

Работа с API через RabbitMQ

- [Описание](#)
- [Очереди](#)
- [Структура отправляемого JSON сообщения](#)
- [Структура принимаемого JSON сообщения](#)
- [Пример работы с API на Go](#)
- [Пример работы с API через интерфейс RabbitMQ Management Plugin](#)

Описание

В данной статье рассматриваются способы отправки API запросов через RabbitMQ.

 Для идентификации запросов посылается уникальный параметр `request_id`.

Очереди

Во время работы с API в RabbitMQ используются две очереди:

1. `api_req` - очередь для входящих запросов;
2. `api_resp` - очередь, в которую попадают ответы на запросы (только если указан `request_id`)

Структура отправляемого JSON сообщения

Параметр	Тип	Пример	Обязательный	Описание
<code>request_id</code>	string	"abcd1234"	Да, если нужен ответ на запрос	Идентификатор запроса
<code>request</code>	string	<code>api/v1.1/campaigns/triggers/import_and_start_batch</code>	Да	Путь API запроса
<code>body</code>	JSON объект	<pre>"body": { "data": { "_fname": "Fname1", "_lname": "Lname1", "email": "profile1@example.com" }, "email": "profile1@example.com", "db_id": 1, "detect_geo": true, "token": "abcdefghijklmnopqrstuvwxyABCD" }</pre>	Да	Тело API запроса

Структура принимаемого JSON сообщения

Параметр	Тип	Пример	Описание
----------	-----	--------	----------

body	JSON объект	<pre>"body" : { "error":0, "error_text":"Successful operation", "profile_id":"60f039e830b8bcb28392f8eb" }</pre>	Тело ответа на запрос
request_id	string	"abcd1234"	Идентификатор запроса

Пример работы с API на Go

Послать API запрос можно с помощью скрипта, обращающегося к RabbitMQ:

- Написать и выполнить скрипт

Пример

```
package main

import (
    "encoding/json"
    "log"

    "github.com/streadway/amqp"
)

const accID = 1
const resourceID = 3
const dbID = 23
const msgID = 17
const segmentID = 85
const amountOfPushMsgs = 1
const emailDomain = "example.com"

const req = `{
    "account_id": 1,
    "request_id": "db1894e4-1a2c-4021-8233-9cca0b96b79e",
    "request": "api/v1.1/campaigns/triggers/import_and_start_batch",
    "body": {
        "token": "abcdefghijklmnopqrstuvwxyABC",
        "format": "json",
        "trigger_id": 240,
        "skip_triggers": false,
        "detect_geo": true,
        "matching": "custom",
        "field_name": "CustomF",

        "custom_data":{
            "some": "some0"
        },
        "content":{
            "someCont": "someCont0"
        },
        "data": [
            {
                "data":{
                    "_fname": "NUMBER13",
                    "_lname": "Lambert",
                    "phones": "790000000013",
                    "email": "profile1@example.com",
                    "CustomF": 18
                },
                "custom_data":{
                    "some": "some1"
                }
            }
        ]
    }
}
```

```

        },
        "content":{
            "someCont":"someCont1"
        }
    },
    {
        "data":{
            "_fname":"NUMBER14",
            "_lname":"Hard",
            "phones":"790000000014",
            "email": "profile2@example.com",
            "CustomF":14
        }
    }
]
}
}
}

func failOnError(err error, msg string) {
    if err != nil {
        log.Fatalf("%s: %s", msg, err)
    }
}

func main() {
    var err error

    var conn *amqp.Connection
    conn, err = amqp.Dial("amqp://example:abcdefghijklmnopqrstuvwxy127.0.0.1:5672/")
    failOnError(err, "Failed to connect to RabbitMQ")
    defer conn.Close()

    var ch *amqp.Channel
    ch, err = conn.Channel()
    failOnError(err, "Failed to open RabbitMQ channel")
    defer ch.Close()

    qUeueImport, err := ch.QueueDeclare(
        "api_req", // name
        true,      // durable
        false,     // delete when unused
        false,     // exclusive
        false,     // no-wait
        nil,      // arguments
    )
    failOnError(err, "Failed to declare RabbitMQ queue")

    for i := 0; i < amountOfPushMsgs; i++ {
        var bodyMap = make(map[string]interface{})
        err = json.Unmarshal([]byte(req), &bodyMap)
        failOnError(err, "Failed json.Unmarshal to bodyMap")

        for k, v := range bodyMap {
            log.Println(k, v)
        }

        body, err := json.Marshal(bodyMap)
        failOnError(err, "Failed to json.Marshal bodyMap")

        err = ch.Publish(
            "", // exchange
            qUeueImport.Name, // routing key
            false, // mandatory
            false, // immediate
            amqp.Publishing{
                ContentType: "text/plain",
                Body: []byte(body),
            })
        failOnError(err, "Failed to publish a message")
    }
}

```

```

    log.Printf("[v] Sended %d requests to %s", amountOfPushMsgs, qUeueImport.Name)
}

```

- После выполнения скрипта, зайти в RabbitMQ во вкладку **Queues**

Overview Connections Channels Exchanges **Queues** Admin Clust

Queues

▼ All queues (85)

Pagination

Page 1 of 1 - Filter: Regex ? Display

Overview					Messages			Message rates		
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/	ak_form_data	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s
/	ak_form_events	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s
/	api_req	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s
/	api_resp	classic	D	idle	1	0	1	0.00/s	0.00/s	0.00/s
/	database_import	classic	D	idle	0	0	0		0.00/s	0.00/s
/	database_import_result	classic	D TTL	idle	0	0	0		0.00/s	0.00/s
/	evgen_3	classic	D	idle	0	0	0			
/	glb_tasks	classic	D	idle	0	0	0			
/	hook_events	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s
/	integ_statseg	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s
/	oxy_background_procleadsaver	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s

- Выбрать **api_resp** и получить ответ на API запрос

Message 3

The server reported 0 messages remaining.

Exchange	(AMQP default)
Routing Key	api_resp
Redelivered	●
Properties	message_id: 9279c236-5be1-4872-a947-63ffafe9f5a7 delivery_mode: 2 content_type: application/json
Payload 161 bytes Encoding: string	<pre> {"request":"","body":{"error":0,"error_text":"Successful operation","profile_id":"60f639e830b8bcb28392f8eb"},"request_id":"db1894e4-1a2c-4021-8233-9cca0b96b79e"} </pre>

Пример работы с API через интерфейс RabbitMQ Management Plugin

Послать API запрос можно прямо в RabbitMQ через **api_req**:

- В RabbitMQ необходимо выбрать вкладку **Queues**. Далее - **api_req** и **Publish message**.

▼ Publish message

Message will be published to the default exchange with routing key **api_req**, routing it to this queue.

Delivery mode: **1 - Non-persistent** ▾

Headers: ? = **String** ▾

Properties: ? =

Payload:

```
"request": "api/v1.1/profiles/import",  
"body": {  
  "data": {  
    {  
      "_fname": "dst",  
      "_lname": "dst",  
      "email": "dst@example.com"  
    },  
    "email": "dst@example.com",  
    "db_id": 42,  
    "detect_geo": true,  
    "token": "abcdefghijklmnopqrstuvwxy"z"  
  }  
}
```

Publish message

- После того как запрос будет отправлен, посмотреть его результат можно будет так же как и в способе №1.