

Processor Documentation

version 1.4

Juan Torrente

2025-05-31

Contents

Processor Documentation	1
Processor package	1
Database module	1
Processor module	2
Scheduler module	3
Logger module	4
Module contents	4
Subpackages	5
Schema package	5
WeatherRecord module	5
DailyRecord module	7
MonthlyRecord module	10
WeatherStation module	12
ProcessorThread module	13
MonthlyUpdateQueue module	13
Module contents	14
Builders package	22
Submodules	22
BaseBuilder module	22
DailyBuilder module	23
MonthlyBuilder module	23
Module contents	24
Index	27
Python Module Index	33

Processor Documentation

Documentation file for the processor package.

Processor package

Database module

`class processor.database.Database`

Bases: `object`

Database class for managing PostgreSQL connections.

`classmethod close_all_connections()`

Close all connections in the pool.

Return type: `None`

`classmethod delete_monthly_update_queue_item(item_id)`

Delete an item from the monthly update queue.

Return type: `None`

`classmethod get_all_stations()`

Get all active weather stations from the database.

Returns: A list of active WeatherStation objects.

Return type: `List[WeatherStation]`

`classmethod get_connection()`

Get a connection from the pool.

Returns: A PostgreSQL database connection.

Return type: `_connection`

Raises: `psycopg2.OperationalError` – If the connection pool is not initialized.

`classmethod get_daily_records_for_station_and_interval(station_id, start_date, end_date)`

Get all daily records for a specific station and date range.

Parameters:

- **station_id** (`str`) – The ID of the weather station.
- **start_date** (`datetime.date`) – Start date for retrieving records.
- **end_date** (`datetime.date`) – End date for retrieving records.

Returns: A list of DailyRecord objects for the station in the date range.

Return type: `List[DailyRecord]`

`classmethod get_monthly_update_queue_items()`

Get all items in the monthly update queue.

`classmethod get_present_timezones()`

Get all unique timezones from the weather stations.

Return type: `List[str]`

`classmethod get_single_station(station_id)`

Get a single weather station by ID.

Parameters: `station_id` (`str`) – The ID of the station to retrieve.

Returns: The weather station object, or None if not found.

Return type: [WeatherStation](#)

```
classmethod get_weather_records_for_station_and_interval (station_id, date_from, date_to)
```

Get all weather records for a specific station and date range.

Parameters:

- **station_id** (*str*) – The ID of the weather station.
- **date_from** (*datetime.datetime*) – Start datetime for retrieving records.
- **date_to** (*datetime.datetime*) – End datetime for retrieving records.

Returns: A list of WeatherRecord objects for the station in the date range.

Return type: [List\[WeatherRecord\]](#)

Raises: **AssertionError** – If date_from and date_to do not have the same timezone info.

```
classmethod initialize (connection_string)
```

Initialize the database connection pool.

Parameters: **connection_string** (*str*) – PostgreSQL connection string.

Return type: [None](#)

```
classmethod return_connection (connection)
```

Return a connection to the pool.

Parameters: **connection** (*_connection*) – The connection to return to the pool.

Return type: [None](#)

```
classmethod save_daily_record (record)
```

Save a daily record to the database.

Return type: [str](#)

```
classmethod save_monthly_record (record)
```

Save a monthly record to the database.

Return type: [str](#)

```
classmethod save_processor_thread (processor_thread)
```

Save a processor thread to the database.

Return type: [None](#)

```
classmethod set_monthly_record_id_for_daily_records
```

(*daily_record_ids*,

monthly_record_id)

Set the monthly record ID for daily records in a specific date range.

Return type: [None](#)

```
classmethod transaction ()
```

Context manager for running multiple statements in a single transaction.

Yields: *_cursor* – A database cursor for executing SQL statements.

Raises: **Exception** – Propagates any exception that occurs during transaction execution.

Processor module

```
class processor.processor.Processor (dry_run, process_date, mode, process_pending, all_stations=False, station_id=None)
```

Bases: [object](#)

Main class for weather record processing.

`__init__(dry_run, process_date, mode, process_pending, all_stations=False, station_id=None)`
Initialize the Processor instance.

Parameters:

- **dry_run (bool)** – Whether to run in dry-run mode without saving to database.
- **process_date (date)** – Date to process data for.
- **mode (str)** – Processing mode ('daily' or 'monthly').
- **process_pending (bool)** – Whether to process records from the pending queue.
- **all_stations (bool, optional)** – Whether to process all stations. Defaults to False.
- **station_id (str, optional)** – ID of a specific station to process. Defaults to None.

Raises: `ValueError` – If both all_stations and station_id are specified.

`fill_up_daily_queue()`

Fill up the processing queue with DailyBuilder instances for each station.

For each station in the specified date, creates a DailyBuilder instance and adds it to the processing queue. Skips stations with no records.

`fill_up_monthly_queue()`

Fill up the processing queue with MonthlyBuilder instances for each station.

For the specified month, creates a MonthlyBuilder instance for each station and adds it to the processing queue. Skips stations with no daily records.

`fill_up_queue_with_pending()`

Fill up the processing queue with records from the pending queue.

Fetches items from the monthly update queue and creates appropriate MonthlyBuilder instances. Deletes processed queue entries on success.

`get_all_stations()`

Retrieve all active weather stations from the database.

Returns: List of WeatherStation objects representing all active stations. Returns empty list if no active stations are found.

Return type: list

`get_single_station(station_id)`

Retrieve a single station by its ID.

Parameters: `station_id (str)` – ID of the station to retrieve.

Returns: A list containing the WeatherStation object if found, or an empty list if not found.

Return type: list

`process_queue()`

Process all items in the processing queue.

Takes builder objects from the queue and runs them one by one. Logs success or failure for each processing operation.

`run()`

Main entry point for weather record processing.

Initializes the scheduler, fills the queue according to the specified mode, and processes all queued items. Saves processor thread information to database.

Scheduler module

```
class processor.scheduler.Scheduler(process_date)
Bases: object
```

Scheduler for weather record processing intervals.

`__init__(process_date)`

`get_full_day_intervals()`

Get start and end datetimes for the full day in each timezone.

`get_month_interval()`

Get start a n-d end datetimes for the month interval in UTC timezone.

Return type: dict

Logger module

This file configures the logger for the application.

```
class processor.logger.ColoredFormatter(fmt=None, datefmt=None, style='%', validate=True, *, defaults=None)
```

Bases: `Formatter`

Custom formatter to add colors to log messages based on their level.

Applies different ANSI color codes to log messages depending on their severity level.

```
COLORS = {'CRITICAL': '\x1b[1;31m', 'DEBUG': '\x1b[0;96m', 'ERROR': '\x1b[31m', 'INFO': '', 'WARNING': '\x1b[33m'}
```

```
RESET = '\x1b[0m'
```

`format(record)`

Format the log record with appropriate color coding.

Parameters: `record` – The log record to format.

Returns: The formatted log message with color codes applied.

Return type: str

```
processor.logger.config_logger(debug=False)
```

Configures the logger to use a custom formatter with colors for different log levels.

Parameters: `debug (bool, optional)` – If True, sets logging level to DEBUG; otherwise sets it to INFO.
Defaults to False.

Return type: None

Module contents

This module stores the model for the processor package.

```
class processor.Processor(dry_run, process_date, mode, process_pending, all_stations=False, station_id=None)
```

Bases: `object`

Main class for weather record processing.

`fill_up_daily_queue()`

Fill up the processing queue with DailyBuilder instances for each station.

For each station in the specified date, creates a DailyBuilder instance and adds it to the processing queue. Skips stations with no records.

`fill_up_monthly_queue()`

Fill up the processing queue with MonthlyBuilder instances for each station.

For the specified month, creates a MonthlyBuilder instance for each station and adds it to the processing queue. Skips stations with no daily records.

`fill_up_queue_with_pending()`

Fill up the processing queue with records from the pending queue.

Fetches items from the monthly update queue and creates appropriate MonthlyBuilder instances. Deletes processed queue entries on success.

`get_all_stations ()`

Retrieve all active weather stations from the database.

Returns: List of WeatherStation objects representing all active stations.Returns empty list if no active stations are found.

Return type: list

`get_single_station (station_id)`

Retrieve a single station by its ID.

Parameters: `station_id (str)` – ID of the station to retrieve.

Returns: A list containing the WeatherStation object if found,or an empty list if not found.

Return type: list

`process_queue ()`

Process all items in the processing queue.

Takes builder objects from the queue and runs them one by one. Logs success or failure for each processing operation.

`run ()`

Main entry point for weather record processing.

Initializes the scheduler, fills the queue according to the specified mode, and processes all queued items. Saves processor thread information to database.

Subpackages

Schema package

WeatherRecord module

```
class processor.schema.weather_record.WeatherRecord (id, station_id, source_timestamp,
temperature, wind_speed, max_wind_speed, wind_direction, rain, humidity, pressure, flagged,
taken_timestamp, gatherer_thread_id, cumulative_rain, max_temperature, min_temperature,
wind_gust, max_wind_gust)
```

Bases: `object`

Represents a single weather record from an exact point in time.

`id`

Unique identifier for the weather record.

Type: `uuid.UUID`

`station_id`

Identifier for the weather station.

Type: `uuid.UUID`

`source_timestamp`

Timestamp when the data was sourced.

Type: `datetime.datetime`

`temperature`

Temperature at the time of the record.

Type: float

wind_speed

Wind speed at the time of the record.

Type: float

max_wind_speed

Maximum wind speed recorded.

Type: float

wind_direction

Wind direction at the time of the record.

Type: float

rain

Rainfall amount at the time of the record.

Type: float

humidity

Humidity level at the time of the record.

Type: float

pressure

Atmospheric pressure at the time of the record.

Type: float

flagged

Indicates if the record has been flagged as suspicious.

Type: bool

taken_timestamp

Timestamp when the data was taken.

Type: datetime.datetime

gatherer_thread_id

Identifier of the thread that gathered this data.

Type: uuid.UUID

cumulative_rain

Total rainfall accumulated up to this point.

Type: float

max_temperature

Maximum temperature recorded up to this point.

Type: float

min_temperature

Minimum temperature recorded up to this point.

Type: float

wind_gust

Wind gust speed at the time of the record.

Type: float

`max_wind_gust`

Maximum wind gust speed recorded up to this point.

Type: float

```
__init__ (id, station_id, source_timestamp, temperature, wind_speed, max_wind_speed,
wind_direction, rain, humidity, pressure, flagged, taken_timestamp, gatherer_thread_id,
cumulative_rain, max_temperature, min_temperature, wind_gust, max_wind_gust)
```

`cumulative_rain: float`

`flagged: bool`

`gatherer_thread_id: UUID`

`humidity: float`

`id: UUID`

`max_temperature: float`

`max_wind_gust: float`

`max_wind_speed: float`

`min_temperature: float`

`pressure: float`

`rain: float`

`source_timestamp: datetime`

`station_id: UUID`

`taken_timestamp: datetime`

`temperature: float`

`wind_direction: float`

`wind_gust: float`

`wind_speed: float`

DailyRecord module

```
class processor.schema.daily_record.DailyRecord (id, station_id, date, max_temperature,
min_temperature, max_wind_gust, max_wind_speed, avg_wind_direction, max_pressure,
min_pressure, rain, flagged, finished, processor_thread_id, avg_temperature, max_humidity,
avg_humidity, min_humidity, timezone, monthly_record_id, meta_construction_data)
```

Bases: `object`

Represents a single daily weather record, which is an aggregation of multiple WeatherRecord instances.

`id`

Unique identifier for the daily record.

Type: uuid.UUID

station_id

Identifier for the weather station.

Type: uuid.UUID

date

The date of the record.

Type: datetime.date

max_temperature

Maximum temperature recorded for the day.

Type: float

min_temperature

Minimum temperature recorded for the day.

Type: float

max_wind_gust

Maximum wind gust recorded for the day.

Type: float

max_wind_speed

Maximum wind speed recorded for the day.

Type: float

avg_wind_direction

Average wind direction for the day.

Type: float

max_pressure

Maximum atmospheric pressure recorded for the day.

Type: float

min_pressure

Minimum atmospheric pressure recorded for the day.

Type: float

rain

Total rainfall recorded for the day.

Type: float

flagged

Indicates if the record has been flagged as suspicious.

Type: bool

finished

Indicates if the record processing is complete.

Type: bool

processor_thread_id

Identifier of the processor thread that created this record.

Type: uuid.UUID

avg_temperature

Average temperature for the day.

Type: float

max_humidity

Maximum humidity recorded for the day.

Type: float

avg_humidity

Average humidity for the day.

Type: float

min_humidity

Minimum humidity recorded for the day.

Type: float

timezone

Timezone of the weather station.

Type: zoneinfo.ZoneInfo

monthly_record_id

Identifier of the associated monthly record, if it exists.

Type: uuid.UUID

meta_construction_data

Additional metadata related to construction data, if any.

Type: str

```
__init__ (id, station_id, date, max_temperature, min_temperature, max_wind_gust,
max_wind_speed, avg_wind_direction, max_pressure, min_pressure, rain, flagged, finished,
processor_thread_id, avg_temperature, max_humidity, avg_humidity, min_humidity, timezone,
monthly_record_id, meta_construction_data)
```

avg_humidity: float

avg_temperature: float

avg_wind_direction: float

date: date

finished: bool

flagged: bool

id: UUID

max_humidity: float

max_pressure: float

max_temperature: float

```

max_wind_gust: float
max_wind_speed: float
meta_construction_data: str
min_humidity: float
min_pressure: float
min_temperature: float
monthly_record_id: UUID
processor_thread_id: UUID
rain: float
station_id: UUID
timezone: ZoneInfo

```

MonthlyRecord module

```

class processor.schema.monthly_record.MonthlyRecord (id, station_id, date,
avg_max_temperature, avg_min_temperature, avg_avg_temperature, avg_humidity,
avg_max_wind_gust, avg_pressure, max_max_temperature, min_min_temperature,
max_max_humidity, min_min_humidity, max_max_pressure, max_max_wind_gust, min_min_pressure,
cumulative_rainfall, processor_thread_id, finished=True)

```

Bases: `object`

Represents a single monthly weather record, which is an aggregation of multiple DailyRecord instances.

`id`

Unique identifier for the monthly record.

Type: `uuid.UUID`

`station_id`

Identifier for the weather station.

Type: `uuid.UUID`

`date`

The date of the record, typically the first day of the month.

Type: `datetime.date`

`avg_max_temperature`

Average of maximum temperatures recorded for the month.

Type: `float`

`avg_min_temperature`

Average of minimum temperatures recorded for the month.

Type: `float`

`avg_avg_temperature`

Average of average temperatures recorded for the month.

Type: `float`

avg_humidity

Average humidity recorded for the month.

Type: float

avg_max_wind_gust

Average of maximum wind gusts recorded for the month.

Type: float

avg_pressure

Average atmospheric pressure recorded for the month.

Type: float

max_max_temperature

Maximum of maximum temperatures recorded for the month.

Type: float

min_min_temperature

Minimum of minimum temperatures recorded for the month.

Type: float

max_max_humidity

Maximum of maximum humidity recorded for the month.

Type: float

min_min_humidity

Minimum of minimum humidity recorded for the month.

Type: float

max_max_pressure

Maximum atmospheric pressure recorded for the month.

Type: float

max_max_wind_gust

Maximum wind gust recorded for the month.

Type: float

min_min_pressure

Minimum atmospheric pressure recorded for the month.

Type: float

cumulative_rainfall

Total rainfall accumulated over the month.

Type: float

processor_thread_id

Identifier of the processor thread that handled this record.

Type: uuid.UUID

finished

Indicates if the record processing is complete.

Type: bool

```

__init__ (id, station_id, date, avg_max_temperature, avg_min_temperature,
avg_avg_temperature, avg_humidity, avg_max_wind_gust, avg_pressure, max_max_temperature,
min_min_temperature, max_max_humidity, min_min_humidity, max_max_pressure,
max_max_wind_gust, min_min_pressure, cumulative_rainfall, processor_thread_id,
finished=True)

avg_avg_temperature: float

avg_humidity: float

avg_max_temperature: float

avg_max_wind_gust: float

avg_min_temperature: float

avg_pressure: float

cumulative_rainfall: float

date: date

finished: bool = True

id: UUID

max_max_humidity: float

max_max_pressure: float

max_max_temperature: float

max_max_wind_gust: float

min_min_humidity: float

min_min_pressure: float

min_min_temperature: float

processor_thread_id: UUID

station_id: UUID

```

WeatherStation module

class processor.schema.weather_station.WeatherStation (id, location, local_timezone)
Bases: **object**
Represents a weather station.

id
Unique identifier for the weather station.

Type: uuid.UUID

location
Location of the weather station.

Type: str

```
local_timezone
    Timezone of the weather station.

    Type: zoneinfo.ZoneInfo

__init__(id, location, local_timezone)

id: UUID

local_timezone: ZoneInfo

location: str
```

ProcessorThread module

```
class processor.schema.processor_thread.ProcessorThread (thread_id, thread_timestamp,
command, processed_date)
Bases: object
Represents a thread that processes weather data.

thread_id
    Unique identifier for the processing thread.

    Type: uuid.UUID

thread_timestamp
    Timestamp when the thread was created.

    Type: datetime.datetime

command
    Console command that launched the processing thread.

    Type: str

processed_date
    Date when the data was processed.

    Type: datetime.date

__init__(thread_id, thread_timestamp, command, processed_date)

command: str

processed_date: date

thread_id: UUID

thread_timestamp: datetime
```

MonthlyUpdateQueue module

```
class processor.schema.monthly_update_queue.MonthlyUpdateQueue (id, station_id, year,
month)
Bases: object
Represents an entry in the monthly update queue.

id
    Unique identifier for the monthly update queue entry.
```

Type: uuid.UUID

station_id

Identifier for the weather station associated with this entry.

Type: uuid.UUID

year

The year of the monthly record to be processed.

Type: int

month

The month of the monthly record to be processed (1-12).

Type: int

__init__(id, station_id, year, month)

id: UUID

month: int

station_id: UUID

year: int

Module contents

Module containing the schema definitions for the data processing component.

class processor.schema.DailyRecord(id, station_id, date, max_temperature, min_temperature, max_wind_gust, max_wind_speed, avg_wind_direction, max_pressure, min_pressure, rain, flagged, finished, processor_thread_id, avg_temperature, max_humidity, avg_humidity, min_humidity, timezone, monthly_record_id, meta_construction_data)

Bases: object

Represents a single daily weather record, which is an aggregation of multiple WeatherRecord instances.

id

Unique identifier for the daily record.

Type: uuid.UUID

station_id

Identifier for the weather station.

Type: uuid.UUID

date

The date of the record.

Type: datetime.date

max_temperature

Maximum temperature recorded for the day.

Type: float

min_temperature

Minimum temperature recorded for the day.

Type: float

max_wind_gust

Maximum wind gust recorded for the day.

Type: float

max_wind_speed

Maximum wind speed recorded for the day.

Type: float

avg_wind_direction

Average wind direction for the day.

Type: float

max_pressure

Maximum atmospheric pressure recorded for the day.

Type: float

min_pressure

Minimum atmospheric pressure recorded for the day.

Type: float

rain

Total rainfall recorded for the day.

Type: float

flagged

Indicates if the record has been flagged as suspicious.

Type: bool

finished

Indicates if the record processing is complete.

Type: bool

processor_thread_id

Identifier of the processor thread that created this record.

Type: uuid.UUID

avg_temperature

Average temperature for the day.

Type: float

max_humidity

Maximum humidity recorded for the day.

Type: float

avg_humidity

Average humidity for the day.

Type: float

min_humidity

Minimum humidity recorded for the day.

Type: float

timezone

Timezone of the weather station.

Type: zoneinfo.ZoneInfo

monthly_record_id

Identifier of the associated monthly record, if it exists.

Type: uuid.UUID

meta_construction_data

Additional metadata related to construction data, if any.

Type: str

avg_humidity: float

avg_temperature: float

avg_wind_direction: float

date: date

finished: bool

flagged: bool

id: UUID

max_humidity: float

max_pressure: float

max_temperature: float

max_wind_gust: float

max_wind_speed: float

meta_construction_data: str

min_humidity: float

min_pressure: float

min_temperature: float

monthly_record_id: UUID

processor_thread_id: UUID

rain: float

station_id: UUID

timezone: ZoneInfo

```
class processor.schema.MonthlyRecord (id, station_id, date, avg_max_temperature,
avg_min_temperature, avg_avg_temperature, avg_humidity, avg_max_wind_gust, avg_pressure,
max_max_temperature, min_min_temperature, max_max_humidity, min_min_humidity,
```

```
max_max_pressure,      max_max_wind_gust,      min_min_pressure,      cumulative_rainfall,
processor_thread_id, finished=True)
```

Bases: `object`

Represents a single monthly weather record, which is an aggregation of multiple `DailyRecord` instances.

`id`

Unique identifier for the monthly record.

Type: `uuid.UUID`

`station_id`

Identifier for the weather station.

Type: `uuid.UUID`

`date`

The date of the record, typically the first day of the month.

Type: `datetime.date`

`avg_max_temperature`

Average of maximum temperatures recorded for the month.

Type: `float`

`avg_min_temperature`

Average of minimum temperatures recorded for the month.

Type: `float`

`avg_avg_temperature`

Average of average temperatures recorded for the month.

Type: `float`

`avg_humidity`

Average humidity recorded for the month.

Type: `float`

`avg_max_wind_gust`

Average of maximum wind gusts recorded for the month.

Type: `float`

`avg_pressure`

Average atmospheric pressure recorded for the month.

Type: `float`

`max_max_temperature`

Maximum of maximum temperatures recorded for the month.

Type: `float`

`min_min_temperature`

Minimum of minimum temperatures recorded for the month.

Type: `float`

`max_max_humidity`

Maximum of maximum humidity recorded for the month.

Type: `float`

min_min_humidity

Minimum of minimum humidity recorded for the month.

Type: float

max_max_pressure

Maximum atmospheric pressure recorded for the month.

Type: float

max_max_wind_gust

Maximum wind gust recorded for the month.

Type: float

min_min_pressure

Minimum atmospheric pressure recorded for the month.

Type: float

cumulative_rainfall

Total rainfall accumulated over the month.

Type: float

processor_thread_id

Identifier of the processor thread that handled this record.

Type: uuid.UUID

finished

Indicates if the record processing is complete.

Type: bool

avg_avg_temperature: float

avg_humidity: float

avg_max_temperature: float

avg_max_wind_gust: float

avg_min_temperature: float

avg_pressure: float

cumulative_rainfall: float

date: date

finished: bool = True

id: UUID

max_max_humidity: float

max_max_pressure: float

max_max_temperature: float

max_max_wind_gust: float

```

min_min_humidity: float
min_min_pressure: float
min_min_temperature: float
processor_thread_id: UUID
station_id: UUID

class processor.schema.MonthlyUpdateQueue (id, station_id, year, month)
Bases: object
Represents an entry in the monthly update queue.

id
Unique identifier for the monthly update queue entry.
Type: uuid.UUID

station_id
Identifier for the weather station associated with this entry.
Type: uuid.UUID

year
The year of the monthly record to be processed.
Type: int

month
The month of the monthly record to be processed (1-12).
Type: int

id: UUID
month: int
station_id: UUID
year: int

class processor.schema.ProcessorThread (thread_id, thread_timestamp, command,
processed_date)
Bases: object
Represents a thread that processes weather data.

thread_id
Unique identifier for the processing thread.
Type: uuid.UUID

thread_timestamp
Timestamp when the thread was created.
Type: datetime.datetime

command
Console command that launched the processing thread.
Type: str

```

processed_date
 Date when the data was processed.

Type: datetime.date

command: str

processed_date: date

thread_id: UUID

thread_timestamp: datetime

class processor.schema.WeatherRecord (id, station_id, source_timestamp, temperature, wind_speed, max_wind_speed, wind_direction, rain, humidity, pressure, flagged, taken_timestamp, gatherer_thread_id, cumulative_rain, max_temperature, min_temperature, wind_gust, max_wind_gust)

Bases: object
 Represents a single weather record from an exact point in time.

id
 Unique identifier for the weather record.

Type: uuid.UUID

station_id
 Identifier for the weather station.

Type: uuid.UUID

source_timestamp
 Timestamp when the data was sourced.

Type: datetime.datetime

temperature
 Temperature at the time of the record.

Type: float

wind_speed
 Wind speed at the time of the record.

Type: float

max_wind_speed
 Maximum wind speed recorded.

Type: float

wind_direction
 Wind direction at the time of the record.

Type: float

rain
 Rainfall amount at the time of the record.

Type: float

humidity
 Humidity level at the time of the record.

Type: float

pressure

Atmospheric pressure at the time of the record.

Type: float

flagged

Indicates if the record has been flagged as suspicious.

Type: bool

taken_timestamp

Timestamp when the data was taken.

Type: datetime.datetime

gatherer_thread_id

Identifier of the thread that gathered this data.

Type: uuid.UUID

cumulative_rain

Total rainfall accumulated up to this point.

Type: float

max_temperature

Maximum temperature recorded up to this point.

Type: float

min_temperature

Minimum temperature recorded up to this point.

Type: float

wind_gust

Wind gust speed at the time of the record.

Type: float

max_wind_gust

Maximum wind gust speed recorded up to this point.

Type: float

cumulative_rain: float

flagged: bool

gatherer_thread_id: UUID

humidity: float

id: UUID

max_temperature: float

max_wind_gust: float

max_wind_speed: float

```

min_temperature: float
pressure: float
rain: float
source_timestamp: datetime
station_id: UUID
taken_timestamp: datetime
temperature: float
wind_direction: float
wind_gust: float
wind_speed: float

class processor.schema.WeatherStation(id, location, local_timezone)
Bases: object
Represents a weather station.

```

id
Unique identifier for the weather station.

Type: uuid.UUID

location
Location of the weather station.

Type: str

local_timezone
Timezone of the weather station.

Type: zoneinfo.ZoneInfo

id: UUID

local_timezone: ZoneInfo

location: str

Builders package

Submodules

BaseBuilder module

```

class processor.builders.base_builder.BaseBuilder(station, records, run_id)
Bases: ABC
Abstract base class for all builders.

```

__init__(station, records, run_id)
Initialize the base processor with common attributes.

Parameters:

- **station** (*WeatherStation*) – The weather station metadata.
- **records** (*pd.DataFrame*) – DataFrame of raw weather records.
- **run_id** (*str*) – Unique identifier for this processing run.

abstractmethod **run** (*dry_run*)

Process the records and save the resulting DailyRecord or MonthlyRecord.

Parameters: **dry_run** (*bool*) – If True, don't save to database; if False, save record.

Returns: The processed record if successful, None otherwise.

Return type: *DailyRecord* | *MonthlyRecord*

DailyBuilder module

class *processor.builders.daily_builder.DailyBuilder* (*station, records, date, run_id*)

Bases: *BaseBuilder*

Processes raw weather station records for a single day into a DailyRecord summary.

__init__ (*station, records, date, run_id*)

Initialize the DailyBuilder. :type station: *WeatherStation* :param station: The weather station metadata. :type station: WeatherStation :type records: *DataFrame* :param records: DataFrame of raw weather records for the day. :type records: *pd.DataFrame* :type date: *date* :param date: The date for which to process records. :type date: *datetime.date* :type run_id: *str* :param run_id: Unique identifier for this processing run. :type run_id: str

calculate_flagged()

Determine if any record in the day is flagged as problematic. :returns: True if any record is flagged, otherwise False. Returns True if no data. :rtype: bool

calculate_humidity()

Calculate max, min, and average humidity for the day. :returns: (max_humidity, min_humidity, avg_humidity) or (None, None, None) if no data. :rtype: tuple

calculate_pressure()

Calculate the maximum and minimum pressure for the day. :returns: (max_pressure, min_pressure) or (None, None) if no data. :rtype: tuple

calculate_rain()

Calculate the total rain for the day based on cumulative rain values. :returns: The maximum cumulative rain value, or None if no data. :rtype: float

calculate_temperature()

Calculate max, min, and average temperature for the day. :returns: (max_temperature, min_temperature, avg_temperature) :rtype: tuple

calculate_wind()

Calculate wind statistics: max wind speed, max wind gust, and average wind direction. :returns: (max_wind_speed, max_wind_gust, avg_wind_direction) :rtype: tuple

run (*dry_run*)

Process the daily records and save the DailyRecord summary. :returns: True if processing was successful, otherwise False. :rtype: bool

MonthlyBuilder module

class *processor.builders.monthly_builder.MonthlyBuilder* (*station, records, interval, run_id*)

Bases: *BaseBuilder*

Processes a month's worth of weather data for a given weather station and interval.

`__init__(station, records, interval, run_id)`

Initialize the MonthlyBuilder.

Parameters:

- **station** (*WeatherStation*) – The weather station object.
- **records** (*pd.DataFrame*) – DataFrame containing daily weather records.
- **interval** (*tuple*) – Tuple representing the date interval (start, end).
- **run_id** (*str*) – Unique identifier for the processing run.

`calculate_humidity()`

Calculate humidity statistics for the month.

Returns: (max_max_humidity, min_min_humidity, avg_humidity)

Return type: tuple

`calculate_pressure()`

Calculate pressure statistics for the month.

Returns: (max_max_pressure, min_min_pressure, avg_pressure)

Return type: tuple

`calculate_rain()`

Calculate cumulative rainfall for the month.

Returns: Total rainfall for the month, or None if no data.

Return type: float

`calculate_temperature()`

Calculate temperature statistics for the month.

Returns: (max_max_temperature, min_min_temperature, avg_avg_temperature, avg_max_temperature, avg_min_temperature)

Return type: tuple

`calculate_wind()`

Calculate wind gust statistics for the month.

Returns: (max_max_wind_gust, avg_max_wind_gust)

Return type: tuple

`run(dry_run)`

Process the monthly records and save the MonthlyRecord summary.

Returns: True if processing was successful, otherwise False.

Return type: bool

Module contents

Builders module.

`class processor.builders.BaseBuilder(station, records, run_id)`

Bases: ABC

Abstract base class for all builders.

`abstractmethod run(dry_run)`

Process the records and save the resulting DailyRecord or MonthlyRecord.

Parameters: `dry_run (bool)` – If True, don't save to database; if False, save record.

Returns: The processed record if successful, None otherwise.

Return type: `DailyRecord | MonthlyRecord`

```
class processor.builders.DailyBuilder(station, records, date, run_id)
```

Bases: **BaseBuilder**

Processes raw weather station records for a single day into a DailyRecord summary.

calculate_flagged()

Determine if any record in the day is flagged as problematic. :returns: True if any record is flagged, otherwise False. Returns True if no data. :rtype: bool

calculate_humidity()

Calculate max, min, and average humidity for the day. :returns: (max_humidity, min_humidity, avg_humidity) or (None, None, None) if no data. :rtype: tuple

calculate_pressure()

Calculate the maximum and minimum pressure for the day. :returns: (max_pressure, min_pressure) or (None, None) if no data. :rtype: tuple

calculate_rain()

Calculate the total rain for the day based on cumulative rain values. :returns: The maximum cumulative rain value, or None if no data. :rtype: float

calculate_temperature()

Calculate max, min, and average temperature for the day. :returns: (max_temperature, min_temperature, avg_temperature) :rtype: tuple

calculate_wind()

Calculate wind statistics: max wind speed, max wind gust, and average wind direction. :returns: (max_wind_speed, max_wind_gust, avg_wind_direction) :rtype: tuple

run(dry_run)

Process the daily records and save the DailyRecord summary. :returns: True if processing was successful, otherwise False. :rtype: bool

```
class processor.builders.MonthlyBuilder(station, records, interval, run_id)
```

Bases: **BaseBuilder**

Processes a month's worth of weather data for a given weather station and interval.

calculate_humidity()

Calculate humidity statistics for the month.

Returns: (max_max_humidity, min_min_humidity, avg_humidity)

Return type: tuple

calculate_pressure()

Calculate pressure statistics for the month.

Returns: (max_max_pressure, min_min_pressure, avg_pressure)

Return type: tuple

calculate_rain()

Calculate cumulative rainfall for the month.

Returns: Total rainfall for the month, or None if no data.

Return type: float

calculate_temperature()

Calculate temperature statistics for the month.

Returns: (max_max_temperature, min_min_temperature, avg_avg_temperature, avg_max_temperature, avg_min_temperature)

Return type: tuple

calculate_wind()

Calculate wind gust statistics for the month.

Returns: (max_max_wind_gust, avg_max_wind_gust)

Return type: tuple

run(dry_run)

Process the monthly records and save the MonthlyRecord summary.

Returns: True if processing was successful, otherwise False.

Return type: bool

Index

`__init__()` (processor.builders.base_builder.BaseBuilder method)
 (processor.builders.daily_builder.DailyBuilder method)
 (processor.builders.monthly_builder.MonthlyBuilder method)
 (processor.processor.Processor method)
 (processor.scheduler.Scheduler method)
 (processor.schema.daily_record.DailyRecord method)
 (processor.schema.monthly_record.MonthlyRecord method)
 (processor.schema.monthly_update_queue.MonthlyUpdateQueue method)
 (processor.schema.processor_thread.ProcessorThread method)
 (processor.schema.weather_record.WeatherRecord method)
 (processor.schema.weather_station.WeatherStation method)

A

`avg_avg_temperature`
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
 (processor.schema.MonthlyRecord attribute) [1]

`avg_humidity`
(processor.schema.daily_record.DailyRecord attribute) [1]
 (processor.schema.DailyRecord attribute) [1]
 (processor.schema.monthly_record.MonthlyRecord attribute) [1]
 (processor.schema.MonthlyRecord attribute) [1]

`avg_max_temperature`
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
 (processor.schema.MonthlyRecord attribute) [1]

`avg_max_wind_gust`
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
 (processor.schema.MonthlyRecord attribute) [1]

`avg_min_temperature`
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
 (processor.schema.MonthlyRecord attribute) [1]

`avg_pressure`
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
 (processor.schema.MonthlyRecord attribute) [1]

`avg_temperature`
(processor.schema.daily_record.DailyRecord attribute) [1]
 (processor.schema.DailyRecord attribute) [1]

`avg_wind_direction`
(processor.schema.daily_record.DailyRecord attribute) [1]
 (processor.schema.DailyRecord attribute) [1]

B

`BaseBuilder` (class in processor.builders)
 (class in processor.builders.base_builder)

C

`calculate_flagged()`
(processor.builders.daily_builder.DailyBuilder method)
 (processor.builders.DailyBuilder method)

`calculate_humidity()`
(processor.builders.daily_builder.DailyBuilder method)
 (processor.builders.DailyBuilder method)

`calculate_pressure()`
(processor.builders.daily_builder.DailyBuilder method)
 (processor.builders.DailyBuilder method)

`calculate_rain()`
(processor.builders.daily_builder.DailyBuilder method)
 (processor.builders.DailyBuilder method)

`calculate_temperature()`
(processor.builders.daily_builder.DailyBuilder method)
 (processor.builders.DailyBuilder method)

`calculate_wind()`
(processor.builders.daily_builder.DailyBuilder method)

(processor.builders.DailyBuilder method)
(processor.builders.monthly_builder.MonthlyBuilder method)
(processor.builders.MonthlyBuilder method)
close_all_connections() (processor.database.Database class method)
ColoredFormatter (class in processor.logger)
COLORS (processor.logger.ColoredFormatter attribute)
command
(processor.schema.processor_thread.ProcessorThread attribute) [1]
(processor.schema.ProcessorThread attribute) [1]
config_logger() (in module processor.logger)
cumulative_rain
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
cumulative_rainfall
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]

D

DailyBuilder (class in processor.builders)
(class in processor.builders.daily_builder)
DailyRecord (class in processor.schema)
(class in processor.schema.daily_record)
Database (class in processor.database)
date (processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
delete_monthly_update_queue_item()
(processor.database.Database class method)

F

fill_up_daily_queue() (processor.Processor method)
(processor.processor.Processor method)
fill_up_monthly_queue() (processor.Processor method)
(processor.processor.Processor method)
fill_up_queue_with_pending() (processor.Processor method)
(processor.processor.Processor method)
finished (processor.schema.daily_record.DailyRecord attribute) [1]

(processor.schema.DailyRecord attribute) [1]
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
flagged (processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
format() (processor.logger.ColoredFormatter method)

G

gatherer_thread_id
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
get_all_stations() (processor.database.Database class method)
(processor.Processor method)
(processor.processor.Processor method)
get_connection() (processor.database.Database class method)
get_daily_records_for_station_and_interval()
(processor.database.Database class method)
get_full_day_intervals()
(processor.scheduler.Scheduler method)
get_month_interval() (processor.scheduler.Scheduler method)
get_monthly_update_queue_items()
(processor.database.Database class method)
get_present_timezones()
(processor.database.Database class method)
get_single_station() (processor.database.Database class method)
(processor.Processor method)
(processor.processor.Processor method)
get_weather_records_for_station_and_interval()
(processor.database.Database class method)

H

humidity
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]

I

id (processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.monthly_update_queue.MonthlyUpdateQueue attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
(processor.schema.MonthlyUpdateQueue attribute) [1]
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.weather_station.WeatherStation attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
(processor.schema.WeatherStation attribute) [1]
initialize() (processor.database.Database class method)

L

local_timezone
(processor.schema.weather_station.WeatherStation attribute) [1]
(processor.schema.WeatherStation attribute) [1]
location
(processor.schema.weather_station.WeatherStation attribute) [1]
(processor.schema.WeatherStation attribute) [1]

M

max_humidity
(processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
max_max_humidity
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
max_max_pressure
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
max_max_temperature
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
max_max_wind_gust
(processor.schema.monthly_record.MonthlyRecord attribute) [1]

(processor.schema.MonthlyRecord attribute) [1]

max_pressure
(processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
max_temperature
(processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
max_wind_gust
(processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
max_wind_speed
(processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
meta_construction_data
(processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
min_humidity
(processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
min_min_humidity
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
min_min_pressure
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
min_min_temperature
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
min_pressure
(processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]

min_temperature
(processor.schema.daily_record.DailyRecord attribute)
[1]
 (processor.schema.DailyRecord attribute) [1]
 (processor.schema.weather_record.WeatherRecord attribute) [1]
 (processor.schema.WeatherRecord attribute) [1]

module

- processor
- processor.builders
- processor.logger
- processor.schema

month (processor.schema.monthly_update_queue.MonthlyUpdateQueue attribute) [1]
 (processor.schema.MonthlyUpdateQueue attribute) [1]

monthly_record_id
(processor.schema.daily_record.DailyRecord attribute)
[1]
 (processor.schema.DailyRecord attribute) [1]

MonthlyBuilder (class in processor.builders)
 (class in processor.builders.monthly_builder)

MonthlyRecord (class in processor.schema)
 (class in processor.schema.monthly_record)

MonthlyUpdateQueue (class in processor.schema)
 (class
 in
 processor.schema.monthly_update_queue)

P

pressure
(processor.schema.weather_record.WeatherRecord attribute) [1]
 (processor.schema.WeatherRecord attribute) [1]

process_queue() (processor.Processor method)
 (processor.processor.Processor method)

processed_date
(processor.schema.processor_thread.ProcessorThread attribute) [1]
 (processor.schema.ProcessorThread attribute) [1]

processor

- module

Processor (class in processor)
 (class in processor.processor)

processor.builders

- module

processor.logger

- module

processor.schema

- module

processor_thread_id
(processor.schema.daily_record.DailyRecord attribute)
[1]
 (processor.schema.DailyRecord attribute) [1]
 (processor.schema.monthly_record.MonthlyRecord attribute) [1]
 (processor.schema.MonthlyRecord attribute) [1]

ProcessorThread (class in processor.schema)
 (class in processor.schema.processor_thread)

R

rain
 (processor.schema.daily_record.DailyRecord attribute) [1]
 (processor.schema.DailyRecord attribute) [1]
 (processor.schema.weather_record.WeatherRecord attribute) [1]
 (processor.schema.WeatherRecord attribute) [1]

RESET (processor.logger.ColoredFormatter attribute)

return_connection()
 (processor.database.Database class method)

run()
 (processor.builders.base_builder.BaseBuilder method)
 (processor.builders.BaseBuilder method)
 (processor.builders.daily_builder.DailyBuilder method)
 (processor.builders.DailyBuilder method)
 (processor.builders.monthly_builder.MonthlyBuilder method)
 (processor.builders.MonthlyBuilder method)

(processor.Processor method)

(processor.processor.Processor method)

S

save_daily_record()
 (processor.database.Database class method)

save_monthly_record()
 (processor.database.Database class method)

save_processor_thread()
 (processor.database.Database class method)

Scheduler (class in processor.scheduler)

set_monthly_record_id_for_daily_records()
 (processor.database.Database class method)

source_timestamp
(processor.schema.weather_record.WeatherRecord attribute) [1]
 (processor.schema.WeatherRecord attribute) [1]

station_id (processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
(processor.schema.monthly_record.MonthlyRecord attribute) [1]
(processor.schema.monthly_update_queue.MonthlyUpdateQueue attribute) [1]
(processor.schema.MonthlyRecord attribute) [1]
(processor.schema.MonthlyUpdateQueue attribute) [1]
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]

T

taken_timestamp
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
temperature
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
thread_id
(processor.schema.processor_thread.ProcessorThread attribute) [1]
(processor.schema.ProcessorThread attribute) [1]
thread_timestamp
(processor.schema.processor_thread.ProcessorThread attribute) [1]
(processor.schema.ProcessorThread attribute) [1]
timezone (processor.schema.daily_record.DailyRecord attribute) [1]
(processor.schema.DailyRecord attribute) [1]
transaction() (processor.database.Database class method)

W

WeatherRecord (class in processor.schema)
(class in processor.schema.weather_record)
WeatherStation (class in processor.schema)
(class in processor.schema.weather_station)
wind_direction
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]
wind_gust
(processor.schema.weather_record.WeatherRecord attribute) [1]

(processor.schema.WeatherRecord attribute) [1]
wind_speed
(processor.schema.weather_record.WeatherRecord attribute) [1]
(processor.schema.WeatherRecord attribute) [1]

Y

year (processor.schema.monthly_update_queue.MonthlyUpdateQueue attribute) [1]
(processor.schema.MonthlyUpdateQueue attribute) [1]

Python Module Index

p

[processor](#)

[processor.builders](#)

[processor.logger](#)

[processor.schema](#)