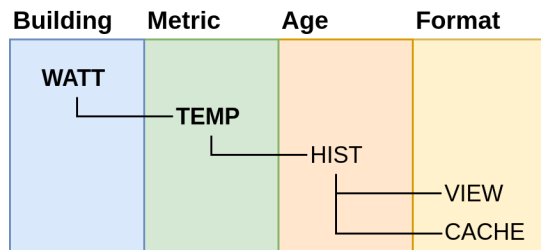


Figure 1: CEVAC Pipes contain collections of views identified as Ages.

CEVAC Pipes are collections of views, some of which are regularly materialized into on-disk cache tables. This design allows for high performance access to pre-aggregated data sets.

Every pipe must have a HIST_VIEW definition in order to be processed (see [Solution 4](#) for more information).

Problems and Solutions

Problem 1: Large Data Volumes

The amount of available data is too large to handle at once.

Solution 1: Partitioning

Pipes partition data into a standard format: by building and metric (Figures 1, 2). This reduces overhead, increases availability, and makes analysis easier.

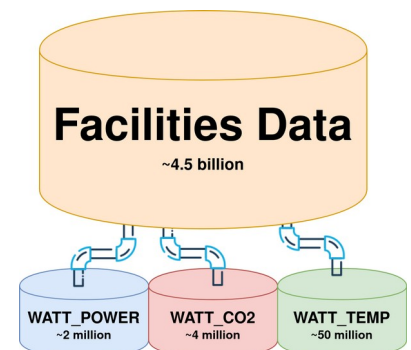


Figure 2: Pipes are fundamentally partitioned by building and metric.

Problem 2: Stress on Infrastructure

Large queries are stressful on critical Facilities infrastructure.

Solution 2: Caching System Scheduler

Pipes are cached by intermittently by the Caching System, thereby spreading the demand of large requests among many small, lightweight queries (Figure 3).

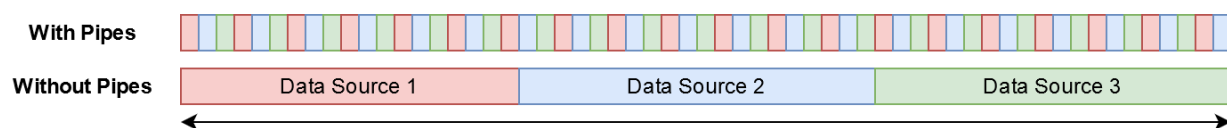


Figure 3: Pipes are cached by the caching scheduler, which balances the load among data sources.

Problem 3: Inconsistent Data Sources

Each data source has a different schema.

Solution 3: CEVAC Interface System

Standard data streams are defined by implementing the CEVAC Interface. Once a data source is applied to the interface, its Pipes inherit the standard Age aggregations and may be cached the same as any other Pipe.

The CEVAC Interface is used to generate Definitions for Pipes (Figure 4) as well as connect and update remote sources (such as pipes from Facilities' Oracle Servers).

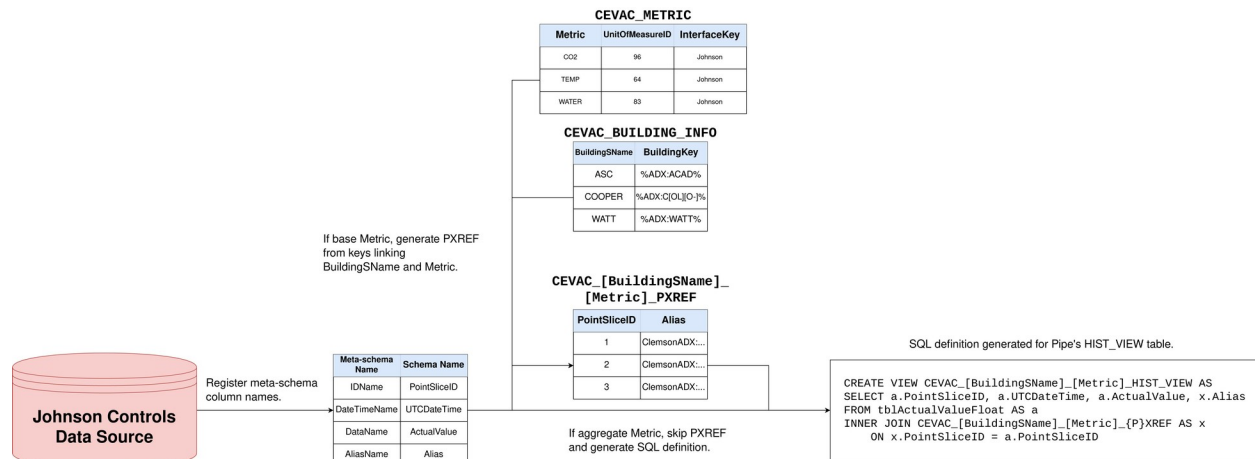
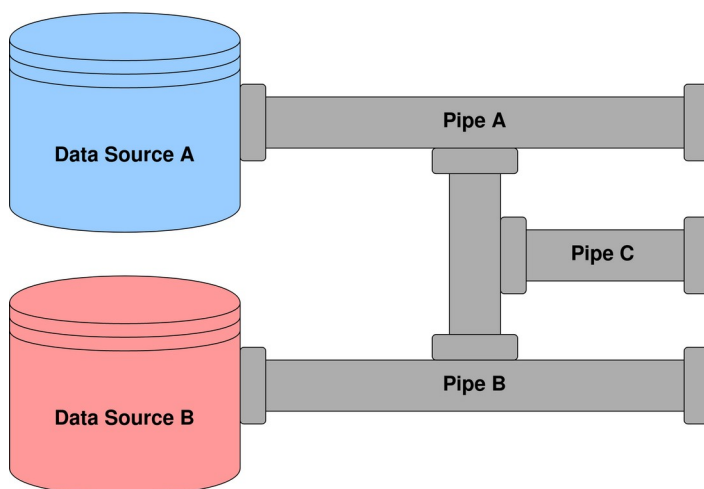


Figure 4: The CEVAC Interface normalizes data sources' schemata so that the CEVAC system can handle all Pipes, regardless of origin.

Problem 4: Raw Data Needs Aggregation

The raw data must be aggregated in order to see relationships and trends.

Solution 4: Custom Pipes



Custom definitions may be used for Pipes, and Pipes may be chained together to easily and efficiently create complex aggregations (Figure 5).

Consider the Pipe **ALL_UTILITIES** (Figure 6, Table 1). The definition aggregates raw meter data from Facilities and Schneider Electric into a comprehensive overview of each building's monthly usage, ranking, and carbon emissions.

Figure 5: Pipes may source from other Pipes. In this example, Pipe C is derived from Pipes A and B, which each derive from separate data sources.

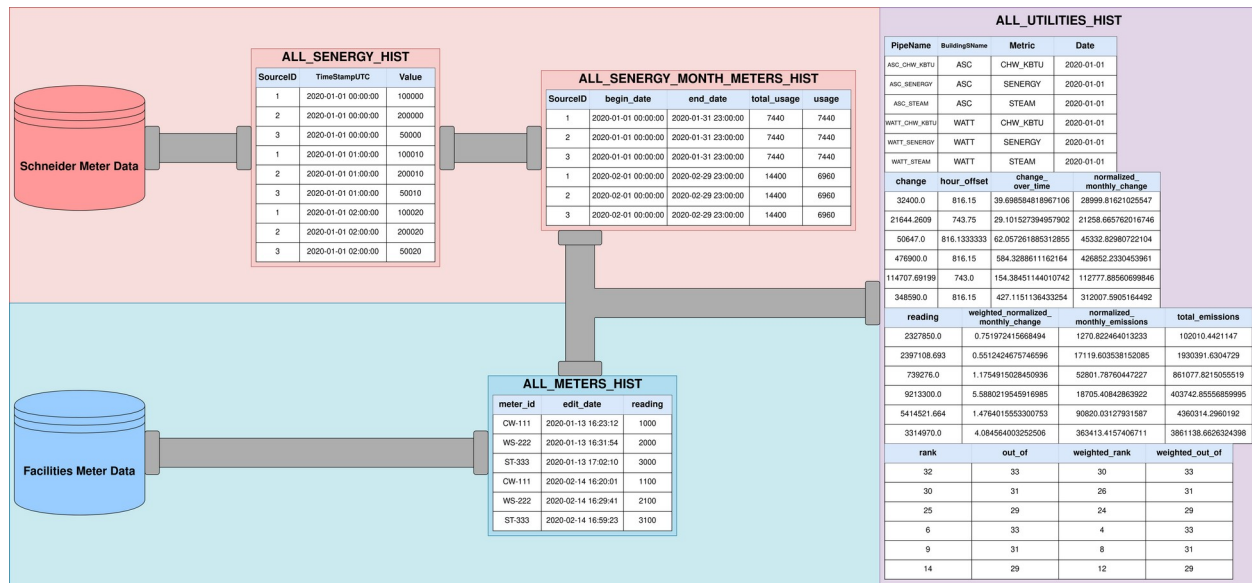


Figure 6: The Pipe ALL_UTILITIES complexly aggregates raw meter data from Schneider Electric and Facilities to calculate normalized and weighted monthly usages, emissions, and rankings for each building and metric.

Table 1: The many aggregations in the custom Pipe ALL_UTILITIES are derived from raw meter data.

Column	Description
BuildingSName / Metric / PipeName	Standard CEVAC building and metric identifiers
Date	Month and year (e.g. 2020-01-01)
change	Difference between this month's meters sum and the previous month's. Accounts for rollovers and resets.
hour_offset	Average number of hours between last month's meter readings and this month's.
change_over_time	Average hourly usage (change / hour_offset).
normalized_monthly_change	Average hourly usage multiplied by the average number of hours in a month (change_over_time * 730.5).
reading	Simulated master meter reading for the building (rolling sum of change).
weighted_normalized_monthly_change	The normalized monthly change weighted by the square footage of the building (normalized_monthly_change / square_feet).
normalized_monthly_emissions	Tons of CO ₂ released per normalized month (normalized_monthly_usage * EmissionScalar).
total_emissions	Total tons of CO ₂ released to date (reading * EmissionScalar).
rank / weighted_rank	Placement of a building's normalized monthly usage in comparison with other buildings (partitioned by Metric and Date). The value weighted_rank is weighted by square footage.
out_of / weighted_out_of	Number of buildings and metrics reporting for the current month and metric.

Problem 5: Administration

Pipes must be managed, maintained, and monitored.

Solution 5a: Actions

The CEVAC Package contains numerous Actions which may be applied to Pipes. Actions share a standard set of arguments and may be triggered via the command-line interface.

For example, the command

```
python -m CEVAC -x update_cache
```

translates to “execute the update_cache method for all existing Pipes”.

Pipes may be explicitly listed with -b and -m flags and filtered with the --params flag. Consider the table below containing several example commands and their explanations (Table 2).

Table 2: The CEVAC Python Package uses a versatile argument system to manage Pipes.

Command	Description
<code>python -m CEVAC -x bootstrap -b ASC,WATT -m WAP</code>	Bootstrap the Pipes ASC_WAP and WATT_WAP.
<code>python -m CEVAC -x live --params isProduction:1</code>	Run the live data scheduler for all production Pipes.

Solution 5b: Administrative Console

The CEVAC Administrative Console provides a graphical interface for managing Pipes. The Console translates the user’s actions into the standard arguments defined above.

For example, in the pictured screenshot (Figure 7), clicking “Build Pipe” translates to the arguments

```
-x bootstrap -b WATT -m CO2
```

Point/SliceID	Alias	UTCDate/Time	ETDate/Time	Actual/Value
8306	RM 110 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	454.405762
8422	AHU-01 Return Air CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	566.168400
8435	AHU-02 Return Air CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	671.819800
8642	RM 106 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	456.962971
8699	RM 112 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	474.102783
8750	RM L103 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	454.128052
8765	RM 218 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	459.551971
8831	RM 216 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	412.933800
8837	RM 211 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	421.683500
8843	RM 209 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	457.844800
8849	RM 207 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	459.047852
8855	RM 205 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	452.167084
8861	RM 203 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	417.048200
8907	RM 333 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	146.627441
8928	RM 323 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	467.891235
8966	RM 313 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	456.299652
9104	RM 310 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	459.597839
9191	RM 344B / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	455.412720
10859	RM 208 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	226.174118
41420	RM 413 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	458.449300
41439	RM 410 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	460.028351
41451	RM 416 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	461.469177
41464	RM 423 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	462.367100
41477	RM 433 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	462.006476
41484	RM 408 / CO2	2020-07-20 14:00:00	2020-07-20 10:00:00	463.034668

Figure 7: The CEVAC Administrative Console allows users to easily interact with the CEVAC Python Package.

Problem 6: Providing Pipes To Developers

Developers need to access CEVAC data to build applications.

Solution 6a: Pipe Objects

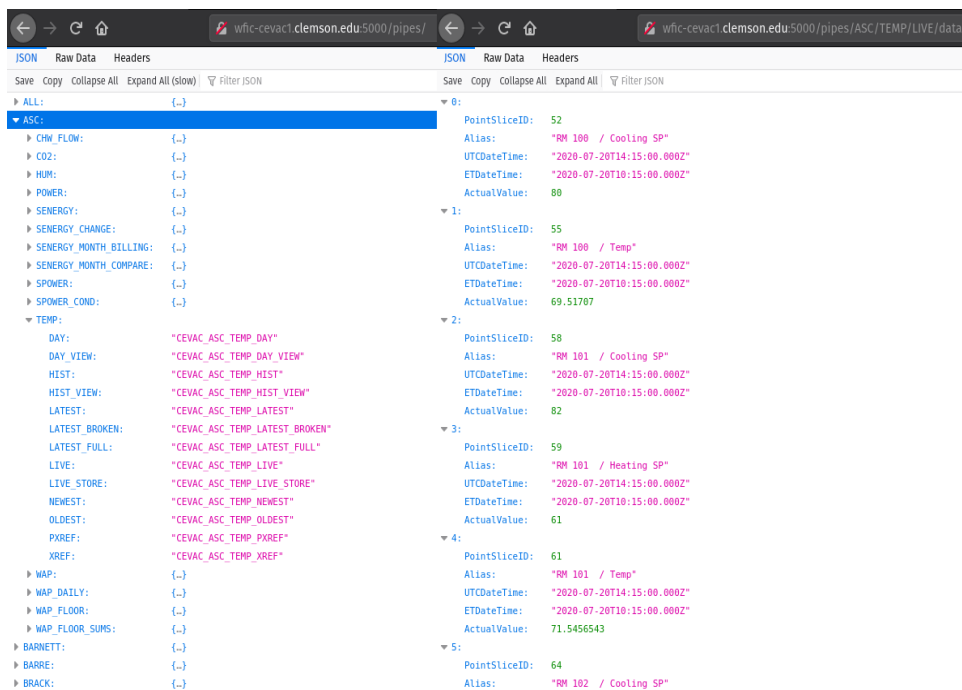
Developers use the CEVAC Python Package to access data through Pipe objects. The package includes a `getPipes()` method for constructing Pipe objects. Referencing the `.data` member of a Pipe's Age fetches and maintains the table's data. Consider the code snippet below:

```
>>> import CEVAC
>>> pipes = CEVAC.getPipes()
>>> pipes['WATT']['SENERGY']['HIST'].data
```

Pipe objects fetch tables on demand and store them as pandas DataFrames (a useful format for data analysis) and index DataFrames by ID and date-time columns. These DataFrames may be stored in-memory or on-disk.

Pipe objects are designed to sync with the CEVAC database to efficiently provide the most up-to-date data on demand. Pipes may be configured to sync data from the [CEVAC REST API \(see Solution 6b\)](#) or directly from the CEVAC SQL Server.

Solution 6b: CEVAC REST API



JSON	Raw Data	Headers
Save	Copy	collapse All (slow)
Filter JSON		
ALL:		
ASC:		
CHW_FLOW:		
CO2:		
HUM:		
POWER:		
SENERGY:		
SENERGY_CHANGE:		
SENERGY_MONTH_BILLING:		
SENERGY_MONTH_COMPARE:		
SPOWER:		
SPOWER_COND:		
TEMP:		
DAY:	"CEVAC_ASC_TEMP_DAY"	
DAY_VIEW:	"CEVAC_ASC_TEMP_DAY_VIEW"	
HIST:	"CEVAC_ASC_TEMP_HIST"	
HIST_VIEW:	"CEVAC_ASC_TEMP_HIST_VIEW"	
LATEST:	"CEVAC_ASC_TEMP_LATEST"	
LATEST_BROKEN:	"CEVAC_ASC_TEMP_LATEST_BROKEN"	
LATEST_FULL:	"CEVAC_ASC_TEMP_LATEST_FULL"	
LIVE:	"CEVAC_ASC_TEMP_LIVE"	
LIVE_STORE:	"CEVAC_ASC_TEMP_LIVE_STORE"	
NEWEST:	"CEVAC_ASC_TEMP_NEWEST"	
OLDEST:	"CEVAC_ASC_TEMP_OLDEST"	
PXREF:	"CEVAC_ASC_TEMP_PXREF"	
XREF:	"CEVAC_ASC_TEMP_XREF"	
WAP:		
WAP_DAILY:		
WAP_FLOOR:		
WAP_FLOOR_SUNGS:		
BARNETT:		
BARRE:		
BRACK:		
PointSliceID:	52	
Alias:	"RM 100 / Cooling SP"	
UTCDateTime:	"2020-07-20T14:15:00.000Z"	
ETDateTime:	"2020-07-20T10:15:00.000Z"	
ActualValue:	80	
PointSliceID:	55	
Alias:	"RM 100 / Temp"	
UTCDateTime:	"2020-07-20T14:15:00.000Z"	
ETDateTime:	"2020-07-20T10:15:00.000Z"	
ActualValue:	69.51707	
PointSliceID:	58	
Alias:	"RM 101 / Cooling SP"	
UTCDateTime:	"2020-07-20T14:15:00.000Z"	
ETDateTime:	"2020-07-20T10:15:00.000Z"	
ActualValue:	82	
PointSliceID:	59	
Alias:	"RM 101 / Heating SP"	
UTCDateTime:	"2020-07-20T14:15:00.000Z"	
ETDateTime:	"2020-07-20T10:15:00.000Z"	
ActualValue:	61	
PointSliceID:	61	
Alias:	"RM 101 / Temp"	
UTCDateTime:	"2020-07-20T14:15:00.000Z"	
ETDateTime:	"2020-07-20T10:15:00.000Z"	
ActualValue:	71.5456543	
PointSliceID:	64	
Alias:	"RM 102 / Cooling SP"	

The CEVAC API is a RESTful API for serving CEVAC data and is analogous to the functionality of `getPipes()` (Figure 8).

The API is designed to be scaled to meet high demand and preserves memory by caching Pipes on disk (Figure 11).

Figure 8: The CEVAC REST API provides easy access to CEVAC Pipes.

Consider the table below which demonstrates the performance improvements of using the CEVAC API over directly accessing the CEVAC database (Table 3, Figure 10).

Table 3: The CEVAC API efficiently serves JSON data.

Test	Time
CEVAC API (warm cache)	13.950 seconds
CEVAC API (cold cache)	42.392 seconds
Direct SQL Query	56.185 seconds

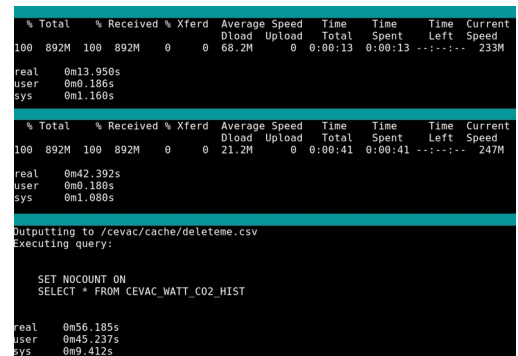


Figure 10: Screenshot of the figures presented in Table 3.

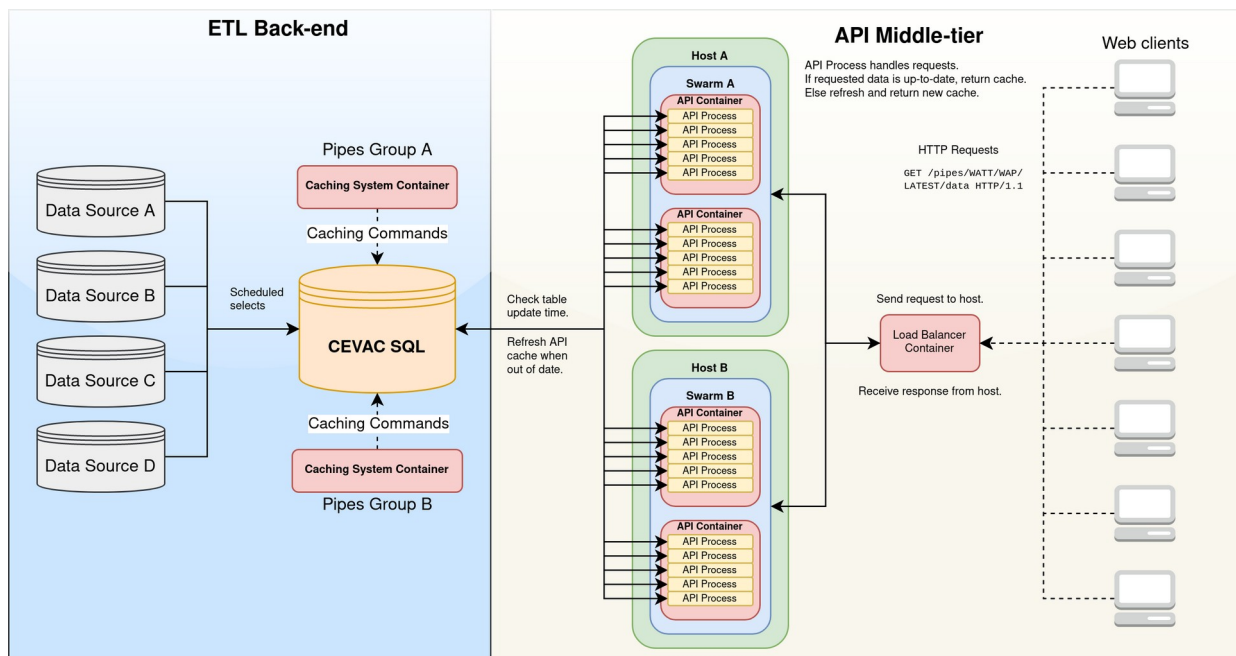


Figure 11: The CEVAC REST API protects the SQL Server by leveraging CEVAC Python to cache Pipes.

Problem 7: Live Data

Customers want to know the real-time status of certain Pipes on-demand. However, Johnson Controls Pipes update irregularly, and the caching system takes approximately 15 minutes to update all Pipes. Therefore data may not be available for several hours (until the Johnson Controls data source contains the latest historical data).

Solution 7: Live System Scheduler

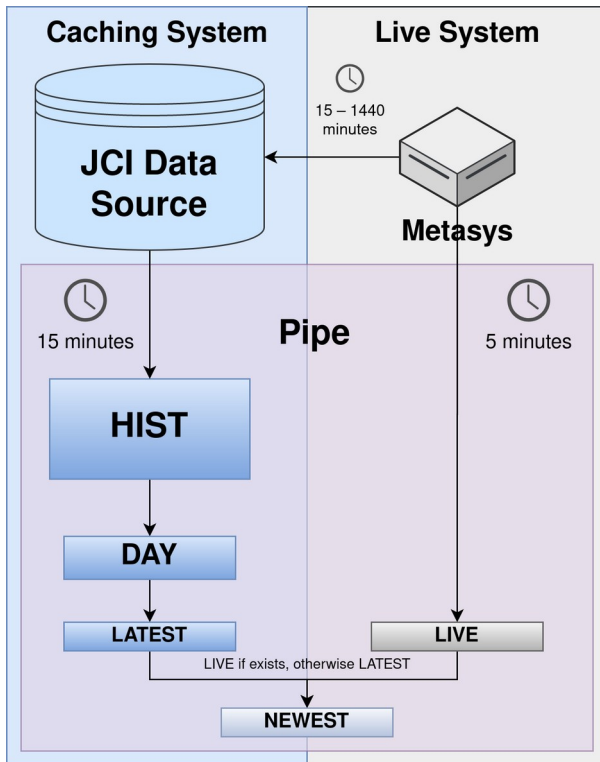


Figure 12: The Caching System updates Pipes' HIST tables, and the Live System creates LIVE tables from the JCI Metasys API.

Johnson Controls Pipes contain the Age LIVE, which is updated every 5 minutes directly from the JCI Metasys API. LIVE tables are created by the Live System (Figure 12), which emits live data via MQTT, a publish-subscribe protocol (Figure 13), and stores live data for 24 hours, reducing the frequency of older stored live data to 1 hour (Figure 14).

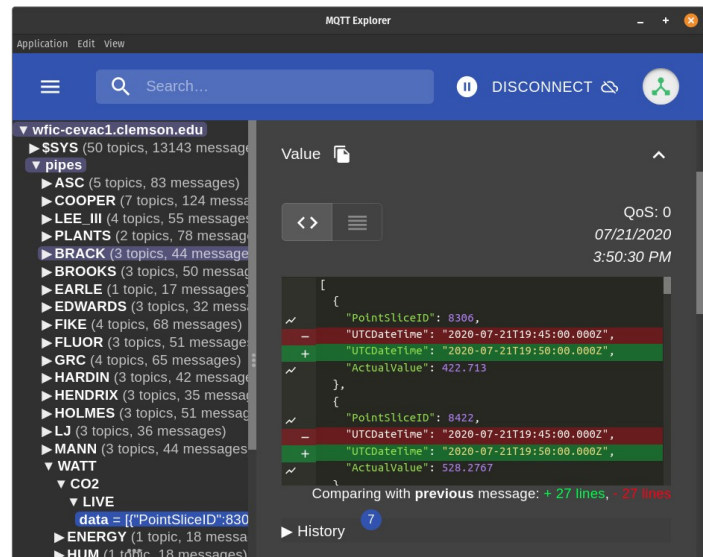


Figure 13: Clients can subscribe to live data through WebSockets using MQTT.

The screenshot shows the CEVAC Administrative Console interface. The console displays various controls and actions for the **WATT** pipe. The **CO2 (ppm)** data is shown. The console includes buttons for **Download Results** and **Download Full Table**. The table below shows the live data for the **WATT_CO2** pipe.

PointSliceID	Alias	UTCDateTime	ETDateTime	ActualValue
8306	RM 110 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	422.713000
8422	AHU-01 Return Air CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	523.341800
8435	AHU-02 Return Air CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	626.923300
8642	RM 106 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	422.840454
8699	RM 112 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	430.892400
8750	RM L103 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	424.128021
8765	RM 218 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	425.249725
8831	RM 216 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	387.602264
8837	RM 211 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	392.383000
8843	RM 209 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	429.365234
8849	RM 207 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	434.501648
8855	RM 205 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	430.711823
8861	RM 203 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	389.894684
8907	RM 333 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	194.857819
8928	RM 323 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	426.986725
8966	RM 313 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	436.171100
9104	RM 310 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	428.302338
9191	RM 344B / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	425.626068
18859	RM 208 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	222.058441
41420	RM 413 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	428.829468
41439	RM 410 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	426.830780
41451	RM 416 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	414.936700
41464	RM 423 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	428.621643
41477	RM 433 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	424.052900
41484	RM 408 / CO2	2020-07-21 20:00:00	2020-07-21 16:00:00	428.420135

Figure 14: An example of LIVE data for the Pipe WATT_CO2.