Jeremy Harrault

Swordfish

– CALS SIM –

API Specification

Objectives of this document

Glossary and Terminology

– A –

API: Application Programming Interface

– R –

REST: Representational State Transfer.

Document Description

|  |  |  |
| --- | --- | --- |
| Title | CALS SIM : API Specification | |
| Creation date | 26/01/2016 | |
| Publication date | 28/01/2016 | |
| Product Owner | Saqib Ahmed |  |
| Authors | Jeremy Harrault | hajr15bp@ju.se |
|  |  |
| Subject | API Specification | |
| Model version | NA | |
| Document version | 1.0 | |

Revisions table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Rev. | Author | Modified Section(s) | Comments |
| 29/02/16 | 1.0 | Jeremy Harrault | All | Define API architecture, function and the static view for each of them. |

Table of Contents

[1. Description of the API 1](#_Toc444535072)

[1.1. REST architecture 1](#_Toc444535073)

[1.2. API keys 2](#_Toc444535074)

[2. Resource View 1](#_Toc444535075)

[3. Implementation 2](#_Toc444535076)

[3.1. Log event 2](#_Toc444535077)

[3.1.1. Interface 2](#_Toc444535078)

[3.1.2. Format and possible values 3](#_Toc444535079)

[3.2. Entity Creation 1](#_Toc444535080)

[3.2.1. Interface 1](#_Toc444535081)

[3.3. Generic errors 2](#_Toc444535082)

List of Tables

[Table 1: Explanation on request and response attributes. 1](#_Toc444535120)

[Table 2: Resource view for CALS-NARMS API 1](#_Toc444535121)

[Table 3: Interface for log event 3](#_Toc444535122)

[Table 4: Interface for entity creation 2](#_Toc444535123)

[Table 5: Generic error of the server 2](#_Toc444535124)

List of Figures

[Figure 1: Composition of HTTP request and response 1](#_Toc444535125)

# Description of the API

## REST architecture

The SAFAPS SIM API fulfil a RESTful architecture. It is reachable using the HTTP protocol. It means that each function offered by the API can be executed by sending an HTTP request and return an HTTP response.

Below is the basic composition of any HTTP response and request.

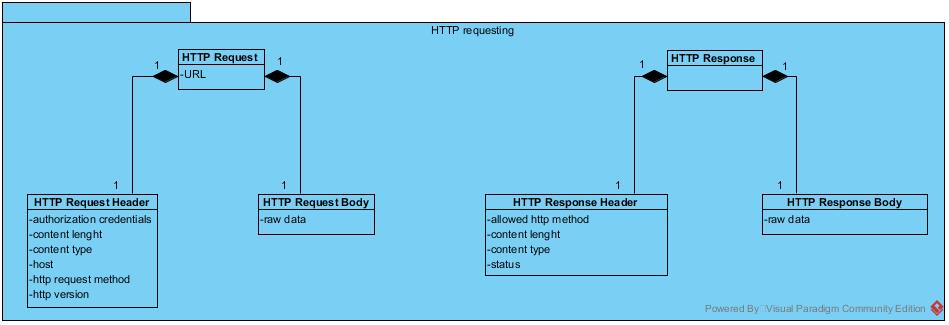


Figure 1: Composition of HTTP request and response

|  |  |  |
| --- | --- | --- |
| Request | Route | This identify the resource on the server to perform an action on |
| Authorization credentials | This field will be used to store the encrypted key allowing the API to authenticate a facility within the system. (cf. 1.2. API keys). |
| Content length | This field contains the number of characters contained in the request body. |
| Content type | This field contains the format of the data sent in the request body. |
| Host | This field contains the host of the requested server |
| HTTP request method | This field specify the action to perform on the route. (GET/POST/PUT/DELETE/etc.) |
| HTTP version | The version the HTTP protocol to be used for the request |
| Raw data | The situational data to send with the request. |
| Response | Allowing HTTP method | The authorized action to perform on the requested resource |
| Content length | This field contains the number of characters contained in the response body. |
| Content type | This field contains the format of the data sent in the response body. |
| Status | This field contains an integer identifying the type of response (OK, redirection, client error, server error). |
| Raw data | The situational data to send back to the caller |

Table 1: Explanation on request and response attributes.

## API keys

In CALS, the facilities will have an API key that will allow them to connect and send request to the NARMS API. This API key shall be stored both on NARMS and CALS side.

# Resource View

|  |  |  |
| --- | --- | --- |
| Resource | HTTP Method | Description |
| /log\_events | POST | Create a log event |
| /users | POST | Create a new CALS user (NARMS worker profile) |
| /workstations | POST | Create a new workstation |

Table 2: Resource view for CALS-NARMS API

# Implementation

## Log event

### Interface

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request | | | Response | |
| Resource | HTTP Method | Body | HTTP Status | Body |
| /log\_events | POST | {  “event\_type”: “[EventType]”,  “controller\_role”: ”[ControllerRole]”,  “controller\_resp”: ”[ControllerResp]”,  “operational\_status”: ”[OperationalStatus],  “date”: “[Date]”,  “user\_id”: “[UserPubId]”,  “facility\_id”: “[FacilityPubId]”,  “workstation\_id”: “[WorkstationPubId]”  } | 200 | {  “type”: “[event\_type]”,  “controller\_role”: ”[controller\_role]”,  “controller\_resp”: ”[controller\_resp]”,  “operational\_status”: ”[operational\_status],  “date”: “[date]”,  “user\_id”: “[user\_id]”,  “[facility\_id]”: “[facility\_id]”,  “[workstation\_id]”: “[workstation\_id]”  } |
| 400 | {  “Error”: “One or several attributes are missing: “[ev\_missing\_attr1]”, “[ev\_missing\_attr2]”, …  }  \*The [ev\_missing\_attrn] refers to the name of the missing attribute. |
| 401 | {  “Error”: “The authorization field is not specified or the API key is not valid”  } |
| 404 | {  “Error”: “The specified user is unknown”  } |
| 404 | {  “Error”: “The specified facility is unknown”  } |
| 404 | {  “Error”: “The specified workstation is unknown”  } |

Table 3: Interface for log event

### Format and possible values

* EventType
  + “login”
  + “logout”
  + “change\_status”
* ControllerRole
  + “PE” for Procedural Enroute
  + “RT” for Radar Terminal
  + “RA” for Radar Arrivals
  + “RD” for Radar Departures
  + “GP” for Ground Procedural
  + “LP” for Local Procedural
* ControllerResp (Responsibility)
  + “Planning”: approving flight strips
  + “Tactical”: managing flights
  + “Weather”: predicting future flight loading
* OperationalStatus
  + “SC” for Solo controller
  + “MCU” for Multi controller – Unsupported
  + “MCS” for Multi controller – Supported
  + “MCM” for Multi controller – Mentoring
  + “MCT” for Multi controller – Trainee
  + “MCI” for Multi controller – instructor
* Date
  + “YYYY-MM-dd hh:mm:ss”
* UserPubId
  + Public ID of the user
* FacilityPubId
  + Public ID of the facility
* WorkstationPubId
  + Public ID of the workstation

## Entity Creation

### Interface

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request | | | Response | |
| Resource | HTTP Method | Body | HTTP Status | Body |
| /users | POST | {  “id”: [UserPubId],  “name”: [UserName],  “facility\_id”: [FacilityPubId]  } | 200 | {  “id”: [UserPubId],  “name”: [UserName]  } |
| 400 | {  “Error”: “One or several attributes are missing: “[ev\_missing\_attr1]”, “[ev\_missing\_attr2]”, …  }  \*The [ev\_missing\_attrn] refers to the name of the missing attribute. |
| 401 | {  “Error”: “The authorization field is not specified or the API key is not valid”  } |
| 404 | {  “Error”: “The specified facility is unknown”  } |
| /workstation | POST | {  “id”: [WorkstationPubId],  “name”: [WorkstationName],  “facility\_id”: [FacilityPubId]  } | 200 | {  “id”: [WorkstationPubId],  “name”: [WorkstationName]  } |
| 400 | {  “Error”: “One or several attributes are missing: “[ev\_missing\_attr1]”, “[ev\_missing\_attr2]”, …  }  \*The [ev\_missing\_attrn] refers to the name of the missing attribute. |
| 401 | {  “Error”: “The authorization field is not specified or the API key is not valid”  } |
| 404 | {  “Error”: “The specified facility is unknown”  } |

Table 4: Interface for entity creation

## Generic errors

Some generic errors can be raised from the server. These errors are not related to any specific route but to the request received but the server.

|  |  |  |
| --- | --- | --- |
| HTTP status | Response body | Explanation |
| 404 | {  “Error”: “The requested resource cannot be found”  } | The resource pointed by the request does not exist. |
| 405 | {  “Error”: “This action cannot be performed on this resource”  } | The HTTP method used in the request cannot be performed on the resource pointed by the request. |

Table 5: Generic error of the server