EUMETSAT

Training Database API - Manual

Doc.No.	:	EUM/USC/TEN/17/929102
Issue	:	V2 Draft
Date	:	5 March 2019
WBS	:	

EUMETSAT
Eumetsat-Allee 1, D-64295 Darmstadt, Germany
Tel: +49 6151 807-7
Fax: +49 6151 807 555
http://www.eumetsat.int

Document Signature Table

	Name	***Function***	***Signature***	***Date***
Prepared by:	Joaquin Rodriguez Guerra	Archive and Data Services Consultant		
Reviewed by:	Mark Higgins	Training Manager		
Approved by:	Harald Rothfuss	Data Centre Operations Manager		

Distribution List

Distribution list	
Name	***No. of Copies***
As indicated in signature table.	Electronic Distribution

Document Change Record

Issue / Revision	***Date***	***DCN. No***	***Summary of Changes***
V1	04/07/2017		Initial version.
V2	05/03/2019		Added sections on security, configuration, email service, authentication and authorization API endpoints

Table of Contents

1 INTRODUCTION

1.1 Purpose

The aim of this document is to provide a detailed description of the Training Database API developed within USC. It currently serves as the data source to different Training Calendar portals, such as the EUMETSAT Training Events site (trainingevents.eumetsat.int)

1.2 Scope

The document is intended for the API clients and developers that have to maintain the tool or that would like to extend it to store new metadata.

1.3 Applicable Documents7

No applicable documents.

1.4 Reference Documents

AD-1	Training Database API - TN on the Web Service Approach	EUM/USC/TEN/16/871419

1.5 Notes

Please understand that the screenshots provided in the User Manual section might correspond to older versions of the tool. Their purpose is to illustrate the operations, but details might not be in accordance with the newest versions.

1.6 Document Structure

Section 1: General information (this section)

Section 2: Describes the design and components used.

Section 3: Describes the build process and the source code particularities.

Section 4: Describes the steps to be followed for installing the tool.

Section 5: Describes the API operations.

Section 6: Describes the current deployments in EUMETSAT.

2 SYSTEM DESIGN

2.1 System overview

The Training Database API is a web-based application composed by a MySQL database for storing training events metadata and a REST web service API to allow thirds parties access the Events data. The goal of this approach is to separate the data and the front-ends, allowing each organisation to build their own front-end or reports compliant with their own requirements.

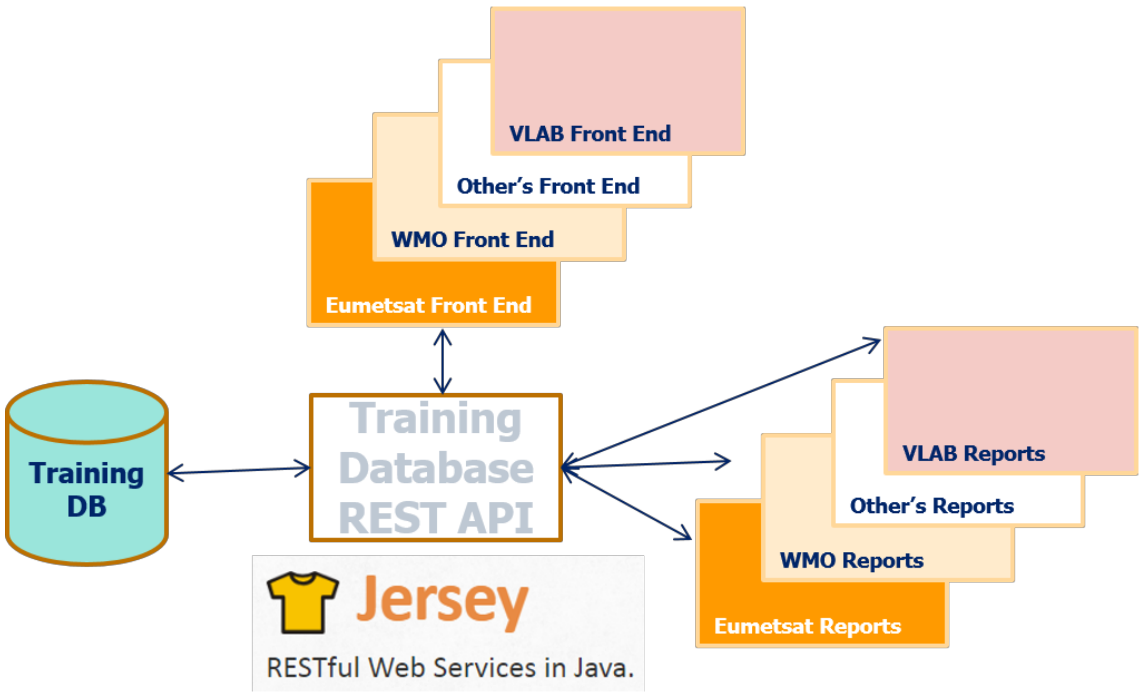
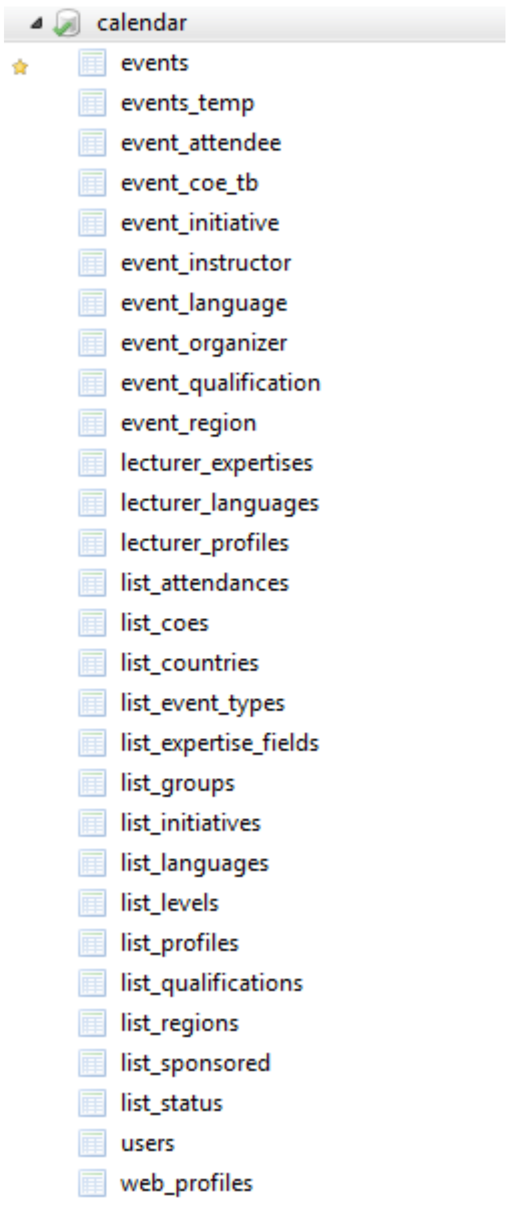


Figure 1 System components

2.2 Training database

The system relies on a MySQL database as the main data store. The schema is quite simple, as it was thought to store only metadata about events and its participants. Calendar, as it is called, is composed of two main tables, events and users, and a set of auxiliary tables.



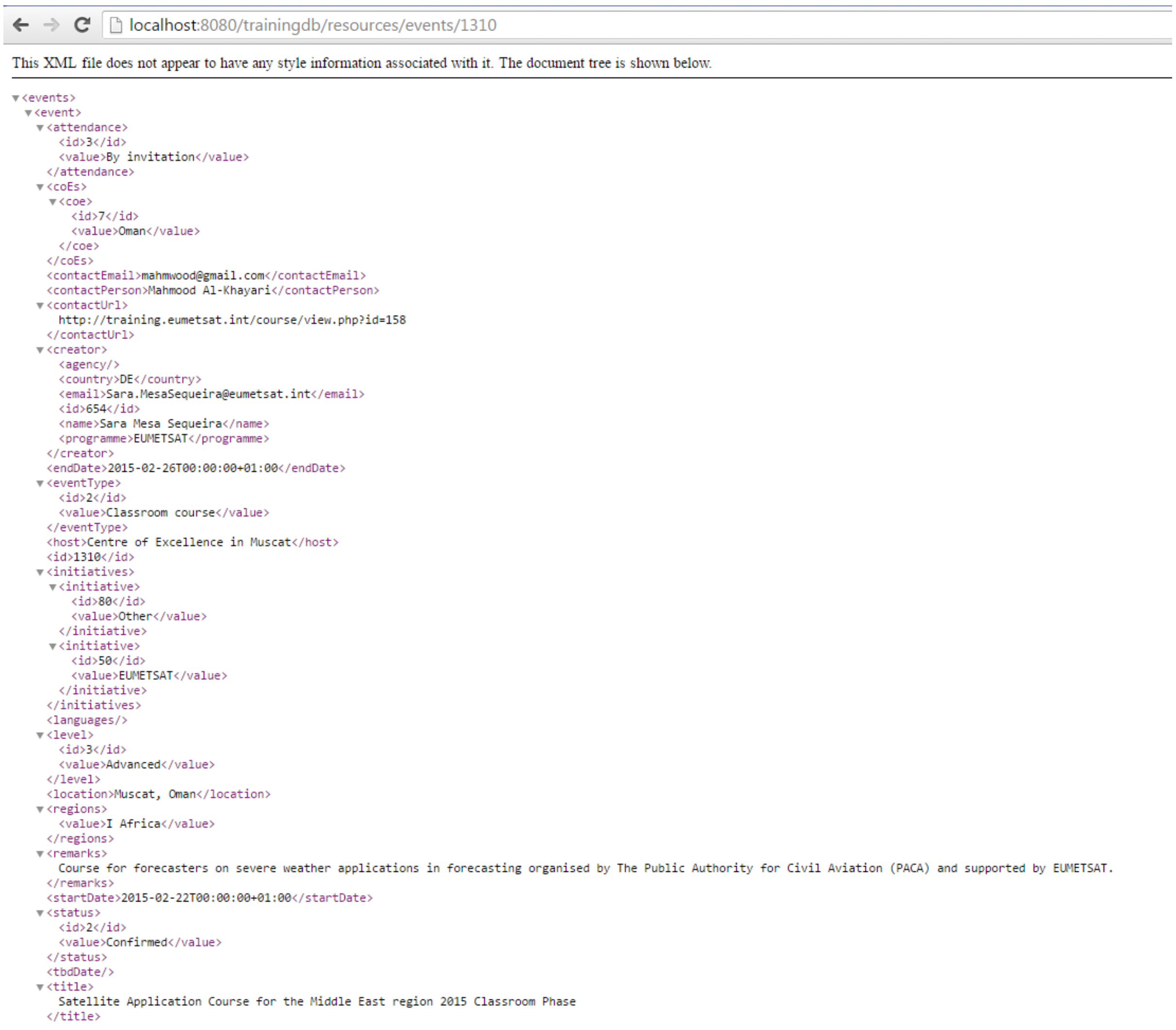
- calendar
  - events
  - events_temp
  - event_attendee
  - event_coe_tb
  - event_initiative
  - event_instructor
  - event_language
  - event_organizer
  - event_qualification
  - event_region
  - lecturer_expertises
  - lecturer_languages
  - lecturer_profiles
  - list_attendances
  - list_coes
  - list_countries
  - list_event_types
  - list_expertise_fields
  - list_groups
  - list_initiatives
  - list_languages
  - list_levels
  - list_profiles
  - list_qualifications
  - list_regions
  - list_sponsored
  - list_status
  - users
  - web_profiles

All tables with the prefix “event_” are one-to-many or many-to-many and are used to store metadata about the events through the use of foreign keys. All tables with the prefix “list_” are independent configuration tables, used to store the lists of valid values for particular metadata fields. Some additional tables were added later for particular purposes, like “web_profiles”, to support the System User Management.

If support for recording additional metadata was to be added into the System, new columns or tables should be added to the proposed schema. An SQL file with the statements for creating the database and populating the configuration tables is delivered within the source code package.

2.3 REST API

The REST API web service module is an interface to the data store made available through HTTP to offer users the possibility of data retrieval or means to execute operations on the data, using popular internet formats such as XML or JSON as the transportation language. As it accepts HTTP requests, any web browser is capable of accessing the service for basic resource retrieval operations, through GET requests (normal URL browsing).


localhost:8080/trainingdb/resources/events/1310

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<events>
  ▼<event>
    ▼<attendance>
       <id>3</id>
       <value>By invitation</value>
     </attendance>
    ▼<coEs>
      ▼<coe>
         <id>7</id>
         <value>Oman</value>
       </coe>
     </coEs>
     <contactEmail>mahmwood@gmail.com</contactEmail>
     <contactPerson>Mahmood Al-Khayari</contactPerson>
    ▼<contactUrl>
       http://training.eumetsat.int/course/view.php?id=158
     </contactUrl>
    ▼<creator>
       <agency/>
       <country>DE</country>
       <email>Sara.MesaSequeira@eumetsat.int</email>
       <id>654</id>
       <name>Sara Mesa Sequeira</name>
       <programme>EUMETSAT</programme>
     </creator>
     <endDate>2015-02-26T00:00:00+01:00</endDate>
    ▼<eventType>
       <id>2</id>
       <value>Classroom course</value>
     </eventType>
     <host>Centre of Excellence in Muscat</host>
     <id>1310</id>
    ▼<initiatives>
      ▼<initiative>
         <id>80</id>
         <value>Other</value>
       </initiative>
      ▼<initiative>
         <id>50</id>
         <value>EUMETSAT</value>
       </initiative>
     </initiatives>
     <languages/>
    ▼<level>
       <id>3</id>
       <value>Advanced</value>
     </level>
     <location>Muscat, Oman</location>
    ▼<regions>
       <value>I Africa</value>
     </regions>
    ▼<remarks>
       Course for forecasters on severe weather applications in forecasting organised by The Public Authority for Civil Aviation (PACA) and supported by EUMETSAT.
     </remarks>
     <startDate>2015-02-22T00:00:00+01:00</startDate>
    ▼<status>
       <id>2</id>
       <value>Confirmed</value>
     </status>
     <tbdDate/>
    ▼<title>
       Satellite Application Course for the Middle East region 2015 Classroom Phase
     </title>
```

It runs on Apache Tomcat, linked to the Calendar database, and the main function is to provide third parties with access to the database in a secure way. Experts highly discourage the use of internet connections directly to the database because of safety reasons and it is a common practice to implement this layer to add access control and security.

2.3.1 Resources

The focus of a REST Web Service is on resources and how to provide access to these resources. In particular, the Training API web service maps resources to database tables. As in the database, the main resources are the events and the users, but some others are also provided to support the main ones: course applications, WMO regions, countries, languages, event types, expertise fields, groups, profiles, email alerts, etc.

In REST, every resource is uniquely identified by a URI (Uniform Resource Identifier), and the Web Service uses a directory hierarchy to address its resources. For instance, the event with id=1 is referred with the following URI: https://<host>/trapi/resources/public/events/1

The operation to execute on a certain resource or collection of resources is determined by the HTTP method or verb. With them, CRUD operations support are provided for the tables in the Training Database.

Verb	**Description**
GET	Fetch a resource
PUT	Create a new resource (with a known ID)
DELETE	Delete a resource
POST	Update/Create a new resource (unknown ID)
OPTIONS	List supported operations on a resource

The following table shows examples of the HTTP requests for the most common operations. Note that there are private and public versions of the resources. The public URLs are only used for retrieval of data, while the private URLs allow the modification of the resources, and thus, require user authentication.

Examples
GET https://usc.tools.eumetsat.int/trapi/resources/public/events
Return the list of all the events in the database
GET https://usc.tools.eumetsat.int/trapi/resources/public/events?location=online
Return the list of all the events with location=online
POST https://usc.tools.eumetsat.int/trapi/resources/private/events
Create a new event
DELETE https://usc.tools.eumetsat.int/trapi/resources/private/users/34
Delete user with id=34
POST https://usc.tools.eumetsat.int/trapi/resources/private/events/233
Update event with id=233

2.3.2 Technologies

The REST API web service is written in Java and runs in the Apache Tomcat web application container. The application is built on top of two Frameworks:

- Hibernate: the object-relational java library is used to map an object-oriented domain model, such as the REST resources, to a relation database, such as Calendar. It takes care of the persistence of the objects, by setting the mapping according to the JPA annotations defined in the POJO classes.

- Jersey: java framework to build REST web services. It helps on mapping an object-oriented domain model, such as the Training Events database objects (POJOs), to resources available through URLs.

2.3.3 Security

The data stored in the Training Database is partially public and partially private and thus needs to be protected with user authentication. Resources under the /public/ tree of URLs are considered public and are open. In the other hand, resources under the /private/ tree of URLs are protected.

2.3.3.1 Authentication

An email-only approach has been chosen for the user authentication. Users can login and obtain access to restricted resources in the API by sending their email address to a specific API endpoint, e.g. via the UI of the tool using TRAPI. An email is then sent to this email address, containing an authentication token. This authentication token is exchanged for an authorization token using a second API endpoint. It needs to be ensured that this authorization token is included with subsequent requests to the API in order to gain access to restricted resources. This password-less approach is based on OAuth2. Consequently, instead of relying on the security of user chosen credentials, this approach relies on the security of the user's email account. This is as safe as any password reset methods using a user's email address. To increase the security, the authentication token is only valid for a short period of time and will not be accepted by the system after it expired.

2.3.3.2 Authorization

In order to control access to the protected URLs, an authorization system is used that consists of permissions, roles and groups. Such a system allows to only give access to certain operations for a specific group of people, either from a specific agency or with a certain role. This system is based on a *role-based access control* system. Figure 2 gives a schematic overview of the authorization system.

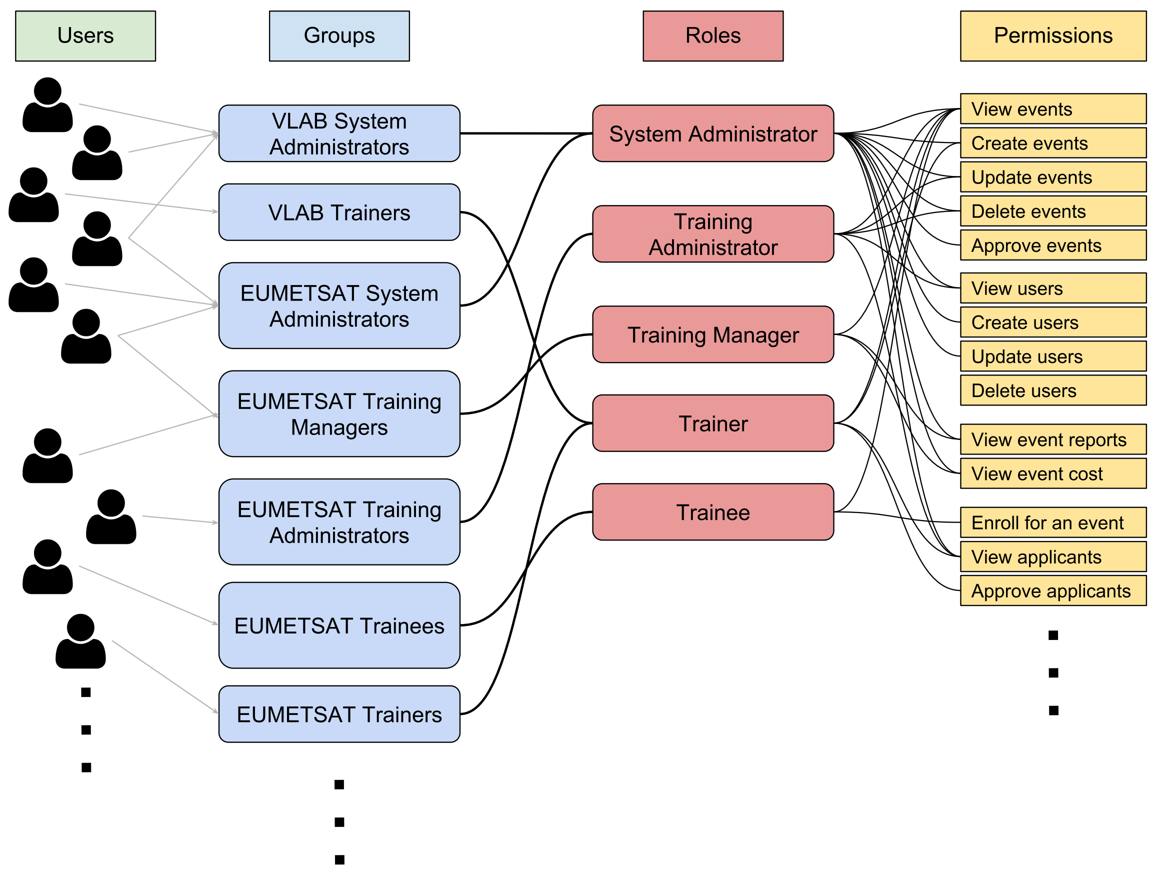


Figure 2 Role-based Access Control

The different components of the role-based access control system are:

- **Permissions**: a permission is a low-level privilege within the system, which defines a particular operation, subject to authorization, that can be performed such as “view event”, “create user” or “generate report”, etc.
- **Roles**: a role is a standard grouping of permissions, e.g. for a Trainer or System Administrator. Roles are not assigned to users directly, but to groups, so that access can be managed on an organisation (e.g. group) level.
- **Groups**: a group limits the permission of a role to certain resources. A group can be equal to an organisation, e.g. VLAB or EUMETSAT. A user in one group can only alter items from the resources of that group, e.g. a System Administrator in the VLAB group can only perform system administration tasks for VLAB items.
- **Users**: a user can be part of multiple groups. A user can be assigned e.g. to both the VLAB system administrator group, and the EUMETSAT trainer.

2.3.3.3 Implementation

There are a few options commonly used to provide security and authentication to REST services: OAuth, OpenId, HTTP Basic Authentication, web.xml security, JAX-RS annotations, Jesery OAuth, etc. The current implementation is based on OAuth2, ensuring security and flexibility.

Authorization requires an authorization token to gain access to restricted resources. Authorizing a user requires the following steps:

1. A specific API endpoint (see section 5.3.1) is called via the client with the user's email address as payload;
2. The authentication token is sent to the provided email address;
3. The client requests an authorization token by calling a second API endpoint (as described in section 5.3.2) using the authentication token as payload;
4. If the authentication token is validated, it is exchanged with an authorization token which is returned to the client.

The authorization token shall be included with requests to restricted resources. It shall be base64-encoded and added to the Authorization header using the 'xBearer' prefix. The following headers need to be used for authorized requests:

```
'WWW-Authenticate': 'xBearer'
'Authorization': 'xBearer <base64-encoded authorizationToken>'
```

In order to base-64 encode a token, it needs to be parsed from a text-based string to a binary string. Many languages have helper functions to do so. For example, to encode the authorization token in JavaScript, the following code can be used:

```
var encodedTok = btoa(unescape(encodeURIComponent(<authorizationoken>))));
```

If the web service receives a request for a protected resource, it rejects the request with an HTTP status code 401 (access denied) and sets the WWW-Authenticate response header. If the web service receives a request for a protected resource, with the Authorization header correctly set, the web service responds with an HTTP status code 200, which indicates that the request succeeded and that the requested information is in the response.

2.3.4 Email Service

The rest API contains a simple SMTP email service that is used by the API internally to send email notifications about certain system aspects to end users. This service either sends a notification to a single recipient, e.g. direct mode, or send an email to a list of recipients, e.g. newsletter mode. The service requires correct SMTP settings provided in the application.properties file. An explanation to the properties file is given in section 4.2.2.

2.4 Third party front ends

EUMETSAT is developing a web front-end to make the trainings events information available to its users. The first phase has already been completed and a prototype is running in the following URL:

http://trainingevents.eumetsat.int/

3 SOURCE CODE MAINTENANCE

3.1 SVN repository

All the code is maintained as an Eclipse project and the control versioning is done through Subversion (EUMETSAT internal).

The URL for accessing the code is: http://tcsvn/USCTT/

As usual, the development version is located under the trunk directory. Tags are created for each version in the tags directory.

3.2 Eclipse project

Eclipse has been used for maintaining the code and the SQL scripts. The project contains the same directory structure as the source code package delivered. The source code package is prepared to be imported in Eclipse as a maven project.

3.3 Building the project

The project uses the build automation tool Maven to build and produce the Web Service WAR file. A pom.xml file is located in the root directory with the build configurations and the list of dependencies.

Three build configuration are defined in the Eclipse project:

- DEV: with the goal: clean install –Ddev, it generates a WAR file ready to be deployed in the DEV environment, which is a Windows machine.
- VAL: with the goal: clean install –Dval, it generates a WAR file ready to be deployed in the VAL environment, which is a Linux machine in the TCE.
- OPE: with the goal: clean install –Dope, it generates a WAR file ready to be deployed in the OPE environment, which is a Linux virtual machine with internet access.

The main difference among the WAR files produced by the different build configurations are the following files:

- Log4j.properties: the log level and paths are set depending on the environment.
- Persistence.xml: the Lucene index base path depends on the environment (OS)
- application.properties: contains application-specific settings, like settings for the email service and authentication timeouts.

4 INSTALLATION

The installation consists of two steps: the database and the web service. Both are independent from each other and can be installed in any order, but both are needed for the system to work.

4.1 Environment Requirements

The installation procedure below assumes MySQL and Apache Tomcat (> v 7.0) are running in the system. Please check their website for installation instructions (https://tomcat.apache.org/ and https://www.mysql.com/). Any Operating System supporting these servers will support the application as well. Linux SLES 12 and Windows 7 have been used in the EUMETSAT environment. The Apache Tomcat version used was 8.0.45, running on top of Java JDK 1.7.

4.2 Installation Procedure

4.2.1 Database Installation

Two SQL scripts are provided in the source code package:

- Database_creation.txt
- Database_vX.Y.sql

The first script contains a sequence of commands to:

1. Create an empty database in the MySQL server called calendar
2. Create the user calendarUser and provide him with privileges for calendar
3. Import the database schema into the calendar database

The second script contains an export of the database schema and is used in the 3rd step of the first script. It contains only tables and configuration values, and no data, like users or events, is provided.

Please follow the instruction in the first script to install the database into the MySQL server.

4.2.2 REST API Installation

The installation of the REST API consists on deploying the application WAR file in Apache Tomcat. With the software delivery, a WAR file is provided, with the name trapi_vX.Y.war. Please rename it to trapi.war before the deployment.

By default it is configured to work in Linux and connect to the calendar database, assuming it has been created on the same host, and with the credentials given in the Database_creations.txt script.

Some configuration parameters can be modified in the WAR file before deployment by editing some files inside. The files containing configuration parameters are listed below. Software like 7-ZIP allows the edition of the WAR files without the need to extract/re-pack the package.

<WAR_FILE>\WEB-INF\classes\META-INF\persistence.xml
• The property hibernate.search.default.indexBase must point to a valid directory. By default, it is pointing to /var/lucene/indexes/trapi_ope. It shall be changed if the WAR will be deployed in Windows.
• Properties starting with the prefix hibernate define the connection to the database. Adjust if the database has been installed in a remote machine.

<WAR_FILE>\WEB-INF\classes\application.properties
• The property auth.token.expirytime.seconds defines after how many seconds authentication tokens, that are sent to users by email, are invalidated
• Properties starting with the prefix email.smtp define the connection to the SMTP host, used by the email service. Adjust these properties to point to an SMTP host that is reachable from the machine onto which TRAPI is installed.
• The property email.from.address defines the email address that is used as a “from” address by emails sent out by the API.
• The property email.whitelist.regex allows to enter a JAVA-based regular expression that filters email addresses to which the API sends emails. E.g. when the property is set to **@eumetsat.int*, emails will only be sent to email addresses that end with the postfix *@eumetsat.int*.
• The property application.datetimeformat defines the dates and times are formatted for display to the user.

The deployment of a WAR file in Tomcat is achieved by copying the WAR file into the directory /<tomcat_installation_dir/webapps. If the deployment is successful the following log message is shown in the log file /<tomcat_installation_dir/logs/catalina.out:

/<tomcat_installation_dir/logs/catalina.out:

```
Deploying web application archive /srv/tomcat/webapps/trapi.war

org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet
contained no TLDs. Enable debug logging for this logger for a complete list of JARs that were
scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve
startup time and JSP compilation time.

org.apache.catalina.startup.HostConfig.deployWAR Deployment of web application archive
/srv/tomcat/webapps/trapi.war has finished in 15,323 ms
```

The WAR file can also be obtained by building the project using the Maven scripts.

4.3 Test Installation

To verify if the installation is successful, open a web browser and type in the URL for the events resource:

http://localhost:8080/trapi/resources/public/events

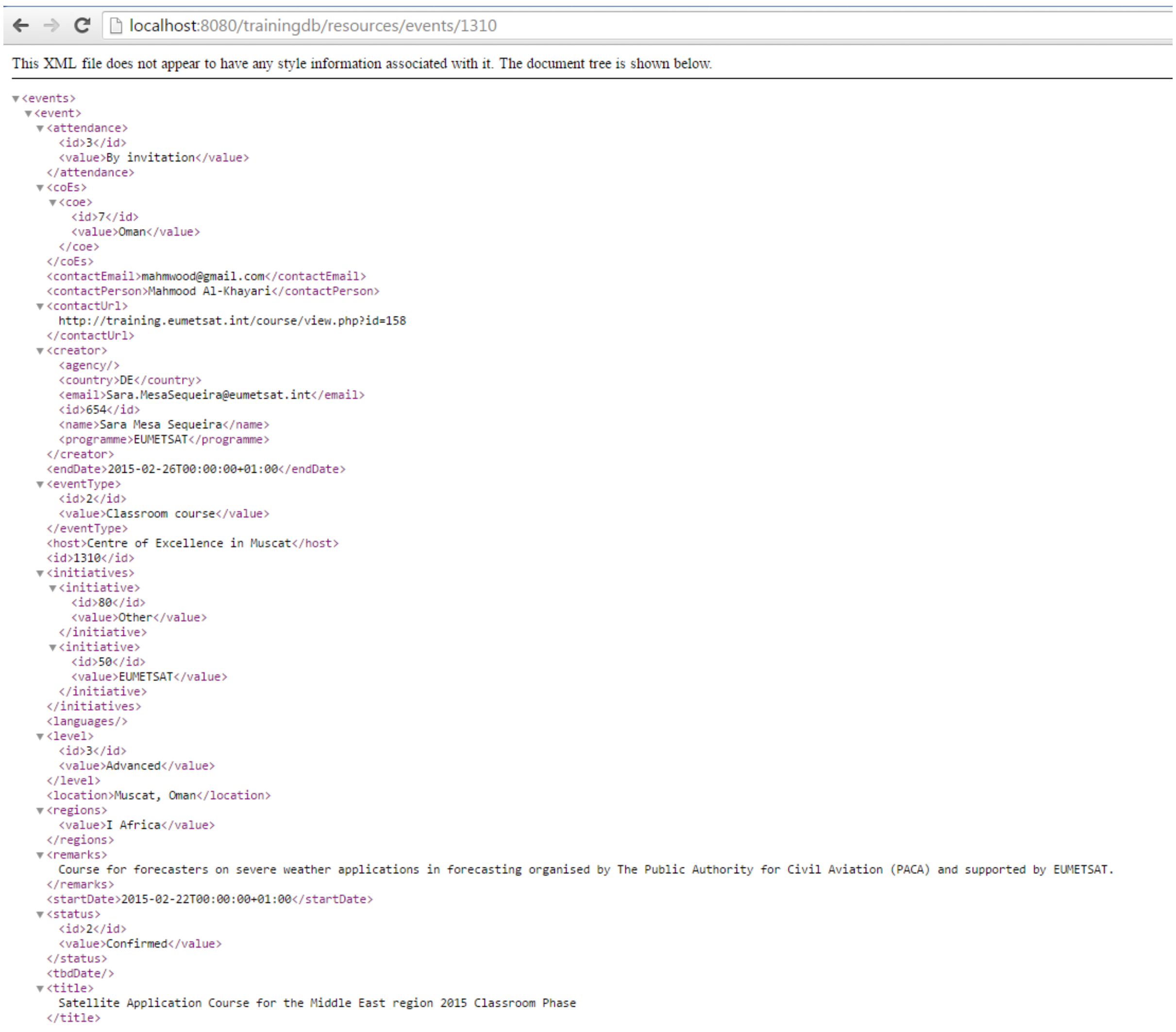
This will trigger a GET request to retrieve the list of events.

NOTE: update the server host and port in the URL if the tomcat server is set in a different way.

5 USER MANUAL

REST Web Services do not always provide a user manual as there is an HTTP request type implemented for this purpose. The OPTIONS request type performed on an resource provides information about the operations allowed for the particular resource. Due to URIs, links, and a uniform interface, the API discovery is simple. Non-standard features supported in this REST API, such as the filtering, are documented below.

The Training Database REST Service accepts HTTP requests like any other REST API. This means that any web browser is capable of accessing the service for basic XML/JSON resource retrieval operations, through GET requests (normal URL browsing).


localhost:8080/trainingdb/resources/events/1310

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<events>
  ▼<event>
    ▼<attendance>
        <id>3</id>
        <value>By invitation</value>
      </attendance>
    ▼<coEs>
      ▼<coe>
          <id>7</id>
          <value>Oman</value>
        </coe>
      </coEs>
      <contactEmail>mahmwood@gmail.com</contactEmail>
      <contactPerson>Mahmood Al-Khayari</contactPerson>
    ▼<contactUrl>
        http://training.eumetsat.int/course/view.php?id=158
      </contactUrl>
    ▼<creator>
        <agency/>
        <country>DE</country>
        <email>Sara.MesaSequeira@eumetsat.int</email>
        <id>654</id>
        <name>Sara Mesa Sequeira</name>
        <programme>EUMETSAT</programme>
      </creator>
      <endDate>2015-02-26T00:00:00+01:00</endDate>
    ▼<eventType>
        <id>2</id>
        <value>Classroom course</value>
      </eventType>
      <host>Centre of Excellence in Muscat</host>
      <id>1310</id>
    ▼<initiatives>
      ▼<initiative>
          <id>80</id>
          <value>Other</value>
        </initiative>
      ▼<initiative>
          <id>50</id>
          <value>EUMETSAT</value>
        </initiative>
      </initiatives>
      <languages/>
    ▼<level>
        <id>3</id>
        <value>Advanced</value>
      </level>
      <location>Muscat, Oman</location>
    ▼<regions>
        <value>I Africa</value>
      </regions>
    ▼<remarks>
        Course for forecasters on severe weather applications in forecasting organised by The Public Authority for Civil Aviation (PACA) and supported by EUMETSAT.
      </remarks>
      <startDate>2015-02-22T00:00:00+01:00</startDate>
    ▼<status>
        <id>2</id>
        <value>Confirmed</value>
      </status>
      <tbdDate/>
    ▼<title>
        Satellite Application Course for the Middle East region 2015 Classroom Phase
      </title>
```

However, for a more advanced access, HTTP request headers need to be set up specifically. Web browsers typically don't allow this by default, but support it with plug-ins. One of the most popular is the Advanced REST Client for Google Chrome. It was the one chosen for the testing, and will be also used in the screenshots provided in this document. It allows users to prepare HTTP requests, providing a nice user interface for setting the headers.

Some other options are command line clients, such as curl, or REST client libraries available for the most common programming languages, such as java, C or python.

5.1 Public operations

For the guest users only the retrieval operation is allowed trough the GET request type. All other operations, such as POST or PUT, to add or delete are restricted to authorized users.

Even though the examples below are performed over the Event resource, they should illustrate also how to access the others. As mentioned before, the OPTIONS request can be helpful for retrieving the list of available operations for a particular resource.

5.1.1 GET: Retrieval of Events

This section provides examples of GET requests to the Event resource, which are used for retrieving the events data.

A few parameters can be configured in the HTTP header of the requests:

- Output Format Type: the "accept" attribute of the HTTP Header request defines what output format is expected from the server. XML (application/xml) and JSON (application/json) are supported at the moment, being XML the default option if nothing is set on the header.

Some other options as URL parameters:

- Filter: the filter is expected to follow the same syntax as per the "where" clauses in the Hibernate Query Language (HQL), including the optional "order by" statement. Some browsers request the URLs with filters to be encoded before transmitted via HTTP (Advanced REST Client supports encoding). For instance, to filter all events located in Darmstadt, the following filter is used:

    - filter=city%3D'darmstadt' (filter: city='darmstadt')

  See section 5.5 for more information on filters.

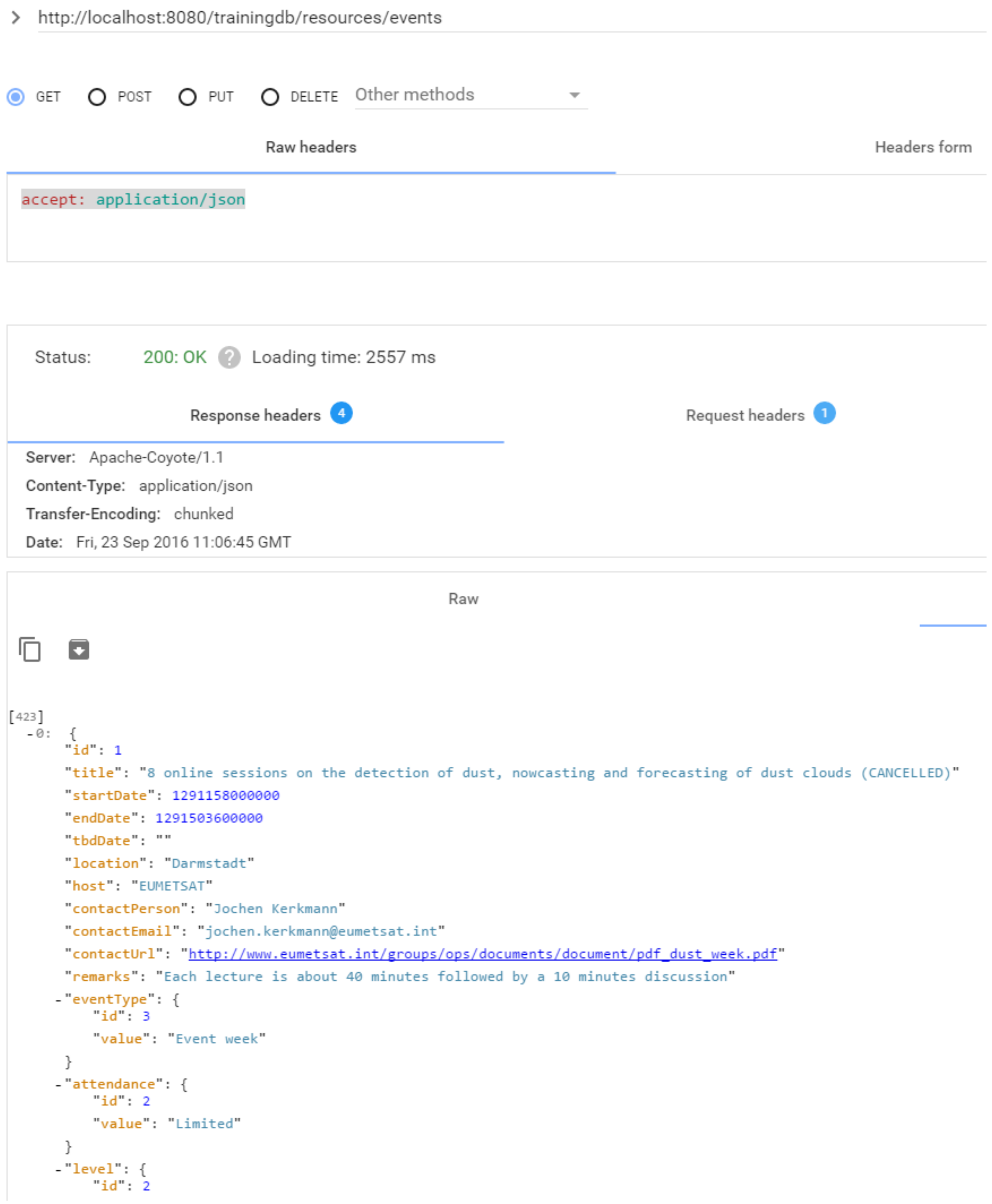
- Search: this options provides full text search, based in Lucene, on the Event resource. The example below would return all events with the word 'Kepler' in any of its fields:

    - Search=Kepler

  Note, the wildcard '*' is supported.

5.1.1.1 Retrieval of all the events

HTTP Request Type	GET
URL	http://<host>/trapi/resources/public/events
HTTP Header	accept: application/json
Data	None



http://localhost:8080/trainingdb/resources/events

[✓ GET]

```
accept: application/json
```

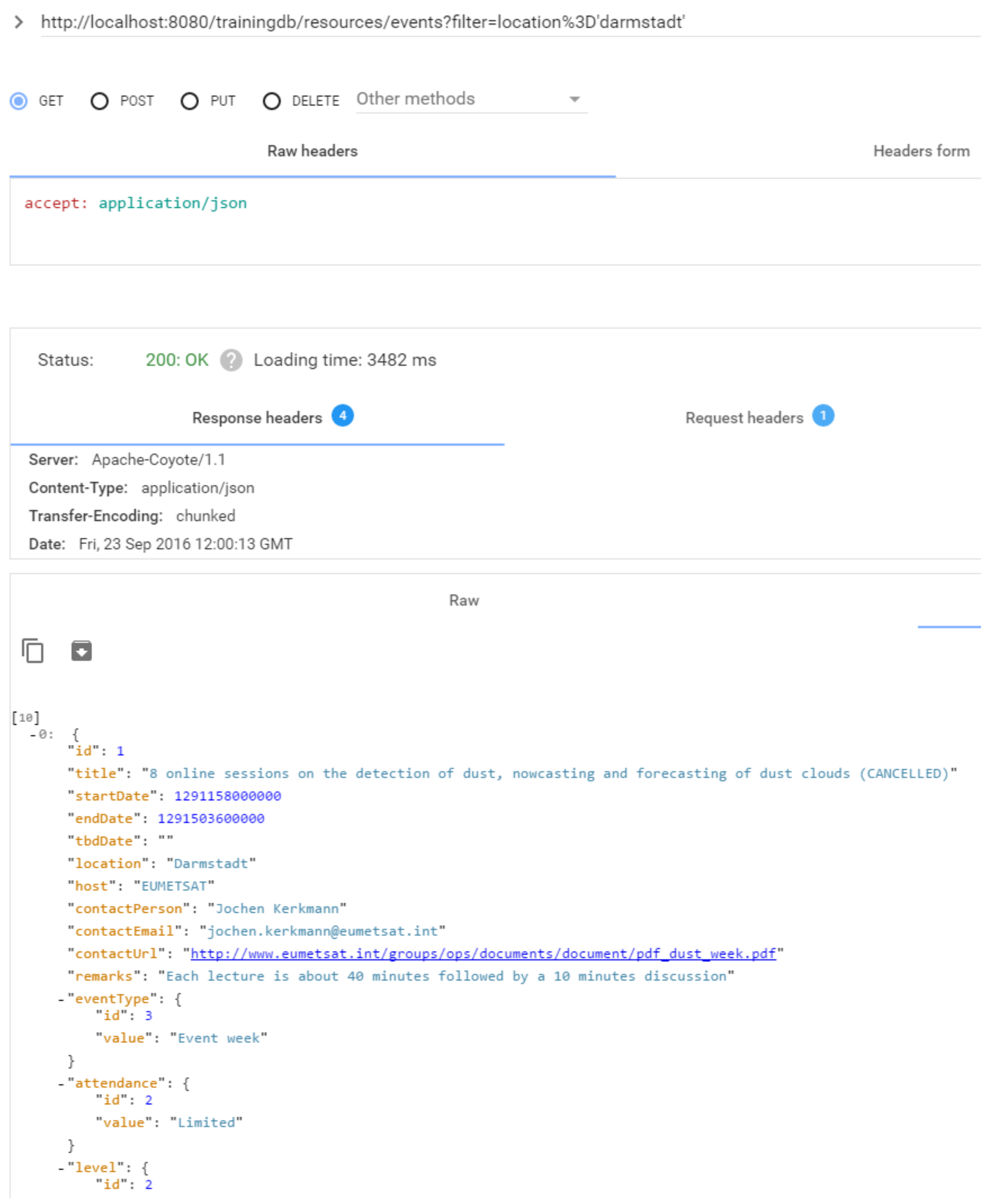
Status: 200: OK Loading time: 2557 ms

Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 23 Sep 2016 11:06:45 GMT

```
[423]
 -0:  {
        "id": 1
        "title": "8 online sessions on the detection of dust, nowcasting and forecasting of dust clouds (CANCELLED)"
        "startDate": 1291158000000
        "endDate": 1291503600000
        "tbdDate": ""
        "location": "Darmstadt"
        "host": "EUMETSAT"
        "contactPerson": "Jochen Kerkmann"
        "contactEmail": "jochen.kerkmann@eumetsat.int"
        "contactUrl": "http://www.eumetsat.int/groups/ops/documents/document/pdf_dust_week.pdf"
        "remarks": "Each lecture is about 40 minutes followed by a 10 minutes discussion"
      -"eventType": {
            "id": 3
            "value": "Event week"
        }
      -"attendance": {
            "id": 2
            "value": "Limited"
        }
      -"level": {
            "id": 2
```

5.1.1.2 Retrieval of events with filter

HTTP Request Type	GET
URL	http://<host>/trapi/resources/public/events?filter=city%3D'darmstadt' (filter: city='darmstadt')
HTTP Header	accept: application/json
Data	None



http://localhost:8080/trainingdb/resources/events?filter=location%3D'darmstadt'

```
accept: application/json
```

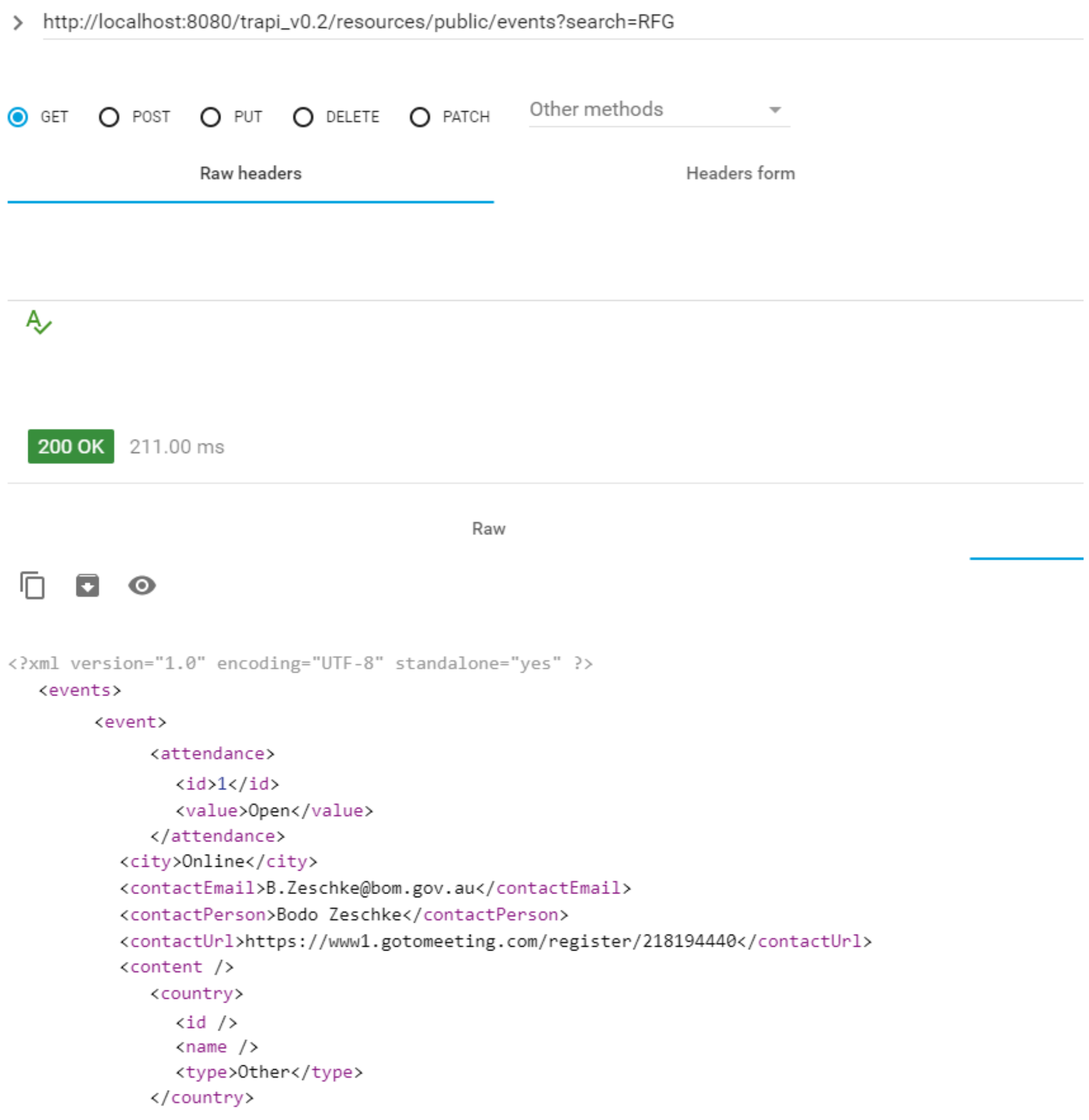
Status: 200: OK Loading time: 3482 ms

Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 23 Sep 2016 12:00:13 GMT

```
[10]
 -0:  {
      "id": 1
      "title": "8 online sessions on the detection of dust, nowcasting and forecasting of dust clouds (CANCELLED)"
      "startDate": 1291158000000
      "endDate": 1291503600000
      "tbdDate": ""
      "location": "Darmstadt"
      "host": "EUMETSAT"
      "contactPerson": "Jochen Kerkmann"
      "contactEmail": "jochen.kerkmann@eumetsat.int"
      "contactUrl": "http://www.eumetsat.int/groups/ops/documents/document/pdf_dust_week.pdf"
      "remarks": "Each lecture is about 40 minutes followed by a 10 minutes discussion"
     -"eventType": {
         "id": 3
         "value": "Event week"
      }
     -"attendance": {
         "id": 2
         "value": "Limited"
      }
     -"level": {
         "id": 2
```

5.1.1.3 Retrieval of events with search

HTTP Request Type	GET
URL	http://<host>/trapi/resources/public/events?search=REF (search=RFG)
HTTP Header	accept: application/xml
Data	None



```
http://localhost:8080/trapi_v0.2/resources/public/events?search=RFG
```

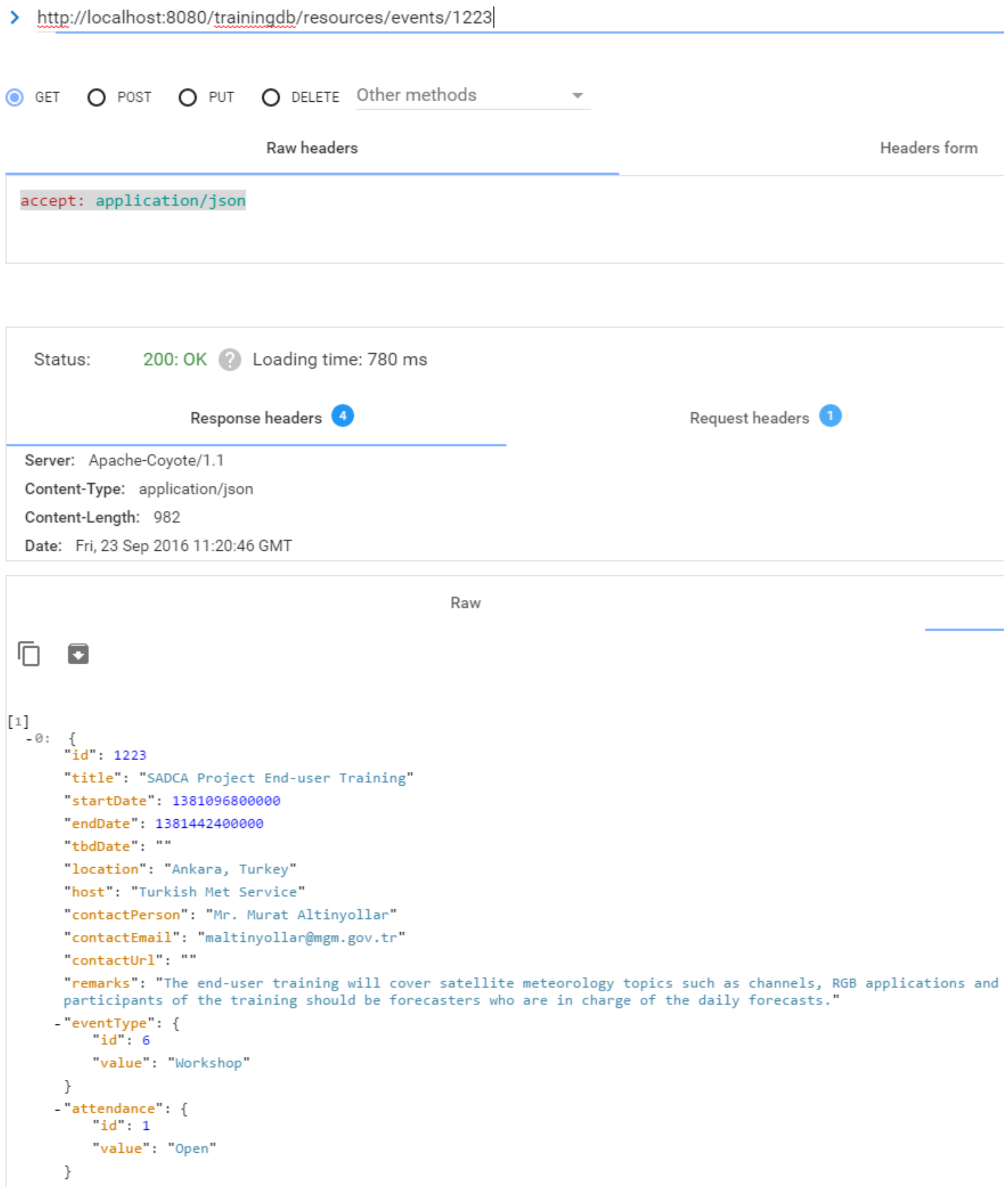
[✓ GET] POST PUT DELETE PATCH

200 OK 211.00 ms

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
   <events>
        <event>
             <attendance>
               <id>1</id>
               <value>Open</value>
             </attendance>
          <city>Online</city>
          <contactEmail>B.Zeschke@bom.gov.au</contactEmail>
          <contactPerson>Bodo Zeschke</contactPerson>
          <contactUrl>https://www1.gotomeeting.com/register/218194440</contactUrl>
          <content />
             <country>
               <id />
               <name />
               <type>Other</type>
             </country>
```

5.1.1.4 Retrieval of the event with ID

HTTP Request Type	GET
URL	http://<host>/trapi/resources/public/events/<id>
HTTP Header	accept: application/json
Data	None



http://localhost:8080/trainingdb/resources/events/1223

```
accept: application/json
```

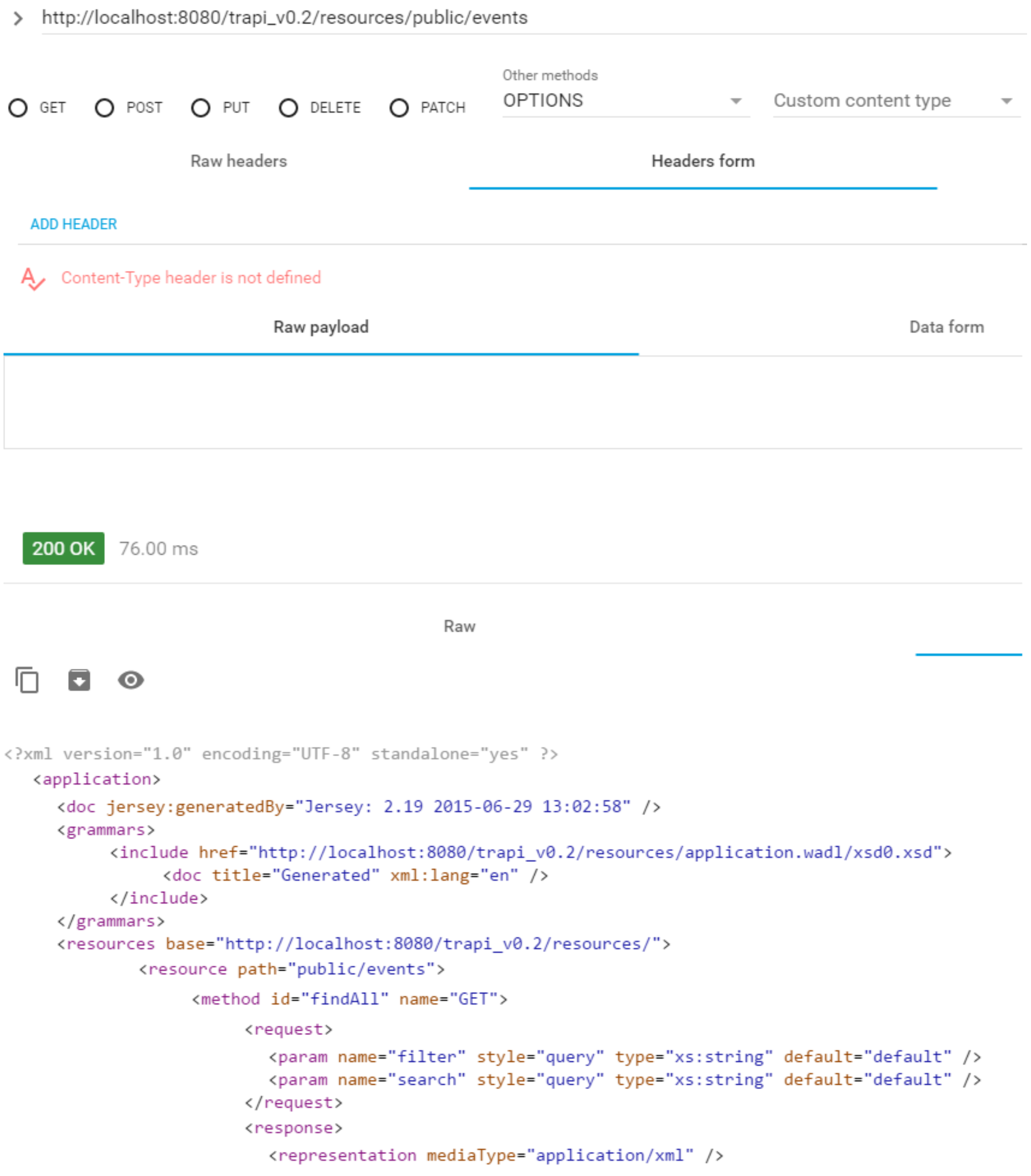
Status: 200: OK Loading time: 780 ms

```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 982
Date: Fri, 23 Sep 2016 11:20:46 GMT
```

```
[1]
 -0: {
      "id": 1223
      "title": "SADCA Project End-user Training"
      "startDate": 1381096800000
      "endDate": 1381442400000
      "tbdDate": ""
      "location": "Ankara, Turkey"
      "host": "Turkish Met Service"
      "contactPerson": "Mr. Murat Altinyollar"
      "contactEmail": "maltinyollar@mgm.gov.tr"
      "contactUrl": ""
      "remarks": "The end-user training will cover satellite meteorology topics such as channels, RGB applications and
      participants of the training should be forecasters who are in charge of the daily forecasts."
     -"eventType": {
          "id": 6
          "value": "Workshop"
      }
     -"attendance": {
          "id": 1
          "value": "Open"
      }
```

5.1.2 OPTIONS: Events API Description

HTTP Request Type	OPTIONS
URL	http://<host>/trapi/resources/public/events
HTTP Header	accept: application/json
Data	None



```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
  <application>
    <doc jersey:generatedBy="Jersey: 2.19 2015-06-29 13:02:58" />
    <grammars>
        <include href="http://localhost:8080/trapi_v0.2/resources/application.wadl/xsd0.xsd">
            <doc title="Generated" xml:lang="en" />
        </include>
    </grammars>
    <resources base="http://localhost:8080/trapi_v0.2/resources/">
          <resource path="public/events">
              <method id="findAll" name="GET">
                  <request>
                    <param name="filter" style="query" type="xs:string" default="default" />
                    <param name="search" style="query" type="xs:string" default="default" />
                  </request>
                  <response>
                    <representation mediaType="application/xml" />
```

5.2 Private operations

Operations involving creation, update or removal of events are subject to user authentication. This version implements the methods GET for retrieval, POST for creation, PUT for updates and DELETE for removal, for the Event resource and those connected to it.

Even though the examples below are performed over the Event resource, they should illustrate also how to access the others. As mentioned before, the OPTIONS request can be helpful for retrieving the list of available operations for a particular resource.

5.2.1 GET: Retrieval of Events

This section provides examples of GET requests to the Event resource, which are used for retrieving the events data. The private GET request extends the visibility given by the public GET.

A few parameters can be configured in the HTTP header of the requests:

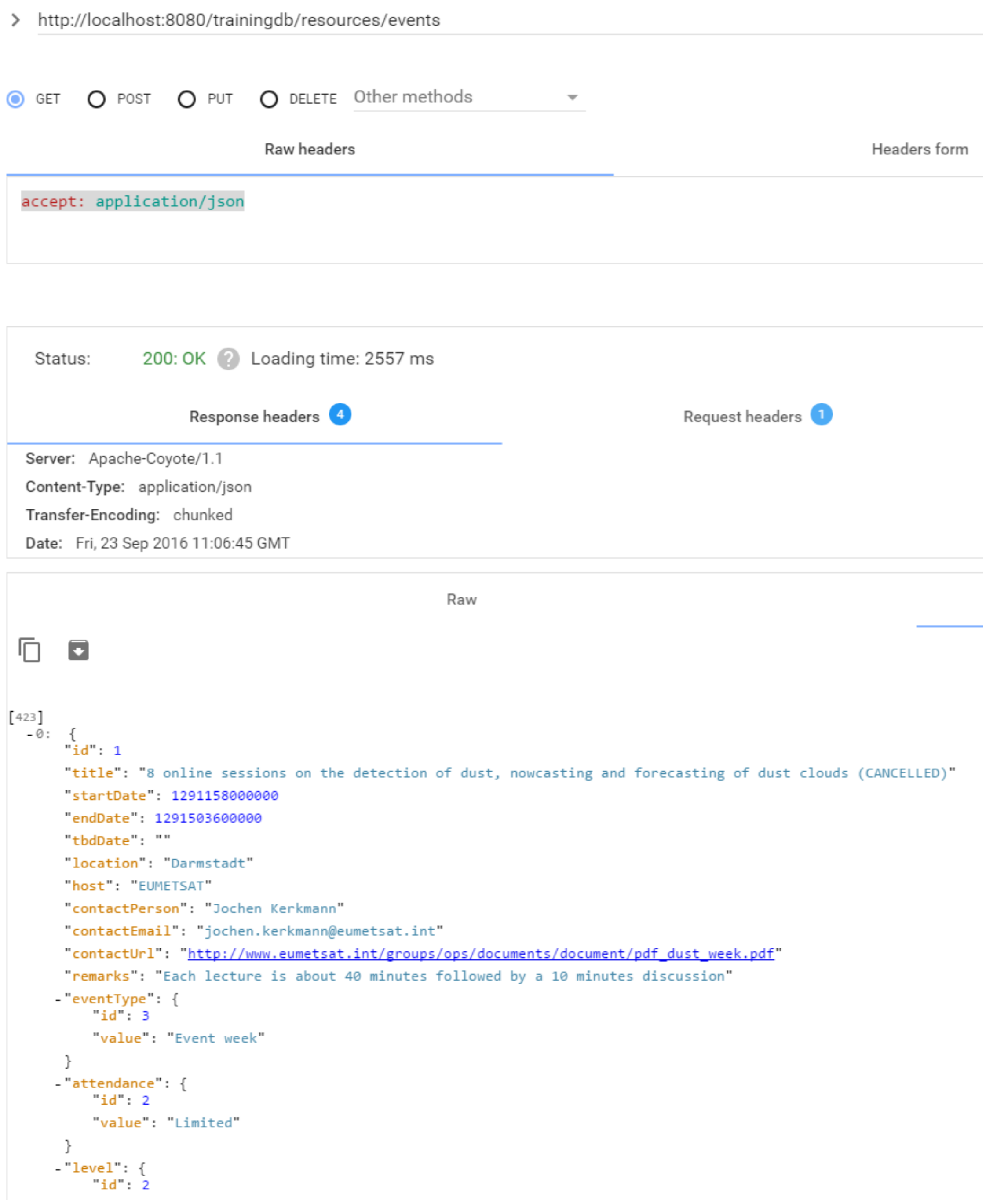
- Output Format Type: the “accept” attribute of the HTTP Header request defines what output format is expected from the server. XML (application/xml) and JSON (application/json) are supported at the moment, being XML the default option if nothing is set on the header.
- Authorization: The “authorization” parameter allows users to identify themselves. It is required for accessing private URLs, and access is granted according to user permissions. The value of the parameter must be constructed according to the procedure in section 2.3.3.3.

Some other options as URL parameters:

- Filter and search: same functionality as with the Public operations. Please see Section 5.1.

5.2.1.1 Retrieval of all the events

HTTP Request Type	GET
URL	http://<host>/trapi/resources/private/events
HTTP Header	accept: application/json authorization: xBearer ************
Data	None



http://localhost:8080/trainingdb/resources/events

[✓ GET] POST PUT DELETE

```
accept: application/json
```

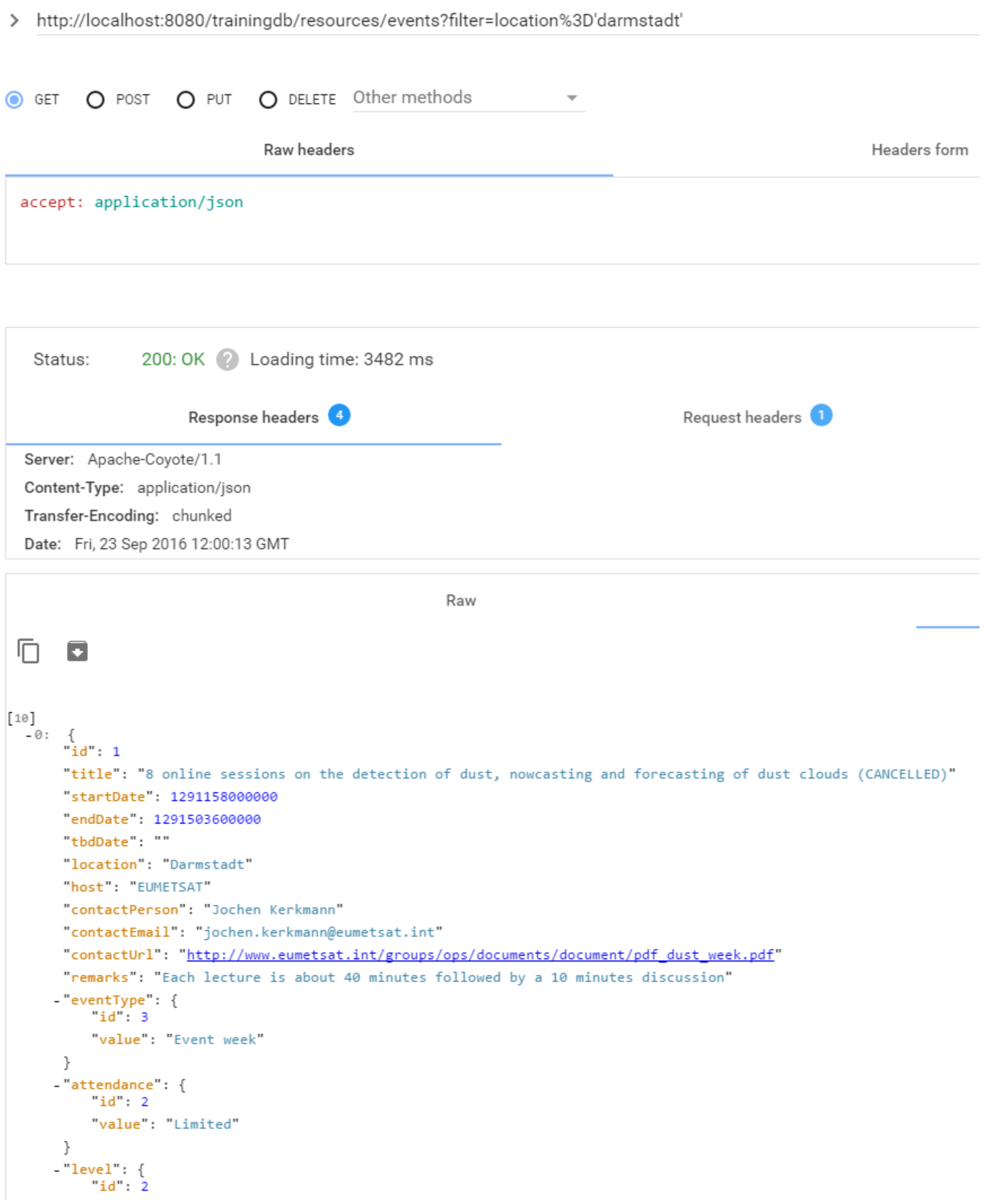
Status: 200: OK Loading time: 2557 ms

Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 23 Sep 2016 11:06:45 GMT

```
[423]
 -0:  {
      "id": 1
      "title": "8 online sessions on the detection of dust, nowcasting and forecasting of dust clouds (CANCELLED)"
      "startDate": 1291158000000
      "endDate": 1291503600000
      "tbdDate": ""
      "location": "Darmstadt"
      "host": "EUMETSAT"
      "contactPerson": "Jochen Kerkmann"
      "contactEmail": "jochen.kerkmann@eumetsat.int"
      "contactUrl": "http://www.eumetsat.int/groups/ops/documents/document/pdf_dust_week.pdf"
      "remarks": "Each lecture is about 40 minutes followed by a 10 minutes discussion"
     -"eventType": {
         "id": 3
         "value": "Event week"
      }
     -"attendance": {
         "id": 2
         "value": "Limited"
      }
     -"level": {
         "id": 2
```

5.2.1.2 Retrieval of events with filter

HTTP Request Type	GET
URL	http://<host>/trapi/resources/private/events?filter=city%3D'darmstadt' (filter: city='darmstadt')
HTTP Header	accept: application/json authorization: xBearer ************
Data	None



http://localhost:8080/trainingdb/resources/events?filter=location%3D'darmstadt'

GET POST PUT DELETE Other methods

Raw headers Headers form

```
accept: application/json
```

Status: 200: OK Loading time: 3482 ms

Response headers 4 Request headers 1

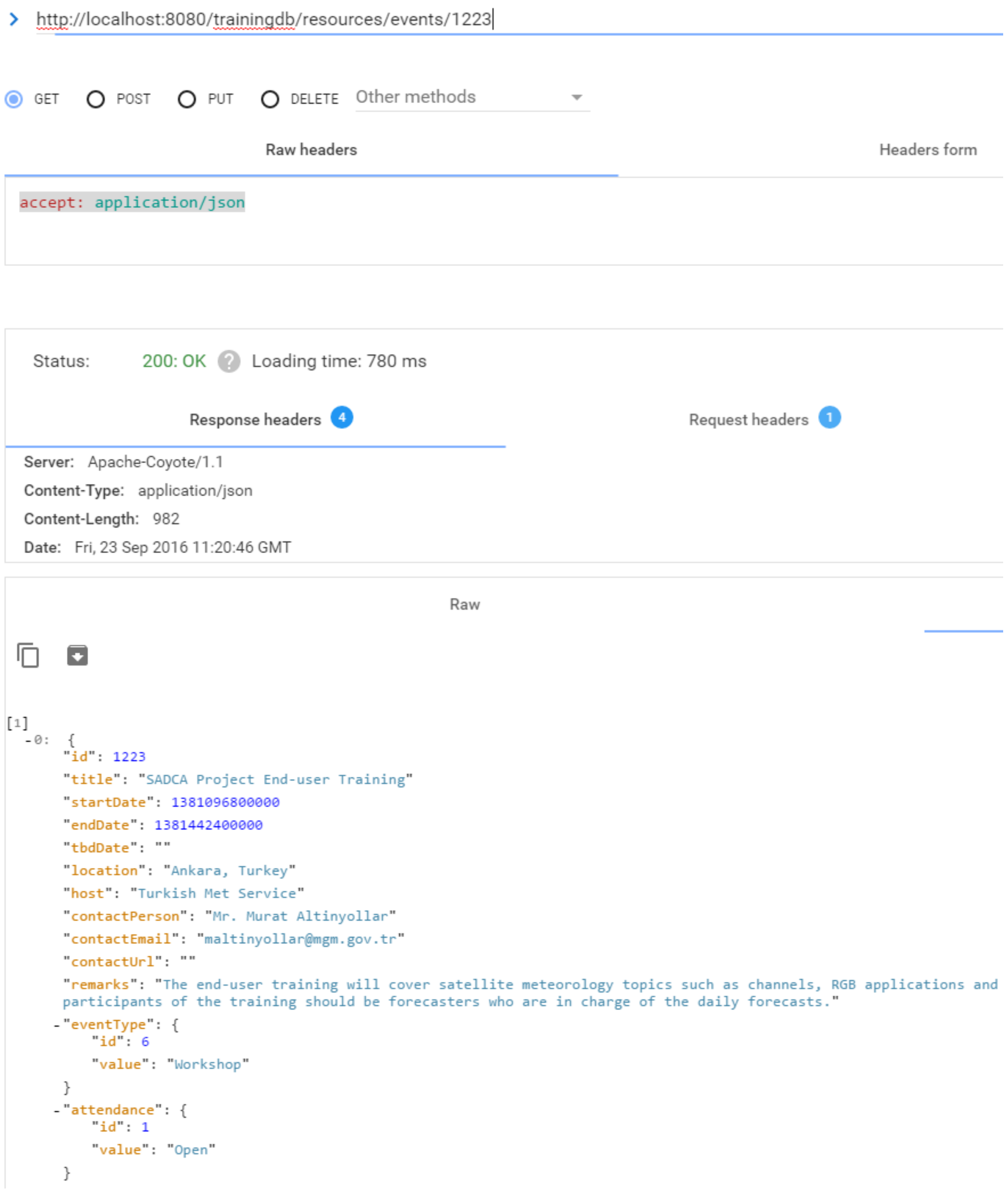
```
Server: Apache-Coyote/1.1
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 23 Sep 2016 12:00:13 GMT
```

Raw

```
[10]
 -0:  {
      "id": 1
      "title": "8 online sessions on the detection of dust, nowcasting and forecasting of dust clouds (CANCELLED)"
      "startDate": 1291158000000
      "endDate": 1291503600000
      "tbdDate": ""
      "location": "Darmstadt"
      "host": "EUMETSAT"
      "contactPerson": "Jochen Kerkmann"
      "contactEmail": "jochen.kerkmann@eumetsat.int"
      "contactUrl": "http://www.eumetsat.int/groups/ops/documents/document/pdf_dust_week.pdf"
      "remarks": "Each lecture is about 40 minutes followed by a 10 minutes discussion"
     -"eventType": {
          "id": 3
          "value": "Event week"
      }
     -"attendance": {
          "id": 2
          "value": "Limited"
      }
     -"level": {
          "id": 2
```

5.2.1.3 Retrieval of the event with ID

HTTP Request Type	GET
URL	http://<host>/trapi/resources/private/events/<id>
HTTP Header	accept: application/json authorization: xBearer ************
Data	None



http://localhost:8080/trainingdb/resources/events/1223

[✓ GET] POST PUT DELETE Other methods

Raw headers | Headers form

```
accept: application/json
```

Status: 200: OK Loading time: 780 ms

Response headers 4 | Request headers 1

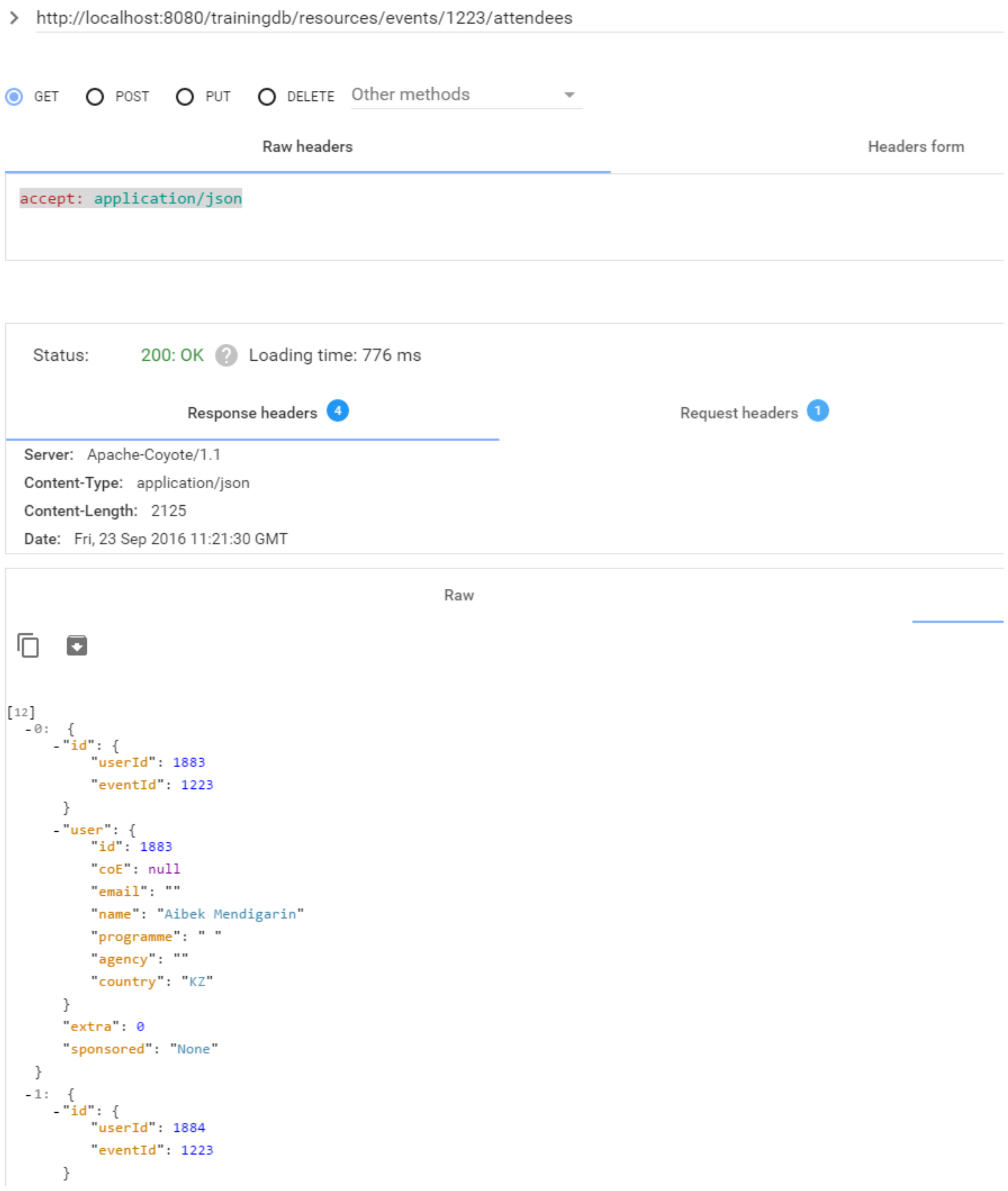
```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 982
Date: Fri, 23 Sep 2016 11:20:46 GMT
```

Raw

```
[1]
 -0: {
      "id": 1223
      "title": "SADCA Project End-user Training"
      "startDate": 1381096800000
      "endDate": 1381442400000
      "tbdDate": ""
      "location": "Ankara, Turkey"
      "host": "Turkish Met Service"
      "contactPerson": "Mr. Murat Altinyollar"
      "contactEmail": "maltinyollar@mgm.gov.tr"
      "contactUrl": ""
      "remarks": "The end-user training will cover satellite meteorology topics such as channels, RGB applications and
      participants of the training should be forecasters who are in charge of the daily forecasts."
     -"eventType": {
          "id": 6
          "value": "Workshop"
      }
     -"attendance": {
          "id": 1
          "value": "Open"
      }
```

5.2.1.4 Retrieval of the attendees for the event with ID

HTTP Request Type	GET
URL	http://<host>/trapi/resources/private/events/<id>/attendees
HTTP Header	accept: application/json authorization: xBearer ************
Data	None


http://localhost:8080/trainingdb/resources/events/1223/attendees
GET
POST
PUT
DELETE
Other methods
Raw headers
Headers form

```
accept: application/json
```

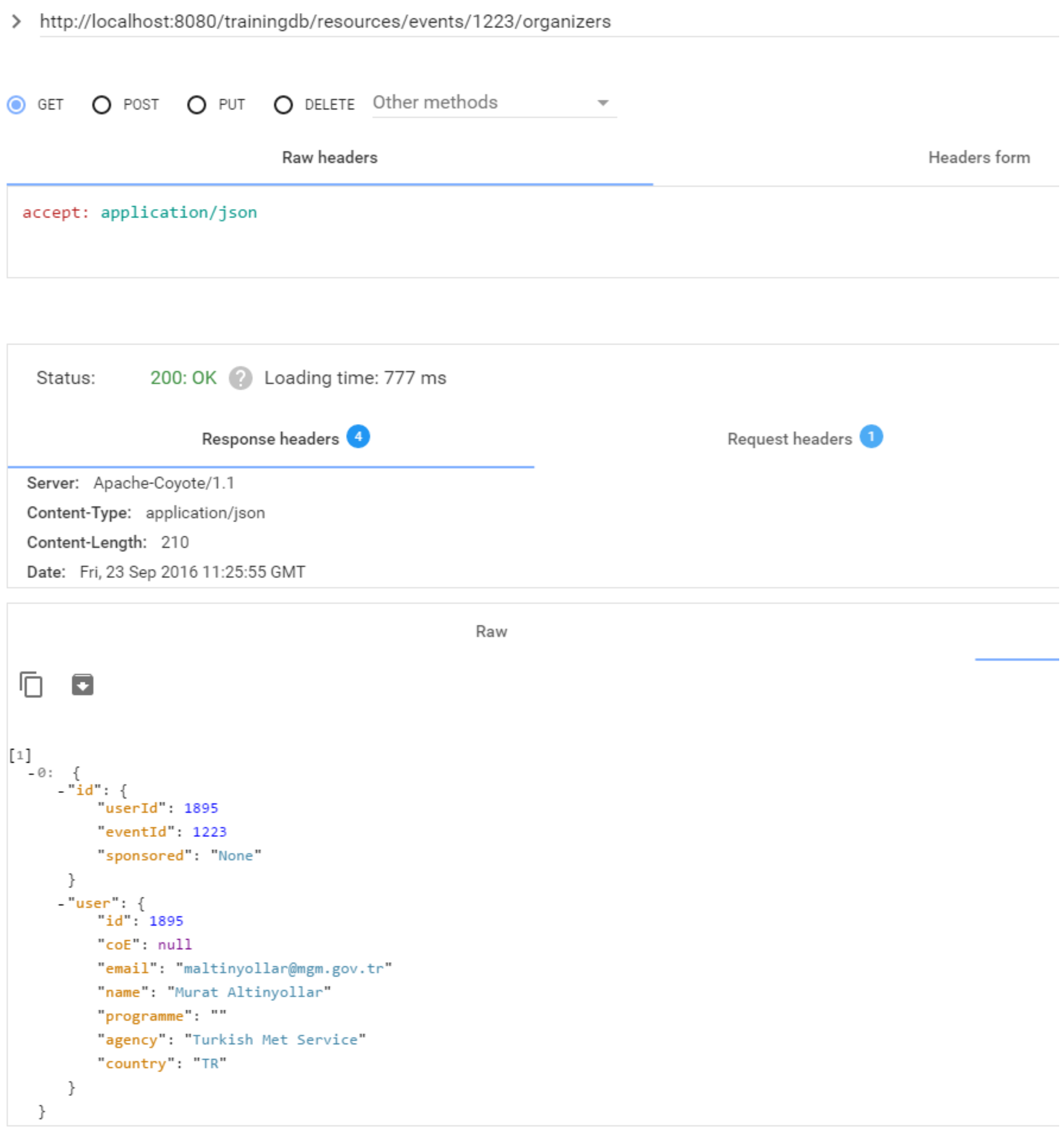
Status: 200: OK Loading time: 776 ms

```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 2125
Date: Fri, 23 Sep 2016 11:21:30 GMT
```

```
[12]
 -0:  {
    -"id": {
         "userId": 1883
         "eventId": 1223
    }
    -"user": {
         "id": 1883
         "coE": null
         "email": ""
         "name": "Aibek Mendigarin"
         "programme": " "
         "agency": ""
         "country": "KZ"
    }
     "extra": 0
     "sponsored": "None"
 }
 -1:  {
    -"id": {
         "userId": 1884
         "eventId": 1223
    }
```

5.2.1.5 Retrieval of the organizers for the event with ID

HTTP Request Type	GET
URL	http://<host>/trapi/resources/private/events/<id>/organizers
HTTP Header	accept: application/json authorization: xBearer ************
Data	None



http://localhost:8080/trainingdb/resources/events/1223/organizers

```
accept: application/json
```

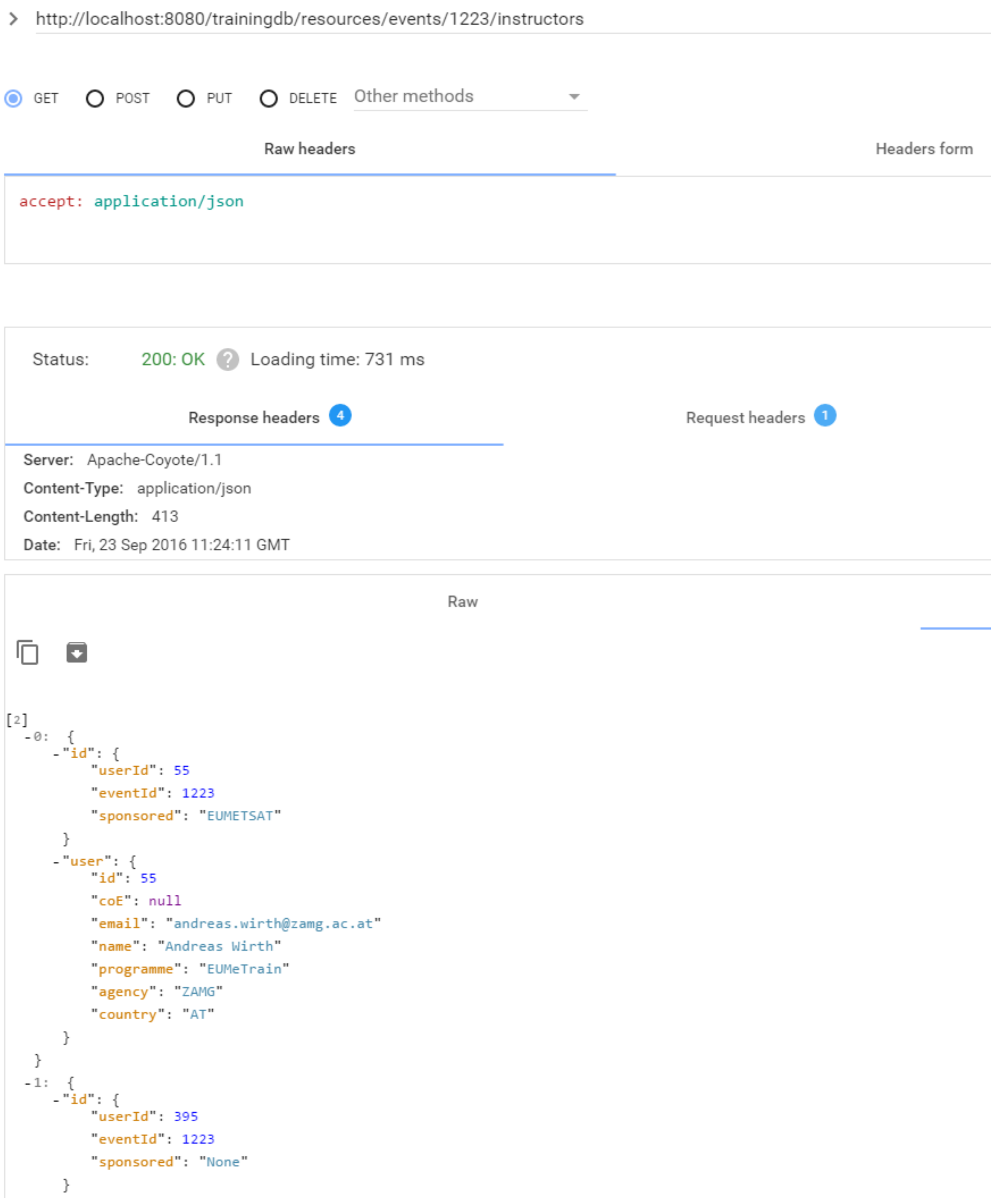
Status: 200: OK Loading time: 777 ms

```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 210
Date: Fri, 23 Sep 2016 11:25:55 GMT
```

```
[1]
 -0: {
   -"id": {
       "userId": 1895
       "eventId": 1223
       "sponsored": "None"
    }
   -"user": {
       "id": 1895
       "coE": null
       "email": "maltinyollar@mgm.gov.tr"
       "name": "Murat Altinyollar"
       "programme": ""
       "agency": "Turkish Met Service"
       "country": "TR"
    }
 }
```

5.2.1.6 Retrieval of the instructors for the event with ID

HTTP Request Type	GET
URL	**Error! Hyperlink reference not valid.**
HTTP Header	accept: application/json authorization: xBearer ************
Data	None



http://localhost:8080/trainingdb/resources/events/1223/instructors

GET POST PUT DELETE Other methods

Raw headers Headers form

```
accept: application/json
```

Status: 200: OK Loading time: 731 ms

Response headers 4 Request headers 1

```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 413
Date: Fri, 23 Sep 2016 11:24:11 GMT
```

Raw

```
[2]
 -0:  {
    -"id": {
        "userId": 55
        "eventId": 1223
        "sponsored": "EUMETSAT"
     }
    -"user": {
        "id": 55
        "coE": null
        "email": "andreas.wirth@zamg.ac.at"
        "name": "Andreas Wirth"
        "programme": "EUMeTrain"
        "agency": "ZAMG"
        "country": "AT"
     }
  }
 -1:  {
    -"id": {
        "userId": 395
        "eventId": 1223
        "sponsored": "None"
     }
```

5.2.2 POST: Creation of Events

This section provides examples of POST requests to the Event resource and also those connected to it. These are used for creating new events and also adding links between them and users to indicate relationships of attendance/instructors/organizers. For Events, the POST operation is supported for the following Resource URLs:

- resources/private/events
- resources/private/events/<id>/attendees
- resources/private/events/<id>/instructors
- resources/private/events/<id>/organizers

A few parameters can be configured in the HTTP header of the requests:

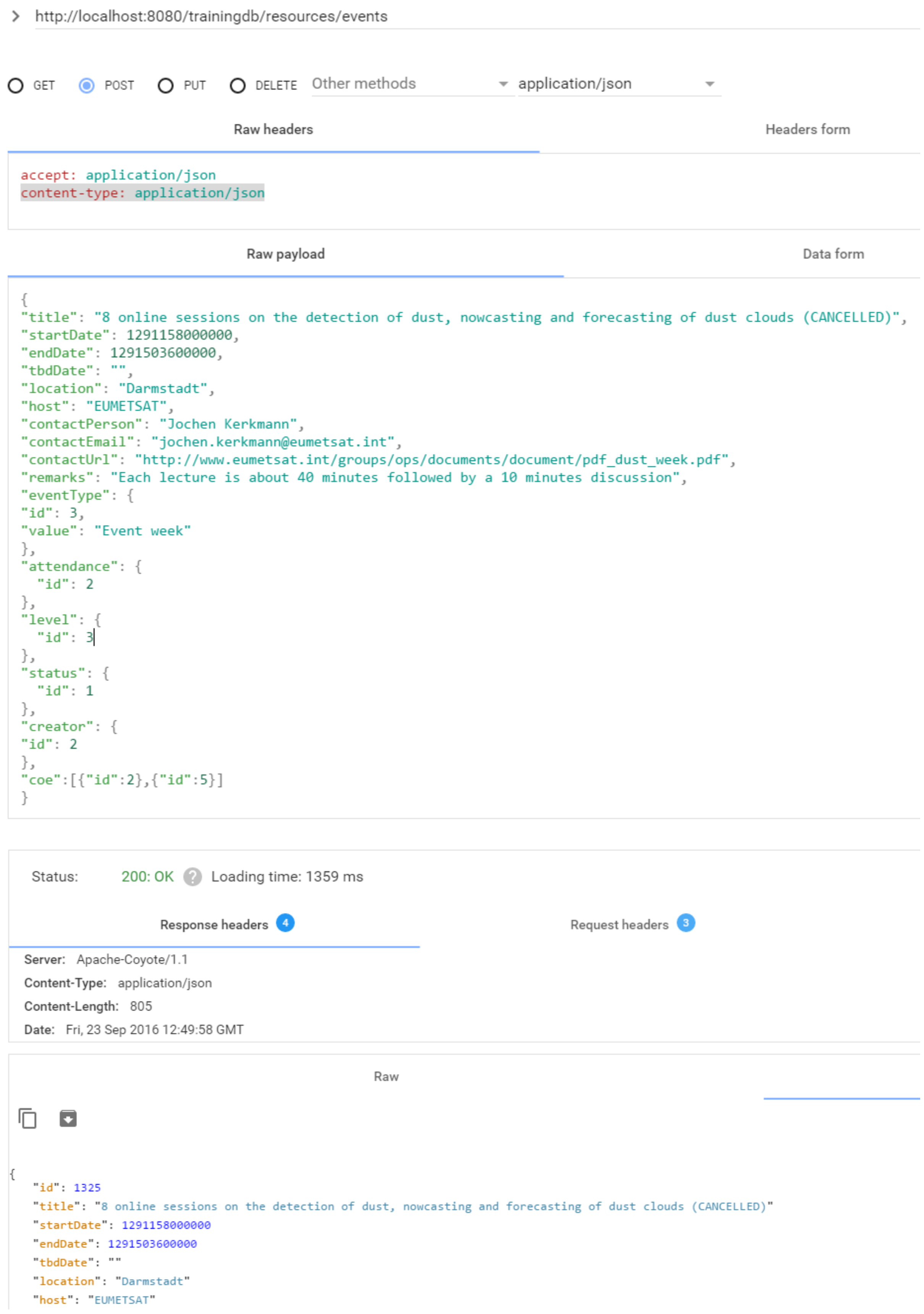
- Output Format Type: the "accept" attribute of the HTTP Header request defines what output format is expected from the server. XML (application/xml) and JSON (application/json) are supported at the moment, being XML the default option if nothing is set on the header.
- Input Format Type: the "content-type" attribute of the HTTP Header request defines what input format is sent to the server. XML (application/xml) and JSON (application/json) are supported at the moment, being XML the default option if nothing is set on the header.
- Authorization: The "authorization" parameter allows users to identify themselves. It is required for accessing private URLs, and access is granted according to user permissions. The value of the parameter must be constructed according to the procedure in section 2.3.3.3.

Differently from the GET requests, POST requests require that the data object to be created in the database is sent to the server. As output, if the response is successful (code 200), the new object is returned.

5.2.2.1 Creation of new events

HTTP Request Type	POST
URL	http://<host>/trapi/resources/private/events
HTTP Header	accept: application/json content-type: application/json authorization: xBearer ************

Data	{
	"title": "8 online sessions on the detection of dust, nowcasting and forecasting of dust clouds (CANCELLED)",
	"startDate": 1291158000000,
	"endDate": 1291503600000,
	"tbdDate": "",
	"city": "Darmstadt",
	"host": "EUMETSAT",
	"contactPerson": "Jochen Kerkmann",
	"contactEmail": "jochen.kerkmann@eumetsat.int",
	"contactUrl": "http://www.eumetsat.int/groups/ops/documents/document/pdf_dust_week.pdf",
	"eventType": {
	"id": 3,
	"value": "Event week"
	},
	"attendance": {
	  "id": 2
	},
	"status": {
	  "id": 1
	},
	"creator": {
	"id": 2
	},
	…



http://localhost:8080/trainingdb/resources/events

GET [✓ POST] PUT DELETE Other methods application/json

Raw headers | Headers form

```
accept: application/json
content-type: application/json
```

Raw payload | Data form

```
{
"title": "8 online sessions on the detection of dust, nowcasting and forecasting of dust clouds (CANCELLED)",
"startDate": 1291158000000,
"endDate": 1291503600000,
"tbdDate": "",
"location": "Darmstadt",
"host": "EUMETSAT",
"contactPerson": "Jochen Kerkmann",
"contactEmail": "jochen.kerkmann@eumetsat.int",
"contactUrl": "http://www.eumetsat.int/groups/ops/documents/document/pdf_dust_week.pdf",
"remarks": "Each lecture is about 40 minutes followed by a 10 minutes discussion",
"eventType": {
"id": 3,
"value": "Event week"
},
"attendance": {
  "id": 2
},
"level": {
  "id": 3
},
"status": {
  "id": 1
},
"creator": {
"id": 2
},
"coe":[{"id":2},{"id":5}]
}
```

Status: 200: OK Loading time: 1359 ms

Response headers 4 | Request headers 3

```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 805
Date: Fri, 23 Sep 2016 12:49:58 GMT
```

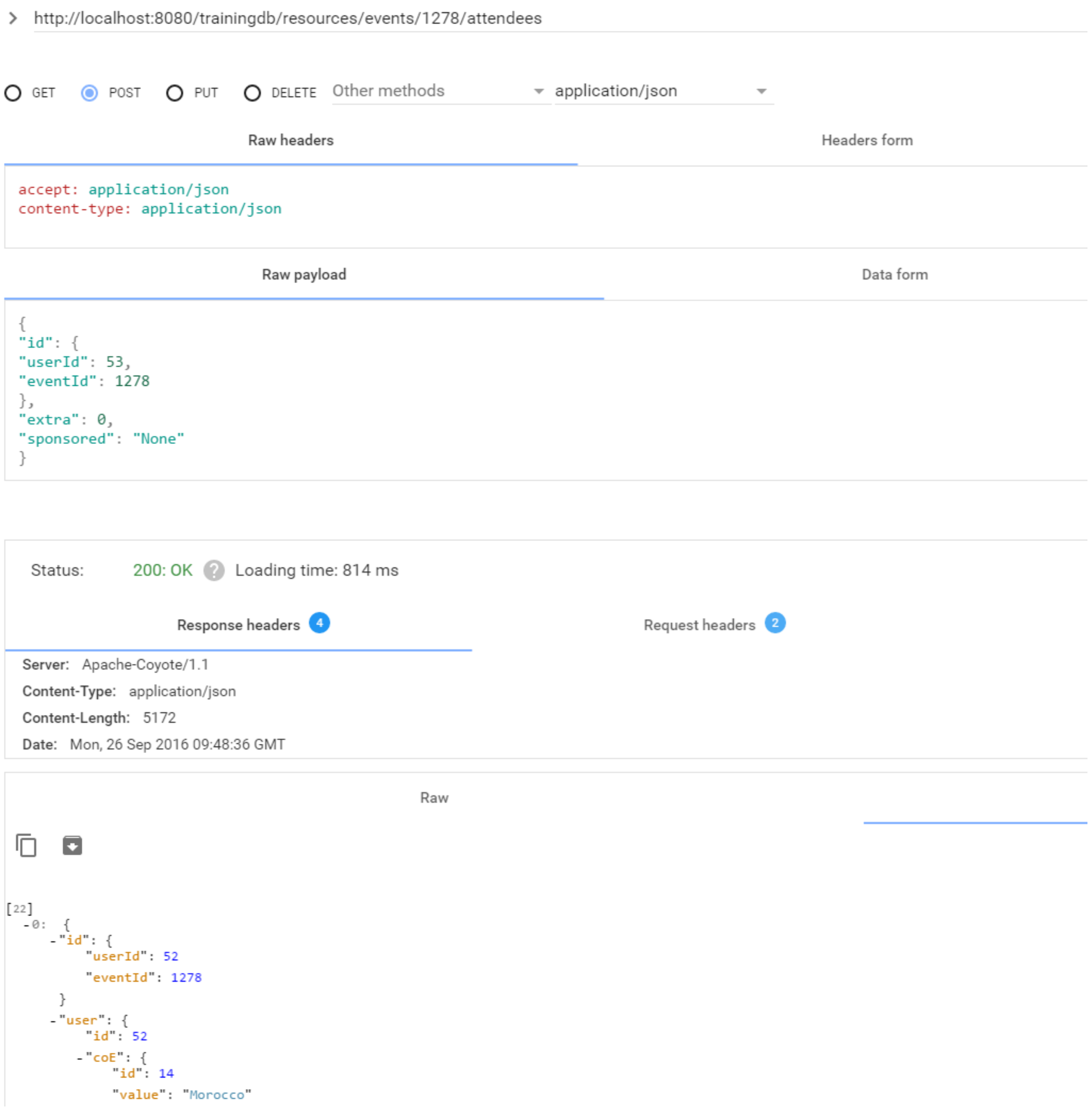
Raw

```
{
  "id": 1325
  "title": "8 online sessions on the detection of dust, nowcasting and forecasting of dust clouds (CANCELLED)"
  "startDate": 1291158000000
  "endDate": 1291503600000
  "tbdDate": ""
  "location": "Darmstadt"
  "host": "EUMETSAT"
```

5.2.2.2 Add attendees/instructors/organizers to an event

HTTP Request Type	POST
URL	http://<host>/trapi/resources/private/events/<id>/attendees
HTTP Header	accept: application/json content-type: application/json authorization: xBearer ************

Data	{
	"id": {
	"userId": 53,
	"eventId": 1278
	},
	"extra": 0,
	"sponsored": "None"
	}


http://localhost:8080/trainingdb/resources/events/1278/attendees

GET [✓ POST] PUT DELETE Other methods application/json

Raw headers | Headers form

```
accept: application/json
content-type: application/json
```

Raw payload | Data form

```
{
"id": {
"userId": 53,
"eventId": 1278
},
"extra": 0,
"sponsored": "None"
}
```

Status: 200: OK Loading time: 814 ms

Response headers 4 | Request headers 2

```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 5172
Date: Mon, 26 Sep 2016 09:48:36 GMT
```

Raw

```
[22]
 -0:  {
    -"id": {
         "userId": 52
         "eventId": 1278
     }
    -"user": {
         "id": 52
       -"coE": {
            "id": 14
            "value": "Morocco"
```

5.2.3 POST: Modify Event Status using Reason

When the event status is changed, event attendees and other users related to that event receive a notification about the change. The following methods trigger a specific change, send out a notification, and require a Reason object as the payload data to use in the notification.

5.2.3.1 Cancel the event with ID

HTTP Request Type	POST
URL	http://<host>/trapi/resources/private/events/<id>/cancel
HTTP Header	accept: application/json

	authorization: xBearer ************

Data	{
	“reason”: “a reason meassage”
	}

5.2.3.1.1 Reject draft event with ID

HTTP Request Type	POST
URL	http://<host>/trapi/resources/private/events/<id>/reject
HTTP Header	accept: application/json authorization: xBearer ************

Data	{
	“reason”: “a reason meassage”
	}

5.2.4 PUT: Update of Events

This section provides examples of PUT requests to the Event resource, which are used for updating events. In this case the operator only works with resources given by ID, /resources/private/events/<id> in this case.

A few parameters can be configured in the HTTP header of the requests:

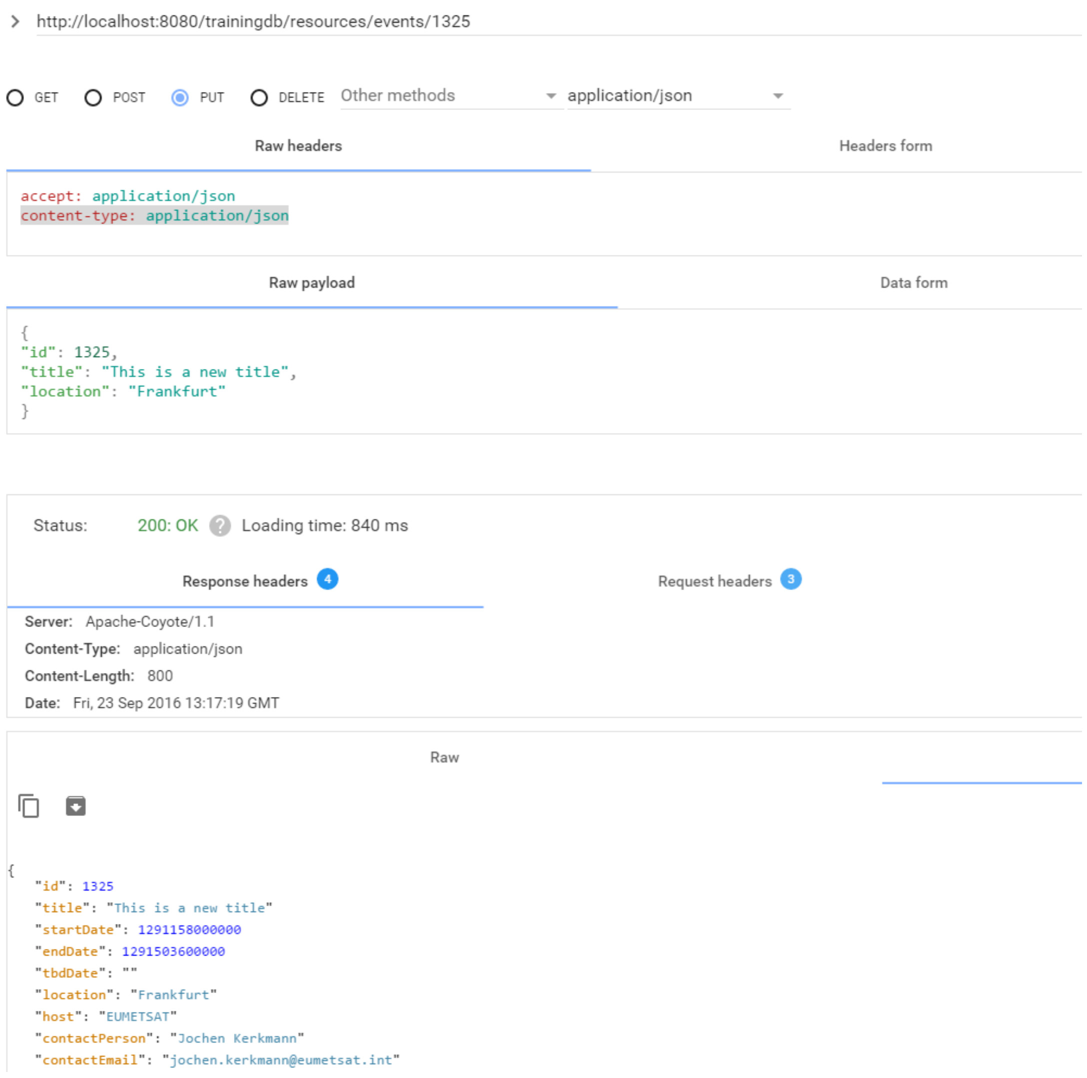
- Output Format Type: the “accept” attribute of the HTTP Header request defines what output format is expected from the server. XML (application/xml) and JSON (application/json) are supported at the moment, being XML the default option if nothing is set on the header.
- Input Format Type: the “content-type” attribute of the HTTP Header request defines what input format is sent to the server. XML (application/xml) and JSON (application/json) are supported at the moment, being XML the default option if nothing is set on the header.
- Authorization: The “authorization” parameter allows users to identify themselves. It is required for accessing private URLs, and access is granted according to user permissions. The value of the parameter must be constructed according to the procedure in section 2.3.3.3.

As with the POST requests, PUT requests require that the data object to be updated in the database is sent to the server. In this case, only the ID of the object and the attributes which need the update need to be sent. As output, if the response is successful (code 200), the updated object is returned.

5.2.4.1 Update the event with ID

HTTP Request Type	PUT
URL	http://<host>/trapi/resources/private/events/<id>
HTTP Header	accept: application/json content-type: application/json authorization: xBearer ************

Data	{
	"id": 1325,
	"title": "This is a new title",
	"city": "Frankfurt"
	}


http://localhost:8080/trainingdb/resources/events/1325

GET POST [✓ PUT] DELETE Other methods application/json

Raw headers | Headers form

```
accept: application/json
content-type: application/json
```

Raw payload | Data form

```
{
"id": 1325,
"title": "This is a new title",
"location": "Frankfurt"
}
```

Status: 200: OK Loading time: 840 ms

Response headers 4 | Request headers 3

```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 800
Date: Fri, 23 Sep 2016 13:17:19 GMT
```

Raw

```
{
   "id": 1325
   "title": "This is a new title"
   "startDate": 1291158000000
   "endDate": 1291503600000
   "tbdDate": ""
   "location": "Frankfurt"
   "host": "EUMETSAT"
   "contactPerson": "Jochen Kerkmann"
   "contactEmail": "jochen.kerkmann@eumetsat.int"
```

5.2.5 DELETE: Removal of Events

This section provides examples of DELETE requests to the Event resource, which are used for removing events. In this case, the operator works with resources given by ID, /resources/private/events/<id>, or with the Resource URL /resources/private/events, if all objects need to be removed, or those resulting from a filter.

A few parameters can be configured in the HTTP header of the requests:

- Output Format Type: the "accept" attribute of the HTTP Header request defines what output format is expected from the server. XML (application/xml) and JSON (application/json) are supported at the moment, being XML the default option if nothing is set on the header.
- Authorization: The "authorization" parameter allows users to identify themselves. It is required for accessing private URLs, and access is granted according to user permissions. The value of the parameter must be constructed according to the procedure in section 2.3.3.3.

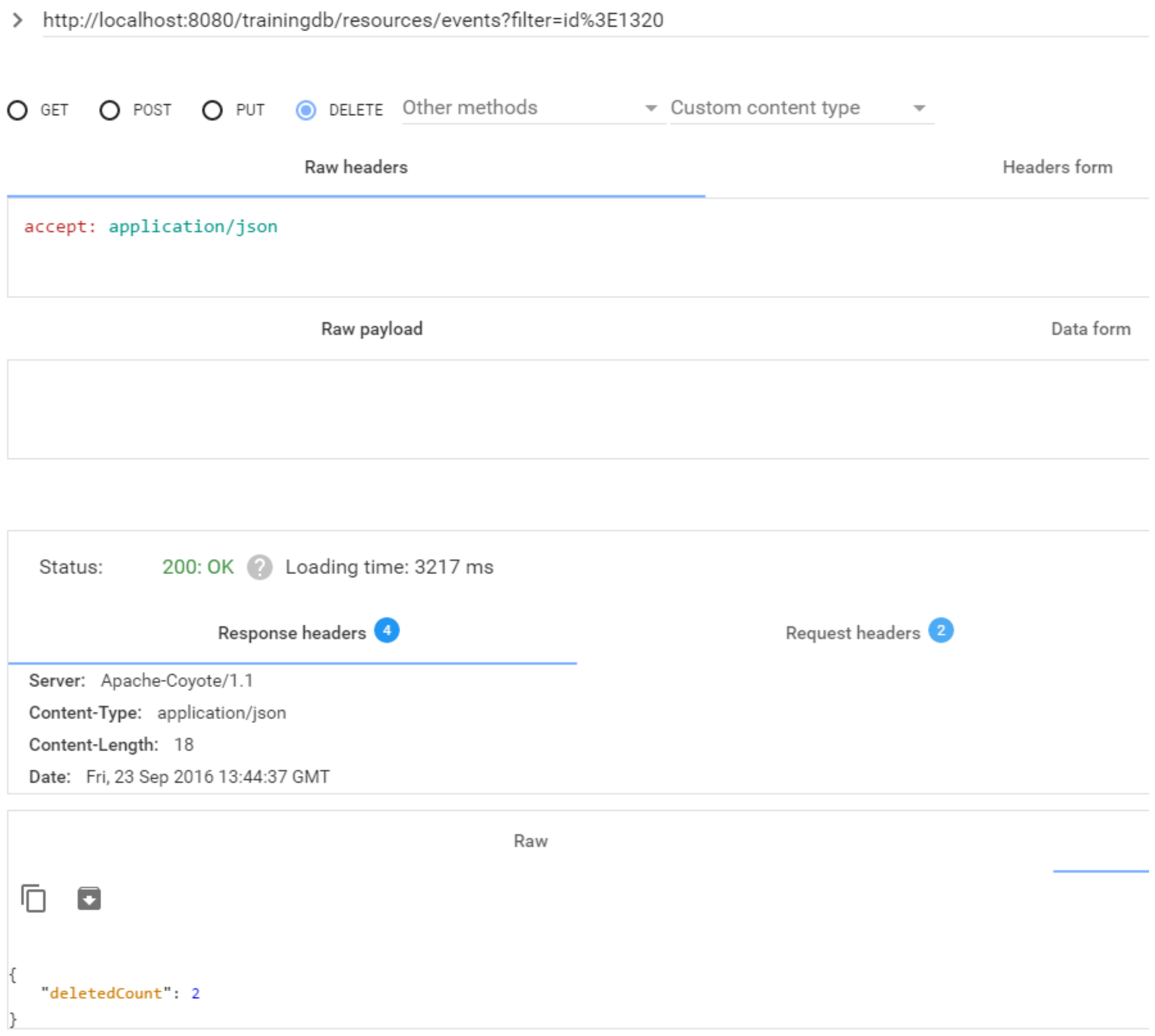
Some other options as URL parameters:

- Filter: the filter is expected to follow the same syntax as per the "where" clauses in Hibernate Query Language (HQL). It needs to be encoded in order to be transferred correctly via HTTP. Objects resulting from the filter will be deleted.

As output, if the response is successful (code 200), the number of deleted items is returned.

5.2.5.1 Removal of events with filter

HTTP Request Type	DELETE
URL	http://<host>/trapi/resources/private/events?filter=id%3E1320 (filter: id>1320)
HTTP Header	accept: application/json authorization: xBearer ************
Data	



```
accept: application/json
```

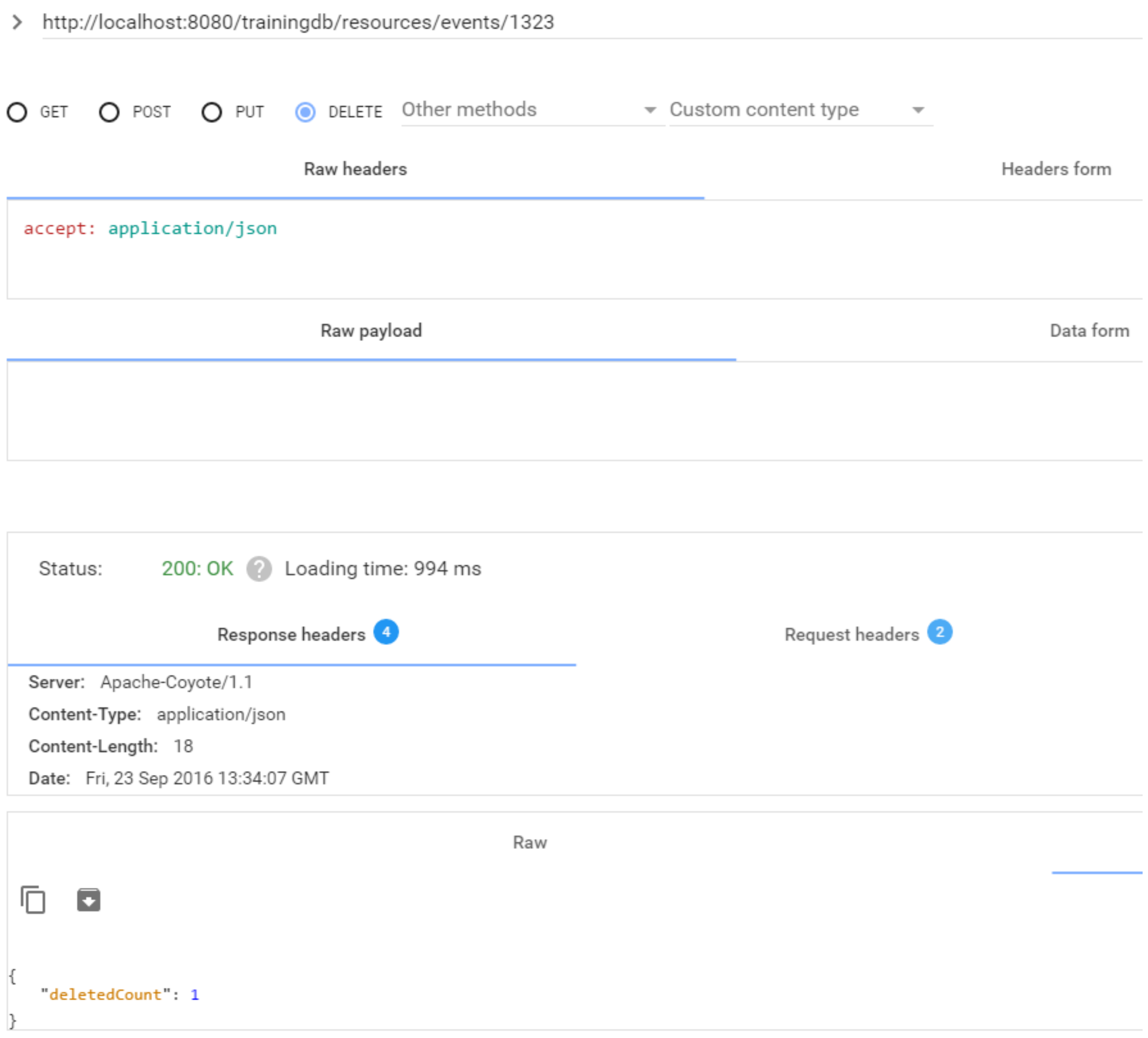
```
Status:    200: OK   Loading time: 3217 ms

Server:   Apache-Coyote/1.1
Content-Type:   application/json
Content-Length:   18
Date:   Fri, 23 Sep 2016 13:44:37 GMT
```

```
{
  "deletedCount": 2
}
```

5.2.5.2 Removal of the event with ID

HTTP Request Type	DELETE
URL	http://<host>/trapi/resources/private/events/<id>
HTTP Header	accept: application/json authorization: xBearer ************
Data	


http://localhost:8080/trainingdb/resources/events/1323

GET POST PUT [✓ DELETE] Other methods Custom content type

Raw headers Headers form

```
accept: application/json
```

Raw payload Data form

Status: 200: OK Loading time: 994 ms

Response headers 4 Request headers 2

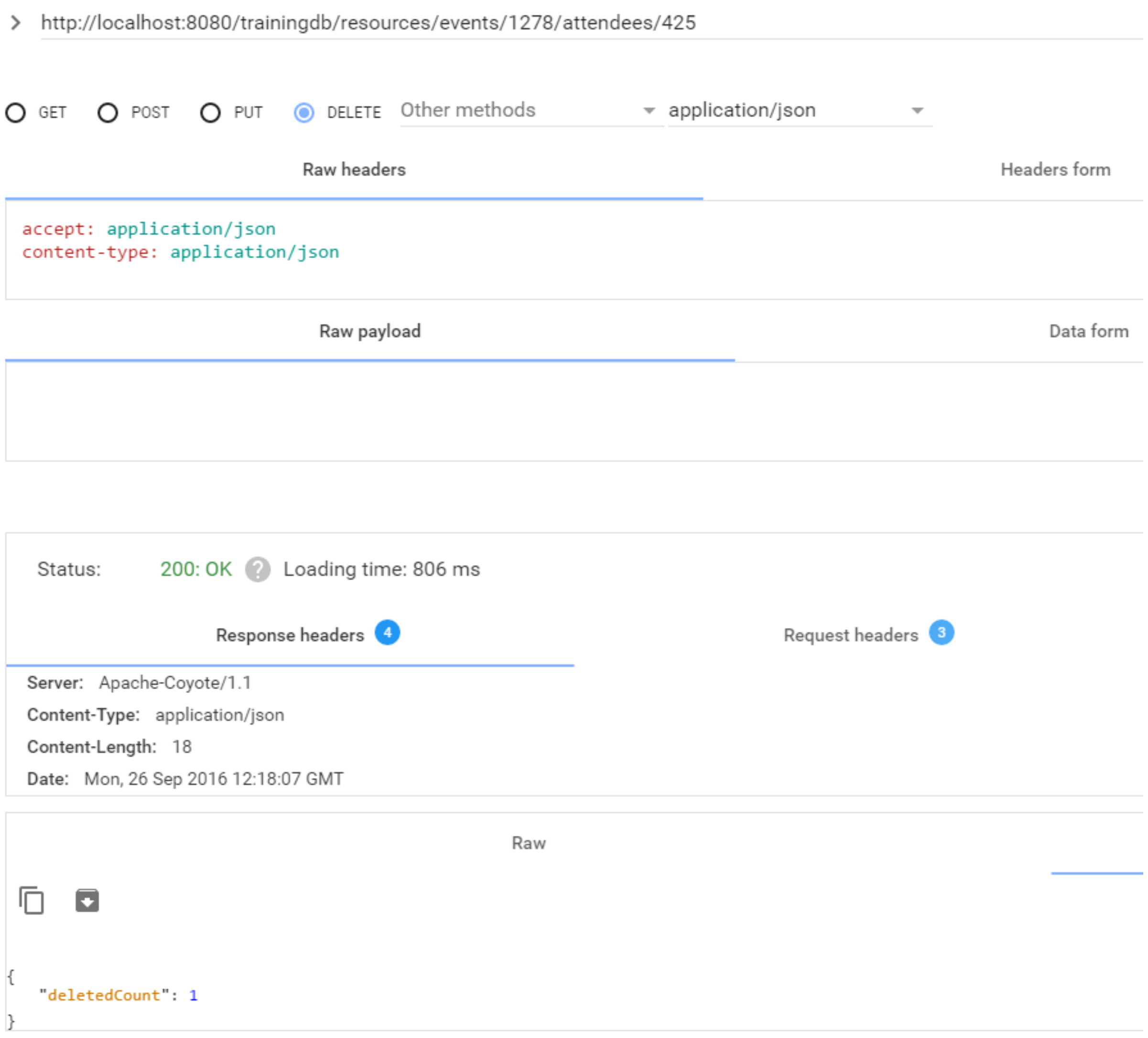
```
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 18
Date: Fri, 23 Sep 2016 13:34:07 GMT
```

Raw

```
{
  "deletedCount": 1
}
```

5.2.5.3 Removal of attendees/instructors/organizers from an event

HTTP Request Type	DELETE
URL	http://<host>/trapi/resources/private/events/<id>/attendees/<userId>
HTTP Header	accept: application/json authorization: xBearer ************
Data	



http://localhost:8080/trainingdb/resources/events/1278/attendees/425

[✓ DELETE] application/json

```
accept: application/json
content-type: application/json
```

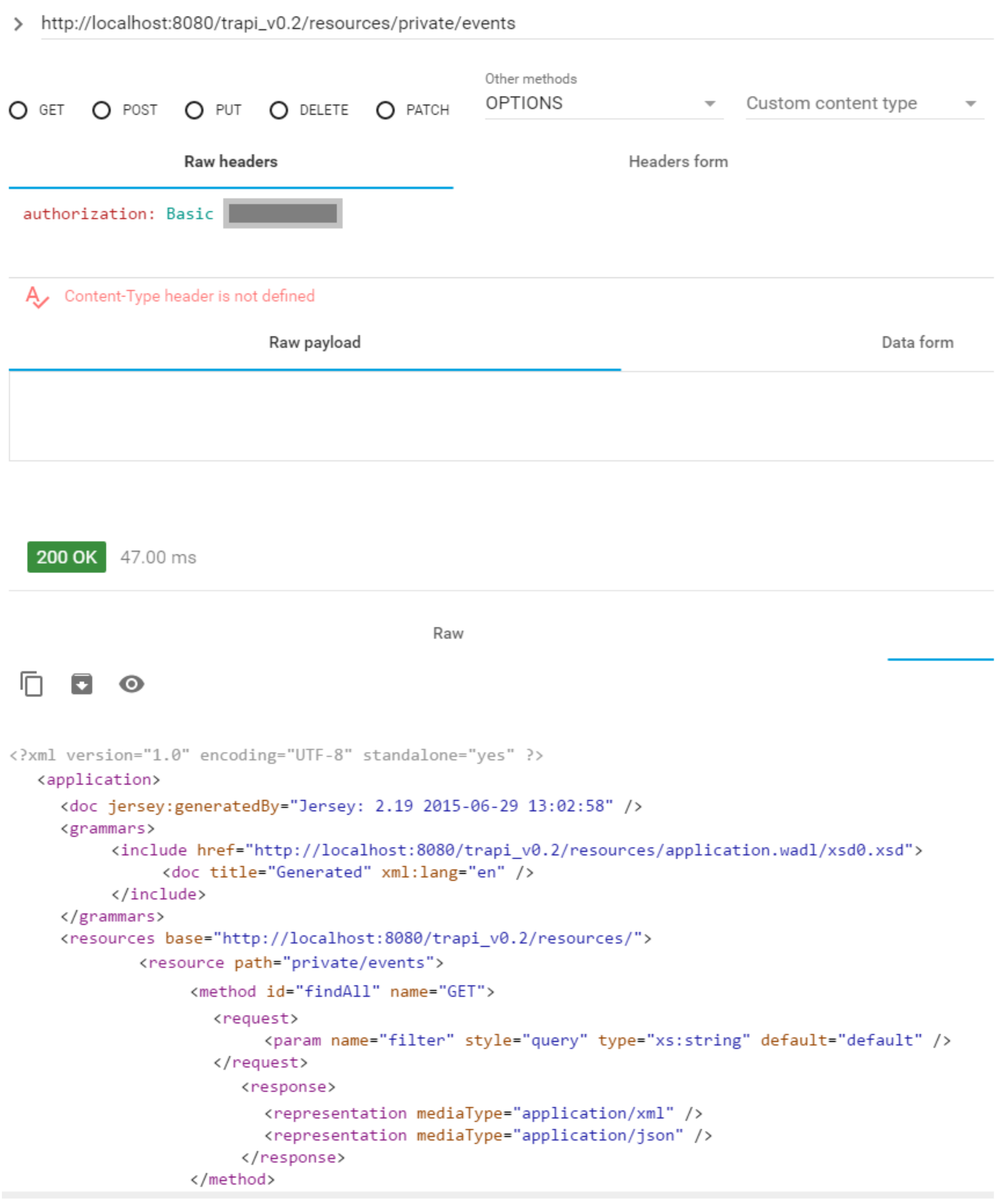
Status: 200: OK Loading time: 806 ms

Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 18
Date: Mon, 26 Sep 2016 12:18:07 GMT

```
{
  "deletedCount": 1
}
```

5.2.6 OPTIONS: Events API Description

HTTP Request Type	OPTIONS
URL	http://<host>/trapi/resources/private/events
HTTP Header	accept: application/xml authentication: authorization ************
Data	None


http://localhost:8080/trapi_v0.2/resources/private/events

```
authorization: Basic [redacted]
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
  <application>
    <doc jersey:generatedBy="Jersey: 2.19 2015-06-29 13:02:58" />
    <grammars>
         <include href="http://localhost:8080/trapi_v0.2/resources/application.wadl/xsd0.xsd">
              <doc title="Generated" xml:lang="en" />
         </include>
    </grammars>
    <resources base="http://localhost:8080/trapi_v0.2/resources/">
            <resource path="private/events">
                 <method id="findAll" name="GET">
                   <request>
                        <param name="filter" style="query" type="xs:string" default="default" />
                   </request>
                      <response>
                        <representation mediaType="application/xml" />
                        <representation mediaType="application/json" />
                      </response>
                 </method>
```

5.3 Authentication and Authorization

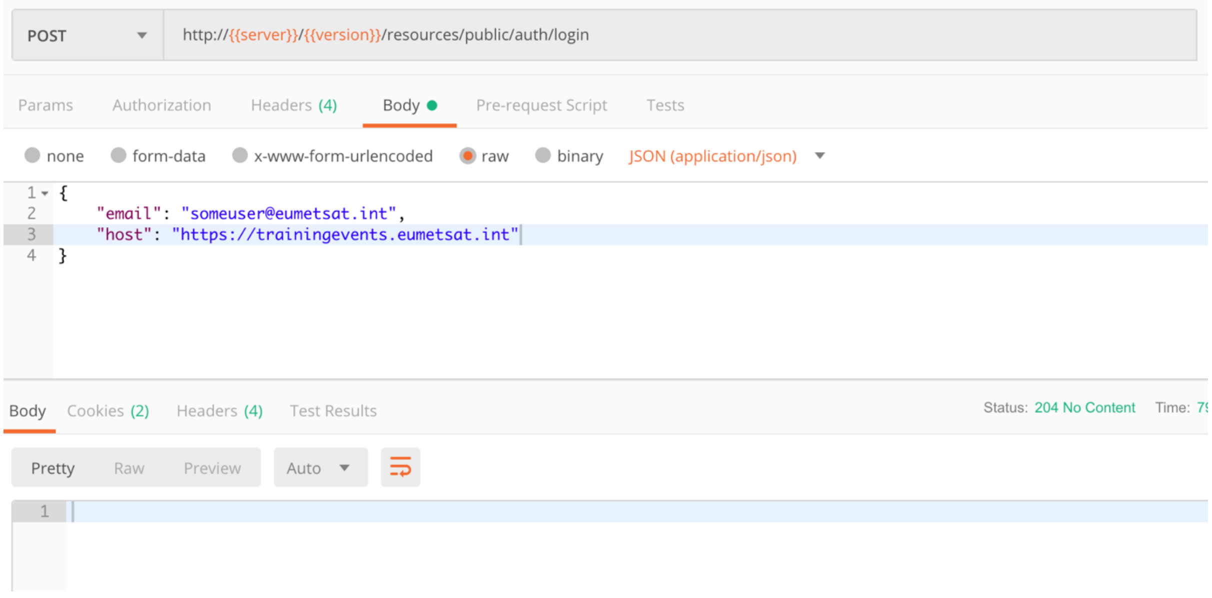
The authentication and authorization system is used through a collection of API endpoints, that are described in this section.

5.3.1 POST: Login User

This endpoint is used to obtain an authentication token for a certain user. The payload data requires an email and a host parameter. The email is used to verify if the user exists in the database, and an email is sent to that email address containing an authentication token that is valid for a limited time. The time how long a token is valid can be configured in the application.properties, as described in section 4.2.2. The host parameter points to the website at which the training calendar client is running.

The response body of this request is empty.

HTTP Request Type	POST
URL	http://<host>/trapi/resources/public/auth/login
HTTP Header	accept: application/json
Data	{ “email”: “example@email.com”, “host”: “http://trainingevents.eumetsat.int” }


POST http://{{server}}/{{version}}/resources/public/auth/login

Params Authorization Headers (4) Body ● Pre-request Script Tests

none form-data x-www-form-urlencoded [✓ raw] binary JSON (application/json)

```
{
    "email": "someuser@eumetsat.int",
    "host": "https://trainingevents.eumetsat.int"
}
```

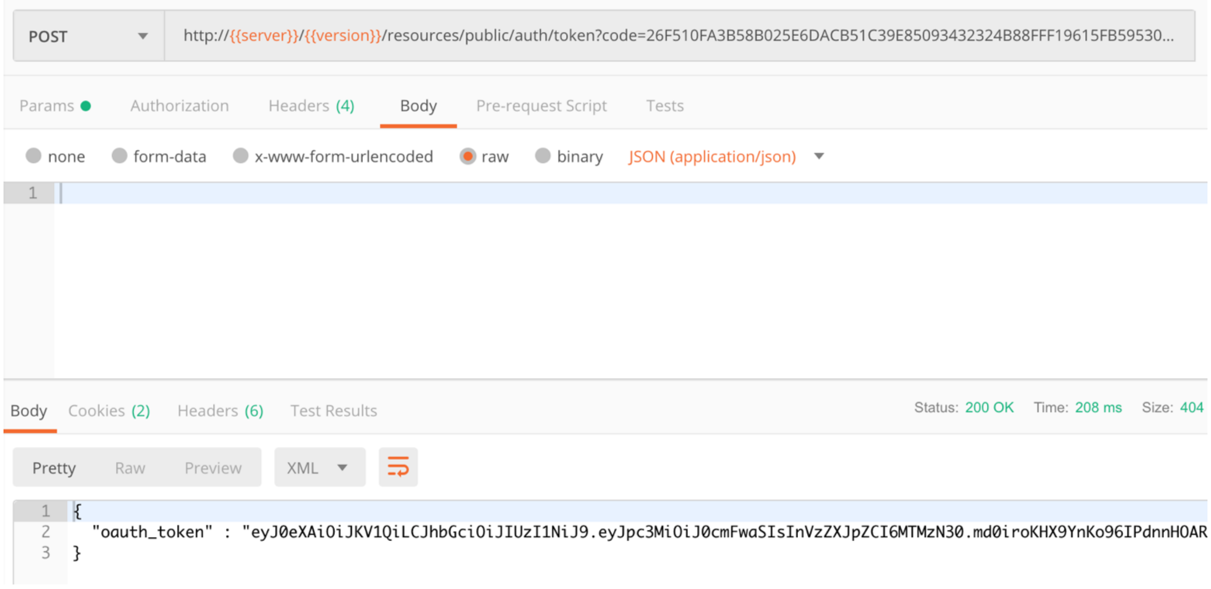
Body Cookies (2) Headers (4) Test Results Status: 204 No Content Time: 7

Pretty Raw Preview Auto

5.3.2 POST: Obtain Authorization Token

This endpoint is used to exchange the authentication token for an authorization token, which can be used in subsequent requests to restricted resources. The payload only requires the authentication token, sent to the user by email. Upon successful validation, an authorization token is returned in the body of the response. This token needs to be added to subsequent requests in order to give the requestee access to restricted resources, as

HTTP Request Type	POST
URL	http://<host>/trapi/resources/public/auth/token?code=<token>
HTTP Header	accept: application/json
Data	


POST http://{{server}}/{{version}}/resources/public/auth/token?code=26F510FA3B58B025E6DACB51C39E85093432324B88FFF19615FB59530...

Body: raw, JSON (application/json)

Status: 200 OK Time: 208 ms Size: 404

```
{
  "oauth_token" : "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJ0cmFwaSIsInVzZXJpZCI6MTMzN30.md0iroKHX9YnKo96IPdnnHOAR
}
```

5.4 Other Resources

The events database is additionally storing other secondary resources for holding the configuration and storing additional metadata for the Event main resource.

- http://<host>/trapi/resources/public/countries
- http://<host>/trapi/resources /public/trainingpartners
- http://<host>/trapi/resources /public/statuses
- http://<host>/trapi/resources /public/sponsoredtypes
- http://<host>/trapi/resources /public/regions
- http://<host>/trapi/resources /public/qualifications
- http://<host>/trapi/resources /public/profiles
- http://<host>/trapi/resources /public/languages
- http://<host>/trapi/resources /public/initiatives
- http://<host>/trapi/resources /public/expertisefields
- http://<host>/trapi/resources /public/eventtypes
- http://<host>/trapi/resources /public/attendancetypes
- http://<host>/trapi/resources /private/webusers
- http://<host>/trapi/resources/private/users
- http://<host>/trapi/resources/private/aclgroup
- http://<host>/trapi/resources/private/aclroles
- http://<host>/trapi/resources/private/aclpermissions

Please use the OPTIONS type HTTP requests to find out the available operations for each resource. All public resource can also be accessed from its private URL.

5.5 Filters

The filter is expected to follow the same syntax as per the "where clauses" in the Hibernate Query Language (HQL), including the optional "order by" statement. Some browsers request the URLs with filters to be encoded before transmitted via HTTP (Advanced REST Client supports encoding).

Filtering by basic attributes of the resource can be done as in the examples below with the Event resource:

- Filter all events located in Darmstadt:

filter=city='darmstadt'

- Filter all events located in Darmstadt or Madrid, ordered by date:

Error! Hyperlink reference not valid.','madrid') order by startDate*

- Filter all events which have been cancelled:

Error! Hyperlink reference not valid.

Filtering by list attributes, such as the languages attribute of the event resource, is also possible with the syntax described in the following examples:

- Filter by Languages: Events with language English or Spanish in the future

filter=languages.value in ('English','Spanish') and endDate>'2017-03-23'

- Initiatives: Events with initiatives 100 or 90 in the future:

filter=initiatives.id in (90,100) and endDate>'2017-03-23'

- WMO Regions: Events with WMO Region Europe in the future:

filter=regions.value='VI Europe' and endDate>'2017-03-23'

- Languages + Initiatives: Events in English with Initiatives 90 or 100 in the future

filter=languages.id=7 and initiatives.id in (90,100) and endDate>'2017-03-23'

5.6 Limiting queries

URLs can be appended with query parameters to limit and offset the results that are returned from the API. The following optional GET parameters can be appended to an URL:

- Limit: limits the number of results. Return only 10 results:

  limit=10

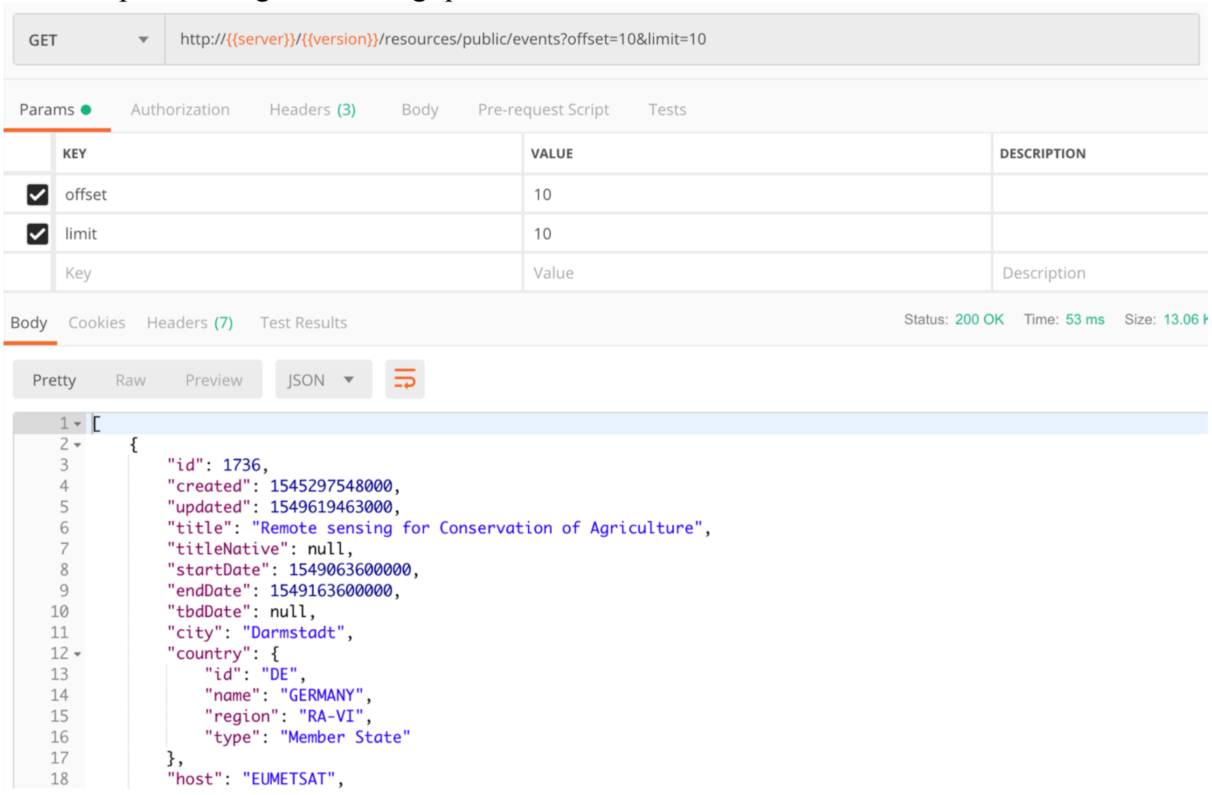
- Offset: skips the first x results. Useful for paging. Skip the first 10 results:

  offset=10

- Both parameters can be combined. To return the second set of 30 results, skipping the first 30 results, use:

  limit=30&offset=30

An example of using the limiting queries is shown in the screenshot below.



```
http://{{server}}/{{version}}/resources/public/events?offset=10&limit=10
```

	KEY	VALUE	DESCRIPTION
✓	offset	10	
✓	limit	10	
	Key	Value	Description

Status: 200 OK   Time: 53 ms   Size: 13.06

```
[
    {
        "id": 1736,
        "created": 1545297548000,
        "updated": 1549619463000,
        "title": "Remote sensing for Conservation of Agriculture",
        "titleNative": null,
        "startDate": 1549063600000,
        "endDate": 1549163600000,
        "tbdDate": null,
        "city": "Darmstadt",
        "country": {
            "id": "DE",
            "name": "GERMANY",
            "region": "RA-VI",
            "type": "Member State"
        },
        "host": "EUMETSAT",
```

6 DEPLOYMENT IN EUMETSAT

The tools is deployed in the TCE environment for internal use and in the usc.tools.eumetsat.int virtual machine for external access. The external access server is monitored regularly and anomalies are fixed in best effort basis.

The service is currently running in the URLs below and listening for requests:

- Internal Use: http://10.11.15.11/trapi/resources/public/events
- External Use: https://usc.tools.eumetsat.int/trapi/resources/public/events/
- External Use (VAL): https://usc.tools.eumetsat.int/trapi-val/resources/public/events/