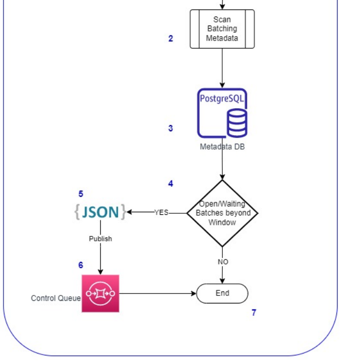
# Batch Monitoring Services

## version

**Vigenesh Raj**

September 13, 2023


Scan Batching Metadata
2
PostgreSQL
3
Metadata DB
4
Open/Waiting Batches beyond Window
YES
NO
5
JSON
Publish
6
Control Queue
End
7

# Co

# Welcome to Batch Monitoring Services’s documentation!

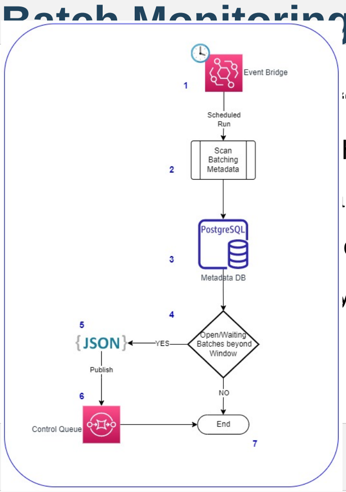
The Batch Monitor monitors the batch at the defined frequency and captures all the open and delayed branches. It will prepare the respective json message and publish it to the Control service queue.

---

## Documentation

The Batch Monitor monitors the batch at the defined frequency and captures all the open and delayed branches. It will prepare the respective json message and publish it to the Control service queue.

## Batch Monitoring Flow



“Batch Monitoring” function at the defined frequency.

hing transaction metadata tables.

nd Delayed batches beyond the window end date, from transaction tables.

elayed batches. If no End the process.

yed bactches, prepare respective JSON message.

service queue.

## Installing Dependent Modules

If you’re using a recent version of Debian or Ubuntu Linux, you can install with the system package manager:

```
$ apt-get install python-module`
```

package name is published through PyPi, so if you can’t install it with the system packager, you can install it with `easy_install` or `pip`. The package name is `packagename`, and the same package works on Python 2 and Python 3.

```
$ easy_install packagename
```

```
$ pip install packagename
```

To install necessary packages:

```
$ pip install uuid
```

```
$ pip install unittest
```

```
$ pip install psycopg2
```

```
$ pip install boto3
```

```
$ pip install botocore
```

```
$ pip install jsonschema
```

If all else fails, the license for package allows you to package the entire library with your application.

I use Python 2.7 and Python 3.2 to develop package, but it should work with other recent versions.

# acuity_de_batchingmonitor

## acuity_de_batchingmonitor package

### Subpackages

### acuity_de_batchingmonitor.commons package

### Submodules

### acuity_de_batchingmonitor.commons.CONSTANTS module

### acuity_de_batchingmonitor.commons.connect_pg module

*class* `acuity_de_batchingmonitor.commons.connect_pg.`**`connect_pg`**

Bases: **`object`**

**`commit_pg_txn`** **()**

**`get_conn_params`** **()**

### acuity_de_batchingmonitor.commons.gen_sql module

### acuity_de_batchingmonitor.commons.json_validator module

`acuity_de_batchingmonitor.commons.json_validator.`**`validate_json`** (`schema`, `json_dict`)

This method will validate validate an instance under a given schema.

**Parameters:**

- **schema** (*str*) – The schema to validate with.
- **json_dict** (*dict*) – The instance to validate.

**Returns:** The Validation boolean.

**Return type:** boolean

### acuity_de_batchingmonitor.commons.log4j_logger module

This module contains a class that wraps the log4j object instantiated by the active SparkContext, enabling Log4j logging for PySpark using.

*class* `acuity_de_batchingmonitor.commons.log4j_logger.`**`Log4j`** (`spark`)

Bases: **`object`**

**`debug`** **(**`message`**)**

It prints messages with the level (Level.DEBUG).

**Parameters:** **message** (*obj*) – The message object.

**Returns:** None.

**`error`** **(**`message`**)**

It prints messages with the level (Level.ERROR).

**Parameters:** **message** (*obj*) – The message object.

**Returns:** None.

**info** (`message`)
It prints messages with the level (Level.INFO).

**Parameters:** **message** (*obj*) – The message object.
**Returns:** None.

**warn** (`message`)
It prints messages with the level (Level.WARN).

**Parameters:** **message** (*obj*) – The message object.
**Returns:** None.

## acuity_de_batchingmonitor.commons.publish_sqs module

*class* `acuity_de_batchingmonitor.commons.publish_sqs.`**publish_sqs**
Bases: **object**

**gen_msg** (`meta_dict: dict`, `msg_dict: dict`, `exceptionType: str`, `evtMsg: str`)
This method will Create SQS client

**Parameters:**

- **queue_name** (*str*) – The Queue Name.
- **pub_msg** (*dict*) – The Public Message.

**Returns:** The response message.
**Return type:** json

**pub_sqs** (`pub_msg: dict`)
This method will Create SQS client

**Parameters:**

- **queue_name** (*str*) – The Queue Name.
- **pub_msg** (*dict*) – The Public Message.

**Returns:** The response message.
**Return type:** json

## Module contents

# acuity_de_batchingmonitor.monitor package

## Submodules

## acuity_de_batchingmonitor.monitor.monitor_main module

`acuity_de_batchingmonitor.monitor.monitor_main.`**b_mon** ()
This method will be used to create public directory.

**Parameters:**

- **ctrl_pub_msg** (*str*) – Event Message.
- **config_nm** (*str*) – Configuration Name.
- **extract_dt** (*str*) – Extraxt Date.
- **trgt_obj_nm** (*str*) – Target Object Name.

**Returns:** The JSON Response.
**Return type:** dict

`acuity_de_batchingmonitor.monitor.monitor_main.`**create_pub_dict** (`ctrl_pub_msg: str`, `config_nm: str`, `extract_dt: str`, `trgt_obj_nm: str`)
This method will be used to create public directory.

**Parameters:**

- **ctrl_pub_msg** (*str*) – Event Message.
- **config_nm** (*str*) – Configuration Name.
- **extract_dt** (*str*) – Extraxt Date.
- **trgt_obj_nm** (*str*) – Target Object Name.

**Returns:** The JSON Response.

**Return type:** dict

### Module contents

## Module contents

# Indices and tables

- **`genindex`**
- **`modindex`**
- **`search`**

# Index

# Python Module Index