CALL RECORDING

REST API Admin

Reference Guide

MOMENTUM

Powered By: MiaRec

UC24Q2

Table of contents

1. REST API

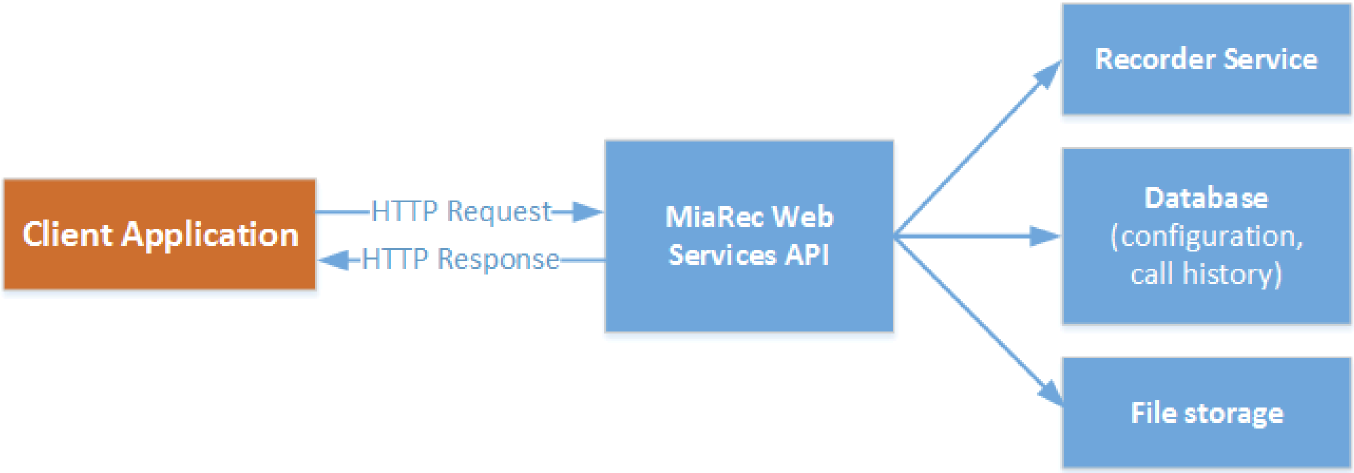
This is documentation for the Call Recording REST API from MiaRec

The API allows third-party applications to operate on Call Recording resources (for example, call recordings, users, groups) using only HTTP and JSON.

2. Architecture

The REST API is a single point of entry to all resources of the MiaRec platform. Client application communicates to MiaRec API to retrieve resources (for example, calls or users) and create/update/delete resources (for example, create new user, delete call).

Client application sends HTTP requests to the MiaRec API. HTTP response is sent back to the client with data in JSON format.



Example of request:

```
GET  http://<host:port>/api/v2/users/<user-id>.json
```

Example of response:

```
{
  "user": {
    "user_id": "546340bf-7b47-11e4-85a4-e03f497dbdff",
    "name": "John Smith",
    "group_id":  "256740bf-7b47-11e4-85a4-e03f497dbd33"
  }
}
```

2.1 Message encoding

2.1.1 HTTP Methods

MiaRec API supports the following HTTP methods:

HTTP Method	**Description**
GET	Retrieve details about resource or a list of resources (users, groups, calls etc). Examples: `https://miarec.example.com/api/v2/users.json`
http://miarec.example.com:8080/api/v2/users/e00f043e-f288-11e4-aa29-e03f497dbdff.json	
POST	Create new resource object
PUT	Modify the existing resource object
DELETE	Delete the resource object

2.1.2 HTTP Headers

- **HTTP Basic Authentication header**

  Each request should contain HTTP Basic Authentication header with a valid user name and password. The user should exist in MiaRec and have permission to access API. * **Accept header**

  Each request should contain `Accept: application/json` header because it is supposed that MiaRec API sends response in JSON format. Response from MiaRec API may contain text/plain response in case of error. * **Content-Type header**

  On PUT and POST operations the client application should supply header `Content-Type: application/json` and format the submitted data in JSON.

2.1.3 HTTP Body

The body of API request or response can optionally carry data in JSON format. The body can be sent by the client on a PUT or POST, or returned to the client on a GET.

2.1.4 Character Sets

Submitted JSON data should be always encoded with UTF-8 character set.

2.1.5 Format of date/time values with timezone

All date/time values are returned in ISO8601 format with timezone (like `20`). It is responsibility of client application to convert time to appropriate timezone before displaying to user.

The default timezone is configured on system level, although each API user account may have unique timezone settings.

2.2 Collections

2.2.1 Paging

Each response has `next_url` attribute like:

```
{
    "calls": [ ... ],
    "next_url": "/api/v2/calls.json?start=200"
}
```

`next_url` is set to **null** if there are no more pages. Otherwise the client application should use this URL to sent retrieve next portion of records.

When requesting a list of objects, by default MiaRec returns 20 records per page. The client application may request up to 1000 items per page with URI parameter `limit=X`, for example:

```
/api/v2/users.json?limit=500
```

2.2.2 Access Scope

The returned list of objects contains only those elements, which are in a scope of API user permissions.

For example, a client application requests a list of calls:

```
/api/v2/calls.json
```

If API request is made with credentials of system administrator role, then all calls will be returned to the client application.

If API request is made with credentials of group's administrator role, then only calls in his group will be returned to the client application.

Some lists can be ordered by transmitting a `sc _` [illegible]or `sort_order=asc` parameter to the end point. Whether a specific list can be ordered is specified in the documentation for that specific resource.

2.2.3 Search

The returned list of objects may be filtered to narrow the search results according to different attributes, like date range, user, group, search term etc.

Filtering parameters should be specified as URI parameters.

Examples:

- **Filter by data range**:

  ```
  /api/v2/calls.json?daterange=2014/11/01-2014/12/01
  ```

  Format of daterange is YYYY/MM/DD-YYYY/MM/DD

- **Filter by user id**:

  ```
  /api/v2/calls.json?user_id=546340bf-7b47-11e4-85a4-e03f497dbdff
  ```

  Such query will return all calls recorded for particular user. `user_id` is a unique ID of user created in MiaRec.

- **Filter by group id**:

  ```
  /api/v2/calls.json?group_id=d1d83c40-eec7-11e4-8558-e03f497dbdff
  ```

  Such query will return all calls recorded for users in particular group.

- **Filter by search term**:

  ```
  /api/v2/calls.json?search_term=1234
  ```

  Such query will return all calls which have `1234` text in phone number, phone name or call notes.

- **Filter by BroadWorks user id**:

```
/api/v2/calls.json?broadworks_user_id=user@broadworks.com
```

For BroadWorks SIPREC recording method such query will return all calls recorded for particular broadworks user id.

It is possible to specify multiple filter parameters simultaneously, for example:

```
/api/v2/calls.json?daterange=2014/11/01-2014/12/01&search_term=12345
```

2.3 Error responses

Unsuccessful requests are responded with HTTP status codes in the 400 range. The response may be content type `text/plain` for API level error messages (e.g when trying to call the API without valid user credentials). The error response has content type `application/json` for business level error messages. The latter contains a JSON formatted error messages to supplement the HTTP status code.

For example:

```
{
  “error”:          “RecordInvalid”,
  “description”:   “RecordValidation errors”,
  “details”: {
     “fields”: [
       {
          “field_name”:   “first_name”,
          “error”:         “blank”,
          “description”:  “can’t be blank”
       },
       {
          “field_name”:   “last_name”,
          “error”:         “blank”,
          “description”:  “can’t be blank”
       }
     ]
   }
}
```

HTTP status code in the 500 range may be returned in case of internal server issue. Contact system administrator to troubleshoot the issue.

2.4 Security and Authentication

MiaRec REST API requests are secured by the following methods:

- **Authentication**

  All requests to be processed must be authenticated before being processed. A valid user name and password should be provided in HTTP Basic Authentication header. The user should be a valid user as pre-configured in MiaRec by using the Web based graphical interface. Requests without authentication are rejected with a 401 Unauthorized error response.

- **Encryption**

  When accessing MiaRec REST API over public networks, it is highly recommended to use Hypertext Transfer Protocol Secure Sockets (HTTPS) for communication between client application and MiaRec REST API so that user names, passwords and contents are protected from snooping.

- **Role based permissions**

  Administrator must configure which resources/operations are accessible by users. When sending API request, if the appropriate permission is not enabled for the required user, an error response is returned with status code 403 Access is denied.

The configuration of users and their permissions is done by using the Web based graphical interface.

Example of authentication using curl:

```
curl -u {login}:{password} https://{your-miarec-server}/api/v2/users.json
```

2.5 REST API clients for development

During development it is useful to use REST API client application to send various requests to MiaRec server and check responses.

The most popular console application is **curl** (see screenshot below). Also, there are many REST API client plugins available for such browsers as Chrome and Firefox.

2.5.1 cURL

cURL is a command-line tool for transferring data using various protocols. It can be used to interact with the Redmine REST API.

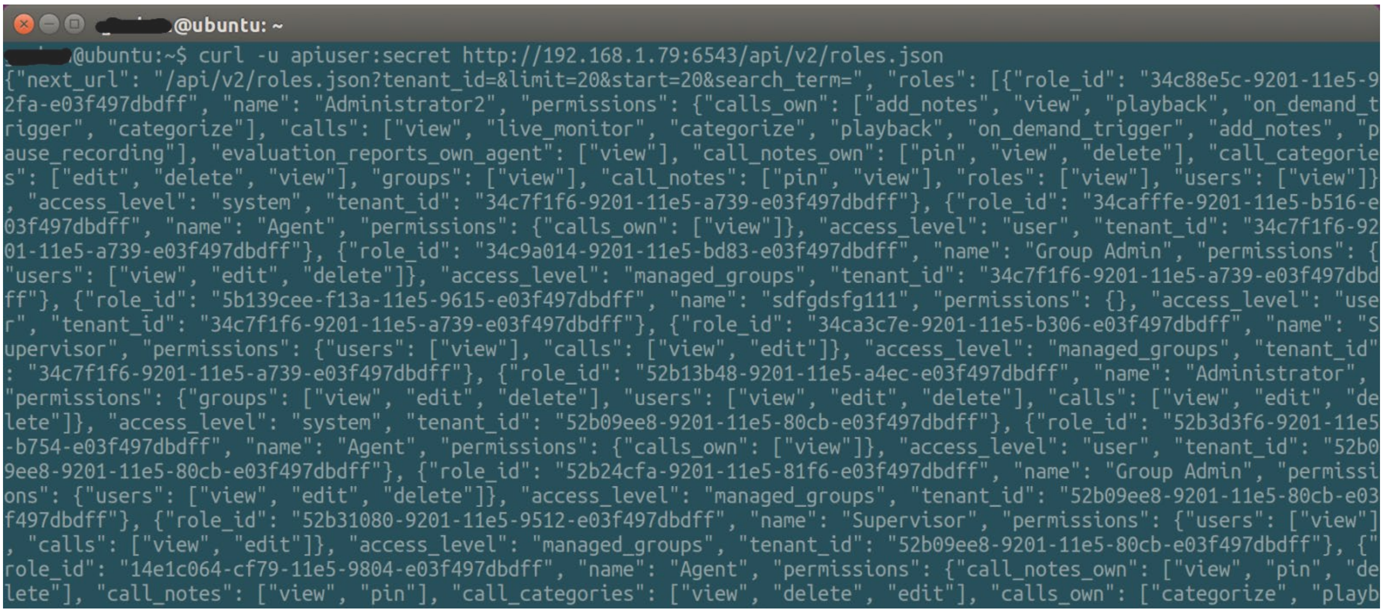
Here is a simple example of a command that can be used to retrieve a list of users from MiaRec server:

```
curl -u apiuser:secret http://{host}:{port}/api/v2/users.json
```

Sample response:

```
{
    "users": [{
        "name": "John Smith",
        "user_id":  "e011c408-f288-11e4-9b73-e03f497dbdff",
        ...
    },{
        "name": "Marry Smith",
        "user_id":  "34c7f1f6-9201-11e5-a739-e03f497dbdff",
        ...
    },
    ... MORE DATA ...
}
```

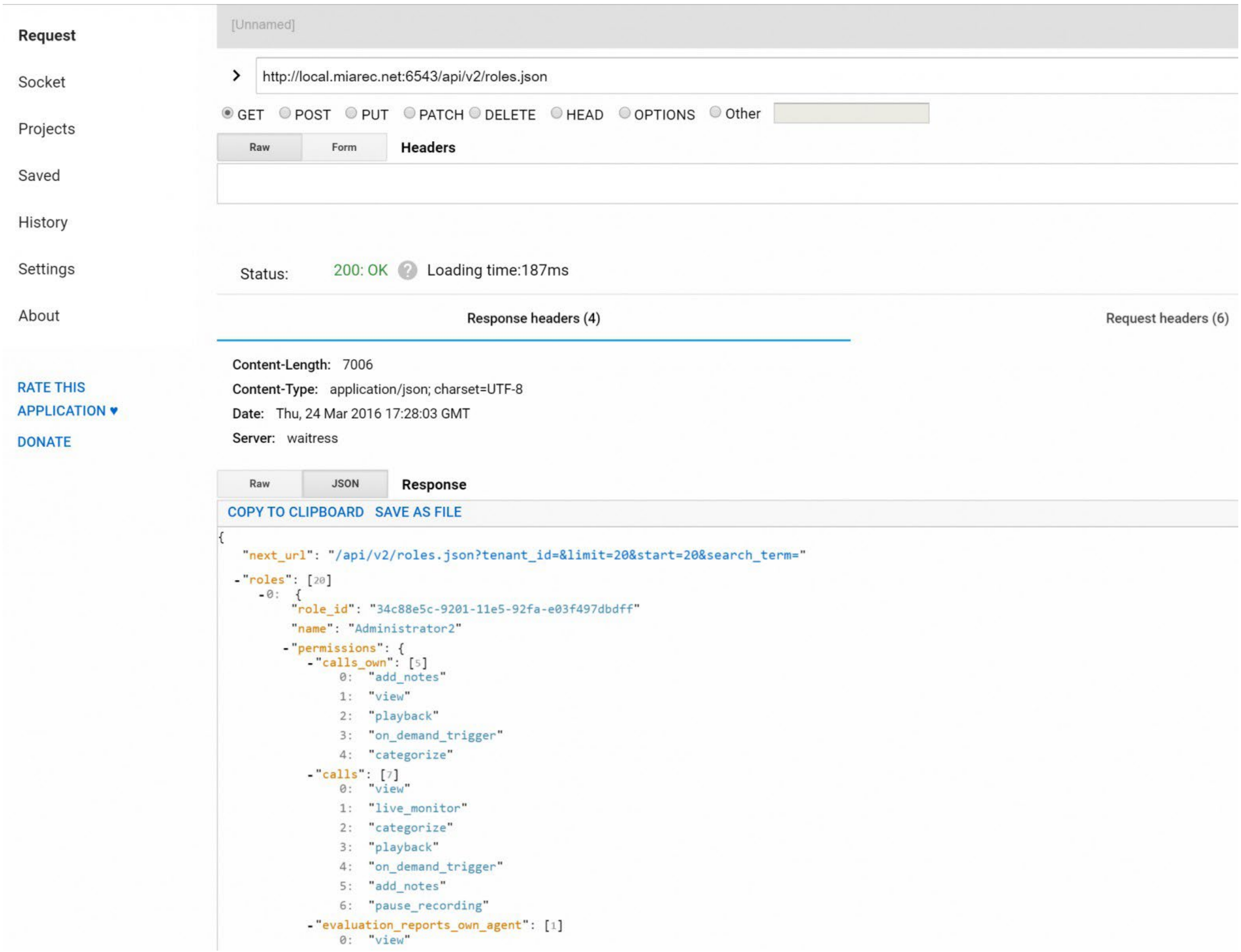
Screenshot of cURL utility:



```
[redacted]@ubuntu:~$ curl -u apiuser:secret http://192.168.1.79:6543/api/v2/roles.json
{"next_url": "/api/v2/roles.json?tenant_id=&limit=20&start=20&search_term=", "roles": [{"role_id": "34c88e5c-9201-11e5-9
2fa-e03f497dbdff", "name": "Administrator2", "permissions": {"calls_own": ["add_notes", "view", "playback", "on_demand_t
rigger", "categorize"], "calls": ["view", "live_monitor", "categorize", "playback", "on_demand_trigger", "add_notes", "p
ause_recording"], "evaluation_reports_own_agent": ["view"], "call_notes_own": ["pin", "view", "delete"], "call_categorie
s": ["edit", "delete", "view"], "groups": ["view"], "call_notes": ["pin", "view"], "roles": ["view"], "users": ["view"]}
, "access_level": "system", "tenant_id": "34c7f1f6-9201-11e5-a739-e03f497dbdff"}, {"role_id": "34cafffe-9201-11e5-b516-e
03f497dbdff", "name": "Agent", "permissions": {"calls_own": ["view"]}, "access_level": "user", "tenant_id": "34c7f1f6-92
01-11e5-a739-e03f497dbdff"}, {"role_id": "34c9a014-9201-11e5-bd83-e03f497dbdff", "name": "Group Admin", "permissions": {
"users": ["view", "edit", "delete"]}, "access_level": "managed_groups", "tenant_id": "34c7f1f6-9201-11e5-a739-e03f497dbd
ff"}, {"role_id": "5b139cee-f13a-11e5-9615-e03f497dbdff", "name": "sdfgdsfg111", "permissions": {}, "access_level": "use
r", "tenant_id": "34c7f1f6-9201-11e5-a739-e03f497dbdff"}, {"role_id": "34ca3c7e-9201-11e5-b306-e03f497dbdff", "name": "S
upervisor", "permissions": {"users": ["view"], "calls": ["view", "edit"]}, "access_level": "managed_groups", "tenant_id"
: "34c7f1f6-9201-11e5-a739-e03f497dbdff"}, {"role_id": "52b13b48-9201-11e5-a4ec-e03f497dbdff", "name": "Administrator",
"permissions": {"groups": ["view", "edit", "delete"], "users": ["view", "edit", "delete"], "calls": ["view", "edit", "de
lete"]}, "access_level": "system", "tenant_id": "52b09ee8-9201-11e5-80cb-e03f497dbdff"}, {"role_id": "52b3d3f6-9201-11e5
-b754-e03f497dbdff", "name": "Agent", "permissions": {"calls_own": ["view"]}, "access_level": "user", "tenant_id": "52b0
9ee8-9201-11e5-80cb-e03f497dbdff"}, {"role_id": "52b24cfa-9201-11e5-81f6-e03f497dbdff", "name": "Group Admin", "permissi
ons": {"users": ["view", "edit", "delete"]}, "access_level": "managed_groups", "tenant_id": "52b09ee8-9201-11e5-80cb-e03
f497dbdff"}, {"role_id": "52b31080-9201-11e5-9512-e03f497dbdff", "name": "Supervisor", "permissions": {"users": ["view"]
, "calls": ["view", "edit"]}, "access_level": "managed_groups", "tenant_id": "52b09ee8-9201-11e5-80cb-e03f497dbdff"}, {"
role_id": "14e1c064-cf79-11e5-9804-e03f497dbdff", "name": "Agent", "permissions": {"call_notes_own": ["view", "pin", "de
lete"], "call_notes": ["view", "pin"], "call_categories": ["view", "delete", "edit"], "calls_own": ["categorize", "playb
```

2.5.2 Advanced REST client for Google Chrome

Advanced REST client is running within Google Chrome browser and allows to create and test REST API requests.



http://local.miarec.net:6543/api/v2/roles.json

● GET ○ POST ○ PUT ○ PATCH ○ DELETE ○ HEAD ○ OPTIONS ○ Other

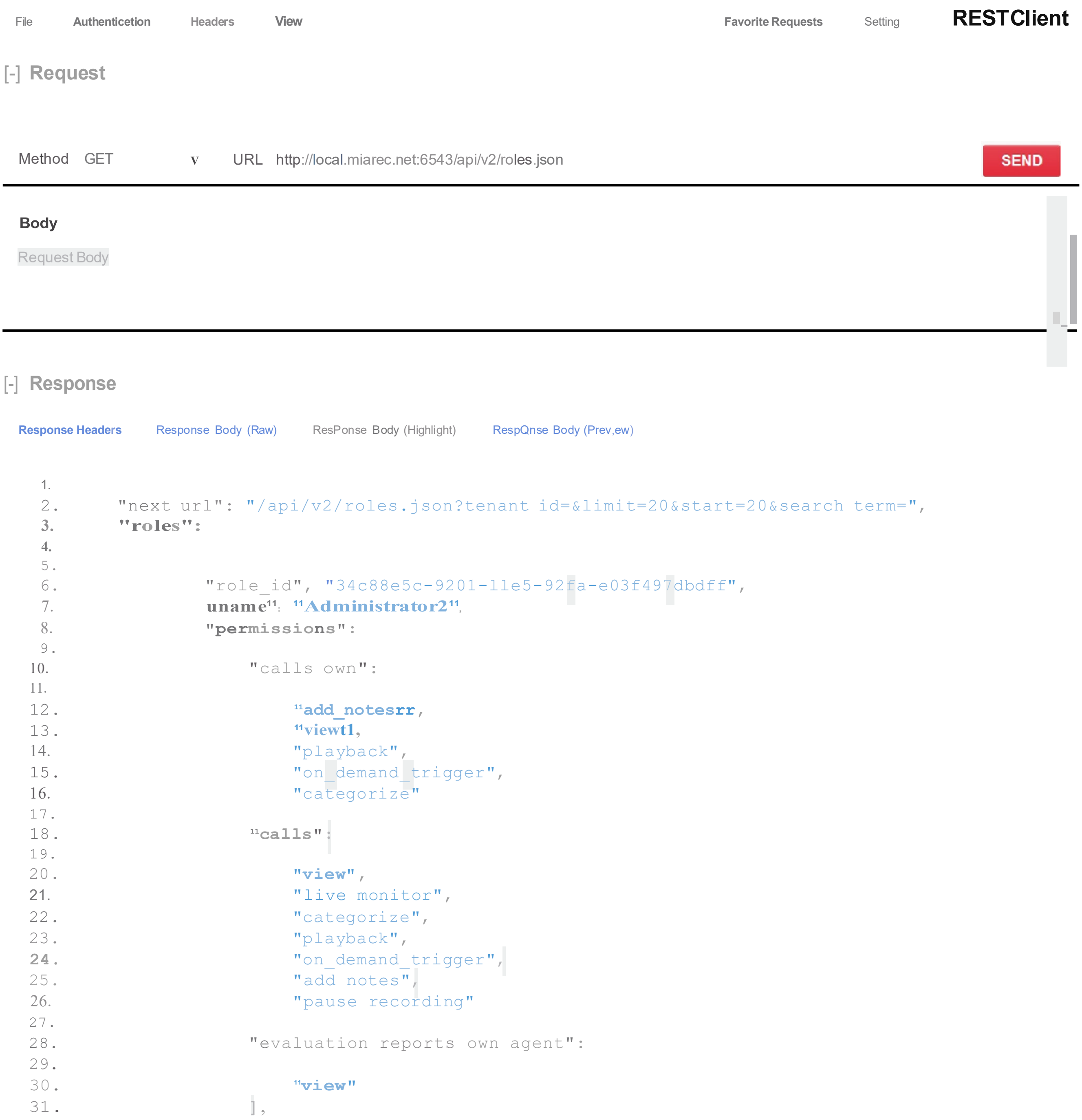
Status: 200: OK Loading time:187ms

Content-Length: 7006
Content-Type: application/json; charset=UTF-8
Date: Thu, 24 Mar 2016 17:28:03 GMT
Server: waitress

```
{
   "next_url": "/api/v2/roles.json?tenant_id=&limit=20&start=20&search_term="
 -"roles": [20]
    -0:  {
          "role_id": "34c88e5c-9201-11e5-92fa-e03f497dbdff"
          "name": "Administrator2"
         -"permissions": {
           -"calls_own": [5]
                0:  "add_notes"
                1:  "view"
                2:  "playback"
                3:  "on_demand_trigger"
                4:  "categorize"
           -"calls": [7]
                0:  "view"
                1:  "live_monitor"
                2:  "categorize"
                3:  "playback"
                4:  "on_demand_trigger"
                5:  "add_notes"
                6:  "pause_recording"
           -"evaluation_reports_own_agent": [1]
                0:  "view"
```

2.5.3 REST Client plugin for Firefox

RESTClient is an extension for Firefox, which you can use to create and test REST API requests.



Method GET v URL http://local.miarec.net:6543/api/v2/roles.json

Body

Request Body

```

    "next url": "/api/v2/roles.json?tenant id=&limit=20&start=20&search term=",
    "roles":


            "role_id", "34c88e5c-9201-lle5-92fa-e03f497dbdff",
            uname": "Administrator2".
            "permissions":

                "calls own":

                    "add_notesrr,
                    "viewt1,
                    "playback",
                    "on_demand_trigger",
                    "categorize"

                "calls":

                    "view",
                    "live monitor",
                    "categorize",
                    "playback",
                    "on_demand_trigger",
                    "add notes",
                    "pause recording"

                "evaluation reports own agent":

                    "view"
                ],
```

3. Resources

3.1 Calls

3.1.1 Call object attributes

Calls are represented as JSON objects, with the following keys:

call_id (UUID)

Unique ID, assigned by MiaRec application to this call record.

parent_call_id (UUID)

This field has value only for some voip protocols.

For Avaya H.323 Passive recording, when call is put on hold and then resumed, a new call recording instance is created. This new instance links to the original call via `parent_call_id` field.

interaction_id (UUID)

ID of interaction if this call is a part of multi-call interaction.

tenant_id

ID of tenant, which this calls is associated to

is_conference (boolean)

If true, then this cal is a conference with more than 2 participants

recorder_id (UUID)

Unique ID of the server, which recorded this call. In multi-site recording setup, this field allows to distinguish calls between locations

protocol_call_id (string)

Unique call id as provided by phone system.

For example, for SIP protocol this field is a "Call-Id" header in SIP INVITE message.

protocol_call_direction (integer)

Call direction as provided by phone system.

Possible values:

- `0` : A call direction is not provided by phone system, or not supported for this phone system (Unknown),
- `1` : Outbound
- `2` : Inbound

call_state (integer)

A state of the call.

Possible values:

- `1`: INITIATED, The caller sent invitation to the callee (e.g. `SIP INVITE` is sent)
- `2`: ACCEPTED, The callee received invitation and confirmed this (e.g. `SIP Trying` is sent)
- `3`: ALERTING, The callee started ringing (e.g. `SIP 183 Session Progress` is sent)
- `4`: CONNECTED, The call is answered (e.g. `SIP 200 OK` is sent)
- `5`: DISCONNECTING, One of parties initiated call disconnect (e.g. `SIP BYE` is sent)
- `6`: DISCONNECTED, The call is completed (e.g. `SIP BYE` is confirmed with `200 OK`)
- : HOLD, The call has been put on hold. For some voip protocols this call state is a final state, meaning that call recording is completed when call is put on hold. When call is resumed, a new call instance is started. For others protocols this state is an intermediate state, meaning that it will switch to CONNECTED state when call is resumed from hold.
- `8`: TRANSFERRED, The call has been transferred to third party. Note, for some voip protocols even if the call has been transferred, it's call state is stored as DISCONNECTED rather than TRANSFERRED. This occurs because on protocol level the reason of call disconnect is not provided by phone system.

on_demand_state (integer)

The state of on-demand recording:

- `0`: DISABLED, On-demand triggers are not allowed for this call (e.g. call is configred as "always record")
- `1`: KEEP_RECORDING, On-demand trigger was received and call will be kept
- : WAITING_TRIGGER, Waiting for on-demand trigger. If not received, then call will be deleted automatically upon completio

record_state (integer)

Recording state:

- `10`: ACTIVE, Call is in process of normal recording
- : LICENSE_OVERUSAGE, Call is recorded, but it is not possible to playback it due to license over-usage. In this case, the audio file is encrypted. Contact vendor to decrypt such files. This state is valid for both active calls and diconnected.
- `30`: FINISHED, Call recording is finished normally
- : IGNORED, Call is ignored by recording filters. Only call metada is stored in database. The audio file is not created for such calls

voip_protocol (integer)

Voip signaling protocol of the call:

- `0` : UNKNOWN, Unrecognized protocol. Call is recorded from RTP packets only
- `1` : SIP
- `2` : H.323
- `4` : SCCP (Cisco Skinny)
- `5` : MGCP
- `6` : Avaya (H.323 protocol with proprietary extensions)
- `7` : Nortel UNISTIM
- `8` : TAPI
- `9` : MGCP PRI Backhaul (it is used between Cisco UCM and Voice Gateway)
- `10` : Alcatel (proprietary protocol used by Alcatel OmniPCX - partially supported)
- `11` : Avaya (passive RTP protocol)
- `12` : Avaya TSAPI + passive RTP
- `13` : SIPREC
- `14` : Cisco Built-in-Bridge (active recording)
- `15` : NEC SIP (proprietary protocol)
- `16` : SIP ED137 radio (passive recording)
- `17` : Cisco Built-in-Bridge(passive recording)

setup_time (datetime)

Date/time when call was initiated. Format is ISO8601 with timezone, for example `2014-06-10T13:45:51+0800`

connect_time (datetime)

Date/time when call was answered. Format is ISO8601 with timezone.

disconnect_time (datetime)

Date/time when call was disconnected. Format is ISO8601 with timezone.

from_ip, **to_ip** (string)

Ip-address of caller/called party. Format is `x.x.x.x`, for example `192.168.0.10`

from_port, **to_port** (integer)

Ip port of caller/called party

from_mac, **to_mac** (string)

Mac-address of caller/called party. Format is `XX-XX-XX-XX-XX-XX

from_number, **to_number**, **from_name**, **to_name**, **from_id**, **to_id** (string)

Number/name/id of caller/called party.

This value is provided by phone system.

For SIP protocol these values are extracted from the "From" and "To" headers of SIP INVITE message.

Example of SIP INVITE mesage:

```
INVITE sip:102@192.168.0.10 SIP/2.0
From: "John Smith" <sip:100@192.168.0.10>
To: "Emy" <sip:102@192.168.0.10>
```

In this case:

- from_number = "100"
- from_name = "John Smith"
- from_id = "100@192.168.0.10"
- to_number = "102"
- to_name = "Emy"
- to_id = "102@192.168.0.10"

Note, for SIP protocol, the values of these fields may be extracted from SIP headers "Remote-Party-ID" and "P-Asserted-Identity".

Example of SIP message:

```
INVITE sip:102@192.168.0.10 SIP/2.0
From: "John Smith" <sip:100@192.168.0.10>
To: "Emy" <sip:102@192.168.0.10>
Remote-Party-ID: "John" <sip:77@ex.com>;party=calling
```

In this case:

- from_number = "77"
- from_name = "John"
- from_id = "77@ex.com"

transferred_from_number, **transferred_from_name**, **transferred_from_id** (string)

Number/name/id of phone, from which the call was transferred.

This value depends on voip signaling protocol.

For example, for Cisco Skinny protocol these fields are the same as "Last Redirecting Party Name/Number"

transferred_to_number, **transferred_to_name**, **transferred_to_id** (string)

Number/name/id of phone, to which the call was transferred.

This value depends on voip signaling protocol.

agent_id, **agent_name** (string)

For Avaya TSAPI protocol these fields are id and name of agent.

broadworks_user_id, **broadworks_group_id**, **broadworks_sp_id** (string)

Broadworks-specifid Ids for SIPREC protocol

metaswitch_user, **metaswitch_group**, **metaswitch_system** (string)

Metaswitch-specifid Ids for SIPREC protocol

cisco_nearend_guid, **cisco_farend_guid** (string)

Cisco near-end and far-end GUIDs for Cisco Built-in-Bridge recording

cisco_nearend_refci, **cisco_farend_refci** (string)

Cisco near-end and far-end REFCI values for Cisco Built-in-Bridge recording

cisco_phone_ip (string)

IP-address of Cisco phone for Cisco Built-in-Bridge recording

cisco_nearend_partition, **cisco_farend_partition** (string)

Cisco near-end and far-end partition info for Cisco Built-in-Bridge recording

participants (list of objects Participant)

A list of call participants. See Participant object attributes for details.

Normally, there are only two participants of each call.

But for a conference call it may be more than 2 participants.

files (list of objects File)

A list of audio files. See File object attributes for details.

Normally, only one audio file exists per call instance. But in some cases it may be multiple audio files per call:

- When MiaRec is configured to detect log silence periods in a call, the recording is split on multiple audio files when silence is detected.
- When there is license issue (not valid or not enough), then a part of call is encrypted. In this case, the call will have at least two audio files. First one will be relatively short (usually less than 15 seconds) and the second one will contain the remaing part of a conversation. The second file will be encrypted.

3.1.2 File object fields

Audio files are represented as JSON objects which have the following keys:

Field	Type	Description
file_id	string	Unique file id It is unique only within a scope of this call instance.
start_time	datetime	Date/time when audio recording for this particular file has been started. Format is ISO8601 with timezone, for example, `2014-06-10T13:45:51+0800`
stop_time	datetime	Date/time when audio recording for this particular file has been stopped
file_size	integer	Size of audio file in bytes
file_path	string	Location of audio file on disk.
watermark	string	SHA1 hash of audio file content. It is used to validate data integrity

3.1.3 Participant object fields

Each call has at least 2 call participants (caller and called party). Conference calls may have more than 2 participants (such scenarios are not supported at the moment, but may be added in future).

Call participants are represented as JSON objects with the following attributes:

Field	Type	Description
participant_id	string	Unique participant id It is unique only within a scope of this call instance.
join_time	datetime	Date/time when the participant joined the call. Format is ISO8601 with timezone, for example, `2014-06-10T13:45:51+0800`
leave_time	datetime	Date/time when the participant left the call
user_id	string	If such call participant is known to MiaRec, then `user_id` field points to existing MiaRec user account. This is field can be `null` when participant is not know
party_direction	integer	Participant direction: • 0 - UNKNOWN • 1 - CALLER • 2 - CALLED
party_type	integer	Participant type: • 0 - UNKNOWN • 1 - AGENT • 2 - GATEWAY • 3 - CONFERENCE_BRIDGE
party_number **party_name** **party_caller_id**	string	Name, number and id of participant. These values are provided by phone system

\

3.1.4 List and search calls

List and search calls

GET A LIST OF ALL CALLS:

```
GET /api/v2/calls.json
```

Example of JSON response:

```
{
    "next_url": null,
    "calls":[{
        "call_id": "e03f497d-bdff-1019-102b-f5caab54f081",
        "setup_time": "2013-10-12T16:32:22-0800"
        "from_number": "102",
        "to_number": "101",
        ...
    },{
        "call_id": "e03f497d-bdff-1019-1023-694f40413c84",
        "setup_time": "2013-10-12T18:45:02-0800"
        "from_number": "103",
        "to_number": "105",
         ...
    }]
}
```

PAGING

Each response has `next_url` attribute like:

```
{
    "calls": [ ... ],
    "next_url": "/api/v2/calls.json?start=200"
 }
```

`next_url` is set to **null** if there are no more pages. Otherwise the client application should use this URL to sent retrieve next portion of records.

When requesting a list of objects, by default MiaRec returns 20 records per page. The client application may request up to 1000 items per page with URI parameter `limit=X`, for example:

```
/api/v2/calls.json?limit=500
```

CALCULATING A TOTAL NUMBER OF RECORDS

A calculation of a total number of records matching to the search criteria can be very expensive operation. For optimization purposes, the MiaRec application doesn't calculate such value. The `next_url` attribute is used to signal if more data is available beyond the currently requested page. On the very last page, the application sets `next_url` attribute to **null** and includes the `total` attribute, which is equal to a total number of records of all pages.

The following example represents the last page of data, i.e. no more data is available:

```
{
    "calls": [ ... ],
    "next_url": null,
    "total": 513
 }
```

How does the application know that there is more data available beyond the current page?

It is quite easy. The application queries the database for `limit+1` records. The last records, if available, is discarded, i.e. the application always returns up to `limit` records. A presence of the last record is a signal that more data is available beyond the currently requested page. Yet, it is not known if there is only 1 extra record available or a lot more.

The MiaRec web portal uses one trick to improve user experience when paginating data.

It shows a pagination counter like "0-20 of many" to signal that more data is available. Word "many" means that there are more than 1,000 records available beyond the current page, otherwise, it shows the pagination counter like "0-20 of 893" telling the end user that only 893 records are available in database. How does it do that?

Actually, the MiaRec web portal queries 1,000 records instead of the page size (`limit` attribute). For example, when end user wants to see the first 20 records, the application actually queries 1,000 records. It calculates a total number of the returned records. If it is less than 1,000, then it shows that exact number to end user (for example, 0-20 of 813), otherwise it shows "many" word telling that a lot more records are available (more than 1,000).

To replicate the same user experience in your application, you can pass the URI parameter `max_total_calc` to REST API. If `max_total_calc` is set to 1,000, then the application queries 1,000 records even if `limit` is set, for example, to 50.

A response still contains 50 or less records, but the `total` attribute will be set to a non-null value no more than 1,000 records available beyond the current page.

Note, for performance reasons, a maximum accepted value for `max_total_calc` parameter is limited to 1,000.

Example request:

```
/api/v2/calls.json?limit=50&max_total_calc=1000
```

Response:

```
{
    "calls": [ ... ],
    "next_url": "/api/v2/calls.json?start=50&limit=50&max_total_calc=1000"
    "total": 813
 }
```

In this example, the `total` attribute signals that only 813 records are available in DB (less than 1,000 beyond the currently requested page 0-50). Note, the `next_url` attribute is not **null**, i.e. more pages are available beyond page 0-50.

SEARCH

The returned list of objects may be filtered to narrow the search results according to different attributes, like date range, user, group, search term etc.

Filtering parameters should be specified as URI parameters.

REST API supports two search engines:

- Basic search
- Advanced search

The Basic search supports filtering recordings by the most frequently used attributes.

The Advanced search supports more attributes for searching as well as various comparison operators like "ends with", "is empty", etc. This search capability is the same as available in the MiaRec web portal on the **Advanced Search** tab.

CHILD PAGES

- Basic search
- Advanced search

Basic search

Basic search is activated when URI attribute `advanced_search` is missing or set to `0`, like:

```
/api/v2/users.json?advanced_search=0
```

Example of basic search using `daterange` and `search_term` attributes (note, multiple filtering parameters can be specified at the same time):

```
/api/v2/calls.json?daterange=2014/11/01-2014/12/01&search_term=12345
```

The following table lists the attributes available for basic search.

Parameter	Description

daterange	Filter by date range. Format of daterange is either **YYYY/MM/DD-YYYY/MM/DD** or **YYYY/MM/DD**
	For example, the following query will return all recordings from 2014/11/01 00:00:00 to 2014/12/01 24:00:00:
	`/api/v2/calls.json?daterange=2014/11/01-2014/12/01`
	The following query will return all recordings from 00:00:00 to 24:00:00 of 2014/11/01
	`/api/v2/calls.json?daterange=2014/11/01`
	Note, the query will be executed with REST API user's timezone. If user's timezone is not configured, then groups, tenants or system default timezone will be used.

user_id	Filter by user. The following query will return all recordings associated with the user identified by ID. `user_id` is a unique ID of user created in MiaRec: `/api/v2/calls.json?user_id=546340bf-7b47-11e4-85a4-e03f497dbdff`
user_login	Filter by user login. The following query will return all recordings associated with the user identified by login: `/api/v2/calls.json?user_login=john.smith`
group_id	Filter by group. The following query will return all recordings associated with the specified group: `/api/v2/calls.json?group_id=d1d83c40-eec7-11e4-8558-e03f497dbdff`
tenant_id	Filter by tenant. The following query will return all recordings associated with the specified tenant: `/api/v2/calls.json?tenant_id=d1d83c40-eec7-11e4-8558-e03f497dbdff`
search_term	Search a text in the phone number, phone name and notes. Example: `/api/v2/calls.json?search_term=1234`
category_id	Filter recordings by category. Example: `/api/v2/calls.json?category_id=de8440f6-f291-11e4-9476-e03f497dbdff`
broadworks_user_id	Filter recordings by Broadworks User ID. Example: `/api/v2/calls.json?broadworks_user_id=user@broadworks.com`
broadworks_group_id	Filter recordings by Broadworks Group ID. Example: `/api/v2/calls.json?broadworks_group_id=GroupA`
active_only	Return active calls only.

Parameter	Description
	Example: `/api/v2/calls.json?active_only=1`

Advanced search

Advanced search is activated when URI attribute `advanced_search` is set to `1`, like:

```
/api/v2/calls.json?advanced_search=1
```

To support various comparison operators like "ends with", "is not empty", the URI parameters should be formatted as:

```
[PARAM_NAME]__[OPERATOR]=[VALUE]
```

Where:

- **[PARAM_NAME]** is an attribute name, like "user_id", "from_number", etc.
- **[OPERATOR]** is a comparison operator, like "is_empty", "lower_than", etc.
- **[VALUE]** is the value to compare the attribute to. Note, the value is always required, even for operators like "is_empty". In this case, supply `1` or any other value, which will be ignored in the end.

Note, two underscore characters (__) are used as a separator between attribute name and operator.

Example of the advanced search URI query:

```
/api/v2/calls.json?advanced_search=1&phone_number__includes=1234&user_name__equal_to=J%20Smith
```

This is equivalent to:

```
("phone_number"  INCLUDES  "1234")  AND  ("user_name"  EQUAL_TO  "J Smith")
```

Note, a space character is encoded with %20 in URI

Table 1. Supported operators

Parameter	Type (see Table 2)	Description
date	`date`	Date of call recording
datetime	`datetime`	Date/Time of call recording
duration	`duration`	Duration of call recording
direction	`set`	Call Direction
voip_protocol	`set`	Voip Protocol
user_id	`set`	User ID
user_name	`string`	User Name
group_id	`set`	Group ID
tenant_id	`set`	Tenant ID
category_id	`set`	Category ID
client_id	`set`	Client ID
notes	`string`	Notes

notes_count	`number`	Notes Count
call_id	`string_exact`	Call ID
pbx_call_id	`string`	PBX Call ID
pbx_tracking_id	`string`	PBX Tracking ID
call_state	`set`	Call State
record_state	`set`	Recording State
phone_number	`string`	Phone Number
phone_number_from	`string`	Phone Number (FROM only)
phone_number_to	`string`	Phone Number (TO only)
phone_name	`string`	Phone Name
phone_name_from	`string`	Phone Name (FROM only)
phone_name_to	`string`	Phone Name (TO only)
phone_id	`string`	Phone ID
phone_id_from	`string`	Phone ID (FROM only)
phone_id_to	`string`	Phone ID (TO only)
orig_calling_number	`string`	Orig Calling Number
orig_dialed_number	`string`	Orig Dialed Number
acd_number	`string`	ACD Number
acd_name	`string`	ACD Name
acd_id	`string`	ACD ID
redirected_from_number	`string`	Redirected From Number
redirected_from_name	`string`	Redirected From Name
redirected_from_id	`string`	Redirected From ID

Parameter	Type (see Table 2)	Description
redirected_to_number	`string`	Redirected To Number
redirected_to_name	`string`	Redirected To Name
redirected_to_id	`string`	Redirected To ID
phone_ip_address	`string`	IP Address
phone_ip_address_from	`string`	IP Address (FROM only)
phone_ip_address_to	`string`	IP Address (TO only)
broadworks_sp_id	`string`	Broadworks SP ID
broadworks_group_id	`string`	Broadworks Group ID
broadworks_user_id	`string`	Broadworks User ID
cisco_phone_ip	`string`	Cisco Phone IP Address
cisco_refci	`string_exact`	Cisco xRefCi
cisco_ucce_agent_id	`set`	Cisco UCCE Agent
cisco_ucce_skill_group_id	`set`	Cisco UCCE Skill Group
cisco_ucce_recovery_key	`number`	Cisco UCCE Call ID
metaswitch_system	`string`	Metaswitch System Name
metaswitch_group	`string`	Metaswitch Group Name
metaswitch_user	`string`	Metaswitch User Name
metaswitch_extension	`string`	Metaswitch User Extension
agent_id	`string`	Avaya Agent ID
agent_name	`string`	Avaya Agent Name
evaluation_report_score	`number`	Evaluation Report Score
evaluation_report_status	`set`	Evaluation Report Status
evaluation_reports_count	`number`	Evaluation Reports Count
screen_recordings_count	`number`	Screen Recordings Count
file_path	`string`	File Path
encrypt_fingerprint	`string_exact`	Encrypt Fingerprint
confidential	`bool`	Confidential Flag

Table 2. Supported comparison operators

Parameter type	Supported operators
string	• `equal_to` - attribute's value matches exactly a text • `not_equal_to` - attribute's value doesn't match a text • `starts_with` - attribute's value starts with a text • `ends_with` - attribute's value ends with a text • `includes` - attribute's value includes a text • `is_empty` - attribute's value is an empty string or not specified • `not_empty` - attribute's value is not an empty string • `pattern` - attribute's value matches a simple pattern (similar to SQL LIKE) using `_` symbol to match exactly one character and `%` symbol to match zero or more characters. • `regex` - attribute's value matches a Regular Expression pattern.
string_exact	• `is` - attribute's value matches exactly a text • `is_not` - attribute's value doesn't match a text • `is_empty` - attribute's value is an empty string or not specified • `not_empty` - attribute's value is not an empty string
string_query	• `query` - attribute's value matches to a query expression • `is_empty` - attribute's value is an empty string or not specified • `not_empty` - attribute's value is not an empty string
number	• `equal_to` - attribute's value equals to a number • `not_equal_to` - attribute's value doesn't equal a number • `greater_than` - attribute's value is greater than a number • `lower_than` - attribute's value is lower than a number • `between` - attribute's value is between two numbers • `is_empty` - attribute's value is not available (NULL) • `not_empty` - attribute's value is available (NOT NULL)
date	• `equal_to` - attribute's value equals to a date • `older_than` - attribute's value is before a date • `newer_than` - attribute's value is after a date • `between` - attribute's value is between dates, separated by " - " • `older_than_days` - attribute's value is older than a specified number of days (integer) • `newer_than_days` - attribute's value is newer than a specified number of days (integer) Format of date is: • **YYYY/MM/DD** • **YYYY/MM/DD - YYYY/MM/DD** - a range of date values (note, the space characters around the separator '-' are not required)
datetime	• `older_than` - attribute's value is before a date+time • `newer_than` - attribute's value is after a date+time • `between` - attribute's value is between two date+time values, separated by "/" • `older_than_minutes` - attribute's value is older than a specified number of minutes • `newer_than_minutes` - attribute's value is newer than a specified number of minutes

Parameter type	Supported operators
	Format of datetime is ISO8601. Example values: • **2019-04-30T06:00:00.000Z** • **2019-04-30T06:00:00.000Z/2019-04-30T06:30:00.000Z** - a range of datetime values (note, a separator character is "/")
duration	• `greater_than` - attribute's value is lower than a duration • `lower_than` - attribute's value is after a duration • `between` - attribute's value is between two duration values, separated by " - " Format of duration is: • **SS** - a number of seconds • **MM:SS** - minutes + seconds • **HH:MM:SS** - hours + minutes + seconds • **SS - SS** or **MM:SS - MM:SS** or **HH:MM:SS - HH:MM:SS** - a range of duration values (note, the space characters around the separator '-' are not required)
set	• `is` - attribute's value equals to a text • `is_not` - attribute's value doesn't equal to a text
bool	• `is_true` - attribute's value is TRUE • `is_false` - attribute's value is FALSE or not specified (NULL)

3.1.5 View one call

Request:

```
GET  /api/v2/calls/<call-id>.json
```

Response is a JSON formatted object with call metadata:

```
{
  "call": {
    "call_id": "20f5acd7-bfbd-58c3-9a81-e453065fa4fe",
    "parent_call_id": null,
    "secondary_parent_call_id": null,
    "tenant_id": "068b518c-d311-11ed-9cd6-d45d64081e0f",
    "interaction_id": null,
    "protocol_call_id": null,
    "protocol_call_direction": 1,
    "protocol_tracking_id": null,
    "recorder_id": null,
    "call_state": 6,
    "on_demand_state": null,
    "record_state": 30,
    "voip_protocol": 13,
    "confidential": null,
    "setup_time": "2023-08-30T17:16:20-07:00",
    "connect_time": "2023-08-30T17:16:20-07:00",
    "disconnect_time": "2023-08-30T17:18:28-07:00",
    "duration": 128,
    "from_ip": null,
    "from_port": null,
    "from_mac": null,
    "from_number": "123000001",
    "from_name": null,
    "from_id": null,
    "to_ip": null,
    "to_port": null,
    "to_mac": null,
    "to_number": "8001000001",
    "to_name": null,
    "to_id": null,
    "redirected_from_number": null,
    "redirected_from_name": null,
    "redirected_from_id": null,
    "redirected_to_number": null,
    "redirected_to_name": null,
    "redirected_to_id": null,
    "orig_from_number": null,
    "orig_from_name": null,
    "orig_to_number": null,
    "orig_to_name": null,
    "agent_id": null,
    "agent_name": null,
    "acd_number": null,
    "acd_name": null,
    "acd_id": null,
    "broadworks_user_id": null,
    "broadworks_group_id": null,
    "broadworks_sp_id": null,
    "metaswitch_extension": null,
    "metaswitch_user": null,
    "metaswitch_group": null,
    "metaswitch_system": null,
    "cisco_nearend_guid": null,
    "cisco_farend_guid": null,
    "cisco_nearend_refci": null,
    "cisco_farend_refci": null,
    "cisco_nearend_partition": null,
    "cisco_farend_partition": null,
    "cisco_phone_ip": null,
    "participants": [
      {
        "participant_id": "00",
        "join_time": "2023-08-30T17:16:20-07:00",
        "leave_time": "2023-08-30T17:18:28-07:00",
        "user_id": "6e058dc0-48ed-11ee-a044-0015172d2a7d",
        "party_direction": 1,
        "party_type": 0,
        "party_number": "123000001",
        "party_name": null,
        "party_caller_id": null
      },
      {
        "participant_id": "01",
        "join_time": "2023-08-30T17:16:20-07:00",
        "leave_time": "2023-08-30T17:18:28-07:00",
        "user_id": null,
        "party_direction": 2,
        "party_type": 0,
        "party_number": "8001000001",
        "party_name": null,
        "party_caller_id": null
      }
    ],
    "categories": [
      {
        "category_id": "2b3982ec-8295-11e7-bda7-e03f497dbdff",
        "name": "Sales",
        "private_user_id": null,
        "parent_id": null
      },
      {
        "category_id": "be2917ee-ed3b-11e5-af83-e03f497dbdff",
        "name": "Evaluate",
        "private_user_id": null,
        "parent_id": null
      }
    ],
    "files": [
      {
        "file_id": "00",
        "start_time": "2023-08-30T17:16:20-07:00",
        "stop_time": "2023-08-30T17:18:28-07:00",
        "file_size": 2059200,
        "file_path": "/var/miarec/recordings/20230830/2023-08-30_171620__123000001__8001000001__20f5acd7bfbd58c39a81e453065fa4fe.mp3",
        "watermark": null,
        "encrypt_key": null,
        "encrypt_tag": null,
        "encrypt_fingerprint": null
      }
    ],
    "custom_fields": [
      {
        "field_id": "138fb20c-4c3b-11ee-8cbb-0015172d2a7d",
        "name": "Payment status",
        "value": "Unknown"
      },
      {
        "field_id": "9079d60a-5192-11ee-aa89-0015172d2a7d",
        "name": "Call type",
        "value": "Sales call"
      },
      {
        "field_id": "063e593e-5188-11ee-a855-0015172d2a7d",
        "name": "Call Summary",
        "value": "During the call, the agent, Samantha Smith,  introduces herself as calling from Acme's Store Support on behalf of Acme's. She confirms the
customer's name and informs her that the call is being recorded for quality purposes. The agent mentions that she noticed the customer's interest in a laptop
and informs her about the great deals available in-store and online, specifically mentioning a 50% off on selected bedroom items. The customer clarifies that
he is only interested in the laptop for his business. The agent offers to send a leasing application through text or email, which will determine how much the
customer can lease up to. The customer agrees to receive the application via text and confirms his mobile number. The agent assures the customer that he can
contact her for any questions or concerns, either by visiting the store or calling back. The call ends with the agent thanking the customer for choosing
Acme's and wishing him a great day. The customer expresses gratitude in return. "
      }
    ]
  }
}
```

3.1.6 Set or clear a custom field value

Request to update the values for call custom fields:

```
PUT  /api/v2/calls/<call-id>.json
```

The custom field can be identified by either its `field_id` or `name`.

Update custom field by ID

HTTP body should contain JSON formatted fields that must be updated.

For example:

```
{
    "call" {
        "custom_fields": [
            {
                "field_id": "138fb20c-4c3b-11ee-8cbb-0015172d2a7d",
                "value": "Unknown"
            },
            {
                "field_id": "9079d60a-5192-11ee-aa89-0015172d2a7d",
                "value": "Sales call"
            },
        ]
    }
}
```

Update custom field by ID

HTTP body should contain JSON formatted fields that must be updated.

For example:

```
{
    "call" {
        "custom_fields": [
            {
                "name": "Payment status",
                "value": "Unknown"
            },
            {
                "name": "Call type",
                "value": "Sales call"
            },
        ]
    }
}
```

If both `field_id` and `name` are supplied in the JSON body request, then `name` is ignored.

Prerequisites

1. REST API user must have permission Edit for the corresponding call record.
2. Custom field must be configured as Editable by administrator

How to clear a custom field value?

Note, this request will update only the custom fields that are listed in the JSON body. Any existing custom fields, that the call record has, but which are not listed in the PUT request, will not be affected.

To remove the value for custom fields, set it to null or empty string, like:

```
{
    "call" {
        "custom_fields": [
            {
                "field_id": "138fb20c-4c3b-11ee-8cbb-0015172d2a7d",
                "name": "Payment status",
                "value": null
            }
        ]
    }
}
```

Responses

Response contains HTTP status code as shown in the following list.

200 OK

Call record has been successfully updated.

A response contains a JSON formatted call's data after update.

403 Forbidden

The request cannot be completed because API user has no permission to update call records

400 Bad Request

The request cannot be completed because supplied JSON object has invalid data.

When response has content type `application/json` , then it contains more detailed description of error in JSON format like:

``` { "error": "InvalidRecord", "description": "Record Validation errors", "details": { "custom_fields.0.value": "Value is too long" } }

### 3.1.7 Delete one call

Request:

```
DELETE /api/v2/calls/<call-id>.json
```

Response contains HTTP status code as shown in the following table.

| Response | Description |
|---|---|
| **200 OK** | Call has been successfully deleted |
| **403 Forbidden** | The request cannot be completed because API user has no permission to delete calls |
```

3.1.8 Retrieve file for playback

The audio/video file could be delieved to end-user web browser using three different ways:

1) Your web appliation requests the signed file URL using MiaRec REST API, then it re-sends that URL to end-user web browser. The web-browser retrieves the file directly from MiaRec web portal using the signed file URL.

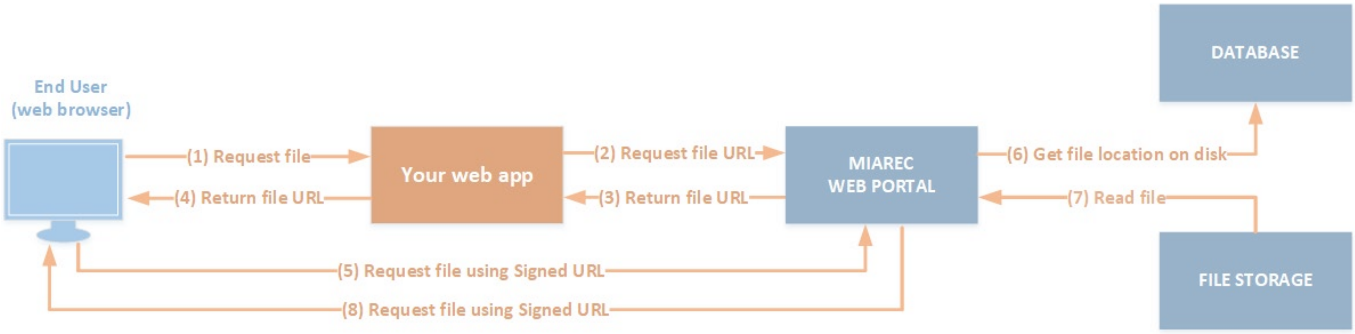
2) Your web application retrieves file using MiaRec REST API and resends the file back to end-user web browser.

3) Your web application retrieves call metadata using MiaRec REST API, then gets access to MiaRec file storage directly and streams the file to end-user web browser.

Method #1 (recommended)

Your web appliation requests the signed file URL using MiaRec REST API. The generated URL is returned to end-user web browser. The web-browser uses the signed URL to request the file directly from MiaRec web portal.

HOW IT WORKS



- **Step 1**. The end-user web browser connects to your web application and requests a playback of particular call. For example, you could render the following HTML web page to end-user:

```
<audio controls>
    <source src="https://YOUR-WEB-SERVER/recordings/CALL-ID" type="audio/mpeg">
</audio>
```

* **Step 2**. Your web appliation sends the "Generate File URL" request to MiaRec web server using REST API:

```
/api/v2/calls/{call-id}.json/file_url.json?expires=3600
```

The `expires` query parameter specifies for how long the URL should be valid (in seconds). The end-user will not be able to access the URL after expiration.

Optionally, you can pass `file_id` query parameter to retrieve the particular file instance of the call:

```
/api/v2/calls/{call-id}.json/file_url.json?expires=3600&file_id=01
```

The call may have multiple files in cases when automatic silence detection is enabled. In these cases, the recording is split on multiple files at silence periods. By default, silence detection is disabled. If the `file_id` is missing, then MiaRec web portal automatically concatenates multiple files into one on flight.

- **Step 3**. MiaRec web portal returns JSON-formatted response containing the secure signed URL, like:

```
{
   "signed_url":  "https://miarec.example.com/calls/file/e03f497d-bdff-11e7-2790-4b6ab9967d89/signed?expires=1493100525&sign=NMxBcIFB6t2M...<TRUNCATED>"
}
```

The URL is signed by MiaRec web server using encryption methods. MiaRec validates the signature when it receives such URL back from the end-user (see step 5). If signature is invalid, then the URL is rejected. The signature protects the important URL parameters from modification (call-id, expires, host name, etc.). I.e. this URL is valid only for particular call and only for a limited permiod of time (see `expires` parameter above). * **Step 4**. Your web application needs to return URL to the end-user web browser, for example, inside HTTP Location header, like:

```
HTTP/1.1 302 Found
Location: https://miarec.example.com/calls/file/e03f...<TRUNCATED>
```

Caution!

You may generate the signed URL in advance when generating HTML web page (not recommended), like:

```
<audio>
    <source src="https://miarec.example.com/calls/file/e03f...<TRUNCATED>" type="audio/mpeg">
</audio>
```

We do not recommend to do that because it creates unecessary load on MiaRec web server. In this example, URL has to be generated for each HTML page display even if the user doesn't playback the call. If you display 50 call recordings on page (in table, for example), then you need to request 50 Signed URLs in advance from MiaRec web server. It is time consuming because you need to send 50 HTTP requests to MiaRec.

Better solution is to route file requests to your own web application first and then redirect to MiaRec web server when necessary, like:

```
<audio controls>
    <source src="https://YOUR-WEB-SERVER/recording/CALL-ID" type="audio/mpeg">
</audio>
```

This URL should point to your web application. If end-user clicks "play" button in media player, then his/her web-browser automatically opens that URL. On your web application side you receive such request, parse "call-id", validate user's permissions and then ask MiaRec web portal to generate the signed URL for that particular call (see step 2). * **Step 5**. When web-browser receives **HTTP 302 Found** response with **Location** header, it automatically tries to open the returned URL (for user it is transparent). * **Step 6**. The MiaRec web portal verifies the signature and expiration parameters. If everyting is ok, then it connects to database and reads the file location from there. * **Step 7**. MiaRec web portal reads the file from the file storage. * **Step 8**. The file content is streamed directly to the end-user web browser.

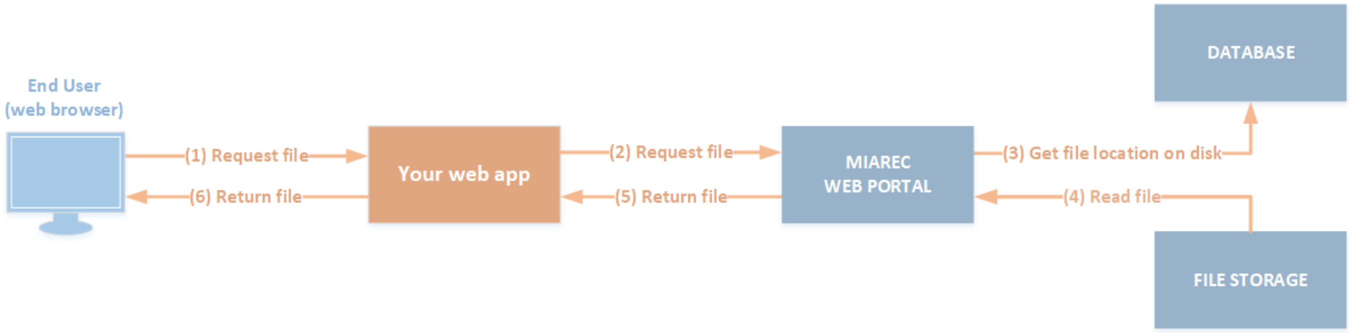
THINGS TO CONSIDER

- It is a responsibility of your web application to check user permissions. You should verify user's role/group and credentials before you return the signed URL to user. MiaRec web portal gracefully generate signed URLs for any call that is accessible by your REST API account. Although, on MiaRec side you could limit access rights for your REST API account, i.e. you could grant your web application permissions only to particular recordings identified by tenant/group/user.
- If both MiaRec web-portal and your web application are located in the same private network, then you could use HTTP (non-encrypted) protocol for REST API connection for optimization purposes. But for end-user to MiaRec web-portal communication, you need to use HTTPS (encrypted) protocol, i.e. you should deploy appropriate SSL certificate on MiaRec web server.

Method #2

Your web application retrieves file using MiaRec REST API. The your web application resends the file to end-user web browser.

HOW IT WORKS



- **Step 1**. The end-user web browser connects to your web application and requests playback of particular call.
- **Step 2**. Your web appliation sends the "Get file" request to MiaRec web server using REST API:

  ```
  /api/v2/calls/{call-id}.json/file
  ```

  Optionally, you can pass `file_id` query parameter to retrieve a particular file instance within the call:

  ```
  /api/v2/calls/{call-id}.json/file?file_id=01
  ```

  * **Step 3**. MiaRec web portal reads the file location from own database. * **Step 4**. MiaRec web portal reads the file from the file storage * **Step 5**. MiaRec web portal streams the file to your web application in a response to the request in step 2. Normally, at this moment, your web application should store temporary the file locally before it can pass that file further to end-user. * **Step 6**. Your web application streams the file to the web browser in a response to the request in step 1.

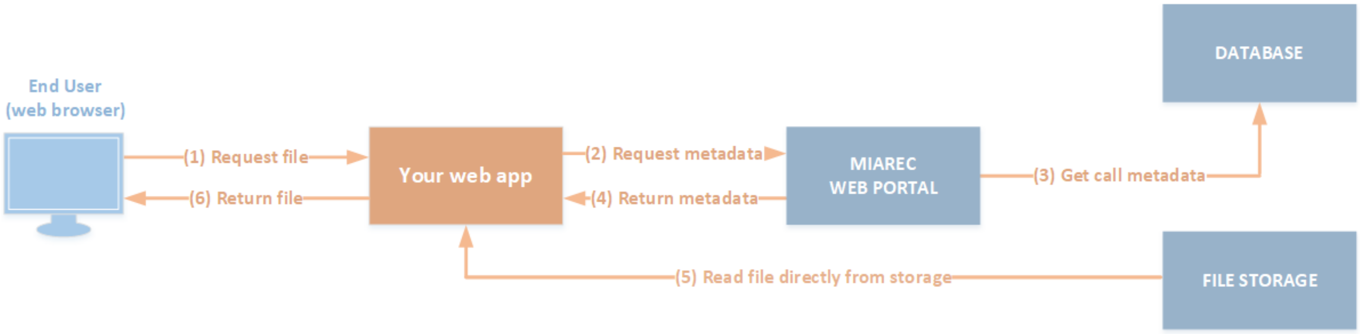
THINGS TO CONSIDER:

- The network connection between end-user and your web application may be significantly slower than the connection between your web app and MiaRec server(s). This means that the file streaming operation (step 6) may time out eventually, especially on big files. Your web application should support resumable file download using HTTP Range headers.
- If your web application and MiaRec web portal are located in different datacenters, then such solution incurs additional bandwidth usage. For each file, your web application has to download it first from MiaRec web portal (inbound traffic), then re-transfer it to the end-user (outbound traffic).

Method #3

Your web application retrieves call metadata using MiaRec REST API. The call metadata contains the file path. This file path is used to access the file directly on MiaRec file storage and stream it to end-user web browser.

HOW IT WORKS



- **Step 1**. The end-user web browser connects to your web application and requests playback of particular call.
- **Step 2**. Your web appliation sends the "Get Call Metadata" request to MiaRec web server using REST API:

  ```
  /api/v2/calls/{call-id}.json
  ```

  * **Step 3**. MiaRec web portal reads call metadata from own database. * **Step 4**. Your web application receives JSON-formatted response from MiaRec containing call metadata like:

```
{
"call": {
 "call_id":           "e03f497d-bdff-1019-102d-68be3896f081",
 "from_number":       "102",
 "to_number":         "101",
 ...
 "files": [{
    "file_id": "01",
    "start_time": "2013-10-12T16:32:22-0800",
    "stop_time": "2013-10-12T16:38:56-0800",
    "file_size": 132000,
    "file_path":  "/var/recordings/20131012/20131012163222-534346ad00000003.mp3",
 }]
}
```

The call metadata conteins the file location. In the example above, the `file_path` attribute is `/var/recordings/20131012/20131012163222-534346ad00000003.mp3` . * **Step 5**. Your web application needs to access the file directly on MiaRec storage.

Note, the `file_path` in JSON response points to the file location accessible from the MiaRec servers. In order to access these files from your own server, you could use NFS file sharing, i.e. mount the local directory `/var/recordings` to remote NFS share. Instructions of how to configure NFS is out of scope of this document. Alternatively, you could use SSHFS instead of NFS. * **Step 6**. Your web application streams the file to the web browser in a response to the request in step 1.

THINGS TO CONSIDER:

- You need to have physical access and appropriate permissions to the MiaRec file storage in order to enable NFS file sharing there.
- The NFS traffic is not encrypted. This means, that you could use NFS only within your private network only. You probably, could use VPN connection between your web app and MiaRec file storage, but this complicates the overall architecture. If a secure file transmission is required between your web application and MiaRec file server, then you should consider other methods described in this document.
- This solution doesn't works with encrypted files. MiaRec may optionally encrypt the files on storage. When you access file storage directly from your web application, then the resulting audio/video files are not playable (they are kept in encrypted format). Other methods do support on-flight decryption (see below).

ADVANTAGES:

- Resumable download is easier to achieve comparing to the previous method, because you can use your web server capabilities to serve files directly. It is a responsibility of the web-server to handle HTTP Range header, i.e. you do not need to worry about processing Range header.

Compatibility with file encryption

Methods 1 and 2 do support file encryption. The file is automatically decrypted on flight by MiaRec web portal during file streaming. The REST API account credentials are used to decrypt the file. This means that your REST API account has to be granted access to the appropriate file encryption key.

In method 3 (when your web application does access file storage directly), the read file remains encrypted (obviously). If you use this method, then you need to disable file encryption in MiaRec.

3.1.9 Pause and resume recording

Pause recording (mute):

```
POST  /api/v2/calls/<call-id>.json/muting?action=mute
```

Resume recording (unmute):

```
POST  /api/v2/calls/<call-id>.json/muting?action=unmute
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Call recording has been paused/resumed successfully
403 Forbidden	The request cannot be completed because API user has no permission to pause/resume recording. Verify role permissions.
404 Not Found	Call recording with such ID is not found.

3.2 Clients

3.2.1 Client object fields

Example of JSON representation:

```
{
    "client":
    {
        "name": "MiaRec",
        "client_id": "34cbea90-9201-11e5-a932-e03f497dbdff",
        "tenant_id": "34c7f1f6-9201-11e5-a739-e03f497dbdff",
        "contacts": [{
            "phone_number": "+14085800150",
            "name": "Main number"
        }, {
            "phone_number": "+14085800105",
            "name": "Fax"
        }]
    }
}
```

Field	Type	Description
name	string	Client name.
client_id	UUID	Unique ID of client assigned by MiaRec when client is created.
tenant_id	UUID	ID of parent tenant object. This field is available only when multi-tenancy is enabled in MiaRec.
contacts	list	A of contacts (phone numbers). See object `Contact` below. At least one contact is required.

CONTACT OBJECT FIELDS

Field	Type	Description
phone_number	string	Phone number associated with contact. MiaRec application uses phone number to associate call recordings to clients. Required.
name	string	Contact name, for example a name of person this phone number belongs to. Optional.

3.2.2 List and search clients

List all groups:

```
GET /api/v2/clients.json
```

Example of response:

```
{
    "next_url": null,
    "clients": [{
        “client_id”:          “b74e35b0-b6e3-11e8-b938-e03f497dbdff”,
        “tenant_id”:         “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:              “MiaRec”,
        "contacts": [{
            "phone_number": "+14085800150",
            "name": "Main number"
        }, {
            "phone_number": "+14085800105",
            "name": "Fax"
        }]
    },{
        “client_id”:          “ccb5957a-b6e5-11e8-b66b-e03f497dbdff”,
        “tenant_id”:         “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:              “Contoso”,
        "contacts": [{
            "phone_number": "1234567890",
            "name": null
        }]
    }]
}
```

See also Paging through collections

Search clients:

- **Search by client name, contact name, contact phone number and tenant name (in multi-tenant version)**

  ```
  GET /api/v2/clients.json?search_term=Contoso
  ```

- **Search by tenant id**

  ```
  GET  /api/v2/clients.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff
  ```

- **Search by multiple parameters (tenant_id + search_term)**

  ```
  GET  /api/v2/clients.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff&search_term=Contoso
  ```

3.2.3 View a client

Request to view a client:

```
GET  /api/v2/clients/<client-id>.json
```

Example of response:

```
{
    "client": {
        “client_id”:           “b74e35b0-b6e3-11e8-b938-e03f497dbdff”,
        “tenant_id”:          “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:               “MiaRec”,
        "contacts": [{
            "phone_number": "+14085800150",
            "name": "Main number"
        }, {
            "phone_number": "+14085800105",
            "name": "Fax"
        }]
    }
}
```

3.2.4 Create a client

Request to create new client:

```
POST /api/v2/clients.json
```

HTTP body should contain JSON formatted profile of client to create.

For example:

```
{
    "client": {
        “tenant_id”:          “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:               “Contoso”,
        "contacts": [{
            "phone_number": "1234567890",
            "name": null
        }]
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
201 Created	Client has been successfully created. HTTP header `Location` contains URL by which the newly created object should be know. For example: `HTTP/1.1 201 Created Location: /api/v2/clients/e011c408-f288-11e4-9b73-e03f497dbdff.json`
403 Forbidden	The request cannot be completed because API user has no permission to create clients
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "Client with such name exists already"]}`

3.2.5 Update a client

Request to update existing client:

```
PUT /api/v2/clients/<client-id>.json
```

HTTP body should contain JSON formatted profile of client to update.

For example:

```
{
    "client": {
        "name": "New Client Name"
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Client has been successfully updated. A response contains a JSON formatted client's data after update.
403 Forbidden	The request cannot be completed because API user has no permission to edit groups
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "Client with such name exists already"]}`

3.2.6 Delete a client

Request to delete a particular client:

```
DELETE  /api/v2/clients/<client-id>.json
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Client has been successfully deleted
403 Forbidden	The request cannot be completed because API user has no permission to delete clients

\

3.3 Custom fields

3.3.1 Custom field attributes

Example of JSON representation:

```
{
    "custom_field": {
      "field_id": "f07ca4b0-51b0-11ee-a424-0015172d2a7d",
      "enable": true,
      "tenant_id": "9079d60a-5192-11ee-aa89-0015172d2a7d",
      "name": "Call type",
      "is_global": false,
      "description": "Type of the call, like "Sales, Collection, etc.",
      "type": "option",
      "display_as": "default",
      "str_max_len": 64,
      "editable": true,
      "search_options": [
        "advanced"
      ],
      "options": [
        "Collection call",
        "Sales call"
      ],
      "ai_assist": false
    }
}
```

field_id (UUID, read-only)

Unique ID of custom field assigned by MiaRec application when a custom field is created. Read-only.

This field is returned in `GET` request only. It is ignored in `PUT` and `POST` requests.

tenant_id (UUID, optional)

ID of the parent tenant's object. This field is available only when all the following conditions are true:

- Multi-tenancy is enabled in the MiaRec application,
- The REST API user belongs to the System tenant,
- The REST API user has permissions to view the corresponding tenant, and
- The custom field is not global.

If REST API user is a tenant user, and such a user creates a new custom field, then user's tenant_id is implicitely used as the custom field's tenant_id.

If a custom field is global, then `tenant_id` is `null`.

name (string, required, maximum 64 characters)

Name of the custom field

description (string, optional, maximum 4096 characters)

Human-friendly description of the field

enable (bool, required)

A boolean flag to enable the custom field. If a custom field is disabled, end users will not be able to view, search or edit the custom fiel values.

is_global (bool, optional)

A boolean flag to make the custom field visible to all tenants. This flag is ignored unless the following conditions are true:

- Multi-tenancy is enabled in the MiaRec application,
- The REST API user belongs to System tenant, and
- The REST API user has access scope `Unrestricted` or `System`.

Tenant users cannot create global custom fields.

type (string, required)

A type of field, available options:

- `string` : A free-text value. In UI, users will be able to enter any text into this field.
- `date` : A date value, for example, shipping date. In UI, users will be able to choose a valid date for this field.
- `option` : A categorical value (see attribute `options` below). In UI, users must choose a value from a list of pre-defined options.

options (list of strings, optional)

A list of options for custom field of type `option`. The options must be pre-programmed by administrator ahead of time. Each option is a string up to 256 characters.

Example:

```
{
    "custom_field": {
        "name": "Call type",
        "type": "option",
        "options": [
            "Sales call",
            "Technical support call",
            "Voicemail",
        ]
    }
}
```

display_as (string, optional)

A display widget for the custom field.

Permitted values:

- `default` : Use a default display widget depending on the field type.
- `label` : Display as a label
- `multiline` : Display as a multi-line field. Normally used for long texts, for a call summary.

str_max_len (number, optional)

A maximum text length (up to 8096 characters). Applicable to field type `string` only.

editable (bool, optional)

Allow authorized users to edit values of this field via UI or REST API. Users must have permission `Edit` for the correspoding call record.

search_options (set of strings, optional)

Allow users to use this custom field in search of call records.

Allowed options (any combination):

- `advanced` : Users can use this custom field in Advanced Search of call records.
- `free_text` : Application searches in this field's values when users are searching a text in Basic Search input. Not recommended as it affects peformance of basic search.

Example:

```
{
    "custom_field": {
        "name": "Ordered product",
        "type": "string",
        "search_options": [
            "advanced",
            "free_text"
        ]
    }
}
```

ai_assist (bool, optional)

Allow administrators to populate values of this field with AI Assistant job. When enabled, such a field will be available for selection in AI Assist jobs (requires voice analytics license).

3.3.2 List and search clients

List all groups:

```
GET /api/v2/custom_fields.json
```

Example of response:

```
{
    {
      "tenant_id": null,
      "is_global": true,
      "name": "Call Summary",
      "description": null,
      "type": "string",
      "enable": true,
      "ai_assist": false,
      "display_as": "multiline",
      "str_max_len": 1024,
      "editable": true,
      "search_options": [
        "advanced"
      ],
      "field_id":  "063e593e-5188-11ee-a855-0015172d2a7d"
    },
    {
      "tenant_id": null,
      "is_global": true,
      "name": "Call type",
      "description": null,
      "type": "option",
      "enable": true,
      "ai_assist": true,
      "display_as": "default",
      "editable": true,
      "search_options": [
        "advanced"
      ],
      "options": [
        "Collection call",
        "Sales call"
      ],
      "field_id":  "9079d60a-5192-11ee-aa89-0015172d2a7d"
    },
    {
      "tenant_id": null,
      "is_global": true,
      "name": "Product ordered",
      "description": null,
      "type": "string",
      "enable": true,
      "ai_assist": true,
      "display_as": "default",
      "str_max_len": 64,
      "editable": true,
      "search_options": [
        "advanced"
      ],
      "field_id":  "2d020436-5189-11ee-a647-0015172d2a7d"
    },
    "total": 3,
    "next_url": null
}
```

See also Paging through collections

Search custom fields:

- **Search by custom field name and tenant name (in multi-tenant version)**

```
GET /api/v2/custom_fields.json?search_term=Contoso
```

- **Search by tenant id**

```
GET  /api/v2/custom_fields.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff
```

- **Search by multiple parameters (`tenant_id` and)**

```
GET  /api/v2/custom_fields.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff&search_term=Contoso
```

3.3.3 View a custom field

Request to view a custom field:

```
GET  /api/v2/custom_fields/<field-id>.json
```

Note

This API endpoint is used to administer custom fields configuration. To assign values to custom fields on call resource, use the `PUT /api/v2/calls.json` API endpoint.

Example of response:

```
{
    "custom_field": {
      "tenant_id": null,
      "is_global": true,
      "name": "Product ordered",
      "description": null,
      "type": "string",
      "enable": true,
      "ai_assist": true,
      "display_as": "default",
      "str_max_len": 64,
      "editable": true,
      "search_options": [
        "advanced"
      ],
      "field_id":  "2d020436-5189-11ee-a647-0015172d2a7d"
    }
}
```

3.3.4 Create a custom field

Request to create new custom field:

```
POST /api/v2/custom_fields.json
```

> **Note**
>
> This API endpoint is used to administer custom fields configuration. To assign values to custom fields on call resource, use the `PUT /api/v2/calls.json` API endpoint.

HTTP body should contain JSON formatted profile of custom field to create.

For example:

```
{
    "custom_field": {
      "enable": true,
      "tenant_id": "9079d60a-5192-11ee-aa89-0015172d2a7d",
      "name": "Call type",
      "is_global": false,
      "description": "Type of the call, like "Sales, Collection, etc.",
      "type": "option",
      "display_as": "default",
      "str_max_len": 64,
      "editable": true,
      "search_options": [
        "advanced"
      ],
      "options": [
        "Collection call",
        "Sales call"
      ],
      "ai_assist": false
    }
}
```

Response contains HTTP status code as shown in the following list.

`201 Created`

Custom field has been successfully created. HTTP header `Location` contains URL by which the newly created object should be know.

For example:

```
HTTP/1.1 201 Created
Location:  /api/v2/custom_fields/e011c408-f288-11e4-9b73-e03f497dbdff.json
```

`403 Forbidden`

The request cannot be completed because API user has no permission to create custom fields

`400 Bad Request`

The request cannot be completed because supplied JSON object has invalid data.

When response has content type `application/json` , then it contains more detailed description of error in JSON format like:

```
{
    "error": "InvalidRecord",
    "description": "Record Validation errors",
    "details": {
        "name": "Such name exists already"
    }
}
```

3.3.5 Update a custom field

Request to update existing custom field:

```
PUT  /api/v2/custom_fields/<field-id>.json
```

Note

This API endpoint is used to administer custom fields configuration. To assign values to custom fields on call resource, use the `PUT /api/v2/calls.json` API endpoint.

HTTP body should contain JSON formatted profile of custom field to update.

For example:

```
{
    "custom_field": {
        "name": "New name"
    }
}
```

Response contains HTTP status code as shown in the following list.

200 OK

Custom field has been successfully updated.

A response contains a JSON formatted client's data after update.

403 Forbidden

The request cannot be completed because API user has no permission to update custom fields

400 Bad Request

The request cannot be completed because supplied JSON object has invalid data.

When response has content type `application/json` , then it contains more detailed description of error in JSON format like:

``` { "error": "InvalidRecord", "description": "Record Validation errors", "details": { "name": "Such name exists already" } }
```

3.3.6 Delete a client

Request to delete a particular custom field:

```
DELETE  /api/v2/custom_fields/<field-id>.json
```

> **Note**
>
> This API endpoint is used to administer custom fields configuration. To assign values to custom fields on call resource, use the `PUT /api/v2/calls.json` API endpoint.

Response contains HTTP status code as shown in the following list.

200 OK

Client has been successfully deleted

403 Forbidden

The request cannot be completed because API user has no permission to delete custom fields.

3.4 Encryption keys

3.4.1 List and search encryption keys

List all encryption keys:

```
GET /api/v2/encrypt_keys.json
```

See also Paging through collections

Search users:

- **Search by key name**

```
GET /api/v2/encrypt_keys.json?search_text=SomeKeyName
```

- **Search by tenant id**

```
GET  /api/v2/encrypt_keys.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff
```

- **Search by multiple parameters (tenant_id + search_term)**

```
GET  /api/v2/encrypt_keys.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff&search_text=SomeKeyName
```

3.4.2 View encrypt key

Request:

```
GET  /api/v2/encrypt_keys/<encrypt-key-id>.json
```

Example response:

```
{
  "encrypt_key": {
    "key_id": "e9780258-f061-11e5-b9d7-e03f497dbdff",
    "is_active": false,
    "fingerprint": "cb874f07fe39656cf88150e569b5b9e8",
    "tenant_id": "11111111-1111-1111-1111-111111111111",
    "public_key": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNA...",
    "name": "Encrypt key new"
  }
}
```

3.4.3 Create encrypt key (generate or import)

Generate new encryption key

To automatically generate new encryption key, submit the following POST request with JSON-formatted data.

```
POST /api/v2/encrypt_keys.json
```

HTTP body should contain JSON-formatted data with the following parameters:

Field	Type	Description
name	string	Human-readable encrypt key name.
tenant_id	UUID	ID of tenant, for which the encrypt key will be created. This field is ignored when multi-tenancy is disabled in MiaRec.
protection_mode	string	This parameter specified whether the key is protected with user's credentials or application credentials. When a key is protected with user's credentials, it is necessary to explicitly grant users access to this key. App-protected mode is required when SAML 2.0 Single Sign-On or speech analytics is used. Supported values: • **user** - protect the key with user's credentials (default) • **app** - protected the key with application credentials
add_type	string	This parameter specified whether the key is generated or imported. Supported values: • **generate** - generate new random key • **import** - import existing key
is_active	boolean	If **true**, then the new key will be used for encrypting of all on-going recordings for that tenant. If **false**, then the key will be used only for accessing previously encrypted recordings with that key.
key_length	integer	Length of encryption key in bits. Supported values: • **1024** • **2048** (default, recommended) • **4096**

Example of JSON data to submit:

```
{
    "encrypt_key":
    {
        "name": "New encrypt key"
        "is_active": true,
        "add_type": "generate",
        "key_lenght": 2048,
    }
}
```

Import encryption key

To import existing encryption key, submit the following POST request with JSON-formatted data.

```
POST /api/v2/encrypt_keys.json
```

HTTP body should contain JSON-formatted data with the following parameters:

Field	Type	Description
name	string	Human-readable encrypt key name.
tenant_id	UUID	ID of tenant, for which the encrypt key will be created. This field is ignored when multi-tenancy is disabled in MiaRec.
add_type	string	This parameter specified whether the key is generated or imported. Supported values: • **generate** - generate new random key • **import** - import existing key
is_active	boolean	If **true**, then the new key will be used for encrypting of all on-going recordings for that tenant. If **false**, then the key will be used only for accessing previously encrypted recordings with that key.
public_key	string	RSA public key formatted in Base64 encoding (PEM format).
private_key	string	RSA private key formatted in Base64 encoding (PEM format). This parameter is optional. If you do not provide private key, then the imported encryption key will be used only for encryption of audio files without ability to decrypt them. Users will not be able to decrypt these recordings on that server. To playback such recordings, you will need to transfer these recordings to another MiaRec server, which has the corresponding private key. This is an advanced feature of MiaRec - it allows to deploy a recording server in one location and a playback server in another location. For example, the hosted service provider may record customer calls directly into encrypted format and nobody on service provider site will be able to playback those recordings, including root administrators. Data should be uploaded to customer premises, where only authorized persons will be able to playback them.
private_key_password	string	Password for decrypting private key, if the latter has been exported previously with password protection.

Example of JSON data to submit:

```
{
    "encrypt_key":
    {
        "name": "New encrypt key"
        "is_active": true,
        "add_type": "import",
        "public_key": "MIGfMA0GCSqGSIb3DQEBAQ...",
        "private_key": "RheQwd3Y6cdLyH4MFMxN61K6K/lyoyB...",
        "private_key_password": "secret"
    }
}
```

Response values

Response contains HTTP status code as shown in the following table.

Response	Description
201 Created	Encrypt key record has been successfully created. HTTP header `Location` contains URL by which the newly created object should be know. For example: `HTTP/1.1 201 Created Location: /api/v2/encrypt_keys/e011c408-f288-11e4-9b73-e03f497dbdff.json`
403 Forbidden	The request cannot be completed because API user has no permission to create encrypt keys
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["key_length": "ke_lenght should one of 1024, 2048 or 4086"]}`

3.4.4 Update encrypt key

To modify the encryption key, submit the following PUT request with JSON-formatted data.

```
PUT  /api/v2/encrypt_keys/<encrypt-key-id>.json
```

HTTP body should contain JSON-formatted data with the following parameters:

Field	Type	Description
name	string	Human-readable encrypt key name.
is_active	boolean	If **true**, then the new key will be used for encrypting of all on-going recordings for that tenant. If **false**, then the key will be used only for accessing previously encrypted recordings with that key.

Example of JSON data to submit:

```
{
    "encrypt_key":
    {
        "name": "New name for encrypt key"
        "is_active": false,
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Encryption key has been successfully updated
403 Forbidden	The request cannot be completed because API user has no permission to edit encryption keys
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord", "description": "Record Validation errors","details": ["is_active": "Should be boolean type"]}`

3.4.5 Delete encrypt key

Request to delete particular encryption key:

```
DELETE  /api/v2/encrypt_keys/<encrypt-key-id>.json
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Encryption key has been successfully deleted
403 Forbidden	The request cannot be completed because API user has no permission to delete encryption keys

3.4.6 Grant access to encrypt key

To grant access to encryption key, submit the following post request:

```
POST  /api/v2/encrypt_keys/<encrypt-key-id>/authorized_users.json
```

HTTP body should contain JSON-formatted user-id.

For example:

```
{
    "user": {
        "user_id":  "5b139cee-f13a-11e5-9615-e03f497dbdff"
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
201 Created	User has been successfully granted access to encryption key. HTTP header `Location` contains URL by which the newly created object should be know. For example: `HTTP/1.1 201 Created Location: /api/v2/encrypt_keys/.../5b139cee-f13a-11e5-9615-e03f497dbdff.json`
403 Forbidden	The request cannot be completed because API user has no permission to grant access to encryption keys
409 Conflict	The request cannot be completed due to error. When response has content type `application/json` , then it contains more detailed description of error in JSON format like: `{"error": "InvalidState","description": "Such user is authorized already"}`

3.4.7 Revoke access to encrypt key

To revoke access to encryption key, submit the following DELETE request:

```
DELETE  /api/v2/encrypt_keys/<encrypt-key-id>/authorized_users/<user-id>.json
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Access has has been successfully revoked
403 Forbidden	The request cannot be completed because API user has no permission to revoke access to encryption key

3.4.8 List authorized users

To list all authorized users for particular encryption key, submit the following GET request:

```
GET  /api/v2/encrypt_keys/<encrypt-key-id>/authorized_users.json
```

Response message will contain a list of user-ids, which are authorized to access that encryption key. Authorized users may access audio recordings, which were encrypted with that key.

Example of response:

```
{
  "next_url": null,
  "total": 2,
  "users": [
    {
      "user_id":  "2c4b823a-8ce4-11e5-93b6-e03f497dbdff"
    },
    {
      "user_id":  "5a5b525a-7ce5-11e5-83b6-a66f445dbdea"
    }
  ]
}
```

3.5 Groups

3.5.1 Group object fields

Example of JSON representation:

```
{
    "group":
    {
        "tenant_id": "34c7f1f6-9201-11e5-a739-e03f497dbdff",
        "name": "Administrators",
        "timezone": null,
        "group_id":  "34cbea90-9201-11e5-a932-e03f497dbdff"
    }
}
```

Field	Type	Description
name	string	Group name.
group_id	UUID	Unique ID of group assigned by MiaRec when group is created.
tenant_id	UUID	ID of parent tenant object. This field is available only when multi-tenancy is enabled in MiaRec.
timezone	string	Timezone setting for all users within this group. This value is used for displaying date/ time values to users. If it is not specified, then a timezone of parent tenant is used. If timezone is not configured on tenant level, then default system timezone is used.

3.5.2 List and search groups

List all groups:

```
GET /api/v2/groups.json
```

Example of response:

```
{
    "next_url": null,
    "groups": [{
        “group_id”:            “546340bf-8b47-21e4-95a4-e03f497dbd55”,
        “tenant_id”:           “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:                “Sales Department”
    },{
        “group_id”:            “e011c408-f288-11e4-9b73-e03f497dbdff”,
        “tenant_id”:           “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:                “Supervisors”
    }]
}
```

See also Paging through collections

Search groups:

- **Search by group name**

  ```
  GET /api/v2/groups.json?search_term=supervisors
  ```

- **Search by tenant id**

  ```
  GET  /api/v2/groups.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff
  ```

- **Search by multiple parameters (tenant_id + search_term)**

  ```
  GET  /api/v2/groups.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff&search_term=supervisors
  ```

3.5.3 View one group

Request to view one group:

```
GET  /api/v2/groups/<group-id>.json
```

Example of response:

```
{
    "group": {
        “group_id”:          “546340bf-8b47-21e4-95a4-e03f497dbd55”,
        “tenant_id”:         “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:              “Sales Department”
    }
}
```

3.5.4 Create group

Request to create new group:

```
POST /api/v2/groups.json
```

HTTP body should contain JSON formatted profile of group to create.

For example:

```
{
    "group": {
        "name": "Group 1",
        "tenant_id":  "e00a4822-f288-11e4-b559-e03f497dbdff",
         ...
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
201 Created	Group has been successfully created. HTTP header `Location` contains URL by which the newly created object should be know. For example: `HTTP/1.1 201 Created Location: /api/v2/groups/e011c408-f288-11e4-9b73-e03f497dbdff.json`
403 Forbidden	The request cannot be completed because API user has no permission to create groups
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "Group with such name exists already"]}`

3.5.5 Update group

Request to update existing group:

```
PUT /api/v2/groups/<group-id>.json
```

HTTP body should contain JSON formatted profile of group to update.

For example:

```
{
    "group": {
        "name": "New Group Name"
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Group has been successfully updated. A response contains a JSON formatted group's data after update.
403 Forbidden	The request cannot be completed because API user has no permission to edit groups
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "Group with such name exists already"]}`

3.5.6 Delete group

Request to delete particular group:

```
DELETE /api/v2/groups/<group-id>.json
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Group has been successfully deleted
403 Forbidden	The request cannot be completed because API user has no permission to delete groups

3.6 Roles

3.6.1 Role object fields

Example of JSON-formatted role instance:

```
{
    "role":
    {
        "role_id": "34c88e5c-9201-11e5-92fa-e03f497dbdff",
        "name": "Administrator",
        "access_level": "system",
        "tenant_id": "34c7f1f6-9201-11e5-a739-e03f497dbdff"
        "permissions":
        {
            "roles": ["view"],
            "users": ["view", "edit", "delete"],
            "groups": ["view"],
            "calls": ["view", "live_monitor", "categorize", "playback", "add_notes"],
            "call_categories": ["view", "edit", "delete"],
            "call_notes": ["view", "pin"]
        }



    }
}
```

Field	Type	Description
name	string	Role name.
role_id	UUID	Unique ID of role assigned by MiaRec when role is created.
tenant_id	UUID	ID of parent tenant object. This field is available only when multi-tenancy is enabled in MiaRec.
access_level	string	Access level setting specifies which resources are accessible by user of such role. Supported values:

		• **root** - User with such role has unrestricted access to the system. • **system** - User with such role has access to all resources on the system (users, groups, calls), but the operations are restricted by permissions. One exception from this rule is when multi-tenancy is enabled and user belongs to particular tenant account. In this case access is limited to tenant resources only. • **managed_groups** - User with such role has access only to resources within the managed groups. A list of managed groups is configured in user's profile. The group manager may see only users and their calls, for which he/she is a manager. Other users/calls are not visible to group manager. • **user** - User with such role has access only to own call recordings.

permissions	dictionary	Permissions setting specifies what operations are permitted on the accessible resources. These operations include view, edit, delete, playback etc.
		Format:
		`RESOURCE_NAME: [PERMISSION_1, PERMISSION_2, ...]`
		Example:
		`"permissions": {"groups": ["view"],"users": ["view", "edit", "delete"],}`
		In above example, user has **read-only** (view) access to **groups** recources and full access rights (view, edit and delete) to **users** resources.
		MiaRec supports very granular configuration of role permissions. A list of resources and supported permissions may be extended in each new software release. We recommend to use web portal to create a reference role with all checkboxes checked and then retrieve such role via REST API. In the response message you will be able to see a list of all supported resources and permissions on your version.

3.6.2 List and search roles

List all roles:

```
GET /api/v2/roles.json
```

Example of response:

```
{
    "next_url": null,
    "roles": [{
        "role_id": "546340bf-8b47-21e4-95a4-e03f497dbd55",
        "tenant_id": "156340bf-8b47-21e4-95a4-e03f497dbd44",
        "name": "Administrator Role",
        "permissions": {
            "system": ["view", "edit"],
            "users": ["view", "edit", "delete"],
            "calls": ["view", "categorize", "on_demand_trigger", "delete"],
            ...
        }
    },{
        "role_id": "e011c408-f288-11e4-9b73-e03f497dbdff",
        "tenant_id": "156340bf-8b47-21e4-95a4-e03f497dbd44",
        "name": "Agent Role",
        "permissions": {
            "system": [],
            "users": ["view"],
            "calls": ["view", "categorize", "on_demand_trigger"],
            ...
        }
    }]
}
```

See also Paging through collections

Search roles:

- **Search by role name**

```
GET /api/v2/roles.json?search_term=Agent
```

- **Search by tenant id**

```
GET  /api/v2/roles.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff
```

- **Search by multiple parameters (tenant_id + search_term)**

```
GET  /api/v2/roles.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff&search_term=Agent
```

3.6.3 View one role

Request to view one role:

```
GET  /api/v2/roles/<role-id>.json
```

Example of response:

```
{
    "role": {
        "role_id": "546340bf-8b47-21e4-95a4-e03f497dbd55",
        "tenant_id": "156340bf-8b47-21e4-95a4-e03f497dbd44",
        "name": "Supervisor Role",
        "access_level": "managed_groups",
        "permissions": {
            "system": [],
            "users": ["view"],
            "calls": ["view", "categorize", "on_demand_trigger"],
            ...
        }
    }
}
```

3.6.4 Create role

Request to create new role:

```
POST /api/v2/roles.json
```

HTTP body should contain JSON formatted profile of role to create.

For example:

```
{
    "role": {
        "name": "Supervisor Role",
        "tenant_id": "e00a4822-f288-11e4-b559-e03f497dbdff",
        "permissions": {
            "system": [],
            "users": ["view"],
            "calls": ["view", "categorize", "on_demand_trigger"],
        }
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
201 Created	Role has been successfully created. HTTP header `Location` contains URL by which the newly created object should be know. For example: `HTTP/1.1 201 Created Location: /api/v2/roles/e011c408-f288-11e4-9b73-e03f497dbdff.json`
403 Forbidden	The request cannot be completed because API user has no permission to create roles
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "Role with such name exists already"]}`

3.6.5 Update role

Request to update existing role:

```
PUT /api/v2/roles/<role-id>.json
```

HTTP body should contain JSON formatted profile of role to update.

For example:

```
{
    "role": {
        "name": "Supervisor Role",
        "tenant_id": "e00a4822-f288-11e4-b559-e03f497dbdff",
        "permissions": {
            "system": [],
            "users": ["view"],
            "calls": ["view", "categorize", "on_demand_trigger"],
        }
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Role has been successfully updated
403 Forbidden	The request cannot be completed because API user has no permission to edit roles
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "Role with such name exists already"]}`

3.6.6 Delete role

Request to delete particular role:

```
DELETE  /api/v2/roles/<role-id>.json
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Role has been successfully deleted
403 Forbidden	The request cannot be completed because API user has no permission to delete roles

3.7 Tenants

3.7.1 Tenant object fields

Example of JSON-formatted tenant instance:

```
{
    "tenant":
    {
        "tenant_id": "34c7f1f6-9201-11e5-a739-e03f497dbdff",
        "name": "Flexus",
        "timezone": null,
        "encrypt_data": null,
        "fieldset_licensing":
        {
            "recording_seats": null,
            "screen_recording_seats": null,
            "monitoring_seats": null,
            "evaluation_seats": 50,
            "speech_analytics": 10
        },
        "fieldset_associate_calls":
        {
            "extension_uniqueness": "systemwide",

            "associate_calls": "broadworks_group",
            "associate_calls_condition": "BroadWorks-serviceProviderID = \"FlexusProvider\" AND BroadWorks-groupID = \"FlexusGroup\"",
            "broadworks_sp_id": "FlexusProvider",
            "broadworks_group_id": "FlexusGroup",
            "sip_uri_host": "",
            "cisco_partition": "",
            "metaswitch_system": "",
            "metaswitch_group": "",

            "record_unknown_users": "record",

            "auto_provision": true,
            "provision_role_id": "34cafffe-9201-11e5-b516-e03f497dbdff",
            "provision_group_id": "47f53fcc-9201-11e5-b4ef-e03f497dbdff",
        }
    }
}
```

Field	Type	Description
name	string	Tenant name.
tenant_id	UUID	Unique ID of tenant assigned by MiaRec when tenant is created.
timezone	string	Default timezone setting for all users within this tenant. This value is used for displaying date/time values to users. If not specified, then system default timezone is used. Note, the timezone setting may be overriden on group and/or user level.
encrypt_data	boolean	If **true**, then audio files for this tenant will be encrypted using tenant's unique public key.
fieldset_licensing	dictionary	A group of licensing-related settings. See below.
fieldset_associate_calls	dictionary	A group of auto-provision settings. See below.

Fieldset "fieldset_licensing"

This section configures a license allocation for tenant.

For any individual license type, it is possible to explicitly pre-allocate a particular number of licenses to tenant (starting from 0). Or set to `null` to use a dynamic license allocation, when tenant is being allocated as many licenses are it has users requiring such license. Licenses are allocated from a global license pool.

Field	Type	Description
recording_seats	integer	Numeric value specifying how many licenses are allocated to that tenant or `null` .
recording_sessions	integer	Numeric value specifying how many licenses are allocated to that tenant or `null` .
monitoring_seats	integer	Numeric value specifying how many licenses are allocated to that tenant or `null` .
evaluation_seats	integer	Numeric value specifying how many licenses are allocated to that tenant or `null` .

Fieldset "fieldset_associate_calls"

Field	Type	Description
extension_uniqueness	string	This setting specifies if extensions within tenant should be system-wide unique or not. Supported values: • **systemwide** - Extensions have to be unique system-wide • **tenant** - Extensions have to be unique within tenant account only. This means that the same extension may appear in multiple tenants.
extension_uniqueness	string	This setting specifies if extensions within tenant should be system-wide unique or not.
associate_calls	string	This setting specifies how to distinguish call recordings of one tenant from another. Such setting is especially useful when different tenant may have ovelapping extensions. Supported values: • **extension** - Associate calls with tenant if they match to user's extension • **broadworks_group** - Associate calls with tenant if they match to particular Broadworks Service Provider ID and Group ID (see fields **broadworks_sp_id** and **broadworks_group_id**) • **metaswitch_group** - Associate calls with tenant if they match to particular Metaswitch System Name and Group Name (see fields **metaswitch_system** and **metaswitch_group**) • **cisco_partition** - Associate calls with tenant if they match to particular Cisco Partition (see field **cisco_partition**) • **sip_uri_host** - Associate calls with tenant if SIP URI host part matches to particular value (see field **sip_uri_host**) • **condition** - Associate calls with tenant if they match to custom condition (see field **associate_calls_condition**).
associate_calls_condition	string	Recording filter condition. Format of this field is described here, but in 99% you can do the following: login to MiaRec web-portal and create a reference tenant account with paricular values. When you save it, the condition field will be populated automatically. Then you can rely on the same format when creating tenant accounts via REST API. Note, this field is ignored when **associate_calls** is equal to **extension**
broadworks_sp_id	string	Broadworks Service Provider ID. Note, this field is ignored when **associate_calls** is not equal to **broadworks_group**
broadworks_group_id	string	Broadworks Group ID. Note, this field is ignored when **associate_calls** is not equal to **broadworks_group**
metaswitch_system	string	Metaswitch System Name. Note, this field is ignored when **associate_calls** is not equal to **metaswitch_group**
metaswitch_group	string	Metaswitch Group Name. Note, this field is ignored when **associate_calls** is not equal to **metaswitch_group**
cisco_partition	string	Cisco Partition name.

Field	Type	Description
		Note, this field is ignored when **associate_calls** is not equal to **cisco_partition**
sip_uri_host	string	SIP URI host part. Note, this field is ignored when **associate_calls** is not equal to **sip_uri_host**
record_unknown_users	string	This setting specifies default rule for call recordings, which match to this tenant, but do not match to any of pre-configured users. Supported values: • **record** - Record such calls and store them within tenant account. Tenant administrators may access these recordings on "Not assigned to users" page. Such recordings may be associated to users at later time. See also field **auto_provision** below. • **ignore** - Ignore such calls
auto_provision	string	This setting specifies if user auto-provisioning is enabled for this tenant. Note, auto-provisioning is activated only when **record_unknown_users** setting is equal to **record**. **How it works:** When call for unknown user is detected and auto-provisioning is enabled, then MiaRec creates automatically new user account inside the specified group (see field **provision_group_id**) and with specified role (see field **provision_role_id**).
provision_group_id	UUID	ID of group, inside which an auto-provisioned user account will be created.
provision_role_id	UUID	ID of role, which will be assigned to newly created users.

3.7.2 List and search tenants

List all tenants:

```
GET /api/v2/tenants.json
```

Example of response:

```
{
    "next_url": null,
    "tenants": [{
        “tenant_id”:          “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:               “Acme”
    },{
        “tenant_id”:          “156340bf-8b47-21e4-95a4-e03f497dbd44”,
        “name”:               “Flexus”
    }]
}
```

See also Paging through collections

Search tenants:

- **Search by tenant name**

  ```
  GET /api/v2/tenants.json?search_term=acme
  ```

3.7.3 Create tenant

Request to create new tenant:

```
POST /api/v2/tenants.json
```

HTTP body should contain JSON formatted profile of tenant to create.

For example:

```
{
    "tenant": {
        "name": "MaxiServe",
        "timezone": "Europe/London"
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
201 Created	Tenant has been successfully created. HTTP header `Location` contains URL by which the newly created object should be know. For example: `HTTP/1.1 201 Created Location: /api/v2/tenants/e011c408-f288-11e4-9b73-e03f497dbdff.json`
403 Forbidden	The request cannot be completed because API user has no permission to create tenants
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "Tenant with such name exists already"]}`

3.7.4 Update tenant

Request to update existing tenant:

```
PUT  /api/v2/tenants/<tenant-id>.json
```

HTTP body should contain JSON formatted profile of tenant to update.

For example:

```
{
    "tenant": {
        "name": "New Tenant Name"
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Tenant has been successfully updated
403 Forbidden	The request cannot be completed because API user has no permission to edit tenants
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "Tenant with such name exists already"]}`

3.7.5 Delete tenant

Request to delete particular tenant:

```
DELETE  /api/v2/tenant/<tenant-id>.json
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	Tenant has been successfully deleted
403 Forbidden	The request cannot be completed because API user has no permission to delete tenants

3.7.6 View one tenant

Request to view one tenant:

```
GET  /api/v2/tenants/<tenant-id>.json
```

Example of response:

```
{
    "tenant": {
        "tenant_id": "156340bf-8b47-21e4-95a4-e03f497dbd44",
        "name": "Acme",
        "timezone": "Europe/London"
    }
}
```

3.8 Users

3.8.1 User object fields

Example of JSON-formatted user instance:

```
{
    "user":
    {
        "user_id": "61e6e6b0-f147-11e5-b8b3-e03f497dbdff",
        "name": "John Smith",
        "group_id": "47f53fcc-9201-11e5-b4ef-e03f497dbdff",
        "role_id": "5b139cee-f13a-11e5-9615-e03f497dbdff",
        "is_active": true,
        "email": "",
        "timezone": "",
        "managed_groups": [
            "34cbea90-9201-11e5-a932-e03f497dbdff",
            "34e1c1ee-9201-11e5-96a0-e03f497dbdff"
        ],
        "fieldset_login":
        {
            "can_login": true,
            "login": "j.smith",
            "password": "secret",
            "authenticate_type": "password",
            "must_change_password": false,
            "valid_till": "2016-10-01T00:00:00-07:00"
        },
        "fieldset_recording":
        {
            "record": "always",
            "extensions": ["12444003002"],
            "confidential": false,
            "record_direction": ["in", "out"],
            "on_demand_default": null
        },
        "fieldset_licensing":
        {
            "evaluation_seat": false,
            "monitoring_seat": true,
            "recording_seat": true
        }
    }
}
```

Fields

Field	Type	Description
name	string	User name.
user_id	UUID	Unique ID of user assigned by MiaRec when user is created.
role_id	UUID	ID of role.
group_id	UUID	ID of parent group object.
email	string	Email address.
timezone	string	Timezone setting for this user. This value is used for displaying date/time values in web portal. If it is not specified, then a timezone of parent group is used.
is_active	boolean	If **false**, then user record is disabled. Login is not permitted to that user and recording settings are ignored.
fieldset_login	dictionary	A group of login-related settings. See below.
fieldset_recording	dictionary	A group of recording-related settings. See below.
fieldset_licensing	dictionary	A group of licensing-related settings. See below.

Fieldset "fieldset_recording"

Field	Type	Description
extensions	list	A list of extensions (string values) associated to this user. Extensions are used to match call recordings to users. This list may contain more than one extension, for example, when user has multiple lines or when phone system may send user's phone number in different formats depending on call direction, like 123456789 and +123456789.
record	string	Recording rule for this user. Supported values: • **always** - Always record calls of this user • **ondemand** - User may swich on/off recording during a call (on-demand recording) • **never** - Disable recording of this user • **default** - Use default recording rule as configured on system level.
record_direction	list	Direction of call. Supported values: • **in** - Inbound call • **out** - Outbound call In order to record both inbound and outbound calls, you need to specify both of them in a list: `["in", "out"]`
on_demand_default	string	When on-demand recording is enabled for this user, then this setting allows to specify what action should be applied to recording when user doesn't take any decision during a call. Supported values: • **keep** - Keep call recording by default • **discard** - Discard call recording upon call completion.
confidential	boolean	Boolean value marking all calls of this user as confidential.

FIELDSET "FIELDSET_LOGIN"

Field	Type	Description
can_login	boolean	This setting specifies whether user has rights to login to MiaRec web portal.
login	string	Web login name used to access MiaRec web portal.
password	string	Password for accessing MiaRec web portal. Note 1: This field is ignored when LDAP authentication is used. Note 2: This field is not available when you retrieve user records from MiaRec database. You may optionally set this field when you create or edit user. In the later case, the password will be overwritten during record update.
authenticate_type	string	Authentication type: • **ldap** - User's login credentials (login/password) are verified on LDAP server • **password** - User's login credentials (login/password) are verified against values stored in MiaRec database.
mush_change_password	boolean	If **true** then user will be asked to change own password on next login.
valid_till	datetime	If specified, then user cannot access MiaRec web portal after that date.

Fieldset "fieldset_licensing"

Field	Type	Description
recording_seat	boolean	If **true**, then recording seat license is allocated to user.
monitoring_seat	boolean	If **true**, then live monitoring seat license is allocated to user.
evaluation_seat	boolean	If **true**, then agent evaluation seat license is allocated to user.

3.8.2 List and search users

List all users:

```
GET /api/v2/users.json
```

See also Paging through collections

Search users:

- **Search by user name, group name, tenant name, login, role or extension (partial match, case insensitive)**

  ```
  GET /api/v2/users.json?search_term=john
  ```

- **Search by name (partial match, case insensitive)**

  ```
  GET /api/v2/users.json?name=John%20Smith
  ```

- **Search by login (exact match)**

  ```
  GET /api/v2/users.json?login=123456
  ```

- **Search by extension**

  ```
  GET /api/v2/users.json?extension=123456
  ```

- **Search by group id**

  ```
  GET  /api/v2/users.json?group_id=d1d83c40-eec7-11e4-8558-e03f497dbdff
  ```

- **Search by tenant id**

  ```
  GET  /api/v2/users.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff
  ```

- **Search by multiple parameters (tenant_id + search_term)**

  ```
  GET  /api/v2/users.json?tenant_id=2bfcefd4-f41d-11e4-983d-e03f497dbdff&search_term=supervisors
  ```

3.8.3 View one user

Request:

```
GET  /api/v2/users/<user-id>.json
```

Example response:

```
{
    "user": {
        "name": "John Smith",
        "group_id":  "e011c408-f288-11e4-9b73-e03f497dbdff",
        ...
    }
}
```

3.8.4 Create user

Request to create new user:

```
POST /api/v2/users.json
```

HTTP body should contain JSON formatted profile of user to create.

For example:

```
{
    "user": {
        "name": "John Smith",
        "role_id": "5b139cee-f13a-11e5-9615-e03f497dbdff",
        "group_id": "47f53fcc-9201-11e5-b4ef-e03f497dbdff",
        "timezone": "Europe/London",
        "fieldset_recording":  {
            "extensions": ["2001", "2002"],
            "record": "always",
            "record_direction": ["in", "out"]
        },
        "fieldset_licensing":
        {
            "recording_seat":  true,
        }
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	**Description**
201 Created	User record has been successfully created. HTTP header `Location` contains URL by which the newly created object should be know. For example: `HTTP/1.1 201 Created Location: /api/v2/users/e011c408-f288-11e4-9b73-e03f497dbdff.json`
403 Forbidden	The request cannot be completed because API user has no permission to create users
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "User with such name exists already"]}`

3.8.5 Update user

Request to update existing user:

```
PUT /api/v2/users/<user-id>.json
```

HTTP body should contain JSON formatted profile of user to update.

For example:

```
{
    "user": {
        "name": "New User Name"
    }
}
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	User has been successfully updated
403 Forbidden	The request cannot be completed because API user has no permission to edit users
400 Bad Request	The request cannot be completed because supplied JSON object has invalid data. When response has content type `application/json`, then it contains more detailed description of error in JSON format like: `{"error": "InvalidRecord","description": "Record Validation errors","details": ["name": "User with such name exists already"]}`

3.8.6 Delete user

Request to delete particular user:

```
DELETE  /api/v2/users/<user-id>.json
```

Response contains HTTP status code as shown in the following table.

Response	Description
200 OK	User has been successfully deleted
403 Forbidden	The request cannot be completed because API user has no permission to delete users

3.8.7 Pause and resume recording for user

Pause recording (mute) for the current active call for user:

```
POST /api/v2/users/<user-id>.json/muting?action=mute
```

Resume recording (unmute):

```
POST /api/v2/users/<user-id>.json/muting?action=unmute
```

Response contains HTTP status code as shown in the following table.

Response	**Description**
200 OK	Call recording has been paused/resumed successfully for the current active call for this user.
403 Forbidden	The request cannot be completed because API user has no permission to pause/resume recording. Verify role permissions.
409 Conflict	The current active call state has been changed from active to completed in a middle of request.
404 Not Found	No active call exist for this user right now.

4. Examples

4.1 Create tenant with roles, groups and users

In this example we will use REST API to create new tenant account with its roles, groups, extensions and users.

We will use cURL utility to send REST API requests to MiaRec server.

4.1.1 **Prerequisites**:

- User account should be created in MiaRec web portal for REST API application. This account should have appropriate role permissions to access and modify MiaRec data. In this example, we use account with login `apiuser` and password `secret` . Its role is `Administrator` will full access to system. The user account should belong to "System" tenant otherwise it will not be able to create tenants.

In this example we will create:

- Tenant account with name 'Acme Tenant'
- Two groups within this tenant with names 'Agents Group' and 'Managers Group'
- Two user roles with names 'Agent Role' and 'Manager Role'
- A few users inside both of groups and with both of roles

4.1.2 Step 1. Verify REST API connection

Execute the following CURL command to retrieve a list of tenants:

```
curl -u apiuser:secret http://{host}:{port}/api/v2/tenants.json
```

Expected result is something like the following. If you see any error instead, then check your REST API user role permissions.

```
{
  "next_url": null,
  "total": 8,
  "tenants": [
    {
      "tenant_id": "34c7f1f6-9201-11e5-a739-e03f497dbdff",
      "timezone": null,
      "name": "Flexus",
      "encrypt_data": false,
      "fieldset_associate_calls": {
        "extension_uniquness": "systemwide",
        "associate_calls": "broadworks_group",
        "broadworks_sp_id": "FlexusProvider",
        "broadworks_group_id": "FlexusGroup",
        "associate_calls_condition": "BroadWorks-serviceProviderID = \"FlexusProvider\" AND BroadWorks-groupID = \"FlexusGroup\"",
        "cisco_partition": "",
        "metaswitch_system": "",
        "metaswitch_group": "",
        "sip_uri_host": "",
        "record_unknown_users": "record",
        "auto_provision": true,
        "provision_role_id": "34cafffe-9201-11e5-b516-e03f497dbdff",
        "provision_group_id":  "47f53fcc-9201-11e5-b4ef-e03f497dbdff"
      },
      "fieldset_licensing": {
        "recording_seats": 20,
        "evaluation_seats":  20,
        "recording_sessions":  0,
        "monitoring_seats": 10,
        "license_mode": "fixed",
        "monitoring_sessions": null
      }
    },
    ... MORE DATA ...
}
```

4.1.3 Step 2. Create tenant account

To create new record we need to submit HTTP POST request with JSON tenant data.

Prepare JSON tenant data in a temporary file with name **add_tenant.json**. Content of this file:

```
{
  "tenant": {
      "name": "Acme Tenant",
      "encrypt_data": false,
      "fieldset_licensing": {
        "license_mode": "fixed",
        "recording_seats": 50,
        "evaluation_seats":  50
      }
    }
}
```

If any tenant configuration parameter is not provided in the submitted JSON data, then a default value will be used. For example, we skipped such parameter like "timezone".

Execute CURL command to add this tenant:

```
curl -u apiuser:secret -H "Content-Type: application/json" -X POST --data "@add_tenant.json" http://{host}:{port}/api/v2/tenants.json
```

If the tenant account is created successfully, then URL to new tenant's profile page is returned in the response message:

```
{"url":  "/api/v2/tenants/00416392-cf76-11e5-a728-e03f497dbdff.json"}
```

Now we can retrieve tenant's data with HTTP GET request:

```
curl  -u  apiuser:secret  http://{host}:{port}/api/v2/tenants/00416392-cf76-11e5-a728-e03f497dbdff.json
```

4.1.4 Step 3. Create user roles

We will create two roles:

- Group Manager Role
- Agent Role

Prepare JSON data in temporary files for each of roles.

Make sure that "tenant_id" is set to the ID for newly created tenant from the previous step. In our example it is **00416392-cf76-11e5-a728-e03f497dbdff**.

Content of file **add_role_manager.json**:

```
{
  "role": {
    "name": "Group Manager Role",
    "access_level": "managed_groups",
    "tenant_id": "00416392-cf76-11e5-a728-e03f497dbdff",
    "permissions": {
      "calls": [
        "categorize",
        "playback",
        "pause_recording",
        "live_monitor",
        "view",
        "on_demand_trigger",
        "add_notes"
      ],
      "call_categories": [
        "view",
        "delete",
        "edit"
      ],
      "call_notes_own": [
        "view",
        "pin",
        "delete"
      ],
      "call_notes": [
        "view",
        "pin"
      ]
    }
  }
}
```

Content of file **add_role_agent.json**:

```
{
  "role": {
    "name": "Agent Role",
    "access_level": "user",
    "tenant_id": "00416392-cf76-11e5-a728-e03f497dbdff",
    "permissions": {
      "calls_own": [
        "view",
        "categorize",
        "playback",
        "pause_recording",
        "on_demand_trigger",
        "add_notes"
      ],
      "call_categories": [
        "view",
        "delete",
        "edit"
      ],
      "call_notes_own": [
        "view",
        "pin",
        "delete"
      ],
      "call_notes": [
        "view",
        "pin"
      ]
    }
  }
}
```

Execute CURL command to add "Group Manager Role" role:

```
curl -u apiuser:secret -H "Content-Type: application/json" -X POST --data "@add_role_manager.json" http://{host}:{port}/api/v2/roles.json
```

If role is created successfully, then the response message contains URL to new role's data:

```
{"url":  "/api/v2/roles/e4195b2c-cf78-11e5-9367-e03f497dbdff.json"}
```

Repeat the same steps for role "Agent Role":

```
curl -u apiuser:secret -H "Content-Type: application/json" -X POST --data "@add_role_agent.json" http://{host}:{port}/api/v2/roles.json
```

Success response:

```
{"url":  "/api/v2/roles/14e1c064-cf79-11e5-9804-e03f497dbdff.json"}
```

4.1.5 Step 4. Create user groups

We will create two groups:

- Managers Group
- Agents Group

Prepare JSON data in temporary files for each of groups.

Make sure that "tenant_id" is set to the ID for newly created tenant from the previous step. In our example it is **00416392-cf76-11e5-a728-e03f497dbdff**.

Content of file **add_group_managers.json**:

```
{
  "group": {
    "tenant_id": "00416392-cf76-11e5-a728-e03f497dbdff",
    "name": "Managers Group"
  }
}
```

Content of file **add_group_agents.json**:

```
{
  "group": {
    "tenant_id": "00416392-cf76-11e5-a728-e03f497dbdff",
    "name": "Agents Group"
  }
}
```

CURL command for adding group "Managers Group":

```
curl -u apiuser:secret -H "Content-Type: application/json" -X POST --data "@add_group_managers.json" http://{host}:{port}/api/v2/groups.json
```

If group is created successfully, then the response message contains URL to group's data:

```
{"url":  "/api/v2/groups/5fce93a8-cf7a-11e5-8e94-e03f497dbdff.json"}
```

Repeat the same steps for group "Agents Group":

```
curl -u apiuser:secret -H "Content-Type: application/json" -X POST --data "@add_group_agents.json" http://{host}:{port}/api/v2/groups.json
```

Success response:

```
{"url":  "/api/v2/groups/7248c8f0-cf7a-11e5-adb9-e03f497dbdff.json"}
```

4.1.6 Step 5. Create user accounts

We will create two users:

- Peter has role "Group Manager Role" and he belongs to group "Managers". Peter is also a manager of group "Agents".
- David has role "Agent Role" and he belongs to group "Agents".

Prepare JSON data file **add_user_peter.json** with the following content:

```
{
    "user": {
        "name": "Peter",
        "is_active": true,
        "role_id": "e4195b2c-cf78-11e5-9367-e03f497dbdff",
        "group_id": "5fce93a8-cf7a-11e5-8e94-e03f497dbdff",
        "managed_groups": [
            "7248c8f0-cf7a-11e5-adb9-e03f497dbdff"
        ],
        "timezone": "Europe/London",
        "fieldset_login": {
            "can_login": true,
            "login": "peter",
            "password": "secret",
            "authenticate_type":  "password"
        },
        "fieldset_recording":  {
            "extensions": ["2001",  "2002"],
            "record": "always",
            "record_direction": ["in", "out"]
        },
        "fieldset_licensing":
        {
            "recording_seat":  true
        }
    }
}
```

Replace in this file the following values:

- `role_id` with ID of "Manager Role" as returned in one of previous steps
- `group_id` with ID of "Managers Group" as returned in one of previous steps
- `managed_groups` should be a list of IDs, which the current user is a manager of. In our example we have a single value in this list and this ID should be replaced with ID of "Agents Group" as returned in one of previous steps.

CURL command:

```
curl -u apiuser:secret -H "Content-Type: application/json" -X POST --data "@add_user_peter.json" http://{host}:{port}/api/v2/users.json
```

If user account is created successfully, then the response message contains URL to user's data:

```
{"url":  "/api/v2/users/34e2ac80-9201-11e5-93e4-e03f497dbdff.json"}
```

Repeat the same steps for user "David".

Content of file **add_user_david.json**:

```
{
    "user": {
        "name": "David",
        "is_active": true,
        "role_id": "14e1c064-cf79-11e5-9804-e03f497dbdff",
        "group_id": "7248c8f0-cf7a-11e5-adb9-e03f497dbdff",
        "timezone": "Europe/London",
        "fieldset_login": {
            "can_login": true,
            "login": "david",
            "password": "secret",
            "authenticate_type":  "password"
        },
        "fieldset_recording": {
            "extensions": ["123456"],
            "record": "always",
            "record_direction": ["in", "out"]
        },
        "fieldset_licensing":
        {
            "recording_seat":  true
        }
    }
}
```

Replace in this file the following values:

- `role_id` with ID of "Agent Role" as returned in one of previous steps
- `group_id` with ID of "Agents Group" as returned in one of previous steps

CURL command:

```
curl -u apiuser:secret -H "Content-Type: application/json" -X POST --data "@add_user_david.json" http://{host}:{port}/api/v2/users.json
```

4.1.7 FAQ

Question: **Where can I find detailed description of all fields that are supported in JSON submitted data?**

Answer: Our product is continuously improved and new configuration parameters may be added at any time. The documentation may be out of date. The best way is to execute the following HTTP GET requests on your MiaRec server and retrieve a complete list of currently supported parameters:

```
curl -u apiuser:secret http://{host}:{port}/api/v2/tenants.json
curl -u apiuser:secret http://{host}:{port}/api/v2/roles.json
curl -u apiuser:secret http://{host}:{port}/api/v2/groups.json
curl -u apiuser:secret http://{host}:{port}/api/v2/users.json
```

If you do not understand meaning of some parameters, for example, inside user's JSON data, then we recommend to edit user's profile settings via web-interface and see how the JSON representation changes after that.

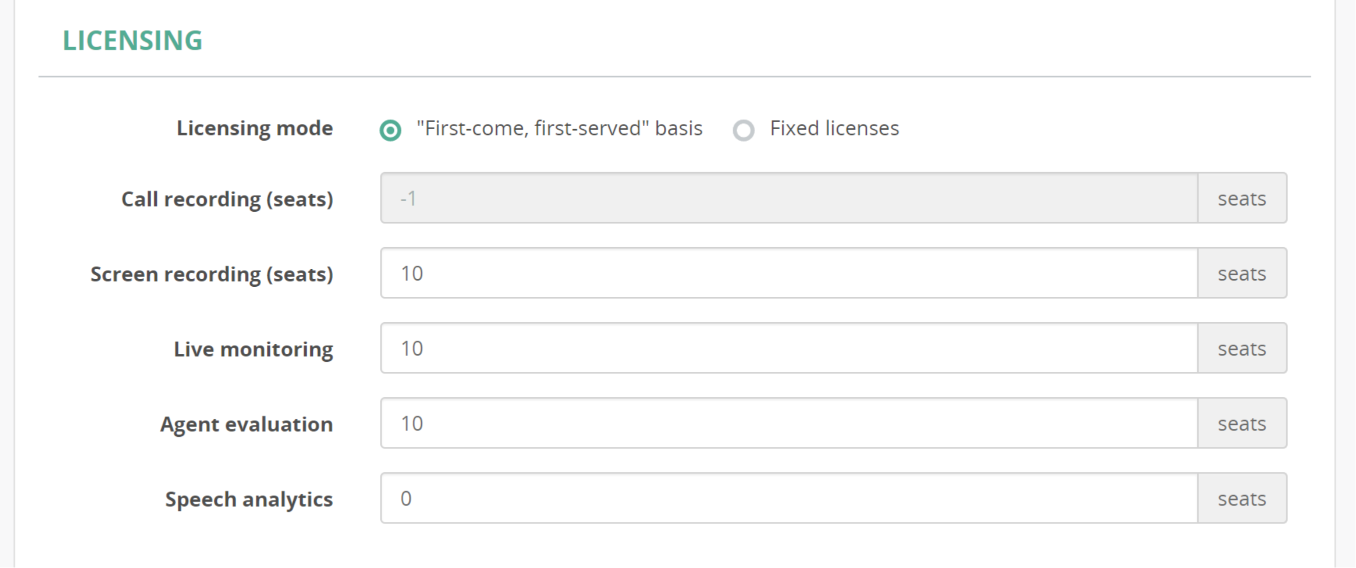
And of course, feel free to contact us if you have any issue or question.

5. History of changes

5.1 Changes to tenant.fieldset_licensing attribute.

Per-tenant license configuration is improved. Previously, it was possible to set "First-come, first-served" licensing mode for "call recording" licenses only. Other licenses were fixed. In this release, all licenses can be be either dynamic or fixed. A configuration section is renamed to "License limits".

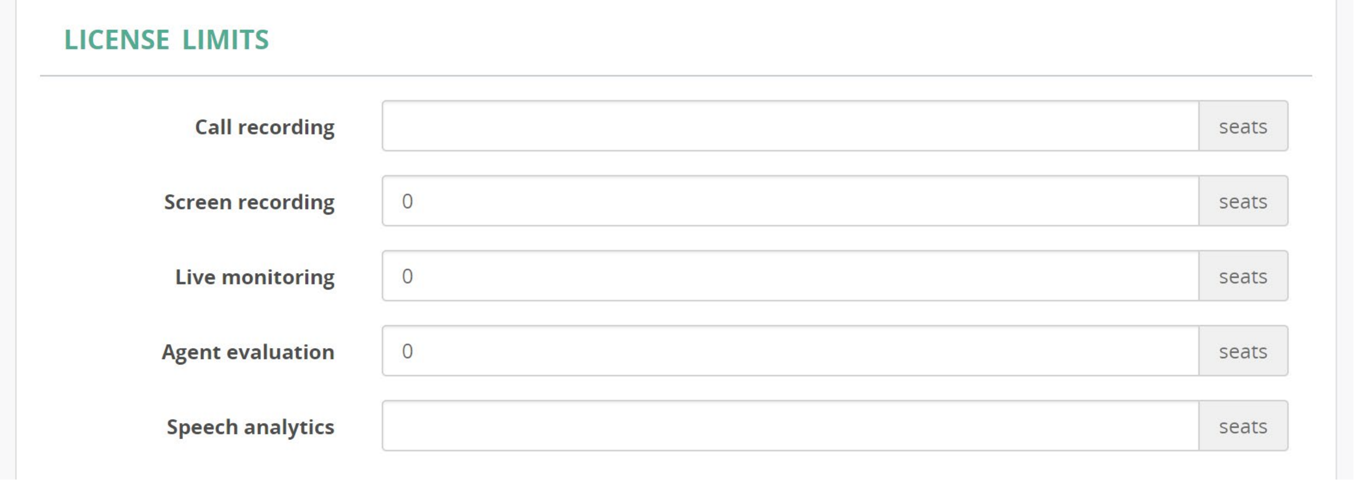
Before:



LICENSING

Licensing mode	● "First-come, first-served" basis ○ Fixed licenses	
Call recording (seats)	-1	seats
Screen recording (seats)	10	seats
Live monitoring	10	seats
Agent evaluation	10	seats
Speech analytics	0	seats

Now:



LICENSE LIMITS

Call recording		seats
Screen recording	0	seats
Live monitoring	0	seats
Agent evaluation	0	seats
Speech analytics		seats

5.1.1 Changes to REST API:

Old version:

```
{
    "tenant":
    {
        "fieldset_licensing":
        {
            "license_mode": "dynamic",
            "recording_seats": -1,
            "screen_recording_seats":  0,
            "monitoring_seats":  10,
            "evaluation_seats":  20,
            "speech_analytics":  20
        },
        ...
```

New version:

```
{
    "tenant":
    {
        "fieldset_licensing":
        {
            "recording_seats": null,
            "screen_recording_seats": 0,
            "monitoring_seats":  10,
            "evaluation_seats":  20,
            "speech_analytics":  20
        },
        ...
```

A tenant's licensing configuration (attribute `fieldset_licensing`) has the following changes:

- Attribute `license_mode` is depreciated. It is ignored when supplied in PUT/POST requests. It is not returned in GET requests.
- `null` can be set to any of license types. `null` value means **no limits**.
- dynamic license (no limits) is supported to any of license types.

In previous versions, a `license_mode` attribute might have one of the following values:

- **fixed** - A particular number of licenses is pre-allocated to this tenant. Tenant cannot use more license then pre-allocated. If the tenant has more users, then licenses, then call recordings for some of users will be unlicensed.
- **dynamic** - First-come, first served (FCFS) strategy. Licenses are dynamically allocated to tenant from a global pool. Tenant is allocated dynamically as many licenses as it has users with appropriate license configuration.

New version still allows to use **First-come, first served** strategy for license distribution, but terminology is different. Instead of **license mode**, a term **license limits** is used. Value `null` or `-1` (for backward compatibility) means **no limits**, values from 0 to N sets an upper limit for tenant for particular license type.

For backward compatibility, you can supply `license_mode` attribute in PUT/POST requests and set license counter to `-1` when a **dynamic** mode is desired. The following example has the same effect on both old and new versions:

```
"fieldset_licensing":{
    "recording_seats": -1,
    "screen_recording_seats":  0,
    "monitoring_seats":  0,
    "evaluation_seats":  0,
    "speech_analytics": 10,
    "license_mode": "dynamic"
}
```