# Scheduler Server Ticker

Task Scheduler ticker microservice.

**Note**: This Service uses a Json file named “config.json” as its configuration file.

Note: This service has no APIs. It only needs to run a task every 5 minutes.

## Technologies

* Back: NodeJS
* Database: MongoDB
* Messaging: Kafka

# Job

## Part 1: Cron Job

The following function should be run every 5 minutes.

Ticker.collect()

Collect records from Datebase and push them on Kafka.

### Steps

1. Get all records from schedule database with all these conditions (“AND” all these conditions):
   1. done == false
   2. queed == false
   3. datetime <= (now + 15minutes)
   4. datetime > now
2. For each record
   1. Publish event on kafka
      1. Topic: “Task\_Queed”
      2. payload: record
   2. Update record field in database with queed=true

/clients/create [POST, JSON]

This API creates a Client and stores it in clients table.

### Parameters

* + masterId
  + masterSecret
  + name: Name of Reader Application
  + title: Title of Reader Application
  + desc: Short description about Reader

### Return

* + clientId
  + clientSecret

### Controller Class

* + clients

### Affected Tables

* + clients

### Notes

* + masterSecret in the config file is Hashed with sha-512. Hash the masterSecret parameter before comparing it with the masterSecret of config file
  + When client created, active parameter should be set to TRUE by default

/ clients/delete [POST, JSON]

This API deletes a Client from clients table.

### Parameters

* + masterId
  + masterSecret
  + clientId or name

### Return

* + <Basic Response>

### Controller Class

* + clients

### Affected Tables

* + clients

### Notes

* + masterSecret in the config file is Hashed with sha-512. Hash the masterSecret parameter before comparing it with the masterSecret of config file

/clients/block [POST, JSON]

This API blocks a Client.

### Parameters

* + masterId
  + masterSecret
  + clientId or name

### Return

* + <Basic Response>

### Controller Class

* + clients

### Affected Tables

* + clients

### Notes

* + masterSecret in the config file is Hashed with sha-512. Hash the masterSecret parameter before comparing it with the masterSecret of config file
  + Mark Client as active=false in clients table

/clients/unblock [POST, JSON]

This API unblocks a Client.

### Parameters

* + masterId
  + masterSecret
  + clientId or name

### Return

* + <Basic Response>

### Controller Class

* + clients

### Affected Tables

* + clients

### Notes

* + masterSecret in the config file is Hashed with sha-512. Hash the masterSecret parameter before comparing it with the masterSecret of config file
  + Mark Reader as active=true in clients table

/clients/list [POST, JSON]

This API blocks a Client.

### Parameters

* + masterId
  + masterSecret
  + clientId

### Return

* + List of all clients including only these items {name, title, desc, clientId, active}

### Controller Class

* + clients

### Affected Tables

* + clients

### Notes

* + masterSecret in the config file is Hashed with sha-512. Hash the masterSecret parameter before comparing it with the masterSecret of config file

## Part 3: Task APIs

/task/create

Creates a new task.

### Parameters

* clientId
* clientSecret
* target: String
* payload: JsonObject
* datetime: Datetime
* endpoints: Json Array Object

### Returns

* request\_code

### Notes

* + clientId and clientSecret should be found in the clients table
  + client shoud be Active=TRUE

### Steps

1. If datetime is older than “now”
   1. Return error: Invalid time range
2. If datetime difference from “now” is less than 30minutes
   1. Return error: Too close
3. Set request\_code = new UUID
4. Store new schedule in database
5. Publish event on Kafka
   1. Topic: “task\_created”
   2. Content:
      1. request\_code
      2. target
      3. payload
      4. datetime
      5. endpoints
      6. created\_at
6. Return Success and request\_code

/task/delete

Deletes a task.

### Parameters

* clientId
* clientSecret
* request\_code

### Returns

* request\_code

### Notes

* + clientId and clientSecret should be found in the clients table
  + client shoud be Active=TRUE

### Steps

1. Get record from database with same request\_code
2. If record.datetime is passed OR record.done==TRUE
   1. Return Error: Task already expired
3. Publish event on Kafka
   1. Topic: “task\_deleted”
   2. Content:
      1. request\_code
      2. record.target
      3. record.payload
      4. record.datetime
      5. record.endpoints
      6. created\_at
4. Return Success