Flight Information System API doc

Confidential - Internal use only

Authors: Pum Walters, Ivo Nijhuis
Version: 4.2
Date: October 1, 2017

Contents

1. Introduction 3
1.1 Flight information system environments 3
1.2 Authentication 3
2. Retrieving information from the flight information system 4
2.1 Passengers 4
2.2 Flights 5
2.3 Tickets 6
3. Storing partner data in the flight information system 7
3.1 Passengers 7
3.2 Tickets 8
4. Generating test data 9
4.1 Add passenger 9
4.2 Add random passengers 10
4.3 Add flight 11
4.4 Add random flights 12
4.5 Add ticket 13
4.6 Add random tickets 14
5. Result codes 15

1. Introduction

The Corendon flight information system provides an API to retrieve and manipulate some of its data. By means of a so-called REST interface, information about passengers, flights and tickets can be retrieved. Furthermore, additional data can be stored for any passenger or ticket. This functionality can be used by Corendon's partners to maintain relevant application-dependent information about passengers or their tickets.

The contents of the REST messages are based on JSON. The REST API processes the JSON embedded in the HTTP request, and responds with a JSON object which represents the result (or information on a possible error if that occurred).

In this document you'll find a listing of the currently available REST requests.

1.1 Flight information system environments

Separate instances of the flight information system are available for development and production purposes. Our development teams can only access the development server in order to develop and test their applications. The production server is critical to support our daily flights and operations, and for that reason, cannot be accessed by developers. It is maintained exclusively by the ICT Operations team.

The development environment of the flight information system can be reached at the following location:

Base URL: **http://fys.securidoc.nl/**

Port number: **11111**

1.2 Authentication

Usage of the flight information system is restricted. Each request to access the system needs to include an authentication, and only authorised requests will be processed. Each development team will receive a username and password in the form of a `teamId` and `teamKey`. These values need to be included in every REST request.

2. Retrieving information from the flight information system

2.1 Passengers

Returns a list of all passengers.

URL: **http://fys.securidoc.nl:11111/Passenger**

Method: **POST**

Function: **List**

Request =

`{`
`  "function": "List",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>"`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request

Reply =

`{`
`  "passengers": {`
`  "<pid-1>": {`
`  "pid": "<unique-id-1>",`
`  "firstName": "<FirstName-1>",`
`  "lastName": "<LastName-1>",`
`  "<teamId>": <PartnerData-1>`
`  },`
`  …`
`  "<pid-N>": {`
`  "pid": "<unique-id-N>",`
`  "firstName": "<FirstName-N>",`
`  "lastName": "<LastName-N>",`
`  "<teamId>": <PartnerData-N>`
`  }`
`  },`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	unique-id of the passenger
	first name of the passenger
	surname of the passenger
	any JSON object, partner-managed
	
	id from corresponding request
	0 is OK

2.2 Flights

Returns a list of all flights.

URL: **http://fys.securidoc.nl:11111/Flight**

Method: **POST**

Function: **List**

Request =

`{`
`  "function": "List",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>"`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request

Reply =

`{`
`  "flights": {`
`  "<flight-id-1>": {`
`  "fid": "<flight-id-1>",`
`  "flightFrom": "<FlightFrom-1>",`
`  "flightTo": "<FlightTo-1>",`
`  "flightNumber": "<FlightNumber-1>"`
`  },`
`  …`
`  "<flight-id-N>": {`
`  "fid": "<unique-id-N>",`
`  "flightFrom": "<FlightFrom-N>",`
`  "flightTo": "<FlightTo-N>",`
`  "flightNumber": "<FlightNumber-N>"`
`  }`
`  },`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	unique-id of the flight
	departure airport
	destination airport
	flight number
	
	id from corresponding request
	0 is OK

2.3 Tickets

Returns a list of all tickets with references to corresponding flight and passenger.

URL: **http://fys.securidoc.nl:11111/Ticket**

Method: **POST**

Function: **List**

Request =

`{`
`  "function": "List",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>"`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request

Reply =

`{`
`  "tickets": {`
`  "<ticket-id-1>": {`
`  "tid": "<ticket-id-1>",`
`  "ticketNumber": "<ticket-num-1>",`
`  "pid": "<psngr-id>",`
`  "fid": "<flight-id>",`
`  "<teamId>": <PartnerData-1>`
`  },`
`  …`
`  "<ticket-id-N>": {`
`  "tid": "<ticket-id-N>",`
`  "ticketNumber": "<ticket-num-N>",`
`  "pid": "<psngr-id>",`
`  "fid": "<flight-id>",`
`  "<teamId>": <PartnerData-N> }`
`  }`
`  },`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	unique-id of the ticket
	human readable id
	unique-id of the passenger
	unique-id of the flight
	any JSON object, partner-managed
	
	id from corresponding request
	0 is OK

3. Storing partner data in the flight information system

Corendon's database allows our ICT development partners to store their own information, relevant for the processes they implement. Every partner has a unique TeamId and TeamKey. In all requests the TeamKey and TeamId must match for the call to be processed. In addition, the TeamId is used to identify proprietary partner data. That is, the partner data is stored in a field in Passenger and Ticket objects. The name of that field is the TeamId, and the value can be any JSON object. Management of this data is the partner's responsibility.

3.1 Passengers

Store partner data in Passenger object for the identified partner.

URL: **http://fys.securidoc.nl:11111/Passenger**

Method: **POST**

Function: **SetPartnerData**

Request =

`{`
`  "function": "SetPartnerData",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>",`
`  "pid" : "<PassengerID>",`
`  "<teamId>" : <PartnerData>`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request
	id of this passenger
	any JSON object, partner-managed

Reply =

`{`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	id from corresponding request
	0 is OK

3.2 Tickets

Store partner data in Ticket object for the identified partner.

URL: **http://fys.securidoc.nl:11111/Ticket**

Method: **POST**

Function: **SetPartnerData**

Request =

`{`
`  "function": "SetPartnerData",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>",`
`  "tid" : "<TicketID>",`
`  "<teamId>" : <PartnerData>`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request
	id of this ticket
	any JSON object, partner-managed

Reply =

`{`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	id from corresponding request
	0 is OK

4. Generating test data

A number of additional requests is available only in the development environment of the flight information system. These requests allow developers to fill the database with data for passengers, flights and tickets. Developers can submit specific data for testing purposes or have the database automatically populated with randomly generated data. (Please note that the development environment for any new project will be initially delivered with an empty database.)

The production environment of the flight information system contains data about real passengers, flights and tickets. This data should not be modified by third parties. The requests described in this chapter are therefore unavailable in the production environment and should not be included in any code that is intended to go live.

4.1 Add passenger

Add a new passenger.

URL: **http://fys.securidoc.nl:11111/Manage**

Method: **POST**

Function: **Add-Passenger**

Request =

`{`
`  "function": "Add-Passenger",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>",`
`  "lastName" : "<LastName>",`
`  "firstName" : "<FirstName>"`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request
	surname of this passenger
	first name of this passenger

Reply =

`{`
`  "pid": "<unique-id>",`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	the unique-id assigned to this passenger
	id from corresponding request
	0 is OK

4.2 Add random passengers

Fill the database with randomly generated new passengers. All previous database entries will be overwritten.

URL: **http://fys.securidoc.nl:11111/Manage**

Method: **POST**

Function: **Random-Fill-Passengers**

Request =

`{`
`  "function": "Random-Fill-Passengers",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>"`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request

Reply =

`{`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	id from corresponding request
	0 is OK

4.3 Add flight

Add a new flight.

URL: **http://fys.securidoc.nl:11111/Manage**

Method: **POST**

Function: **Add-Flight**

Request =

{	
"function": "Add-Flight",	fixed
"teamId": "<teamId>",	team-id for this request e.g. IN109-1
"teamKey": "<teamKey>",	team-key, uniquely supplied key
"requestId": "<request-id>",	id for this request
"flightFrom": "<depart airport>",	airport of departure
"flightTo": "<arrival airport>",	airport of arrival
"flightNumber": "<unique nr>"	unique identifier of flight e.g. 'CND 115'
}	

Reply =

{	
"fid": "<unique-id>",	the unique-id assigned to this flight
"requestId": "<request-id>",	id from corresponding request
"result": "<result-code>",	0 is OK
"resultText": "<result-text>"	
}	

4.4 Add random flights

Fill the database with randomly generated new flights. All previous database entries will be overwritten.

URL: **http://fys.securidoc.nl:11111/Manage**

Method: **POST**

Function: **Random-Fill-Flights**

Request =

`{`
`  "function": "Random-Fill-Flights",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>"`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request

Reply =

`{`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	id from corresponding request
	0 is OK

4.5 Add ticket

Add a new ticket for a specified passenger on a specified flight.

URL: **http://fys.securidoc.nl:11111/Manage**

Method: **POST**

Function: **Add-Ticket**

Request =

<pre>{
"function": "Add-Ticket",
"teamId": "<teamId>",
"teamKey": "<teamKey>",
"requestId": "<request-id>",
"ticketNumber": "<TicketNumber>",
"pid": "<Id-Passenger>",
"fid": "<Id-Flight>"
}</pre>
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request
	unique number for this ticket
	unique-id of passenger
	unique-id of flight

Reply =

<pre>{
"tid": "<unique-id>",
"requestId": "<request-id>",
"result": "<result-code>",
"resultText": "<result-text>"
}</pre>
	the unique-id assigned to this ticket
	id from corresponding request
	0 is OK

4.6 Add random tickets

Fill the database with randomly generated new tickets. All previous database entries will be overwritten.

URL: **http://fys.securidoc.nl:11111/Manage**

Method: **POST**

Function: **Random-Fill-Tickets**

Request =

`{`
`  "function": "Random-Fill-Tickets",`
`  "teamId": "<teamId>",`
`  "teamKey": "<teamKey>",`
`  "requestId": "<request-id>"`
`}`
	fixed
	team-id for this request e.g. IN109-1
	team-key, uniquely supplied key
	id for this request

Reply =

`{`
`  "requestId": "<request-id>",`
`  "result": "<result-code>",`
`  "resultText": "<result-text>"`
`}`
	id from corresponding request
	0 is OK

5. Result codes

For a successful request, the result code will be 0. All other result codes indicate that an error has occurred.

Result code	Status
0	OK
1	IO error
2	Database could not be opened
3	SQL error
4	No connection with database
5	Error closing database
6	Database context error
7	Unknown function
8	Element already existing
9	Function not implemented
10	Unknown request
11	Element not existing
12	Invalid key
13	Unknown error