=

Template microservice specification

Version February 11, 2020, 8:13 PM

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Spec | Summary of Changes | Revised by |
| **24/04/2018** | **1.0** | Initial version for review | Paul Johansen |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Chapter 1: Procedure 4

# 

# Procedure

| **Name:** | **Notification Service** | | |
| --- | --- | --- | --- |
| **Description:** | Provides an API for sending notifications to clients.  Currently it is only e-mail but could extend later to mechanisms like instant messaging if a user has set that as their preference.  It is implemented around templates which the caller can overwrite labels or attach files to. | | |
| **Capabilities** | | | |
| Order Management | | | |
| **Notifications API** | | | |
| Commands | Queries | | Events Published |
| Synchronous:   * createOrder() * reviseOrder() * cancelOrder()   Asynchronous:   * N/A | getOrder() | | * None |
| Non-functional requirements | * 99.95% availability * 1000 orders/second * Submission time <= 100ms | | |
| **Observability** | | | |
| Key metrics | | | |
| * Storage\_provider * Smtp\_available * Temp\_storage * Time\_since\_last\_sent | | | |
| Health check endpoint | | / health | |
| **Implementation** | | | |
| Domain Model | | | |
| * Order aggregate | | | |
| **Dependencies** | | | |
| Invokes | | Subscribes to | |
| * None | | Users Service  Accounts Service | |

Notes

When sending a message to a client, logo’s and colours will be derived from either the first parent dealer account that has ‘while label’ settings enabled or if none do then the default GiacomoRepo schema.

When sending a message to a client the smtp gateways will be derived from either the first parent dealer account that has a ‘while label’ SMTP gateway setting enabled or if none do then the default GiacomoRepo gateway. (Office 365 at this stage).

This is currently configured as a REST POST call which is not 100% optimised microservice but having an event queue with links to temporary blobs also breaks the microservice design of data modelling.

Send by

Userid

Account/role e.g. owner

Params:

* template -- Template ID to send
* recipient
  + address -- The recipient's email address
  + name (optional) -- The recipient's name
* cc (optional) -- An array of CC recipients, of the format {"address":"cc@email.com"}
* bcc (optional) -- An array of BCC recipients, of the format {"address":"bcc@email.com"}
* sender (optional)
  + address -- The sender's email address
  + reply\_to -- The sender's reply-to address
  + name -- The sender's name
* template\_data (optional) -- Object containing email template data (maximum 128KB)
* tags (optional) -- Array of tags (as strings). *Tags are passed to your ESP as Categories, Tags, etc.*
* headers (optional) -- Object contain SMTP headers to be included with the email
* inline (optional) -- Inline attachment object (see example)
* files (optional) -- List of file attachments (combined maximum 7MB, see example)
* esp\_account (optional) -- ID of the ESP Account to send this email through. ex: esp\_1a2b3c4d5e
* locale (optional) -- Template locale to send (ie: en-US)
* version\_name (optional) -- Name of the template version to send (overrides A/B tests)

Sample Request:

{

"template": "tem\_A5RHVP6CnRbS34UysLjYHx",

"recipient": {

"name": "John",

"address": "user@email.com"

},

"template\_data": { "amount": "$12.00" },

"cc": [

{"address": "cc\_one@email.com"},

{"address": "cc\_two@email.com"}

],

"bcc": [

{"address": "bcc\_one@email.com"},

{"address": "bcc\_two@email.com"}

],

"sender": {

"name": "Company",

"address": "company@company.com",

"reply\_to": "info@company.com"

},

"tags": [

"tag1",

"tag2",

"tag3"

],

"headers": {

"X-HEADER-ONE": "header-value"

},

"inline": {

"id": "cat.png",

"data": "{BASE\_64\_ENCODED\_FILE\_DATA}"

},

"files": [

{

"id": "doc.txt",

"data": "{BASE\_64\_ENCODED\_FILE\_DATA}"

},

{

"id": "stuff.zip",

"data": "{BASE\_64\_ENCODED\_FILE\_DATA}"

}

],

"locale": "en-US",

"esp\_account": "esp\_1a2b3c4d5e",

"version\_name": "Version Name"

}

Sample Response:

{

"success": true,

"status": "OK",

"receipt\_id": "log\_asdf1234qwerty",

"email": {

"name": "NAME OF EMAIL",

"version\_name": "NAME OF VERSION",

"locale": "en-US"

}

}

Inline Images

Sample Request Code

{

"template": "tem\_9UPo9P7qMr8qHwHvPhDjP6F6f",

"recipient": {

"name": "John",

"address": "customer@example.com"

},

"inline": {

"id": "happy.png",

"data": "iVBORw0KGgoAAAANSUhEUgAAABQAAAAUCAMAAAC6V+0/AAAANlBMVEX1zHX///////70znj31I799eP20YP53qb879P879V/XjXTrWO5lFSgfUbYsWWOazx0Ui+DYjfNsfnyAAAAfUlEQVQYlXWQ2w6AIAxDK3KTIej//6yboiiZfWjGSZaVYlKE000MHvAhmg6dRZN1N1xmPJqXC7oXY+oEGouPrGEYZaq1e2QYZEipe2DoMcgzHBnwB2W90PWk0tblEG1ZWN6oHTojlT2ta9rLHamFp5yph1e/qReiV6eXPOoAcMkDL0/NWToAAAAASUVORK5CYII="

}

}

Sample Template Code

<img src="cid:happy.png" alt="a happy face">