Q-loud REST API Documentation

Release 1.14.1.test1

Roland Hänel

May 04, 2017

CONTENTS

This document defines and describes the application programming interface (API). All of the functionality is exposed via the API, in fact, even the web-interface is a HTML5 application ("AJAX") that solely relies on the API outlined in this document with no additional features. With this full-featured coverage, the API allows for almost unlimited possibilies to integrate its features into third-party systems, interfaces and products.

The API is completely based on HTTP and follows a REST architecture, using JSON to encapsulate and serialize object data where applicable. This approach enables rapid integration of the API because of a simple, well-understood architecture and information content that can easily be read by both human beings and machines.

Through the use of a "comet-like" functionality within the `GET /api/event` API call, the API also features "push-style" notifications that effectively enable "real-time" capabilities for automated data processing within the platform. A broad range of specific events enable the client to stay informed about events and data object generated by other users. These events are also specified in this document.

CHAPTER

ONE

EXAMPLE API USAGE

The REST API uses JSON as the underlying data serialization mechanism, because JSON is a quick, easy-to-read and fast technology. Parsers and encoders are available for virtually all programming languages and of course it's the native way to express objects in Javascript, which is the major target of in-browser applications.

The following example includes basic code in Python to show a fully function API session. The code can be literally re-used if Python is the language of choice, or just be taken as a further explanation of how to use the API. For Python, we include the HTTP and JSON libraries:

```
import httplib
import json
```

The API follows a session model, which means that for all transactions, you must first create a session with the cospace platform as a context for the following API requests. All session requests must be directed to http://api.cospace.de or https://api.cospace.de, which serves as a "load-balancing" name for individual API servers within the platform.

The *`GET /api/session`* call returns the session ID (`sid`) and the responsible `server` for this session. *Further requests within this session must be addressed to this server only, because only this server has knowledge about the session data.* This mechanism ensures a load-distribution between multiple API serving nodes in the cospace system. The `sid` and `server` information will also be sent by the server in form of HTTP cookies. Further requests within the session rely on re-transmitting `sid` to the server via the `Authorization: Bearer <sid>` header, via `sid` cookie, or it may be given as a `sid` URL parameter.

```
# get session information on api.cospace.de

initialConnection = httplib.HTTPSConnection("api.cospace.de")
initialConnection.request("GET", "/api/session")

# result is JSON encoded

response = json.loads(initialConnection.getresponse().read())

# print response

print "get session response: " + str(response)

# cut https:// prefix (7 characters) from server returned in JSON

apiServer = response["server"][8:]

# remember the session id

sid = response["sid"]

# create the connection for the following API requests
```

```
apiConnection = httplib.HTTPSConnection(apiServer)
```

Just created, the session is not authenticated, which means it is not associated with a system user account. Let's assume that you already create a user with cospace (using the web interface on http://cospace.de), so we can now authenticate the session with this user using the *POST /api/session*. Note that this call is directed to the server returned by the *GET /api/session* call and carries the `Authorization` header:

```
# post credentials into session

body = json.dumps({
    "username": "johndoe47",
    "password": "secret12345"
})

headers = {"Authorization": "Bearer " + sid}

apiConnection.request("POST", "/api/session", body, headers=headers)

response = json.loads(apiConnection.getresponse().read())

print "login response: " + str(response)
```

Now, the session is authenticated and the session ID grants access to the user whose credentials we just logged in with. We might use the following *GET /api/user* API call to find out more about the user `johndoe47` associated with our example session:

```
apiConnection.request("GET", "/api/user", headers=headers)

response = json.loads(apiConnection.getresponse().read())

print "get user details: " + str(response)

# remember the John's "tag_all" to access his objects later

tagAll = response["user"]["tag_all"]
```

The *GET /api/user* call we just made lists the user's "system tags", which serve as the fundamental point of managing the user's objects. Given the UUID of John's `tag_all` tag (the tag that references all of his objects), we might get a list of all his data objects with the *GET /api/tag/(uuid)/object* call:

```
apiConnection.request("GET", "/api/tag/" + tagAll + "/object", headers=headers)

response = json.loads(apiConnection.getresponse().read())

print "user's objects: " + str(response)
```

Before we finish the example, one last note about a special API call, *GET /api/event*. In contrast to all the other API calls, which focus to return as fast as possible after issuing the HTTP request, *GET /api/event* will "hang" until the requested event (i.e., any event with an id equal or higher to the id that was requested within the call) arrives. The event is then delivered instantly by returning the API call. In this way, push-style notifications are accomplished through client-initiated, blocking requests.

The time until the call returns can be tuned with the `timeout` parameter. After timeout, the call will return empty, if no event was seen.

```
apiConnection.request("GET", "/api/event?timeout=10&sid=" + sid)

# this might take up to 10 seconds

response = json.loads(apiConnection.getresponse().read())

print "event response: " + str(response)
```

After a session is no longer used, consider it a good habit to free the server-side resources of the session by deleting it:

```
apiConnection.request("DELETE", "/api/session", headers=headers)

response = json.loads(apiConnection.getresponse().read())

print "session delete status: " + str(response["status"])
```

CHAPTER

TWO

GENERIC API REQUEST CONSIDERATIONS

All API calls operate on sessions, with the only exception of the *GET /api/session* call that is used to create a session in the first place. The session id can be provided by the client in an `Authorization: Bearer <sid>` header, as a URL parameter `sid`, or as a cookie named `sid`. Note that sessions and their ids are only known in the context of a specific, single server, as returned on session creation.

Unless otherwise noted, all API commands require not only a valid session, but also a successful user authentication on this session, as performed with the *POST /api/session* call.

2.1 API Responses

The server will always answer with HTTP code 200 OK to all API URL requests that are supposed to deliver JSON response messages. The JSON response itself carries the information whether the API call was successful or some error has occurred.

The general layout of a JSON response looks like the following:

```
{
    "status": "ok"|"no-session"|"wrong-syntax"|"missing-element"|"fail"|...,
    "message": "a short text about what went wrong"
}
```

The `status` element is always present and carries information about the success of the API call, or a reason why the call was not successful. There are some generic status strings that can occur in possibly all of the API calls. Some API calls define additional, situation-specific strings. The generic status strings are:

Status	Description
ok	The API call was successfully executed and no error has occurred
no-session	The request did not include a valid session id, or it contained an invalid or expired id
no-auth	The session id included in the request is not authenticated with a valid user or partner
wrong-origin	The API call was made from a wrong origin (i.e., the Origin header in the HTTP request does not match the Origin header of the session creation request)
wrong-syntax	Some element in the request does not match the required element syntax format (e.g., the value of a field was too long or included invalid characters)
missing-element	Some mandatory element is missing from the request
fail	A generic failure has occurred. This is usually an unexpected event, more specific status codes are used for specific failure conditions. Note that for any failure condition, partial execution of the API call may have occurred.

If an individual API call specifies additional, specific status strings, they are documented there.

The `message` element might be present in a status "non-ok" condition and contains a short description of the generic failure. This is primarily useful for debugging and logging purposes.

Every API response includes the general response layout described in this section, the status and message fields are not shown in the descriptions of the specific API calls.

2.2 Localization

cospace uses very little server-side localization, mostly for texts in e-mails and sounds in the phone features. The language parameter or field to some of the API calls specifies the desired language. Supported languages in this version of the API are:

Language identifier	Description
de	German
en	English

2.3 Authentication for file downloads

The session ID (`sid`) is the central authentication token that grants access to all API requests once the session is authenticated. However, sometimes a limited access only to certain objects is needed; and an application that obtained a session ID may not want to share the session ID with some other process only to enable that process to access a specific object. For this reason, cospace implements a restricted authentication method for file downloads, the so-called "download id" (`did`).

Certain API calls (usually those API calls that access file-level data of some objects) accept the `did` as a means of authentication in addition to the possibility of using the session ID. If authentication with `did` is requested, two URL parameters need to be present for the respective API call:

- `did` – this parameter carries the did of the session.
- `dkey` – this parameter, called the download key, contains a hashed combination of the requested object UUID and the sid (session ID).

The `dkey` is calculated as follows:

```
dkey = Base64_URL_Safe(MD5(string(object UUID), string(sid)))
```

That is the (lowercase) version of the object UUID's string representation (ASCII/UTF-8) is concatenated with the session ID and then hashed with an MD5 hash function. The binary (!) result of the MD5 hash is encoded with the `base64url` method as defined in RFC 4648 section 5.

2.4 On the use of UUIDs

The system makes extensive use of Universally Unique Identifiers (UUIDs) to identify objects, metadata, parameters and state on the platform. All UUIDs used by the platform are time-based UUIDs (Version 1 according to RFC 4122). If clients need to sort objects or data identified by UUIDs by time, they can exploit this feature and implemented the sorting based on the intrinsic timestamps of these UUIDs. In cases where no explicit timestamp for the object exists, the intrinsic timestamp can also serve as an information about the creation time of the respective object.

However, clients must never rely on the fact that a specific server seems to have a specific clock sequence or node identifier, as these might change any time in a distributed system (for newly created UUIDs).

CHAPTER

THREE

API CALL REFERENCE (USER API)

3.1 /api/session

GET /api/session
Requests a new session with the system or confirms an existing session.

Query Parameters

- **language** – optional: language identifier

  used to localize server-side messages before the session is authenticated.

  defaults to `de` if not given

Response body example

```
{
    "sid": "boajw93bawjckbja2ZdbfdGa84Afib",
    "did": "iaEia83AviaDia943faobpEPRkva98",
    "server": "https://api43.service.de:1234",
    "partner": "mypartner",
    "auth": false,
    "next_event": 0
}
```

Status

- **resource-throttle**: The server did not create a session because of excessive resource usage. A new call to this API might be considered at a later time.
- **wrong-session**: The request contained an `sid` parameter that does not map to a known session on the server.

The HTTP response to this API call will set two HTTP cookies, `sid` and `server`, with exactly the same contents as returned in the JSON response message.

Further API calls that reference the newly created session can only be made towards the server specified in the `server` field, as the session state is only maintained on this server.

If the request to this API call contains an `Authorization` header, a `sid` cookie or URL parameter, the server will try to re-use the given session. If the session cannot be re-used, a `wrong-session` status is returned and the `sid` and `server` cookies will be deleted.

The `auth` response field will be `false` in the case of a newly created session (because no user is yet authenticated with this session), but may have a value of `true` if an existing, already authenticated session was re-used.

The `did` field is the download id of this session that can be used on certain API GET requests to obtain file contents. For more information on file download authentication, see *Authentication for file downloads*.

The `partner` field in the response will only be present if the authenticated user belongs to a partner.

The `next_event` response field indicates the id of the next, not-yet-received event. It might be used as the `next` parameter of the *`GET /api/event`* call to ensure that only new events are received.

`POST /api/session`

Authenticate a user with or without an existing session.

Request body example 1 (user authentication)

```
{
    "username": "johndoe47",
    "password": "secret12345"
}
```

- or-

```
{
    "email": "john@doe.com",
    "password": "secret12345"
}
```

Request body example 2 (partner authentication)

```
{
    "partner": "mypartner",
    "key": "dOajks83AF39ua08uaSDFas"
}
```

Request body example 3 (partner proxy authentication)

```
{
    "partner": "mypartner",
    "key": "dOajks83AF39ua08uaSDFas",
    "user": "52c53e36-65be-11e4-bf4a-201a06c768e1"
}
```

JSON Parameters

- **`username`** – `^[a-zA-Z0-9]{2,20}$`

  only for user authentication

- **`email`** – `<user@domain>`

  only for user authentication

- **`password`** – `^.{5,50}$`

  only for user authentication

- **`partner`** – `^[a-zA-Z0-9]{2,20}$`

  only for partner authentication

- **`key`** – `^.{5,50}$`

  only for partner authentication

- **user** – UUID

  only for partner proxy authentication

Response body example 1 (user authentication)

```
{
    "partner": "mypartner"
}
```

Response body example 2 (user authentication without session)

```
{
    "sid": "boajw93bawjckbja2ZdbfdGa84Afib",
    "did": "iaEia83AviaDia943faobpEPRkva98",
    "server": "https://api43.service.de:1234",
    "partner": "mypartner",
    "auth": true,
    "next_event": 0
}
```

Response body example 3 (partner authentication)

```
{}
```

Response body example 4 (partner proxy authentication)

```
{}
```

Status

- **wrong-credentials**: username and/or password is wrong/unknown
- **already-auth**: the session is already authenticated
- **wrong-user**: the given user does not exist or does not belong to the partner

If this API call was successful (i.e. a valid password and the correct username or email is given), the session is authenticated and ready to accept API calls that require an authenticated session.

The `partner` field in the response will only be present if the authenticated user belongs to a partner.

If user authentication is used, this API may be called without a session to reduce the number of required requests to create an authenticated session.

If the `user` field is given in a partner authentication, this signals a partner proxy authentication for the specific user. Using this feature, a partner can create an authenticated user session for the specific user.

DELETE /api/session

Closes a session and invalidates the session id.

Response body example

```
{}
```

After closing the session, the session id is deleted on the server and can not be used for any further requests. The HTTP response will also invalidate the sid cookie.

This command does require a valid session, but not necessarily a user authentication with that session.

3.2 /api/signup

GET /api/signup/captcha/ (*xxx*) **.png**
Retrieves a captcha picture associated with the given sign-up process.

Response body example

```
The captcha picture in PNG format on success, or HTTP 404 without
response body on failure.
```

Each time called, this API replaces the server-side captcha with a new one.

This command does require a valid session, but not necessarily a user authentication with that session.

POST /api/signup/verification
Requests verification for the sign-up procedure via E-mail in case of sign-up with captcha or via the HTTP response in case of sign-up with hardware code, or initiates partner-controlled sign-up procedure.

Request body example 1 (user sign-up)

```
{
    "captcha": "soLuTiOn",
    "email": "john@doe.com"
}
```

Request body example 2 (partner-controlled sign-up)

```
{
    "info": {
        "some": ["thing", 4711]
    }
}
```

Request body example 3 (sign-up with hardware code)

```
{
    "hardware-code": "ABCDE-FGHIJ-KLMNI-OPQRS",
    "email": "john@doe.com"
}
```

JSON Parameters

- **captcha** – `^[a-zA-Z0-9]{1,20}$`

  The captcha solution for the captcha that was previously requested with *GET /api/signup/captcha/(xxx).png*

  (not required when this operation is done on a partner session or when a sign-up with `hardware-code` is performed)

- **email** – `user@domain`

  The e-mail address to be verified

  (not required when this operation is done on a partner session)

- **info** – `^.{1,10000}$`

  Informational metadata that will later be returned on the corresponding *GET /api/signup/verification/(verification)* call. Can be any valid JSON data type.

- **hardware-code** – `^.{10,50}$`

  the sensor device access code, which is the key to the physical device.

  (in this case, the `captcha` field is not required)

Response body example 1 (user sign-up)

```
{}
```

Response body example 2 (partner-controlled sign-up)

```
{
    "verification": "dijaDia8Aiofjb9aaobasdfDF",
    "uuid": "dfc3e9c4-4e81-11e1-9fe3-0024e8f90cc0",
    "info": {
        "some": ["thing", 4711]
    }
}
```

Response body example 3 (sign-up with hardware code)

```
{
    "verification": "dijaDia8Aiofjb9aaobasdfDF"
}
```

Status

- **wrong-captcha**: The captcha solution is wrong, or no captcha was previously requested
- **email-failure**: The verification e-mail could not be sent out
- **already-exist**: The e-mail address matches an existing user record (and thus cannot be used)
- **wrong-code**: The hardware code is invalid.
- **exceeded-limit**: The use of this hardware code for verification process has reached the maximum permitted count of 5 verifications. This hardware code cannot be used anymore for sign-up.

This API generates a verification key and sends it to the E-mail address in case of a sign-up with captcha, or sends it in the HTTP response in case of a sign-up with hardware code or a partner-controlled sign-up.

This command does require a valid session, but not necessarily a user authentication with that session for the user-style sign-up. In this case, a call to this API will reset the server-side captcha, no matter whether the response is successful or not. That means, that if the client needs to re-try this API call, a new captcha must be requested and solved before a second try can be made.

For a partner-controlled sign-up, the call will return the verification key and a uuid. If a user account is later created with this verification key, the user will be assigned this uuid.

GET **/api/signup/verification/**(*verification*)
Request information for the given E-Mail verification key

Response body example 1 (user sign-up)

```
{
    "email": "john@doe.com"
}
```

Response body example 2 (partner-controlled sign-up)

```
{
    "partner": "mypartner",
    "uuid": "dfc3e9c4-4e81-11e1-9fe3-0024e8f90cc0"
}
```

Status

- **unknown-verification**: The given verification key is unknown

This command does require a valid session, but not necessarily a user or partner authentication with that session.

DELETE /api/signup/verification/ (*verification*)

Deletes a verification key.

Response body example

```
{}
```

Status

- **unknown-verification**: The given verification key is unknown

This command does require a valid session, but not necessarily a user or partner authentication with that session.

POST /api/signup/user

Creates a new user account within the system.

Request body example

```
{
    "username": "johndoe47",
    "firstname": "John",
    "lastname": "Doe",
    "display_name": "Johnny Doe",
    "country_code": "+49",
    "password": "secret12345",
    "language": "de",
    "newsletter": true,
    "verification": "dijaDia8Aiofjb9aaobasdfDF"
}
```

JSON Parameters

- **email** – `user@domain`

  required if the verification key is from a partner-controlled sign-up and does not already include an e-mail address

- **username** – `^[a-zA-Z0-9]{2,20}$`
- **firstname** – `^.{1,50}$`
- **lastname** – `^.{1,50}$`
- **display_name** – `^.{1,100}$`

  optional; the full username as it should be displayed within an application (default is "*firstname lastname*")

- **country_code** – `^\+[0-9]{1,3}$`

  optional; the country code of the user. (default is "+49" for Germany)

- **password** – `^.{5,50}$`
- **language** – optional; a valid language identifier
- **newsletter** – `true|false`

  optional; `true` if the user wants to receive the newsletter, `false` if not. Default `true`.

- **verification** – `^[a-zA-Z0-9]{1,50}$`

  the verification key (received by e-mail or partner-controlled sign-up)

Response body example

```
{}
```

Status

- **duplicate-username**: the selected username is already taken
- **unknown-verification**: the verification key is unknown
- **already-exist**: The e-mail address matches an existing user record

The client must provide a valid verification key (E-Mail verification procedure or partner-controlled sign-up). On success, a new user is created within the system and the E-Mail verification key is deleted.

The `country_code` field is used to enable clients to display a correct phone number. If no `country_code` field is given, the default value "+49" for Germany will be used.

If no `language` field is given, the user is created using the language of the session.

If the response indicates success, the session is automatically authenticated with the newly created user, i.e. the client can immediately issue all API commands that require an authenticated session.

This command does require a valid session without a user authentication.

3.3 /api/password

POST /api/password/recovery

Request E-Mail password recovery procedure (password recovery code).

Request body example

```
{
    "username": "johndoe47",
    "captcha": "soLuTiOn"
}
```

- or-

```
{
    "email": "john@doe.com",
    "captcha": "soLuTiOn"
}
```

JSON Parameters

- **username** – `^[a-zA-Z0-9]{2,20}$`
- **email** – `user@domain`

  alternatively to the username: the email address of the user

- **captcha** – `^[a-zA-Z0-9]{1,20}$`

  the captcha solution for the captcha that was previously requested with *GET /api/signup/captcha/(xxx).png*

Response body example

```
{}
```

Status

- **wrong-captcha**: The captcha solution is wrong, or no captcha was previously requested
- **wrong-username**: The given username or email does not exist
- **email-failure**: The verification e-mail could not be sent out

This API generates an e-mail with a password recovery code and sends it to the e-mail address stored in the user's profile.

This command does require a valid session, but not necessarily a user authentication with that session.

A call to this API will reset the server-side captcha, no matter whether the response is successful or not. That means, the if the client needs to re-try this API call, a new captcha must be requested and solved before a second try can be made.

POST /api/password

Resets a user's password using a valid password recovery code.

Request body example

```
{
    "username": "johndoe47",
    "code": "iausDia8sdfasjb9aaobasdfDF",
    "password": "newpass"
}
```

- or-

```
{
    "email": "john@doe.com",
    "code": "iausDia8sdfasjb9aaobasdfDF",
    "password": "newpass"
}
```

JSON Parameters

- **username** – `^[a-zA-Z0-9]{2,20}$`
- **email** – `user@domain`

  alternatively to the username: the email address of the user

- **code** – `^[a-zA-Z0-9]{1,50}$`

  the password recovery code (received by e-mail)

- **password** – `^.{5,50}$`

  the new password

Response body example

```
{
    "partner": "mypartner"
}
```

Status

- **wrong-username**: The given username or email does not exist or the password recovery code is invalid for this username

The client must provide a valid password recovery code (E-Mail password recovery procedure). On success, the password of the given user is overwritten with password and the E-Mail password recovery code is deleted.

The user account can be identified either by specifiying the username, or the email of the user.

If the response indicates success, the session is automatically authenticated with the user, i.e. the client can immediately issue all API commands that require an authenticated session.

The `partner` field in the response will only be present if the user belongs to a partner.

This command does require a valid session without a user authentication.

3.4 /api/language

GET /api/language
Get the list of languages

Response body example

```
{
    "language": [
        "en",
        "de"
    ]
}
```

This call will return all the languages supported by the API. It does not require a session.

3.5 /api/user

GET /api/user
Get user-related information of current session's user

Response body example

```
{
    "user": {
        "uuid": "1de7257a-4f34-11e0-ab6e-0024e8f90cc0",
        "username": "johndoe47",
        "firstname": "John",
        "lastname": "Doe",
        "display_name": "Johnny Doe",
        "birthday": "1970-12-24",
        "email": "john@doe.com",
        "language": "de",
        "newsletter": true,
        "partner": "somepartner",
        "zip_code": "10557",
        "town": "Berlin",
        "street": "Willy-Brandt-Stra\u00dfe",
        "house_number": "15",
        "country": "de",
        "country_code": "+49",
        "area_code": "30",
        "validation": "ok",
        "tag_all": "56697e6e-4f36-11e0-be81-0024e8f90cc0",
        "tag_unread": "56961546-4f36-11e0-9449-0024e8f90cc0",
        "tag_inbox": "56b8e300-4f36-11e0-873b-0024e8f90cc0",
        "tag_outbox": "56ddc9fe-4f36-11e0-a94b-0024e8f90cc0",
        "tag_trash": "5700ebe6-4f36-11e0-859a-0024e8f90cc0",
        "tag_shared": "5726c410-4f36-11e0-a8c1-0024e8f90cc0",
        "share_read": "9638badc-ab55-11e2-a825-0024e8f90cc0",
        "share_write": "a192a9e2-ab55-11e2-b215-0024e8f90cc0",
        "share_propagate": "a8c7aadc-ab55-11e2-a55f-0024e8f90cc0",
        "contact": "b02a5f0c-9268-11e0-84f8-0024e8f90cc0",
        "feature": {
            "pack": [
                "20GB+",
                "COSPACE-BOX"
            ],
            "quota": {
                "volume": 25000,
                "fax": 50
            },
            "used": {
                "volume": 452,
                "fax": 13
            },
            "feature": {
                "fax_noad": true,
                "call_unrestricted": true,
                "box_enable": true,
                "dialplan_enable": true,
                "control_phone": true
            }
        },
        "fax_ident": "+49 221 9999999",
        "fax_header": "John Doe, Inc.",
        "picture": true,
        "usergroup": "Controlling"
    }
}
```

The `partner` field will only be present if this user was created with a partner-controlled sign-up.

The `feature` section lists the feature configuration for the current user, that is, all of the additional feature packs that this user has, the resulting quota for `volume` (in MB) and `fax` (pages per month). The `used` subsection lists the actually used values (in case of `fax`, for the current month). The `feature` subsection lists boolean feature variables.

The `area_code`, `birthday`, `zip_code`, `town`, `street`, `house_number`, `country`, `fax_ident` and `fax_header` fields will only be present if they were set with *`POST /api/user`*.

The `validation` field can be set to `incomplete`, `checking`, `missmatch`, `baddocument`, `fail` or `ok`.

- `incomplete` address information not set with *`POST /api/user`* or no idcard set with *`POST /api/user/idcard`*.
- `checking` the address information are complete and in validation process
- `missmatch` the address information miss match to the idcard.
- `baddocument` the document dosen't match the requirements.
- `fail` the document is unreadable.
- `ok` the address information match to the idcard.

The `share_xxx` fields contain the user's personal share tags. These auto-generated tags can be used by other users to share objects with this user on a personal basis (i.e., without using manually created tags).

The `usergroup` field will only be present if the user was added to a usergroup by the corresponding partner.

The `picture` flag will have a value of `true` if the user has a profile picture.

`POST /api/user`

Modify user-related information of current session's user

Request body example

```
{
    "firstname": "John",
    "lastname": "Doe",
    "display_name": "Johnny Doe",
    "birthday": "1970-12-24",
    "zip_code": "10557",
    "town": "Berlin",
    "street": "Willy-Brandt-Stra\u00dfe",
    "house_number": "15",
    "country": "de",
    "country_code": "+49",
    "password": "newpass",
    "old_password": "oldpass",
    "pin": "1234",
    "language": "de",
    "newsletter": false,
    "email": "johnsnewmail@foobar.de",
    "fax_ident": "+49 221 9999999",
    "fax_header": "John Doe, Inc."
}
```

JSON Parameters

- **`firstname`** – `^.{1,50}$`

optional

- **lastname** – `^.{1,50}$`

  optional

- **display_name** – `^.{1,100}$`

  optional; the user name as it should display within an application

- **birthday** – `^[\d]{4}-[\d]{2}-[\d]{2}$`

  optional; the user's birthday, format `YYYY-MM-DD`, if present, `zip_code`, `town`, `street`, `house_number` and `country` must also be present

- **zip_code** – `^.{1,50}$`

  optional; if present, `birthday`, `town`, `street`, `house_number` and `country` must also be present

- **town** – `^.{1,50}$`

  optional; if present, `birthday`, `zip_code`, `street`, `house_number` and `country` must also be present

- **street** – `^.{1,50}$`

  optional; if present, `birthday`, `zip_code`, `town`, `house_number` and `country` must also be present

- **house_number** – `^.{1,50}$`

  optional; if present, `birthday`, `zip_code`, `town`, `street` and `country` must also be present

- **country** – `^[a-z]{2}$`

  optional; ISO country code; if present, `birthday`, `zip_code`, `town`, `street` and `house_number` must also be present

- **country_code** – `^\+[0-9]{1,3}$`

  optional; the country code of the user

- **password** – `^.{5,50}$`

  optional; a new password for this user

- **old_password** – `^.{5,50}$`

  optional; must be present if password is present

- **pin** – `^[0-9]{4}$`

  optional; a new phone PIN for this user

- **language** – optional; a valid language identifier

- **email** – `user@domain`

  optional; a new e-mail address (first step)

- **newsletter** – `true|false`

  optional; `true` if the user wants to receive the newsletter

- **code** – `^[a-zA-Z0-9]{1,50}$`

  optional; the e-mail verification code (second step)

- **fax_ident** – `^[\+ 0-9]{0,20}$`

  optional; the default fax transmitting station ID (TSI)

- **fax_header** – `^.{0,40}$`

  optional; the default fax header line

Response body example

```
{}
```

Status

- **wrong-password**: The `old_password` is wrong
- **already-exist**: The e-mail address matches an existing user record
- **email-failure**: The e-mail could not be sent out
- **address-incomplete**: Missing parameter `birthday`, `zip_code`, `town`, `street`, `house_number` or `country`
- **address-wrong**: The combination of the given address-parameters `zip_code`, `town`, `street`, `house_number` and `country` are incorrect, too imprecise or unknown to the system
- **address-forbidden**: The new address is correct, but is located at a different area code than the current address. Please release all phone numbers associated with your current area code and retry.
- **unsupported-country**: The country given is not supported by the system
- **too-young**: The given age (`birthday` field) is too young

To change the user's e-mail address, a two-step process is needed. First, this API call is used to with the `email` field to indicate the new e-mail address. The system will send a verification e-mail to the new address including a verification code. The client can then issue this API call a second time and provide the verification code in the code field to confirm and save the new e-mail address. Thus, a single call to this API must not include both the `email` and `code` fields.

In order to set location data for a user, all 6 parameters `birthday`, `zip_code`, `town`, `street`, `house_number` and `country` must be given. These parameters will be verified and the corresponding `area_code` will be set. If the user already has location data inserted, the new data will only be accepted if the new address is correct and has the same area code or there a no phone numbers associated with the old area code.

POST /api/user/delete

Requests permanent deletion of a user account.

Request body example

```
{
    "username": "johndoe47",
    "password": "secret12345"
}
```

- or-

```
{
    "email": "john@doe.com",
    "password": "secret12345"
}
```

Response body example

```
{}
```

Status

- **wrong-user**: The given username does not match the session's user or the user is controlled by a partner
- **wrong-credentials**: The given password is wrong

The deletion of a user account is a permanent action and cannot be undone.

To delete the user account, the username and password need to be repeated in this call (i.e., the given username must match the session's user name).

On successful completion, this call will invalidate the current user session.

POST /api/user/idcard

Set the current session user's idcard for identification process.

Request body

```
The request body contains binary data in PDF format.
```

Response body example

```
{}
```

Status

- **too-large**: The contents are too large to be handled by the system
- **malformed-file**: The content is malformed.

The data in the request body should be given as a binary format PDF.

GET /api/user/(user-uuid)

Get basic information about a user

Response body example

```
{
    "user": {
        "username": "johndoe47",
        "firstname": "John",
        "lastname": "Doe",
        "display_name": "Johnny Doe",
        "deleted": true,
        "picture": true,
        "share_read": "9638badc-ab55-11e2-a825-0024e8f90cc0",
        "share_write": "a192a9e2-ab55-11e2-b215-0024e8f90cc0",
        "share_propagate": "a8c7aadc-ab55-11e2-a55f-0024e8f90cc0"
    }
}
```

Status

- **wrong-user**: The given `user-uuid` does not exist

This call is typically used to get information about another user of the system, whose UUID is known because it shows up in some other object.

If the `deleted` field is present, this indicates that the user is no longer active in the system.

The `share_xxx` fields will contain the personal sharing tags of the other user. These fields will only be present if the current user is linked with this user.

The `picture` flag will have a value of `true` if the user has a profile picture.

GET /api/user/metadata

Get the metadata information for the current session's user.

Query Parameters

- **keys** – optional: a comma-separated list of metadata keys to get information for
- **domain** – optional: a domain to retrieve metadata for all keys under this domain

Response body example

```
{
    "metadata": {
        "org.mydomain.phone": "+492216698000",
        "org.mydomain.room": "B3"
    }
}
```

Within the `metadata` section, the requested metadata information of the user is given in form of a JSON object.

Without any URL parameters, the call will retrieve all metadata keys for current session's user. If they `keys` parameter is given, only the metadata elements for the given keys are returned. If the `domain` parameter is given, all metadata elements for keys under this domain (not including the domain as a key itself) are returned. That is, if domain is `com.mycompany`, keys like `com.mycompany.application1.element1` and `com.mycompany.entity2` will be returned, but not the key `com.mycompany`.

POST /api/user/metadata

Modify or delete metadata information associated with the current session's user.

Query Parameters

- **noevent** – optional: if present, don't generate a `user_metadata` event (see below)
- **nostore** – optional: if present, don't store the metadata permanently (see below)

Request body example

```
{
    "update": {
        "com.otherdomain.client.nice": true,
        "com.otherdomain.client.timestamp": 1324899645
    },
    "delete": [
        "com.otherdomain.client.something"
    ]
}
```

JSON Parameters

- **key** – `^[a-z0-9]{2,50}(\\.[a-z0-9]{2,50}){1,10}$`
- **value** – any valid JSON type, maximum length: 100KB

Response body example

```
{}
```

Status

- **too-large**: At least metadata value is too big (100 KB maximum size limit).

The metadata keys listed in the `update` section are updated in the database (or added if they don not currently exist).

The metadata keys listed in the `delete` section are deleted from the database.

To avoid conflicts between different applications that might otherwise use the same metadata keys, metadata keys must have the form of a reverse Internet domain, possibly extended with customer-chose domain elements. Applications are encouraged to use officially registered domains for the first part of their metadata key, like `com.mycompany.application1.element1`.

By default, the modification of the user's metadata will generate a `user_metadata` event that is sent towards all sessions currently active for the user. This behavior can be suppressed by including the `noevent` parameter in the request URL.

If the request URL includes the `nostore` parameter, the given metadata will not be stored to the database (i.e. the contents can not be retrieved with *`GET /api/user/metadata`* afterwards). This is useful if the desired behavior is to create the `user_metadata` event only.

3.6 /api/user/picture

`POST /api/user/picture`

Set the current session user's profile picture.

Request body

```
The request body contains binary picture data in JPEG or PNG format.
```

Response body example

```
{}
```

Status

- **too-large**: The picture data is too large to be handled
- **malformed-file**: The picture data is malformed / cannot be decoded

The picture data in the request body should be given as a binary format JPEG or PNG image. The actual format is automatically determined by the system. The profile picture of the user should be square, the optimal resolution is 384x384 picture, larger resolutions are possible but will be down-scaled on the server side. The maximum acceptable picture resolution is 1500x1500 pixels.

DELETE /api/user/picture
Deletes the current session user's profile picture.

Response body example

```
{}
```

GET /api/user/(user-id)/picture/(*size*)**/**
xxx.**jpg** Get a user's profile picture

Response body example

```
The picture contents as a binary JPEG file.
```

The `user-id` portion of the call might have one of the following forms:

- a UUID of an arbitrary user of the system.
- the string "self" to get the profile picture of the current session's user.
- the string "default" to get the default profile picture for users that don't have a profile picture.

If a UUID is given in `user-id` and the user is not found, an HTTP 404 error will be returned.

If a UUID is given in `user-id` and the specified user does not have a profile picture, a temporary HTTP redirection (status code 302) will be returned pointing to the location of the default profile picture.

The `size` selector within the URI is used to select the resolution of the picture returned by the call. The following resolutions are available:

- `small` 48x48 pixels
- `medium` 96x96 pixels
- `large` 192x192 pixels
- `huge` 384x384 pixels

The `xxx` portion of the URI is an arbitrary file name to be chosen by the client.

3.7 /api/user/invitation

GET /api/user/invitation
Get user's invitations

Response body example

```
{
    "invitation": {
        "dlJdod9df2jAoemnvao4u9a": {
            "description": "CeBIT 2011, Stand 42",
            "expire": 1300217017,
            "one-time": false
        },
        "Dfvp9audpd93dtAadvdDvsD": {
            "description": "Einladung f\u00fcr Hans M\u00fcller",
            "one-time": true
        }
    },
    "out": [
        "56697e6e-4f36-11e0-be81-0024e8f90cc0"
```

```
    ],
    "in": [
        "11697e6e-4f36-11e0-be81-0024e8f90cc0"
    ]
}
```

Invitations listed within the `invitation` element are keyed by the invitation's identifier string. The `expire` field specifies the expiration date of the invitation (seconds-since-epoch, only present if an expiration date exists), and `one-time` specifies whether the invitation will immediately expire on usage (if `true`).

The `out` section lists outstanding requests that were made by the current user, i.e. the listed UUIDs are those of the invited users.

The `in` section lists requests that were made by other users to invite the current user, i.e. the listed UUIDs are those of users that originated an invitation towards the current user.

`POST /api/user/invitation`

Modify user's invitations

Request body example

```
{
    "create": [
        {
            "description": "Grillen am See",
            "expire": 2281283821,
            "one-time": true
        },
        {
            "description": "Please get in touch with me",
            "expire": 2381283821,
            "one-time": false
        }
    ],
    "update": {
        "doOsie8dOmdos9732Jdkfos": {
            "description": "Changed description",
            "one-time": true
        },
        "Dfvp9audpd93dtAadvdDvsD": {
            "expire": 2381283821,
            "one-time": false
        }
    },
    "delete": [
        "dlJdod9df2jAoemnvao4u9a"
    ],
    "invite": [
        "56697e6e-4f36-11e0-be81-0024e8f90cc0"
    ],
    "refuse": [
        "11697e6e-4f36-11e0-be81-0024e8f90cc0"
    ]
}
```

JSON Parameters

- **`description`** – `^.{,100}$`

- **expire** – 32-bit integer
- **one-time** – true|false

Response body example

```
{
    "create": [
        "dfijaijADFVLIaosjd34342",
        "aFbouajfASDERGa49ja94wd"
    ]
}
```

Status

- **wrong-invitation**: An invitation in the `update` or `delete` or `refuse` section does not exist.
- **access-denied**: An invitation in the `update` or `delete` section is not owned by the current user.
- **wrong-user**: A user listed in the `invite` section does not exist or is deleted.
- **already-link**: A link already exists towards a user listed in the `invite` section.
- **already-invited**: An invitation to or from a user listed in the `invite` section already exists.

The invitations in the `create` section of the request are added to the user's invitation data. These refer to so-called *external* invitations, i.e. persons that are not yet users of the system. For these invitations, `description` and `one-time` are mandatory fields, `expire` is optional.

Data in the `update` section of the request body is modified in the current user's invitation data, all fields are optional (only the fields given are modified).

The invitations in the `delete` section are deleted from the user's invitation data.

The `invite` section lists UUIDs of other users to be invited via so-called *internal* invitations.

An internal invitation can be deleted by both the initiator and the recipient of the invitation by listing the other party's UUID in the refuse section.

The `create` section of the response lists the identifier strings of the newly created external invitations.

GET /api/invitation/(*invitation*)
Get information about (usually other user's) external invitation `invitation`

Response body example

```
{
    "description": "Please get in touch with me",
    "user": {
        "username": "johndoe47",
        "firstname": "John",
        "lastname": "Doe",
        "display_name": "Johnny Doe"
    }
}
```

Status

- **wrong-invitation**: The given `invitation` identifier is wrong/unknown

This API call is used to "pre-view" an invitation received from another user, identified by its invitation identifier string `invitation`. The response reveals the description of the invitation and the name of the user that issued the invitation.

To actually confirm the invitation and to create a user link, use *`POST /api/user/link`*.

3.8 /api/user/link

`GET /api/user/link`
Get the user's links

Response body example


```
{
    "link": {
        "59f43914-4f3d-11e0-ac91-0024e8f90cc0": {
            "username": "ben",
            "firstname": "Ben",
            "lastname": "Miller",
            "display_name": "Ben Miller",
            "picture": true,
            "share_read": "9638badc-ab55-11e2-a825-0024e8f90cc0",
            "share_write": "a192a9e2-ab55-11e2-b215-0024e8f90cc0",
            "share_propagate": "a8c7aadc-ab55-11e2-a55f-0024e8f90cc0"
        },
        "755c97aa-4f3d-11e0-855e-0024e8f90cc0": {
            "username": "hacker3",
            "firstname": "Steven",
            "lastname": "Kerner",
            "display_name": "Steve Kerner",
            "deleted": true,
            "picture": false,
            "share_read": "364f5c06-ab56-11e2-8b53-0024e8f90cc0",
            "share_write": "3a4f7d68-ab56-11e2-bcf1-0024e8f90cc0",
            "share_propagate": "3d83d772-ab56-11e2-be2a-0024e8f90cc0"
        }
    },
    "know": {
        "bcf23c7c-e7b3-11e1-9de6-0024e8f90cc0": {
            "username": "lg",
            "firstname": "Stefani",
            "lastname": "Germanotta",
            "display_name": "Lady Gaga",
            "picture": true
        },
        "be82b4c2-e7b3-11e1-9b12-0024e8f90cc0": {
            "username": "sme",
            "firstname": "Sid",
            "lastname": "Meyer",
            "display_name": "Sid Meyer",
            "picture": false
        }
    }
}
```


The `link` section contains the links of the current user, the `know` sections lists users that the current user might know according to relationships in tags or other system objects. Users in the `know` section are typical candidates

for future links of the current user.

Users who have the `deleted` field are no longer active in the system. Users listed within the `link` and `know` sections are keyed by the linked user's UUID.

The `link` section lists the personal share tags of the users. These can be used to share objects with those users (*`POST /api/object/(uuid)/tag`*).

The `picture` flag will have a value of `true` if the corresponding user has a profile picture.

`POST /api/user/link`
Modify the user's links

Request body

```
{
    "invitation": [
        "aWdijv84AdivjaRia9dRba3"
    ],
    "accept": [
        "56697e6e-4f36-11e0-be81-0024e8f90cc0"
    ],
    "delete": [
        "755c97aa-4f3d-11e0-855e-0024e8f90cc0"
    ]
}
```

Response body example

```
{}
```

Status

- **wrong-invitation**: There was an invitation identifier string in the `invitation` section or a wrong user in the `accept` section of the request which is wrong/unknown
- **wrong-link**: There was a request to delete a link in the `delete` section which does not exist
- **already-link**: There was an invitation identifier string in the `invitation` section of the request which points to a user that the current user already has an established link with

The `invitation` section lists invitation identifier strings that are accepted to create new user links in the process of this API command.

The `accept` section lists UUID's of other users whose internal invitation are accepted.

The `delete` section specifies UUID's of other users, whose already established links to the current session's user will be abandoned.

3.9 /api/phone

`GET /api/phone`
Get a list of (system-managed) phone numbers associated with current user

Response body example

```
{
    "phone": {
        "+492216698712": {
            "announcement": {
                "uuid": "1de7257a-4f34-11e0-ab6e-0024e8f90cc0",
                "description": "John's personal announcement"
            },
            "dialplan_enable": false,
            "call_enable": false,
            "fax_enable": true,
            "recording_enable": true,
            "play_announcement_only": false,
            "conference_enable": true,
            "notification_email": true,
            "notification_attachment": true,
            "on_hold_music": null
        },
        "+492211110004": {
            "dialplan_enable": true,
            "call_enable": false,
            "fax_enable": false,
            "recording_enable": false,
            "play_announcement_only": false,
            "conference_enable": false,
            "notification_email": true,
            "notification_attachment": false,
            "partner": "mypartner",
            "on_hold_music": null
        },
        "+492118888888": {
            "dialplan_enable": false,
            "call_enable": false,
            "fax_enable": true,
            "recording_enable": true,
            "play_announcement_only": false,
            "conference_enable": true,
            "notification_email": false,
            "notification_attachment": false,
            "unlock_code": "7649"
            "on_hold_music": null
        },
        "+492118888899": {
            "dialplan_enable": false,
            "call_enable": false,
            "fax_enable": false,
            "recording_enable": false,
            "play_announcement_only": false,
            "conference_enable": false,
            "notification_email": false,
            "notification_attachment": false,
            "ivr": "ce19884c-8fed-11e2-93fe-0024e8f90cc2"
            "on_hold_music": null
        },
        "+492118888999": {
            "dialplan": "215a5977-7835-47ed-8795-e9da0540074c",
            "dialplan_enable": false,
            "call_enable": false,
            "fax_enable": false,
```

```
            "recording_enable": false,
            "play_announcement_only": false,
            "conference_enable": false,
            "notification_email": false,
            "notification_attachment": false,
            "notification_email": false,
            "notification_attachment": false,
            "on_hold_music": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
        }
    }
}
```

The `phone` section lists the phone numbers associated with the current session's user.

The `announcement` element exists if an associated announcement is present.

The `partner` element exists if the phone number is controlled by the partner.

The `ivr` field points to an "*Interactive Voice Response API*" object that is associated with this phone number.

If an `unlock_code` field is present, it signals that the corresponding number is reserved, but still inactive. It must be unlocked by calling the number and entering the `unlock_code` (DTMF).

The `on_hold_music` field is either `null` or the UUID of an announcement that should be used as the On-Hold-Music for this number.

`POST /api/phone`

Modify current user's phone configuration

Request body example

```
{
    "create": true,
    "update": {
        "+492216698712": {
            "announcement": "1de7257a-4f34-11e0-ab6e-0024e8f90cc0",
            "conference_enable": false,
            "notification_email": true,
            "notification_attachment": true
        },
        "+491796548354": {
            "announcement": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
            "dialplan_enable": false,
            "call_enable": false,
            "fax_enable": true,
            "recording_enable": true,
            "play_announcement_only": true,
            "conference_enable": false,
            "notification_attachment": false,
            "ivr": "ce19884c-8fed-11e2-93fe-0024e8f90cc2"
        },
        "+492118888899": {
            "dialplan": "215a5977-7835-47ed-8795-e9da0540074c",
            "notification_email": false,
            "notification_attachment": false
            "on_hold_music": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1"
        }
    },
    "delete": [
        "+492211110004",
```

```
        "+492211110006"
    ]
}
```

JSON Parameters

- **phonenumber** – `^\\+[0-9]{3,20}$`
- **ivr** – `UUID | null`
  - `UUID` - associate this IVR to the phone
  - `null` - remove the IVR field from the phone

Response body example

```
{
    "+492118888888": {
        "unlock_code": "7649"
    }
}
```

Status

- **pool-depleted**: There was a request to create a new phone number, but there are no more numbers available in the user's area code
- **address-missing**: There was a request to create a new phone number, but there were no verified address information entered (*`POST /api/user`* & *`POST /api/user/idcard`*) .
- **phone-limit**: There was a request to create a new phone number, but the maximum amount of numbers for this users has been reached.
- **wrong-phone**: The phone number referenced in the update section does not belong to the user.
- **wrong-announcement**: The given announcement does not exist or is not accessible by the current user
- **wrong-dialplan**: The given dialplan does not exist or is not accessible by the current user
- **concurrent-access**: There is another request to create a new phone number running in parallel
- **phone-forbidden**: This user has no rights to create new phone numbers

A new number is added to the user's list of system-maintained phone numbers and will be automatically chosen from a internal pool of available phone numbers for the users `area_code` if `create` is `true`. Numbers can only be requested after a valid address is entered with *`POST /api/user`*. The returned number is reserved for the user but inactive. In order to activate the number, it must be called and the `unlock_code` must be entered via DTMF.

The phone numbers in the `update` section of the request body are modified in the current user's phone data. If no announcement is given in this section, an existing announcement binding is deleted.

The phone numbers in the `delete` section are deleted from the user's phone data.

The `dialplan_enable`, `call_enable`, `fax_enable`, `conference_enable` and `recording_enable` elements in the create and update sections are all optional (`dialplan_enable` and `call_enable` default to `false`, all other switches default to `true`).

Any combination of the `*_enable` switches is possible; however `dialplan_enable` takes priority, i.e. if `dialplan_enable` is `true`, the setting of `call_enable`, `fax_enable`, `conference_enable` and `recording_enable` is ignored for that number. The call API (`call_enable`) will only work if `call_enable` is the only switch that is `true`.

The `dialplan` field enables the "Dialplan v2" feature for this number. Setting the field to `null` will disable the feature, setting it to the UUID of a "Dialplan v2" object will enable it. This field has priority over all other phone flags. See *POST /api/dialplan/v2* for further information.

The `ivr` is activated only if `call_enable` and `dialplan_enable` are disabled.

The `play_announcement_only` field is optional and defaults to `false`. If set to `true`, the system will not record a voice message after the announcement has been played in voice recorder mode.

The `notification_email` and `notification_attachment` elements in the create and update sections are optional (default `false`).

For a given user, only one request to create a new phone number might be running in parallel. If multiple requests are issued at the same time, some will return a `concurrent-access` status.

The `on_hold_music` field should be either `null` or the UUID of an announcement that will be used as the On-Hold-Music for this number.

3.10 /api/tag

GET /api/tag
Get list of tags associated with current user

Response body example


```
{
    "tag": {
        "1de7257a-4f34-11e0-ab6e-0024e8f90cc0": {
            "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
            "label": "private",
            "foreign_use": true,
            "policy": {
                "1de7257a-4f34-11e0-ab6e-0024e8f90cc1": 4,
                "1d333333-4f34-11e0-ab6e-0024e8f90cc1": 2,
                "1d444444-4f34-11e0-ab6e-0024e8f90cc1": 3
            }
        },
        "1de1227a-4f34-11e0-ab6e-0024e8f90cc1": {
            "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
            "label": "Q-loud Team",
            "foreign_use": false,
            "policy": {
                "1de7257a-4f34-11e0-ab6e-0024e8f90cc1": 4,
                "1d333333-4f34-11e0-ab6e-0024e8f90cc1": 1,
                "1d444444-4f34-11e0-ab6e-0024e8f90cc1": 1
            }
        }
    }
}
```


The `tag` section lists all tags that are visible for the current session's user. This includes tags that are owned by the user and tags from other user's where the current user has some sort of access privileges. The result does not include the user's system tags and the user's share tags.

POST /api/tag

Create a new tag for the current user.

Request body example

```
{
    "label": "Project X",
    "foreign_use": true,
    "policy": {
        "1de1227a-4f34-11e0-ab6e-0024e8f90cc1": 1,
        "1d333333-4f34-11e0-ab6e-0024e8f90cc1": 2,
        "1d444444-4f34-11e0-ab6e-0024e8f90cc1": 3
    }
}
```

JSON Parameters

- **label** – `^.{1,50}$`

  Text label of this tag

Response body example

```
{
    "uuid": "1de7257a-4f34-11e0-ab6e-0024e8f90cc0"
}
```

Status

- **wrong-user**: One of the given user UUIDs is unknown or has no link with the current user

In the `policy` section, a mapping of other users UUIDs to access privileges is defined.

The policy levels are:

- `1` (read): user has read access to objects tagged with this tag
- `2` (write): user has read and write access to objects tagged with this tag
- `3` (propagate): user has read and write access to objects tagged with this tag, and may re-tag the object, possibly granting access rights to other users
- `4` (owner): user is the owner of that tag, and has full permissions on any object carrying the tag (in respect to the object, this is equivalent to policy level 3)

The API call returns the UUID of the created tag.

GET /api/tag/(*uuid*)

Get information for tag `uuid`

Response body example

```
{
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "label": "Project X",
    "foreign_use": true,
    "policy": {
        "1de1227a-4f34-11e0-ab6e-0024e8f90cc1": 1,
        "1d333333-4f34-11e0-ab6e-0024e8f90cc1": 2,
        "1d444444-4f34-11e0-ab6e-0024e8f90cc1": 3
```

```
    }
}
```

Status

- **wrong-tag**: The given UUID does not match to a tag accessible by the current user

This API cannot be used to request detail information for one of the user's system tags or share tags (because `label`, `foreign_use` and `policy` do not make sense in the context of those tags).

POST /api/tag/ (*uuid*)
Modify tag `uuid`

Request body example

```
{
    "label": "Project X",
    "foreign_use": false,
    "policy": {
        "update": {
            "1de1227a-4f34-11e0-ab6e-0024e8f90cc1": 3,
            "1d333333-4f34-11e0-ab6e-0024e8f90cc1": 2
        },
        "delete": [
            "1d444444-4f34-11e0-ab6e-0024e8f90cc1"
        ]
    }
}
```

JSON Parameters

- **label** – `^.{1,50}$`

  Text label of this tag

Response body example

```
{}
```

Status

- **wrong-tag**: The given UUID does not match to a tag accessible by the current user
- **wrong-user**: One of the given user UUIDs is unknown or has no link with the current user
- **access-denied**: The user is not the owner of the specified tag

All fields in the request body are optional to allow a subset of fields to be updated. Users listed in the `update` section are added to or modified in the `policy` list of this tag, users listed in the `delete` section are deleted from the `policy` list.

System tags and share tags cannot be modified using this API call.

DELETE /api/tag/ (*uuid*)
Delete tag `uuid`

Response body example

```
{}
```

Status

- **wrong-tag**: The given UUID does not match to a tag accessible by the current user
- **access-denied**: The user is not the owner of the specified tag

System tags and share tags cannot be deleted using this API call.

GET /api/tag/(*uuid*)**/object**
Get objects tagged with the tag <uuid>

Query Parameters

- **filter** – optional: one of {fax, recording, announcement, contact, conference, volume, box, ipquality, presentation, sensor, xobject}
- **from** – optional: start time, in seconds-since-epoch (must have from <= to)
- **to** – optional: end time, in seconds-since-epoch (must have from <= to)
- **start** – optional: the UUID of the first object to return, exclusive, i.e. the object with an exact match UUID is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.
- **order** – optional: If set to `asc` (default), objects are returned in time ascending order (i.e. starting with from or start and ending with stop).

  If set to `desc`, objects are returned in time descending order (i.e. starting with to or start and ending with from).
- **count** – optional, default `10`: limits the number of returned objects (valid range: `1...500`)
- **tag** – optional, a comma-separated list of additional tag UUIDs that also need to be present on the objects that this query will return. This effectively forms a logical `AND` query for objects with multiple tags.
- **search** – optional, a search string containing words that must be present in the content of the objects this query will return. This feature is used to implement a full-text search within tags.

Response body example

```
{
    "object": {
        "550e8400-e29b-41d4-a716-446655440000": {
            "type": "recording",
            "owner": "550e8400-e29b-41d4-a716-446655440000",
            "time": 24532354234,
            "tag": [
                "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
                "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
            ],
            "description": "Ein Anruf von Herrn Mueller",
            "from": "+492421848484",
            "to": "+492216698123"
        },
        "550e8421-e29b-41d4-a716-446655440001": {
```

```
            "type": "fax",
            "owner": "550e8400-e29b-41d4-a716-446655440000",
            "time": 21232145273,
            "tag": [
                "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0",
                "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
                "97d3a606-a961-11e0-9a43-0024e8f90cc0"
            ],
            "description": "Versicherungsvertrag Hausrat",
            "page_count": 12
        },
        "12322321-e29b-41d4-a716-446655440001": {
            "type": "contact",
            "owner": "550e8400-e29b-41d4-a716-446655440000",
            "time": 22634544573,
            "tag": [
                "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0"
            ],
            "firstname": "Dirk",
            "lastname": "Mueller"
        },
        "6546565-e29b-41d4-a716-446655440001": {
            "type": "announcement",
            "owner": "550e8400-e29b-41d4-a716-446655440000",
            "time": 22436234323,
            "tag": [
                "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
                "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0",
                "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
                "97d3a606-a961-11e0-9a43-0024e8f90cc0",
                "9e7c3b80-a961-11e0-b40e-0024e8f90cc0"
            ],
            "description": "Meine Standardansage"
        },
        "c1497ba2-926c-11e0-973a-0024e8f90cc0": {
            "type": "conference",
            "owner": "550e8400-e29b-41d4-a716-446655440000",
            "time": 22436292746,
            "tag": [
                "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
                "97d3a606-a961-11e0-9a43-0024e8f90cc0",
                "9e7c3b80-a961-11e0-b40e-0024e8f90cc0"
            ],
            "description": "Unser t\u00e4gliches Kaffeekr\u00e4nzchen",
            "active": true,
            "pin": "4711"
        }
    }
}
```


Status

- **wrong-tag**: The given UUID does not match to tag accessible by the current user

3.11 /api/trash

DELETE /api/trash
Purge the trash. All objects in the trash will be permanently deleted.

Response body example

```
{}
```

3.12 /api/object

POST /api/object/(*uuid*)**/tag**
Modify tag attachments of object `uuid`

Request body example

```
{
    "add": [
        "3da40b94-591b-11e0-9d92-0024e8f90cc0",
        "43bd397e-591b-11e0-a47e-0024e8f90cc0"
    ],
    "delete": [
        "4d4f6606-591b-11e0-81f3-0024e8f90cc0",
        "53190e02-591b-11e0-be0c-0024e8f90cc0"
    ]
}
```

Response body example

```
{}
```

Status

- **wrong-object**: The given UUID does not match to an object accessible by the current user
- **wrong-tag**: One of the given UUIDs does not match a tag accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the object (not the owner and no propagate rights, or the object is in trash and the request operation does something else than remove the trash tag)
- **already-tag**: A tag listed in the `add` section is already attached to the object
- **no-tag**: A tag listed in the `delete` section is not attached to the object

Tags listed in the `add` section are attached to the object, those listed in the `delete` section are detached from it.

If an object has the trash tag, then the only valid tag modification is the deletion of the trash tag. This operation effectively "undeletes" the object.

GET /api/object/(*uuid*)**/comment**
Get comments of object `uuid`

Query Parameters

- **from** – optional: start time, in seconds-since-epoch (must have from <= to)

- **to** – optional: end time, in seconds-since-epoch (must have from <= to)
- **start** – optional: the UUID of the first object to return, exclusive, i.e. the comment with an exact match UUID is not returned. If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.
- **order** – optional: If set to `asc` (default), objects are returned in time ascending order (i.e. starting with from or start and endin If set to `desc`, objects are returned in time descending order (i.e. starting with to or start and ending with from).
- **count** – optional, default `10`: limits the number of returned objects (valid range: `1...500`)

Response body example

```
{
    "comment": {
        "36b7bbf4-591c-11e0-a7e3-0024e8f90cc0": {
            "owner": "550e8400-e29b-41d4-a716-446655440000",
            "time": 2598392983,
            "text": "This is a comment"
        },
        "54512f9c-591c-11e0-8085-0024e8f90cc0": {
            "owner": "550e8400-e29b-41d4-a716-446655440000",
            "time": 2498234333,
            "text": "Another comment"
        }
    }
}
```

Status

- **wrong-object**: The given UUID does not match to an object accessible by the current user

Comments in the `comment` section are keyed by their UUID and appear ordered with ascending creating time.

POST /api/object/(*uuid*)**/comment**

Create a comment for the object `uuid`

Request body example

```
{
    "text": "This is just another comment"
}
```

JSON Parameters

- **text** – `^.{1,1000}$`

Response body example

```
{
    "uuid": "550e8400-e29b-41d4-a716-446655440001"
}
```

Status

- **wrong-object**: The given UUID does not match to an object accessible by the current user

The call returns the UUID of the created comment.

POST /api/comment/(*uuid*)
Modify the comment `uuid`

Request body example

```
{
    "text": "This is just another comment"
}
```

JSON Parameters

- **text** – `^.{1,1000}$`

Response body example

```
{}
```

Status

- **wrong-comment**: The given UUID does not match to a comment accessible by the current user
- **access-denied**: The user is not the owner of the specified comment

DELETE /api/comment/(*uuid*)
Delete the comment <uuid>

Response body example

```
{}
```

Status

- **wrong-comment**: The given UUID does not match to a comment accessible by the current user
- **access-denied**: The user is neither the owner of the specified comment nor the owner of the object that this comment refers to

GET /api/object/(*uuid*)**/link**
Get all objects that are linked to object (uuid)

Response body example

```
{
    "link": {
        "6546565-e29b-41d4-a716-446655440001": {
            "type": "announcement"
        },
        "c1497ba2-926c-11e0-973a-0024e8f90cc0": {
            "type": "conference"
        }
    }
}
```

Status

- **wrong-object**: The given <uuid> does not match to an object accessible by the current user

The result will only contain objects that the current user has at least read access for.

POST /api/object/(*uuid*)**/link/(other-uuid)**
Create a connection between object `uuid` and object `other-uuid`

Request body example

```
{}
```

Response body example

```
{}
```

Status

- **wrong-object**: At least one of the given `uuid` or `other-uuid` do not match to an object accessible by the current user
- **access-denied**: The user does not have sufficient privileges on at least one of the given objects `uuid` and `other-uuid` (at least write privileges are required)

To create a link between to objects, write privileges are needed on both objects.

DELETE /api/object/(*uuid*)**/link/(other-uuid)**
Delete the connection between object `uuid` and object `other-uuid`

Response body example

```
{}
```

Status

- **wrong-object**: The given `uuid` or `other-uuid` does not match to an object accessible by the current user
- **access-denied**: The user does not have sufficient privileges on object `uuid` (at least write privileges are required)

GET /api/object/(*uuid*)**/metadata**
Get the metadata information for object `uuid`.

Query Parameters

- **keys** – optional: a comma-separated list of metadata keys to get information for
- **domain** – optional: a domain to retrieve metadata for all keys under this domain

Response body example

```
{
    "metadata": {
        "org.mydomain.firstapp.floor": "basement",
        "org.mydomain.firstapp.room": "conference room",
        "com.otherdomain.client.nice": null,
        "com.otherdomain.client.phonelist": [
            "+492216698000",
            "+492216698999"
        ],
        "com.otherdomain.client.timestamp": 1324898273,
```

```
        "com.otherdomain.client.triggers": {
            "this": true,
            "that": false
        }
    }
}
```

Status

- **wrong-object**: The given UUID does not match an object accessible to the current user

Within the `metadata` section, the requested metadata information of the object is given in form of a JSON object.

Without any URL parameters, the call will retrieve all metadata keys for the specified object. If the `keys` parameter is given, only the metadata elements for the given keys are returned. If the `domain` parameter is given, all metadata elements for keys under this domain (not including the domain as a key itself) are returned. That is, if domain is `com.mycompany`, keys like `com.mycompany.application1.element1` and `com.mycompany.entity2` will be returned, but not the key `com.mycompany`.

POST /api/object/(*uuid*)**/metadata**

Modify or delete metadata information associated with the object `uuid`.

Query Parameters

- **noevent** – optional: if present, don't generate an `object_metadata` event (see below)
- **nostore** – optional: if present, don't store the metadata permanently (see below)

Request body example

```
{
    "update": {
        "com.otherdomain.client.nice": true,
        "com.otherdomain.client.timestamp": 1324899645
    },
    "delete": [
        "com.otherdomain.client.something"
    ]
}
```

JSON Parameters

- **key** – `^[a-z0-9]{2,50}(\\.[a-z0-9]{2,50}){1,10}$`
- **value** – any valid JSON type, maximum length: 100KB

Response body example

```
{}
```

Status

- **too-large**: At least metadata value is too big (100 KB maximum size limit)
- **wrong-object**: The given UUID does not match an object accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the object (at least write privileges are required)

The metadata keys listed in the `update` section are updated in the database (or added if they don not currently exist).

The metadata keys listed in the `delete` section are delete from the database.

To avoid conflicts between different applications that might otherwise use the same metadata keys, metadata keys must have the form of a reverse Internet domain, possibly extended with customer-chose domain elements. Applications are encouraged to use officially registered domains for the first part of their metadata key, like `com.mycompany.application1.element1`.

By default, the modification of the object's metadata will generate a `object_metadata` event that is sent towards all users that have access to the object. This behavior can be suppressed by including the `noevent` parameter in the request URL.

If the request URL includes the `nostore` parameter, the given metadata will not be stored to the database (i.e. the contents can not be retrieved with *GET /api/object/(uuid)/metadata* afterwards). This is useful if the desired behavior is to create the `object_metadata` event only.

3.13 /api/fax

POST /api/fax

Create a new fax object

Request body example

```
{
    "description": "A description text"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this fax

Response body example

```
{
    "uuid": "ca9a612c-591d-11e0-a7c8-0024e8f90cc0"
}
```

The UUID of the new fax is returned.

GET /api/fax/ (*uuid*)

Get information about fax `uuid`

Response body example

```
{
    "fax": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145273,
        "description": "Versicherungsvertrag Hausrat",
        "page_count": 12,
        "x_res": 203,
        "y_res": 98,
        "status": "ok",
        "tag": [
```

```
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ],
        "report": {
            "7b149600-5921-11e0-b85f-0024e8f90cc0": {
                "incoming": true,
                "from": "+492216689711",
                "to": "+492216689712",
                "time_start": 2128383234,
                "time_end": 2128384351,
                "status": "ok",
                "sip_status_code": 200,
                "fax_error_code": 0
            },
            "882b6418-5921-11e0-bb15-0024e8f90cc0": {
                "incoming": false,
                "from": "+492216689712",
                "to": "+4925328372322",
                "time_start": 2128383234,
                "status": "transmit"
            }
        }
    }
}
```


Status

- **wrong-fax**: The given UUID does not match a fax accessible by the current user

The `status` field in the fax section can have the following values:

- `record`: Fax is being received. No contents are present yet.
- `ok`: Fax contents available
- `convert`: Fax is being converted (upload). No contents are present yet.
- `empty`: Fax has no contents available
- `fail`: Fax or fax upload has failed, no contents available

The `page_count`, `x_res` and `y_res` fields will not show if the fax doesn't (yet) have a content. In the report section, the following `status` fields are possible:

- `dial`: The destination is dialled (outgoing fax only)
- `transmit`: The fax is transmitting
- `ok`: The fax was transmitted successfully
- `fail`: Fax transmission failed
- `cancel`: Fax transmission was cancelled by the user (outgoing fax only)
- `busy`: Fax transmission failed because opposite site is busy (outgoing fax only)
- `error`: Fax cannot be delivered because of an internal error

The `sip_status_code` field is optional and, if present, delivers information about the status of the underlying telephone call. Likewise, the `fax_error_code`, if present, gives detailed information about the T.38 fax transaction.

Depending on the `status`, the `time_end` field in the `report` section may or may not be present.

The `report` section is keyed by the fax report UUID and ordered by time.

POST /api/fax/ (*uuid*)
Modify fax `uuid`

Request body example

```
{
    "description": "A description text"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  a short description of this fax

Response body example

```
{}
```

Status

- **wrong-fax**: The given UUID does not match a fax accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the fax (at least write privileges are required)

DELETE /api/fax/ (*uuid*)
Delete the fax <uuid>

Query Parameters

- **purge** – optional: if present, delete this fax permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-fax**: The given UUID does not match a fax accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the fax (at least write privileges are required)

GET /api/fax/ (*uuid*) **/**
page/*xxx*`.png` Get contents of page `page` of fax `uuid` in PNG format

Response body example

```
The contents of page <page> of the fax in PNG format on success, or
HTTP 404 without response body on failure.
```

`xxx` is a file name arbitrarily chosen by the user agent.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the fax `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

GET /api/fax/ (*uuid*) **/**
xxx`.pdf` Get contents of fax `uuid` in PDF format

Query Parameters

- **inline** – optional: if present, the response will include an HTTP `Content-Disposition` header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The contents of the fax in PDF format on success, or HTTP 404 without
response body on failure.
```

`xxx` is a file name arbitrarily chosen by the user agent.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the fax `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

POST /api/fax/ (*uuid*) **/**
xxx.**pdf** Post fax `uuid` contents in PDF format (`xxx` is an arbitrary name)

Query Parameters

- **orientation** – optional: if set to landscape, the contents will be rotated 90 degrees before being rendered into a fax
- **mode** – optional: if set to photo, the document will be rastered instead of simply being converted to black and white

Request body

The fax contents in PDF format.

Response body example

```
{}
```

Status

- **wrong-fax**: The given UUID does not match a fax accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the fax (at least write privileges are required)
- **wrong-state**: The contents of this fax cannot be changed, because it is an incoming fax or because it was already transmitted to some destination
- **too-large**: The contents are too large to be handled by the system
- **malformed-file**: The content is malformed.

`xxx` is a file name arbitrarily chosen by the user agent. After the contents are received, the PDF file is converted and the contents of the fax are updated.

GET /api/fax/ (*uuid*) **/**
xxx.**tif** Get contents of fax `uuid` in TIF format

Query Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The contents of the fax in TIF format on success, or HTTP 404 without
response body on failure.
```

`xxx` is a file name arbitrarily chosen by the user agent.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the fax `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

POST /api/fax/(*uuid*)**/send**

Send the fax `uuid` out to a destination (i.e., queue fax transmission)

Request body example

```
{
    "from": "+492216698123",
    "to": "+492216698711"
}
```

JSON Parameters

- **from** – `^\+[0-9]{3,20}$`
- **to** – `^\+[0-9]{3,20}$`
- **fax_ident** – `^[\+ 0-9]{0,20}$`

  Optional; the fax transmitting station ID (TSI)
- **fax_header** – `^.{0,40}$`

  Optional; a fax header line printed to the top of the page

Response body example

```
{
    "uuid": "b2f2a27e-5921-11e0-b923-0024e8f90cc0"
}
```

Status

- **wrong-fax**: The given UUID does not match a fax accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the fax (at least write privileges are required)
- **wrong-phone**: The phone number in the `from` field does not match one of the user's phone numbers in the system
- **locked-phone**: The phone number in the `from` field has not yet been unlocked
- **quota-exceeded**: The send operation would exceed the user's quota
- **no-content**: The fax does not have any content to be sent
- **forbidden-to**: Invalid attempt to send to a "expensive" or "premium" number using a "free" user account

The call returns the UUID of the outgoing fax report.

GET /api/fax/(fax-uuid)/report/(report-uuid)/(*xxx*)**.pdf**

Get a fax sender report in PDF format for the given fax `fax-uuid` and faxreport `report-uuid`.

Query Parameters

- **inline** – optional: if present, the response will include an HTTP `Content-Disposition` header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The contents of the fax report in PDF format on success, or HTTP 404
without response body on failure.
```

`xxx` is a file name arbitrarily chosen by the user agent.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the fax report `report-uuid`. For more information on file download authentication, see *Authentication for file downloads*.

3.14 /api/recording

GET /api/recording/ (*uuid*)
Get information about recording `uuid`

Response body example

```
{
    "recording": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145273,
        "description": "Anruf Herr Mueller",
        "from": "+492216689712",
        "to": "+492216689711",
        "status": "ok",
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```

Status

- **wrong-recording**: The given UUID does not match a recording accessible by the current user

In the `recording` section, the following `status` fields are possible:

- `record`: The recording is just being recorded (in progress). No contents are present yet.
- `ok`: Recording was successful, contents available
- `convert`: Recording is being converted (upload). No contents are present yet.
- `empty`: Recording was successful, only silence detected, no contents available
- `fail`: Recording failed for some reason, no contents available

POST /api/recording/ (*uuid*)
Modify recording `uuid`

Request body example

```
{
    "description": "A description text"
}
```

JSON Parameters

- **description** – ^.{0,200}$

  a short description of this recording

Response body example

```
{}
```

Status

- **wrong-recording**: The given UUID does not match a recording accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the recording (at least write privileges are required)

DELETE /api/recording/ (*uuid*)
Delete recording uuid

Query Parameters

- **purge** – optional: if present, delete this recording permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-recording**: The given UUID does not match a recording accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the recording (at least write privileges are required)

GET /api/recording/ (*uuid*) **/**
xxx.**wav** Get the recording uuid contents in WAV format (xxx is an arbitrary name)

Query Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The recording in WAV format on success, or HTTP 404 without response
body on failure.
```

This API is eligible for use with the session's download id (did). In this case, the object UUID is the recording uuid. For more information on file download authentication, see *Authentication for file downloads*.

GET /api/recording/ (*uuid*) **/**
xxx.**ogg** Get the recording uuid contents in Ogg Vorbis format (xxx is an arbitrary name)

Query Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The recording in Ogg Vorbis format on success, or HTTP 404 without
response body on failure.
```

This API is eligible for use with the session's download id (did). In this case, the object UUID is the recording `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

3.15 /api/announcement

GET /api/announcement/ (*uuid*)
Get information about announcement `uuid`

Response body example

```
{
    "announcement": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145273,
        "description": "Meine Standardansage",
        "status": "ok",
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```

Status

- **wrong-announcement**: The given UUID does not match an announcement accessible by the current user

In the `announcement` section, the following `status` fields are possible:

- `record`: The announcement is just being recorded (in progress). No contents are present yet.
- `ok`: Announcement was successful, contents available
- `convert`: Announcement is being converted (upload). No contents are present yet.
- `empty`: Announcement was successful, only silence detected, no contents available
- `fail`: Announcement failed for some reason, no contents available

POST /api/announcement/ (*uuid*)
Modify announcemet `uuid`

Request body example

```
{
    "description": "A description text"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this announcement

Response body example

```
{}
```

Status

- **wrong-announcement**: The given UUID does not match an announcement accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the announcement (at least write privileges are required)

DELETE /api/announcement/ (*uuid*)
Delete announcement `uuid`

Query Parameters

- **purge** – optional: if present, delete this announcement permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-announcement**: The given UUID does not match an announcement accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the announcement (at least write privileges are required)

GET /api/announcement/ (*uuid*) **/**
xxx`.wav` Get the announcement `uuid` contents in WAV format (`xxx` is an arbitrary name)

Query Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The announcement in WAV format on success, or HTTP 404 without response
body on failure.
```

This API is eligible for use with the session's download id (did). In this case, the object UUID is the announcement <uuid>. For more information on file download authentication, see *Authentication for file downloads*.

GET /api/announcement/ (*uuid*) **/**
xxx`.ogg` Get the announcement <uuid> contents in Ogg Vorbis format (<xxx> is an arbitrary name)

Query Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The announcement in Ogg Vorbis format on success, or HTTP 404 without
response body on failure.
```

This API is eligible for use with the session's download id (did). In this case, the object UUID is the announcement <uuid>. For more information on file download authentication, see *Authentication for file downloads*.

POST /api/announcement/(*uuid*)**/**

xxx.**wav** Upload the announcement `uuid` in WAV format

Request body

```
The announcement contents in WAV format
```

Response body example

```
{}
```

Status

- **wrong-announcement**: The given UUID does not match an announcement accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the announcement (at least write privileges are required)
- **too-large**: The contents are too large to be handled by the system
- **malformed-file**: The content is malformed.

`xxx` is a file name arbitrarily chosen by the user agent. After the contents are received, the WAV file is converted and the contents of the announcement is updated.

POST /api/announcement/(*uuid*)**/**

xxx.**ogg** Upload the announcement `uuid` in Ogg Vorbis format

Request body

```
The announcement contents in Ogg Vorbis format
```

Response body example

```
{}
```

Status

- **wrong-announcement**: The given UUID does not match an announcement accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the announcement (at least write privileges are required)
- **too-large**: The contents are too large to be handled by the system
- **malformed-file**: The content is malformed.

`xxx` is a file name arbitrarily chosen by the user agent. After the contents are received, the Ogg Vorbis file is converted and the contents of the announcement is updated.

POST /api/announcement/(*uuid*)**/text2speech**

Upload the announcement `uuid` in text format.

Request body example

```
{
    "text": "Hello to Johns mailbox.",
    "locale": "en-GB"
}
```

JSON Parameters

- **text** – `^.{1,5000}$`

  the text of this announcement

- **locale** – `^(de-DE|en-GB|en-US|es-ES|fr-FR|it-IT)$`

  optional, default `de-DE`: the language of text

Response body example

```
{}
```

Status

- **wrong-announcement**: The given UUID does not match an announcement accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the announcement (at least write privileges are required)

3.16 /api/contact

POST /api/contact

Create a new contact object

Request body example

```
{
    "firstname": "Peter",
    "lastname": "Mueller"
}
```

JSON Parameters

- **firstname** – *^.{1,50}$‘*
- **lastname** – *^.{1,50}$‘*

Response body example

```
{
    "uuid": "f2e1f97c-592e-11e0-8d4a-0024e8f90cc0"
}
```

The UUID of the newly created contact object is returned.

GET /api/contact/(*uuid*)
Get information about contact `uuid`

Response body example

```
{
    "contact": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145273,
        "firstname": "Peter",
        "lastname": "Mueller",
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ],
        "entry": {
            "98b62706-9269-11e0-8ef6-0024e8f90cc0": {
                "type": "phone",
                "scope": "business",
                "value": "+492216698100"
            },
            "da084df6-9269-11e0-88e3-0024e8f90cc0": {
                "type": "email",
                "scope": "private",
                "value": "hans@wurst.de"
            },
            "b3dbdf94-9269-11e0-bf32-0024e8f90cc0": {
                "type": "other",
                "scope": "private",
                "key": "Haarfarbe",
                "value": "schwarz"
            }
        }
    }
}
```

Status

- **wrong-contact**: The given UUID does not match a contact accessible by the current user

POST /api/contact/(*uuid*)
Modify the contact `uuid`

Request body example

```
{
    "firstname": "Peter",
    "lastname": "Mueller",
    "create": [
        {
            "type": "phone",
            "scope": "business",
            "value": "+492216698100"
        }
    ],
    "update": {
        "98b62706-9269-11e0-8ef6-0024e8f90cc0": {
```

```
            "type": "phone",
            "value": "+492216698200"
        },
        "b3dbdf94-9269-11e0-bf32-0024e8f90cc0": {
            "type": "other",
            "scope": "private",
            "key": "Strandkorb-Nr.",
            "value": "42a"
        }
    },
    "delete": [
        "da084df6-9269-11e0-88e3-0024e8f90cc0"
    ]
}
```

JSON Parameters

- **firstname** – `^.{1,50}$`
- **lastname** – `^.{1,50}$`
- **type** – `^phone|mobile|fax|email|address|other$`
- **scope** – `^private|business$`
- **key** – `^.{,100}$` (only for type other)
- **value** – `^\\+[0-9]{3,20}$` (type phone, mobile, fax)

  `valid email address` (type email)

  `^.{,100}$` (type address, other)

Response body example

```
{
    "create": [
        "04056934-926c-11e0-97ef-0024e8f90cc0"
    ]
}
```

Status

- **wrong-contact**: The given UUID does not match a contact accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the contact (at least write privileges are required)

Newly created entries must have all fields `type`, `scope`, `key` (only for type `other`) and `value`.

Modified entries must have at least the fields `type`, `key` (only for type `other`) and `value`.

The `create` section of the response lists the member UUIDs for the newly created members (`create` section of the request).

DELETE /api/contact/ (*uuid*)

Modify the contact `uuid`

Query Parameters

- **purge** – optional: if present, delete this contact permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-contact**: The given UUID does not match a contact accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the contact (at least write privileges are required)

3.17 /api/conference

POST /api/conference
Create a new conference object

Request body example

```
{
    "description": "Konferenzraum f\u00fcr Team-Besprechung",
    "pin": "4711",
    "active": true,
    "mute_default": true
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this announcement

- **pin** – `^[0-9]{4}$`

  The authentication code (PIN) to access the conference

- **active** – `true|false`

  If true, participants can join the conference

- **mute_default** – `true|false`

  optional: if true, participants joining the conference are muted by default

Response body example

```
{
    "uuid": "2d4d09e0-85de-11e0-844e-0024e8f90cc0"
}
```

Status

- **duplicate-pin**: The PIN is already assigned to another conference object of this user

The UUID of the newly created conference object is returned.

GET /api/conference/(*uuid*)
Get information about conference `uuid`

Response body example

```
{
    "conference": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145373,
        "description": "Konferenzraum f\u00fcr Team-Besprechung",
        "pin": "4711",
        "active": true,
        "mute_default": false,
        "phone": [
            "+492216698000",
            "+493029819232"
        ],
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```

Status

- **wrong-conference**: The given UUID does not match a conference accessible by the current user

POST /api/conference/ (*uuid*)

Modify the conference `uuid`

Request body example

```
{
    "description": "Konferenzraum f\u00fcr Kaffeekr\u00e4nzchen",
    "pin": "4712",
    "active": false,
    "mute_default": false
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  A short description of this conference

- **pin** – `^[0-9]{4}$`

  The authentication code (PIN) to access the conference

- **active** – `true\|false`

  If true, participants can join the conference

- **mute_default** – `true\|false`

  optional: if true, participants joining the conference are muted by default

Response body example

```
{}
```

Status

- **wrong-conference**: The given UUID does not match a conference accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the conference (at least write privileges are required)
- **duplicate-pin**: The PIN is already assigned to another conference object of this user

All fields in the request are optional.

DELETE /api/conference/ (*uuid*)
Delete conference `uuid`

Query Parameters

- **purge** – optional: if present, delete this conference permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-conference**: The given UUID does not match a conference accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the conference (at least write privileges are required)

GET /api/conference/ (*uuid*) **/event**
Subscribe the current session to the event channel of the given conference

Response body example

```
{
    "busy": true
}
```

Status

- **wrong-conference**: The given UUID does not match a conference accessible by the current user

This API call will subscribe the current session to the detailed events of the given conference. The events themselves will be delivered by the *GET /api/event* interface.

If the conference has active participants, the `busy` field will be `true`, and the server is supposed to send a `conference_status` event shortly after receiving this subscription, so that the client can catch up with the current conference state.

If `busy` is `false`, then the conference does not have active participants, however the detailed event subscription still takes place, and conference events will start to flow as soon as the first participant enters the conference.

DELETE /api/conference/ (*uuid*) **/event**
Cancel the subscription of the current session to the event channel of the given conference

Response body example

```
{}
```

Status

- **wrong-conference**: The given UUID does not match a conference accessible by the current user

This API call will stop the flow of conference detail events to the current session.

POST /api/conference/ (*uuid*) **/member**

Modify conference member state

Request body example

```
{
    "update": {
        "9fe1cc2a-8ab6-11e0-b368-0024e8f90cc0": {
            "user": "2d07f7be-8abc-11e0-8166-0024e8f90cc0",
            "mute": true,
            "deaf": false,
            "volume_in": 1,
            "volume_out": -1
        },
        "b1439c46-8ab6-11e0-acdc-0024e8f90cc0": {
            "mute": false
        }
    },
    "delete": [
        "22731d12-8b6e-11e0-9f8c-0024e8f90cc0",
        "23980df6-8b6e-11e0-b616-0024e8f90cc0"
    ]
}
```

Response body example

```
{}
```

Status

- **conference-offline**: The conference is currently offline, so the action cannot be delivered
- **wrong-conference**: The given UUID does not match a conference accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the conference (at least write privileges are required)

The `update` section lists the member UUIDs whose properties are to be modified. All fields within a specific conference member are optional, only those that contain properties to be modified should be present.

The `user` field is used to associate a conference participant with a system user.

A participant can be muted (i.e., the conference cannot hear what she says) by setting the `mute` field to `true`, or made deaf (i.e., she cannot hear the conference) by setting the `deaf` field to `true`.

Volume levels for the participant can be set from `-4` to `4`, whereas `0` is the default level.

The `delete` section lists the member UUIDs to be kicked from the conference.

3.18 /api/volume

POST /api/volume
Create a new volume object

Request body example

```
{
    "description": "files for project X"
}
```

JSON Parameters

- **description** – ^.{0,200}$

  optional: a short description of this volume

Response body example

```
{
    "uuid": "2d4d09e0-85de-11e0-844e-0024e8f90cc0"
}
```

The UUID of the newly created volume object is returned.

GET /api/volume/(*uuid*)
Get information about volume `uuid`

Response body example

```
{
    "volume": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145373,
        "description": "files for project X",
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```

Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user

POST /api/volume/(*uuid*)
Modify volume `uuid`

Request body example

```
{
    "description": "files and more for project Y"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  A short description of this volume

Response body example

```
{}
```

Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)

All fields in the request are optional.

DELETE /api/volume/ (*uuid*)

Delete volume `uuid`

Query Parameters

- **purge** – optional: if present, delete this volume permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)

POST /api/volume/ (*uuid*) **/folder**

Create a new folder in volume `uuid`

Request body example

```
{
    "name": "temp_files",
    "parent": "2d4d09e0-85de-11e0-844e-0024e8f90cc0",
    "ctime": 21232145273,
    "mtime": 21232145373
}
```

JSON Parameters

- **name** – `valid file name`

  The name of the folder

- **parent** – `UUID`

  The UUID of the parent folder, or, if the folder is to be created at the root level of the volume, the UUID of the volume itself

- **ctime** – Optional: the creation time of the folder in seconds-since-epoch, defaults to the current time

- **mtime** – Optional: the modification time of the folder in seconds-since-epoch, defaults to the current time

Response body example


```
{
    "uuid": "8dfb2e5e-45d5-11e1-962b-0024e8f90cc0",
    "commit": "a0801fe0-3632-11e3-9f9f-0024e8f90cc0"
}
```


Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)
- **wrong-parent**: The given parent is not a valid folder in this volume and not the volume itself
- **name-exists**: The parent folder already contains a file or folder with the given name

The uuid of the newly created folder is returned.

The commit UUID of the operation is returned in the `commit` field.

GET /api/volume/(*uuid*)**/folder/(folder-uuid)**
Get information about folder `folder-uuid` in volume `uuid`. If `folder-uuid` is the same as `uuid`, then the information about the volume's root folder is returned

Response body example


```
{
    "name": "some_documents",
    "parent": "2d4d09e0-85de-11e0-844e-0024e8f90cc0",
    "ctime": 21232145273,
    "mtime": 21232145373,
    "user": "bc93f42a-45d6-11e1-b856-0024e8f90cc0",
    "content": {
        "637e470a-45db-11e1-bb2e-0024e8f90cc0": {
            "name": "letter.pdf",
            "type": "file",
            "ctime": 20232142212,
            "mtime": 21232145373,
            "user": "bc93f42a-45d6-11e1-b856-0024e8f90cc0",
            "size": 386231,
            "mime_type": "application/pdf",
            "md5": "92f2e0728cd03376ed13a191734cc065"
        },
        "6540c04a-45db-11e1-b771-0024e8f90cc0": {
            "name": "picture.png",
            "type": "file",
            "ctime": 20232142212,
            "mtime": 21232145373,
            "user": "71f79f78-4664-11e1-b5e8-0024e8f90cc0",
            "size": 4982722,
            "mime_type": "image/png",
            "md5": "1ce642b0a3616a60be0dc5651a4abd73"
        },
        "63f0f6b0-45db-11e1-b56b-0024e8f90cc0": {
```

```
            "name": "old",
            "type": "folder",
            "ctime": 20232142212,
            "mtime": 21232145373,
            "user": "bc93f42a-45d6-11e1-b856-0024e8f90cc0"
        }
    }
}
```

Status

- **wrong-volume**: The given `uuid` does not match a volume accessible by the current user
- **wrong-folder**: The given `folder-uuid` is not a valid folder in this volume

The `parent`, `name`, `ctime`, `mtime` and `user` fields in the response body will not be present if the requested folder is the root folder of the volume. For files without content, the `size`, `mime_type`, and `md5` fields will not be present.

POST /api/volume/(*uuid*)**/folder/(folder-uuid)**

Modify folder `folder-uuid` within volume `uuid`

Request body example

```
{
    "name": "temp_files",
    "parent": "2d4d09e0-85de-11e0-844e-0024e8f90cc0",
    "ctime": 21232145273,
    "mtime": 21232145373
}
```

JSON Parameters

- **name** – `valid file name`

  The name of the folder

- **parent** – `UUID`

  The UUID of the parent folder, or, if the folder is to be moved to the root level of the volume, the UUID of the volume itself

- **ctime** – The creation time of the folder in seconds-since-epoch
- **mtime** – The modification time of the folder in seconds-since-epoch

Response body example

```
{
    "commit": "a0801fe0-3632-11e3-9f9f-0024e8f90cc0"
}
```

Status

- **wrong-volume**: The given `uuid` does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)
- **wrong-folder**: The given `folder-uuid` is not a valid folder in this volume

- **wrong-parent**: The given parent is not a valid folder in this volume and not the volume itself
- **name-exists**: The parent folder already contains a file or folder with the given name

All fields in the request are optional. If the `parent` field is given, the resulting operation is a move of the folder to a new parent folder. If the `name` field is given, the resulting operation is a rename of the folder.

In contrast to *`GET /api/volume/(uuid)/folder/(folder-uuid)`*, this operation is only valid on a sub-folder of the volume, not the volume itself (i.e., `folder-uuid` must not be equal to `uuid`).

The commit UUID of the operation is returned in the the `commit` field.

`DELETE /api/volume/`(*uuid*)**`/folder/(folder-uuid)`**

Delete folder `folder-uuid` within volume `uuid`.

Response body example

```
{
    "commit": "a0801fe0-3632-11e3-9f9f-0024e8f90cc0"
}
```

Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)
- **wrong-folder**: The given `folder-uuid` is not a valid folder in this volume and not the UUID of volume itself

This operation recursively deletes all folders and files contained in the given folder.

If `folder-uuid` is equal to the volume `uuid`, then all folders and files in the volume are deleted (but not the volume itself).

The commit UUID of the operation is returned in the the `commit` field.

`POST /api/volume/`(*uuid*)**`/file`**

Create a new file in volume `uuid`

Request body example

```
{
    "name": "picture.png",
    "parent": "2d4d09e0-85de-11e0-844e-0024e8f90cc0",
    "ctime": 21232145273,
    "mtime": 21232145373
}
```

JSON Parameters

- **`name`** – `valid file name`

  The name of the file

- **`parent`** – `UUID`

  The UUID of the parent folder, or, if the file is to be created at the root level of the volume, the UUID of the volume itself

- **ctime** – Optional: the creation time of the file in seconds-since-epoch, defaults to the current time
- **mtime** – Optional: the modification time of the file in seconds-since-epoch, defaults to the current time

Response body example

```
{
    "uuid": "8dfb2e5e-45d5-11e1-962b-0024e8f90cc0",
    "commit": "a0801fe0-3632-11e3-9f9f-0024e8f90cc0"
}
```

Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)
- **wrong-parent**: The given parent is not a valid folder in this volume and not the volume itself
- **name-exists**: The parent folder already contains a file or folder with the given name

The uuid of the newly created file is returned.

The commit UUID of the operation is returned in the the commit field.

GET /api/volume/(*uuid*)**/file/(file-uuid)**
Get information about file `file-uuid` in volume `uuid`

Response body example

```
{
    "name": "letter.pdf",
    "parent": "2d4d09e0-85de-11e0-844e-0024e8f90cc0",
    "ctime": 21232145273,
    "mtime": 21232145373,
    "user": "bc93f42a-45d6-11e1-b856-0024e8f90cc0",
    "size": 386231,
    "mime_type": "application/pdf",
    "md5": "92f2e0728cd03376ed13a191734cc065"
}
```

Status

- **wrong-volume**: The given `uuid` does not match a volume accessible by the current user
- **wrong-file**: The given `file-uuid` is not a valid file in this volume

For files without content, the `size`, `mime_type`, and `md5` fields will not be present.

POST /api/volume/(*uuid*)**/file/(file-uuid)**
Modify file `file-uuid` in volume `uuid`

Request body example

```
{
    "name": "picture.png",
    "parent": "2d4d09e0-85de-11e0-844e-0024e8f90cc0",
    "ctime": 21232145273,
    "mtime": 21232145373
}
```

JSON Parameters

- **name** – `valid file name`

  The name of the file

- **parent** – `UUID`

  The UUID of the parent folder, or, if the file is to be moved to the root level of the volume, the UUID of the volume itself

- **ctime** – The creation time of the file in seconds-since-epoch, defaults to the current time

- **mtime** – The modification time of the file in seconds-since-epoch, defaults to the current time

Response body example

```
{
    "commit": "a0801fe0-3632-11e3-9f9f-0024e8f90cc0"
}
```

Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)
- **wrong-file**: The given `file-uuid` is not a valid file in this volume
- **wrong-parent**: The given parent is not a valid folder in this volume and not the volume itself
- **name-exists**: The parent folder already contains a file or folder with the given name

All fields in the request are optional. If the `parent` field is given, the resulting operation is a move of the file to a new parent folder. If the `name` field is given, the resulting operation is a rename of the file.

The commit UUID of the operation is returned in the the `commit` field.

POST /api/volume/(*uuid*)**/file/(file-uuid)/**

xxx Upload file contents of file `file-uuid` in volume `uuid`. `xxx` is an arbitrary name chosen by the client.

Request body

```
The file contents
```

Response body example

```
{
    "size": 386231,
    "mime_type": "application/pdf",
    "md5": "92f2e0728cd03376ed13a191734cc065",
```

```
    "commit": "a0801fe0-3632-11e3-9f9f-0024e8f90cc0"
}
```

Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)
- **wrong-file**: The given `file-uuid` is not a valid file in this volume
- **quota-exceeded**: The upload operation would exceed the user's quota
- **upload-fail**: The upload operation failed

The `size`, `mime_type`, and `md5` properties of the file will be derived from the content of the upload, stored in the database and returned in the response.

The `xxx` is an arbitrary name that can be chosen by the client for convenience. It will not change or set the file's name as defined by this API.

The commit UUID of the operation is returned in the the `commit` field.

GET /api/volume/(*uuid*)**/file/(file-uuid)/**
xxx Get file contents of file `file-uuid` in volume `uuid`. `xxx` is an arbitrary name chosen by the client.

Query Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body

```
The file contents, or a HTTP error status (e.g., 404) if the file does
not exist or is not accessible.
```

The `xxx` is an arbitrary name that can be chosen by the client for convenience, for example to select the proposed "download file name" within a browser. It has no influence on which file is selected for download; the file is uniquely addressed by the given `file-uuid`.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the file's `file-uuid`. For more information on file download authentication, see *Authentication for file downloads*.

GET /api/volume/(*uuid*)**/folder/(folder-uuid)/**
xxx Get file contents of folder `folder-uuid` in volume `uuid` as a compressed ZIP file, i.e. all files and sub-folders of this folder. `xxx` is an arbitrary name chosen by the client.

Query Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment. |

Response body

```
The folder contents as a ZIP file, mime type application/zip, or a HTTP
error status (e.g., 404) if the folder does not exist or is not
accessible.
```

The `xxx` is an arbitrary name that can be chosen by the client for convenience, for example to select the proposed "download file name" within a browser (in this case, the extension ".zip" should probably be used to let the user know that the result is actually a ZIP file). It has no influence on which folder is selected for download; the folder is uniquely addressed by the given `folder-uuid`.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the folder's `folder-uuid`. For more information on file download authentication, see *Authentication for file downloads*.

DELETE /api/volume/ (*uuid*) **/file/(file-uuid)**
Delete file `file-uuid` within volume `uuid`.

Response body example

```
{
    "commit": "a0801fe0-3632-11e3-9f9f-0024e8f90cc0"
}
```

Status

- **wrong-volume**: The given UUID does not match a volume accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the volume (at least write privileges are required)
- **wrong-file**: The given `file-uuid` is not a valid file in this volume

The commit UUID of the operation is returned in the the `commit` field.

GET /api/volume/ (*uuid*) **/commit**
Get commit operations for volume `uuid`

Query Parameters

- **start** – optional: the UUID of the first commit entry to return, exclusive, i.e. the entry with an exact match UUID is not returned.
- **count** – optional, default `10`: limits the number of returned entries (valid range: `1...500`)
- **order** – optional: If set to `asc` (default), commit entries are returned in time ascending order. If set to `desc`, commit entries are returned in time descending order.

Response body example

```
{
    "commit": [
        {
            "commit": "383a5354-362f-11e3-837a-0024e8f90cc0",
            "event": "file_new",
            "time": 21232145372,
            "file": "8aaf6000-474f-11e1-b0ba-0024e8f90cc0",
            "volume": "6a8432ec-474f-11e1-8495-0024e8f90cc0",
            "type": "file",
            "name": "letter.pdf",
            "parent": "9e6bde98-474f-11e1-9a62-0024e8f90cc0",
            "ctime": 20232142212,
            "mtime": 21232145372,
            "user": "bc93f42a-45d6-11e1-b856-0024e8f90cc0"
        },
        {
            "commit": "400f9be8-362f-11e3-b868-0024e8f90cc0",
```

```
            "event": "file_modify",
            "time": 21232145373,
            "file": "8aaf6000-474f-11e1-b0ba-0024e8f90cc0",
            "volume": "6a8432ec-474f-11e1-8495-0024e8f90cc0",
            "type": "file",
            "parent": "d5edfe04-4750-11e1-be06-0024e8f90cc0",
            "ctime": 20232142212,
            "mtime": 21232145373,
            "user": "bc93f42a-45d6-11e1-b856-0024e8f90cc0",
            "size": 386231,
            "mime_type": "application/pdf",
            "md5": "92f2e0728cd03376ed13a191734cc065"
        },
        {
            "commit": "46bf0e74-362f-11e3-a637-0024e8f90cc0",
            "event": "file_delete",
            "time": 2182782840,
            "file": "8aaf6000-474f-11e1-b0ba-0024e8f90cc0",
            "volume": "6a8432ec-474f-11e1-8495-0024e8f90cc0"
        }
    ]
}
```


Status

- **wrong-volume**: The given `uuid` does not match a volume accessible by the current user

The commit log of a volume is the amount of all file events, which are stored in the system and can be retrieved with this call, sorted by time according to the order parameter.

Each file event is marked with a commit entry UUID, which serves for time-ordering and can be used as a key for paging through a large set of commit entries with this command by using the start and count parameters. To get the next batch of commit entries, use the last commit UUID as a `start` parameter for the next call of this command.

For a detailed description of the entries, refer to the description of the file events in this document (`file_new`, `file_modify`, `file_delete`). Please note that several fields in the file events are optional, for example the name field will only be present in a `file_modify` event if the file was actually renamed.

3.19 /api/box

`POST /api/box`

Create a new box object and associates the box with a physical device using the device code.

Request body example


```
{
    "description": "box in office basement",
    "code": "ABCDE-FGHIJ-KLMNI-OPQRS"
}
```


JSON Parameters

- **`description`** – `^.{0,200}$`

  optional: a short description of this volume

- **code** – `^.{10,50}$`

  the device access code, which is the key to the physical device that will be attached to the new box

Response body example

```
{
    "uuid": "63013ee2-772a-11e1-a9b8-0024e8f90cc0",
    "device": "64215bea-772a-11e1-ae6c-0024e8f90cc0"
}
```

Status

- **wrong-code**: The given code does not match an available physical device
- **duplicate-box**: The device is already linked to an existing box object

The `uuid` field returns the UUID of the newly created box object. The `device` field carries the UUID of the device that the box was linked to.

GET /api/box/(*uuid*)
Get information about box `uuid`

Response body example

```
{
    "box": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145273,
        "description": "box in office basement",
        "device": "64215bea-772a-11e1-ae6c-0024e8f90cc0",
        "hw_version": "cospace-box",
        "sw_version": "001",
        "ip_addr": "212.202.35.2",
        "mac_addr": "00:24:e8:f9:0c:c0",
        "online": true,
        "license": {
            "key": "EXAMP-LELIC-ENSEK-EY007",
            "begin_time": 1356998400,
            "expire_time": 1388534399
        },
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```

Status

- **wrong-box**: The given UUID does not match a box accessible to the current user

The `device` field carries the UUID of the device associated with the box. The `mac_addr` field contains the last known MAC address of the associated device, if any. The `ip_addr` field contains the last known IP address of the device, if any. The `online` field will have a value of `true` if the box (aka the associated device) is currently connected to the cloud.

The `license` section (which will only be present if a license is attached to the box) will contain the license key as well as the license begin and end times (in seconds-since-epoch).

POST /api/box/(*uuid*)

Modifies box `uuid`, including the possibility to change the underlying physical device.

Request body example

```
{
    "description": "new box in office basement",
    "code": "XYZKL-FGHIJ-KLMNI-OPQRS",
    "license": "EXAMP-LELIC-ENSEK-EY007"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this volume

- **code** – `^.{10,50}$`

  optional: the device access code, which is the key to the new physical device that will be attached to the new box

- **license** – `^.{10,50}$`

  optional: a license key to attach to the box

Response body example

```
{
    "device": "828bfe1c-772c-11e1-90f0-0024e8f90cc0"
}
```

Status

- **wrong-box**: The given UUID does not match a box accessible to the current user
- **duplicate-box**: The given code is already linked to an existing box object
- **access-denied**: The user does not have sufficient privileges on the box (at least write privileges are required)
- **wrong-code**: The given code does not match an available physical device
- **wrong-license**: The given license key is unknown

The `device` field carries the UUID of the new device that the box was linked to and is only present if the device association was changed during this request.

Note that once a license key is attached to a box, it cannot be deleted, only be replaced by another license key.

DELETE /api/box/(*uuid*)

Delete box object `uuid`

Response body example

```
{}
```

Status

- **wrong-box**: The given UUID does not match a box accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the box (at least write privileges are required)

The box object is permanently deleted (no trash option) and the association with the physical device is removed.

3.20 /api/ipquality

POST /api/ipquality

Create a ipquality object

Request body example

```
{
    "description": "data center rack #1",
    "source": "736f7e1c-772d-11e1-bc19-0024e8f90cc0",
    "destination": "80334d0e-772d-11e1-b374-0024e8f90cc0",
    "interval": 10,
    "duration": 5,
    "active": true,
    "dscp": 0,
    "ptime": 30,
    "psize": 100
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this object

- **source** – `UUID`

  optional: the UUID of the box that acts as a source for the quality measurements under this object

- **destination** – `UUID`

  optional: the UUID of the box that acts as a destination for the quality measurements under this object

- **interval** – `^[0-9]{1,4}$`

  the time (in seconds) between the end of a measurement and the start of the next measurement

- **duration** – `^[1-9][0-9]{1,3}$`

  the duration of a measurement (in seconds)

- **active** – `true|false`

  optional: if `true`, measurements are currently scheduled according to the interval/duration parameters. Defaults to `false`.

- **dscp** – `0..63`

  optional: IP DSCP value for the measurement, default `46`

- **ptime** – `10..1000`

  optional: packetization time (packet rate) in ms, default `20`

- **psize** – `80..2000`

  optional: IP packet size, default `200`

Response body example

```
{
    "uuid": "9a955dbc-772e-11e1-ad02-0024e8f90cc0"
}
```

Status

- **wrong-box**: The given source or destination do not match to accessible box device (at least read access is needed to the respective box object to create a measurement on that box)

The uuid of the newly created ipquality object is returned.

GET /api/ipquality/ (*uuid*)
Get information about ipquality object `uuid`

Response body example

```
{
    "ipquality": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145273,
        "description": "data center rack #1",
        "source": {
            "box": "736f7e1c-772d-11e1-bc19-0024e8f90cc0",
            "description": "box in office basement",
            "device": "64215bea-772a-11e1-ae6c-0024e8f90cc0",
            "ip_addr": "212.202.35.2",
            "mac_addr": "00:24:e8:f9:0c:c0",
            "online": true
        },
        "destination": {
            "box": "80334d0e-772d-11e1-b374-0024e8f90cc0",
            "description": "box in the woods",
            "device": "3c445074-abea-11e1-be9e-0024e8f90cc0",
            "ip_addr": "92.192.3.45",
            "mac_addr": "00:c0:e1:12:23:0a"
        },
        "interval": 10,
        "duration": 5,
        "active": true,
        "dscp": 46,
        "ptime": 20,
        "psize": 200,
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```

Status

- **wrong-ipquality**: The given UUID does not match a ipquality object accessible to the current user

The `source` and `destination` sections include information about the source and destination box objects for this ipquality measurement. Their respective description, device UUID, IP address and MAC address will always be included (if they exist), regardless if the user has access to the box or not. The online information will only be included if at least read access to the box is possible for the user, because only in this case a status change might later be recognized in form of a `box_online` / `box_offline` event.

POST /api/ipquality/ (*uuid*)

Modify ipquality object `uuid`

Request body example

```
{
    "description": "data center rack #2",
    "source": "736f7e1c-772d-11e1-bc19-0024e8f90cc0",
    "destination": "80334d0e-772d-11e1-b374-0024e8f90cc0",
    "interval": 0,
    "duration": 10,
    "active": true,
    "psize": 150
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this object

- **source** – `UUID`

  optional: the UUID of the box that acts as a source for the quality measurements under this object. To delete the source, this field must be an empty string.

- **destination** – `UUID`

  optional: the UUID of the box that acts as a destination for the quality measurements under this object. To delete the destination, this field must be an empty string.

- **interval** – `^[0-9]{1,4}$`

  optional: the time (in seconds) between the end of a measurement and the start of the next measurement

- **duration** – `^[1-9][0-9]{1,3}$`

  optional: the duration of a measurement (in seconds)

- **active** – `true|false`

  optional: if `true`, measurements are currently scheduled according to the interval/duration parameters.

- **dscp** – `0..63`

  optional: IP DSCP value for the measurement, default `46`

- **ptime** – `10..1000`

  optional: packetization time (packet rate) in ms, default `20`

- **psize** – 80..2000

  optional: IP packet size, default 200

Response body example

```
{}
```

Status

- **wrong-ipquality**: The given UUID does not match a ipquality object accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the ipquality object (at least write privileges are required)
- **wrong-box**: The given source or destination do not match to accessible box device (at least read access is needed to the respective box object to create a measurement on that box)

GET /api/ipquality/(*uuid*)**/result**
Get information about ipquality object <uuid>

Parameters

- **from** – optional: start time, in seconds-since-epoch (must have from <= to)
- **to** – optional: end time, in seconds-since-epoch (must have from <= to)
- **start** – optional: the UUID of the first result to return, exclusive, i.e. the result with an exact match UUID is not returned.

  If given, overrides the from parameter when asc order is selected or overrides the to parameter when desc order is selected.
- **order** – optional: If set to asc (default), objects are returned in time ascending order (i.e. starting with from or start and ending with stop).

  If set to desc, objects are returned in time descending order (i.e. starting with to or start and ending with from).
- **count** – optional, default 10: limits the number of returned objects (valid range: 1...1000)

Response body example


```
{
    "result": {
        "c7c1e972-7747-11e1-85ea-0024e8f90cc0": {
            "time": 2598392983,
            "duration": 10,
            "source": {
                "p_s": 500,
                "p_r": 480,
                "t_n": 475,
                "t_min": 13,
                "t_max": 56,
                "t_sum": 14970,
                "t_sq": 311875,
                "j_n": 465,
                "j_max": 34,
                "j_sum": 2413,
                "j_sq": 11625
```

```
            },
            "destination": {
                "p_s": 500,
                "p_r": 480,
                "t_n": 475,
                "t_min": 13,
                "t_max": 56,
                "t_sum": 14970,
                "t_sq": 311875,
                "j_n": 465,
                "j_max": 34,
                "j_sum": 2413,
                "j_sq": 11625
            }
        },
        "d8c42a00-7747-11e1-a938-0024e8f90cc0": {
            "time": 2598392993,
            "duration": 10,
            "source": {
                "p_s": 500,
                "p_r": 480,
                "t_n": 475,
                "t_min": 13,
                "t_max": 56,
                "t_sum": 14970,
                "t_sq": 311875,
                "j_n": 465,
                "j_max": 34,
                "j_sum": 2413,
                "j_sq": 11625,
                "l_n": 20,
                "rfac": 87
            },
            "destination": {
                "p_s": 500,
                "p_r": 480,
                "t_n": 475,
                "t_min": 13,
                "t_max": 56,
                "t_sum": 14970,
                "t_sq": 311875,
                "j_n": 465,
                "j_max": 34,
                "j_sum": 2413,
                "j_sq": 11625,
                "l_n": 18,
                "rfac": 79
            }
        }
    }
}
```


Status

- **wrong-ipquality**: The given UUID does not match a ipquality object accessible to the current user

Explanation of fields in the source / destination response:

`p_s`: number of packets sent

`p_r`: number of packets received

`p_e`: number of ICMP errors

`t_n`: number of packets valid for RTT (round-trip time) calculation

`t_min`: minimum RTT (ms)

`t_max`: maximum RTT (ms)

`t_sum`: sum of all RTT values of valid packets (ms)

`t_sq`: sum of all squared RTT values of valid packets (ms^2)

`j_n`: number of packets valid for jitter calculation

`j_max`: maximum jitter (ms)

`j_sum`: sum of all jitter values of valid packets (ms)

`j_sq`: sum of all squared jitter value of valid packets (ms^2)

`l_n`: the number of packet loss sequences

`rfac`: the R-factor quality assessment of the measurement

Note that the `l_n` parameter was introduced later and may not be present on older ipquality results. It specifies the number of gaps in the packet sequence, not the number of lost packets. For example, if `l_n` is 1, it means that all the packet loss happened in one burst error.

The `rfac` result is optional and will only be present if the parameters of the measurement allow for a quality assessment using the ITU-T E-model (R-factor).

GET /api/ipquality/(*uuid*)**/result/**
xxx`.csv` Download ipquality results in CSV format

Parameters

- **from** – start time, in seconds-since-epoch (must have from <= to)
- **to** – end time, in seconds-since-epoch (must have from <= to)
- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
time,duration,src pkts sent,src pkts rcvd,src ICMP errs,src # pkts
RTT,src min RTT,src max RTT,src sum RTT,src square sum RTT,src # pkts
jitter,src max jitter,src sum jitter,src square sum jitter,src # loss
sequences,src r-factor,dst pkts sent,dst pkts rcvd,dst ICMP errs,dst #
pkts RTT,dst min RTT,dst max RTT,dst sum RTT,dst square sum RTT,dst #
pkts jitter,dst max jitter,dst sum jitter,dst square sum jitter,dst #
loss sequences,dst r-factor
1342004417,10,500,500,0,500,0,0,0,0,499,0,0,0,,,500,500,0,499,0,0,0,0,499,0,0,0,,
1342004427,10,500,500,0,500,0,0,0,0,499,0,0,0,,,500,500,0,499,0,0,0,0,499,0,0,0,,
1342004437,10,500,500,0,500,0,0,0,0,499,0,0,0,,,500,500,0,499,0,0,0,0,499,0,0,0,,
1342004447,10,500,500,0,500,0,0,0,0,499,0,0,0,,,500,500,0,499,0,0,0,0,499,0,0,0,,
1342004457,10,500,500,0,500,0,0,0,0,499,0,0,0,,,500,500,0,499,0,0,0,0,499,0,0,0,,
1342004467,10,500,500,0,500,0,0,0,0,499,0,0,0,,,500,500,0,499,0,0,0,0,499,0,0,0,,
```

This API call allows for downloading of ipquality results in comma-separated-values format. The output is streaming, so very large results can be handled well.

The fields in the CSV output correspond to the fields in the *GET /api/ipquality/(uuid)/result* call.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the ipquality `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

DELETE /api/ipquality/(*uuid*)

Delete ipquality object `uuid`

Parameters

- **purge** – optional: if present, delete this object permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-ipquality**: The given UUID does not match a ipquality object accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the ipquality object (at least write privileges are required)

3.21 /api/sip

POST /api/sip

Create a SIP account

Request body example

```
{
    "password": "sEcReT53a",
    "description": "office phone",
    "domain": "sip.cospace.de"
}
```

JSON Parameters

- **password** – `^.{5,50}$`

  a password for the new account

- **description** – `^.{0,100}$`

  optional: a description text for the new account

- **domain** – `sip.solucon.com` or `sip.cospace.de`

  optional: the SIP domain that this account is associated with

  SIP accounts created before the 01.01.2016 are created with `sip.cospace.de` by default. SIP accounts created after that date are using `sip.solucon.com` instead.

Response body example

```
{
    "user": "joe_08",
    "domain": "sip.cospace.de"
}
```

Status

- **quota-exceeded**: The maximum number of SIP accounts for this user is reached
- **wrong-domain**: The specified `domain` is not a valid choice

The `user` and `domain` parts of the newly created SIP accounts are returned.

`GET /api/sip`
Get the user's SIP accounts

Response body example

```
{
    "sip": {
        "joe_01": {
            "domain": "sip.solucon.com",
            "password": "sEcReT53a",
            "description": "office phone",
            "contact": "212.202.0.32:5060",
            "expire": 1340107084
        },
        "joe_08": {
            "domain": "sip.cospace.de",
            "password": "sEcReT55a"
        }
    }
}
```

The `contact` and `expire` fields will only be present if there is currently an active registration by a SIP end device for the respective SIP account. `contact` contains the IP address and port of the registration origin (i.e., usually the IP address and port of the end device), `expire` contains the absolute time of registration expiration in seconds-since-epoch.

`POST /api/sip/(sip-user)`
Modify a SIP account

Request body example

```
{
    "password": "sEcReT54a",
    "description": "PC softphone",
    "domain": "sip.cospace.de"
}
```

JSON Parameters

- **`password`** – `^.{5,50}$`

  optional: a password for the new account

- **`description`** – `^.{0,100}$`

  optional: a description text for the new account

- **domain** – `sip.solucon.com` or `sip.cospace.de`

  optional: the SIP domain that this account is associated with

  SIP accounts created before the 01.01.2016 are created with `sip.cospace.de` by default. SIP accounts created after that date are using `sip.solucon.com` instead.

Response body example

```
{}
```

Status

- **wrong-sip**: The given SIP user name does not match any of the user's SIP accounts
- **wrong-domain**: The specified `domain` is not a valid choice

DELETE /api/sip/(sip-user)

Delete a SIP account

Response body example

```
{}
```

Status

- **wrong-sip**: The given SIP user / account does not exist

POST /api/sip/(sip-user)/notify

Request the system to send a custom notify message to the given sip-user

Request body example

```
{
    "event-string": "check-sync;reboot=true",
    "content-type": "application/simple-message-summary"
}
```

JSON Parameters

- **event-string** – `^.{0,200}$`

  the event-string used in the generated notify

- **content-type** – `^.{0,200}$`

  the content-type for the generated notify

- **content** – `^.{0,2500}$`

  optional: the content set in the generated notify

Response body example

```
{}
```

Status

- **wrong-sip**: The given SIP user / account does not exist
- **not-registered**: The given SIP user / account is not registered

The user can send custom notify messages to own registered sip accounts e.g. to restart the used device or to reload the current configuration.

GET /api/sip/(sip-user)/blf
Get all active phone number subscriptions for the given `sip-user`

Response body example

```
{
    "subscriptions": [
        "492216698711",
        "492216698712"
    ]
}
```

Status

- **wrong-sip**: The given SIP user / account does not exist

POST /api/sip/(sip-user)/blf
Change the subscriptions of phone numbers for the busy lamp field (blf) feature.

Request body example

```
{
    "subscribe": [
        "+492216698711",
        "+492216698712"
    ],
    "unsubscribe": [
        "+492216698307"
    ]
}
```

Response body example

```
{}
```

Status

- **wrong-sip**: The given SIP user / account does not exist
- **wrong-phone**: At least one phone number does not exist
- **access-denied**: The user is not member of an usergroup or at least one owner of the given phone numbers are not in the same usergroup as the user.

The user can subscribe to other users phone numbers and will be notified when these receive calls or are in an active call. The user can also pickup incoming calls from monitored users while they are ringing. In order to use the busy lamp field feature a supported device (SIP-Client) must be used and properly configured.

3.22 /api/presentation

POST /api/presentation
Create a new presentation object

Request body example

```
{
    "description": "Why The Sky Is Black At Nighttime"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this object

Response body example

```
{
    "uuid": "13e46b2c-13a1-11e2-b0eb-0024e8f90cc0"
}
```

The uuid of the newly created presentation object is returned.

GET /api/presentation/ (*uuid*)
Get information about presentation object `uuid`

Response body example

```
{
    "presentation": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 21232145273,
        "description": "Why The Sky Is Black At Nighttime",
        "page_count": 15,
        "busy": false,
        "status": "ok",
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```

Status

- **wrong-presentation**: The given UUID does not match a presentation object accessible to the current user

The `page_count` field returns the number of slides (pages) in this presentation. If no content has been uploaded to the presentation, this field is not present.

The `busy` field is `true` if the presentation has been started, i.e. it has active participants.

The `status` field can have the following values:

- `ok`: Presentation content is available
- `convert`: Presentation is being converted (upload). No contents are present yet.
- `empty`: Presentation has no contents available
- `fail`: Presentation or presentation upload has failed, no contents available

POST /api/presentation/ (*uuid*)
Modify presentation object `uuid`

Request body example

```
{
    "description": "Why The Sky Is Blue At Daytime"
}
```

JSON Parameters

- **description** – ^.{0,200}$

  optional: a short description of this object

Response body example

```
{}
```

Status

- **wrong-presentation**: The given UUID does not match a presentation object accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the presentation object (at least write privileges are required)

GET /api/presentation/(*uuid*)**/***page*/*xxx*.**png** Get contents of page `page` of presentation `uuid` in PNG format

Response body example

```
The contents of page <page> of the presentation in PNG format on
success, or HTTP 404 without response body on failure.
```

`xxx` is a file name arbitrarily chosen by the user agent.

GET /api/presentation/(*uuid*)**/***xxx*.**pdf** Get original contents of presentation `uuid` in PDF format

Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The original contents of the presentation in PDF format on success, or
HTTP 404 without response body on failure.
```

`xxx` is a file name arbitrarily chosen by the user agent.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the presentation `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

POST /api/presentation/(*uuid*)**/***xxx*.**pdf** Upload presentation `uuid` contents in PDF format

Request body

```
The presentation contents in PDF format.
```

Response body example

```
{}
```

Status

- **wrong-presentation**: The given UUID does not match a presentation accessible by the current user
- **access-denied**: The user does not have sufficient privileges on the presentation (at least write privileges are required)
- **too-large**: The contents are too large to be handled by the system
- **malformed-file**: The content is malformed.

`xxx` is a file name arbitrarily chosen by the user agent.

After the contents are received, the PDF file is converted and the individual pages (slides) of the presentation are stored as images (PNG format). The original content is preserved and can be downloaded later.

DELETE /api/presentation/ (*uuid*)
Delete presentation object `uuid`

Parameters

- **purge** – optional: if present, delete this object permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-presentation**: The given UUID does not match a presentation object accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the presentation object (at least write privileges are required)

GET /api/presentation/ (*uuid*) **/event**
Subscribe the current session to the event channel of the given presentation

Response body example

```
{
    "busy": true
}
```

Status

- **wrong-presentation**: The given UUID does not match a presentation accessible by the current user

This API call will subscribe the current session to the detailed events of the given presentation. The events themselves will be delivered by the *GET /api/event* interface.

If the presentation has active participants, the `busy` field will be `true`, and the server is supposed to send a `presentation_status` event shortly after receiving this subscription, so that the client can catch up with the current presentation state.

If `busy` is `false`, then the presentation does not have active participants, however the detailed event subscription still takes place, and presentation events will start to flow as soon as the presentation starts.

`DELETE /api/presentation/`(*uuid*)**`/event`**

Cancel the subscription of the current session to the event channel of the given presentation

Response body example

```
{}
```

Status

- **wrong-presentation**: The given UUID does not match a presentation accessible by the current user

This API call will stop the flow of presentation detail events to the current session.

`POST /api/presentation/`(*uuid*)**`/control`**

Control the presentation `uuid`, modify subscription state for participants or control the presentation parameters for the moderator

Request body example

```
{
    "control": "start"
}
```

Response body example

```
{}
```

Status

- **wrong-presentation**: The given UUID does not match a presentation accessible by the current user
- **illegal-state**: The given control parameter does not match the current presentation state or the state of the current user

The following control command parameters are defined:

Start a presentation

```
{
    "control": "start"
}
```

This command starts a presentation. The current user will join the presentation as the first member and will take the role of the moderator. If the presentation has already been started, an `illegal-state` status will be returned.

Stop a presentation

```
{
    "control": "stop"
}
```

This command ends a presentation. All members of the presentation leave the presentation. If the presentation has not been started, an `illegal-state` status will be returned. Only the moderator can stop a presentation.

Join a presentation

```
{
    "control": "join"
}
```

With this command, the current user will join the presentation as a listening member. If the presentation has not been started, an `illegal-state` status will be returned.

Leave a presentation

```
{
    "control": "leave"
}
```

With this command, the current user will leave the presentation. If the presentation has not been started, an `illegal-state` status will be returned.

Member keepalive

```
{
    "control": "keepalive"
}
```

A presentation member should send this control after receiving a `presentation_keepalive` event. The keepalive mechanism is used by the server to continuously ensure the presence of listening members.

Change the presentation moderator

```
{
    "control": "moderator",
    "moderator": "489fbb76-13a8-11e2-b6aa-0024e8f90cc0"
}
```

This command changes the moderator of a presentation. If the presentation has not been started, an `illegal-state` status will be returned. Only users that are currently members of the presentation can be appointed the moderator. The old moderator will stay a member of the presentation.

Switch page

```
{
    "control": "page",
    "page": 3
}
```

The current page of the presentation is changed to the given number. If the presentation has not been started, an `illegal-state` status will be returned. Only the moderator can control the current page.

Move pointer

```
{
    "control": "pointer",
    "x": 0.47,
    "y": 0.7,
    "visible": true
}
```

This command changes the pointer properties of the presentation. The `x`, `y` and `visible` fields are all optional. They refer to the pointer position (x, y) with values between 0.0 (left, top) and 1.0 (right, bottom) and control the

visibility of the pointer. If the presentation has not been started, an `illegal-state` status will be returned. Only the moderator can control the pointer.

3.23 /api/chat

`GET /api/chat`
Get information about chats that the user participates in

Parameters

- **`from`** – optional: start time, in seconds-since-epoch (must have from <= to)
- **`to`** – optional: end time, in seconds-since-epoch (must have from <= to)
- **`start`** – optional: the UUID of the first chat to return, exclusive, i.e. the chat with an exact match UUID is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.
- **`order`** – optional: if set to `asc` (default), chats are returned in time ascending order (i.e. starting with `from` or `start` and ending with `stop`).

  If set to `desc`, chats are returned in time descending order (i.e. starting with `to` or `start` and ending with `from`).
- **`count`** – optional, default `10`: limits the number of returned chats (valid range: `1...500`)
- **`uuid`** – optional: if set, the chat with the given UUID is returned as a single entry (if it exists and the current user is a member). All other URL parameters are ignored.

Response body example


```
{
    "chat": {
        "1815eef2-84d3-11e2-8d90-0024e8f90cc0": {
            "description": "some nice chat",
            "create_time": 1362587184,
            "message_head": "fd43561e-84d7-11e2-88d8-0024e8f90cc0",
            "message_read": "68ee6d72-84d3-11e2-a343-0024e8f90cc0",
            "user": [
                "126ad5ae-31a2-11e3-bfaa-0024e8f90cc0",
                "2018470e-31a2-11e3-a05a-0024e8f90cc0"
            ]
        },
        "2c71353c-84d3-11e2-a1d3-0024e8f90cc0": {
            "message_head": "39af0574-84d5-11e2-bf25-0024e8f90cc0",
            "create_time": 1362587000,
            "user": [
                "2018470e-31a2-11e3-a05a-0024e8f90cc0"
            ]
        },
        "3a7354c4-84d5-11e2-a795-0024e8f90cc0": {
            "create_time": 1362534343,
            "user": [
                "126ad5ae-31a2-11e3-bfaa-0024e8f90cc0"
            ]
        }
    }
}
```

The `description` field is optional, in fact it might not be set for a majority of chats.

The `create_time` field denotes the creation time of the chat in seconds-since-epoch.

The `message_head` UUID points to the newest message in the chat, while the `message_read` UUID points to the newest message that the current user has marked with *`POST /api/chat/(uuid)/user`*. Together, these fields enable a client to detect whether there are new messages in the chat. In addition, they serve to synchronize the "message-read state" between multiple clients. Both fields are optional (no messages might exist in the chat or no message might have been marked yet).

The user section contains all user UUIDs currently participating in the chat, excluding the current session's user.

`POST /api/chat`
Create a new chat

Request body example

```
{}
```

Response body example

```
{
    "uuid": "1815eef2-84d3-11e2-8d90-0024e8f90cc0"
}
```

The newly created chat will have only one participant, the current user.

The call returns the uuid of the newly created chat.

`GET /api/chat/`(*uuid*)**`/user`**
Get participant information about a specific chat

Response body example

```
{
    "user": {
        "1de7257a-4f34-11e0-ab6e-0024e8f90cc1": {
            "active": 58,
            "message_read": "68ee6d72-84d3-11e2-a343-0024e8f90cc0"
        },
        "1d333333-4f34-11e0-ab6e-0024e8f90cc1": {
            "active": 35,
            "typing": 20,
            "message_read": "68ee6d72-84d3-11e2-a343-0024e8f90cc0"
        },
        "1d444444-4f34-11e0-ab6e-0024e8f90cc1": {}
    }
}
```

Status

- **wrong-chat**: The given UUID does not match a chat that the current user participates in

The `user` section lists all users that participate in the chat. Users are keyed by their user UUID.

Within the individual user's section, the transient states of the user are listed. The represents the state, and the value is a timeout in seconds that denotes the expire time of this state. For example, if the state is `"typing": 20`, this means that user is currently typing text, and if no update is received this state will last for the next 20

seconds. It is the responsibility of a client to expire transient states after the given expire time, so in this case the client must set the internal state to "not typing text" after 20 seconds.

Note that all transient states in the user's section are optional. States that are not listed are supposed to be inactive, i.e. not set.

The following transient states are defined:

- `active` states that the user is currently attending the chat, i.e. has an active view of the chat and is most likely to take note of chat contents.
- `typing` states that the user is currently typing text.

In addition to the transient states, the `user` section also lists the `message_read` field if the corresponding user has read at least one message in the chat.

POST /api/chat/(*uuid*)**/user**

Change the current user's transient chat state or read message pointer

Request body example

```
{
    "active": 60,
    "typing": 10,
    "message_read": "68ee6d72-84d3-11e2-a343-0024e8f90cc0"
}
```

JSON Parameters

- **state** – `0..60`

  optional; the state expire value in seconds, `0` will set the state to inactive immediately

- **message_read** – `UUID`

  optional; the current message-read pointer for this user

Response body example

```
{}
```

Status

- **wrong-chat**: The given UUID does not match a chat that the current user participates in
- **wrong-message**: The given `message_read` does not match a message within the chat

All fields in this request are optional.

With this call, the user can set her own transient states expire values. For a description about transient states, see *GET /api/chat/(uuid)/user*.

Note that since only `expire` values between `0` and `60` seconds are allowed, clients are forced to regularly refresh longer lasting states such as active. This mechanism is used to ensure the detection of dead clients.

In addition to the transient states, the `message_read` pointer for the session's use can also be set with this call.

GET /api/chat/(*uuid*)**/message**

Get chat messages

Parameters

- **from** – optional: start time, in seconds-since-epoch (must have from <= to)
- **to** – optional: end time, in seconds-since-epoch (must have from <= to)
- **start** – optional: the UUID of the first message to return, exclusive, i.e. the message with an exact match UUID is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.
- **order** – optional: If set to `asc` (default), messages are returned in time ascending order (i.e. starting with `from` or `start` and ending with `stop`).

  If set to desc, messages are returned in time descending order (i.e. starting with `to` or `start` and ending with `from`).
- **count** – optional, default `10`: limits the number of returned messages (valid range: `1...500`)

Response body example


```
{
    "message": {
        "c2c35df8-84dd-11e2-933d-0024e8f90cc0": {
            "user": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
            "time": 1362407297,
            "type": "add",
            "member": "fd43561e-84d7-11e2-88d8-0024e8f90cc0"
        },
        "c1aa1566-84d7-11e2-a7a8-0024e8f90cc0": {
            "user": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
            "time": 1362407297,
            "type": "text",
            "text": "what's for lunch today?"
        },
        "ee01a430-84d7-11e2-bab2-0024e8f90cc0": {
            "user": "1d333333-4f34-11e0-ab6e-0024e8f90cc1",
            "time": 1362407293,
            "type": "image"
        },
        "f901a430-84d7-11e2-bab2-0024e8f90cc0": {
            "user": "1d333333-4f34-11e0-ab6e-0024e8f90cc1",
            "time": 1362407297,
            "type": "text",
            "text": "chicken wings \u2013 again :-("
        },
        "fa01a430-84d7-11e2-bab2-0024e8f90cc0": {
            "user": "1d333333-4f34-11e0-ab6e-0024e8f90cc1",
            "time": 1362407293,
            "type": "audio"
        },
        "fd43561e-84d7-11e2-88d8-0024e8f90cc0": {
            "user": "1d444444-4f34-11e0-ab6e-0024e8f90cc1",
            "time": 1362407297,
            "type": "leave"
        }
    }
}
```


Status

- **wrong-chat**: The given UUID does not match a chat that the current user participates in

The returned chat messages are keyed by the message UUID.

Within the `message` section, the `user` field specifies the user who performed the action. For a description of the message types, see *POST /api/chat/(uuid)/message*.

POST /api/chat/(*uuid*)**/message**

Post a chat message or action

Request body example

```
{
    "type": "text",
    "text": "what's for lunch today?"
}
```

- or -

```
{
    "type": "add",
    "member": "1d444444-4f34-11e0-ab6e-0024e8f90cc1"
}
```

JSON Parameters

- **text** – `^.{1,500}$`

  the message text (for type text)

- **description** – `^.{,100}$`

  the new chat description (for type description)

- **member** – `UUID`

  the user UUID of the new member (for type add)

Response body example

```
{
    "uuid": "c1aa1566-84d7-11e2-a7a8-0024e8f90cc0"
}
```

Status

- **wrong-chat**: The given UUID does not match a chat that the current user participates in
- **wrong-user**: Only for type `add`: the given member is unknown or has no link with the current user
- **already-member**: Only for type `add`: the given member is already a member of this chat

The following message types are defined:

- `text` is a simple text message. The content is delivered in the field text.
- `add` adds a new member to the chat. The member field holds the UUID of the new member.
- `leave` is used to exit the chat.

- `description` is used to change the description of the chat. The new description is delivered in the field description.

The call will return the UUID of the newly created chat message.

POST /api/chat/(*uuid*)**/message/image**

Post a chat message with type image

Request body example

```
Binary encoded image data in JPEG or PNG format
```

Response body example

```
{
    "uuid": "c1aa1566-84d7-11e2-a7a8-0024e8f90cc0"
}
```

Status

- **wrong-chat**: The given UUID does not match a chat that the current user participates in
- **too-large**: The image data is too large to be handled
- **malformed-file**: The image data is malformed or cannot be handled

This request will create a chat message with type `image`. The image data is passed as raw image data (format JPEG or PNG) in the request body.

The maximum image size is 10MB and the maximum acceptable dimension is 5000 x 5000 pixels.

The system will down-scale images to a maximum dimension (x- or y-dimension) of 2000 pixels (the large version of the image) and an additional version with a maximum dimension of 500 pixels (the small version of the image). Down-scaling will keep the aspect ration of the image; images smaller than the target dimension will not be up-scaled.

The call will return the UUID of the newly created chat message.

POST /api/chat/(*uuid*)**/message/audio**

Post a chat message with type audio

Request body example

```
Binary encoded audio data in AAC format
```

Response body example

```
{
    "uuid": "c1aa1566-84d7-11e2-a7a8-0024e8f90cc0"
}
```

Status

- **wrong-chat**: The given UUID does not match a chat that the current user participates in
- **too-large**: The audio data is too large to be handled
- **malformed-file**: The audio data is malformed or cannot be handled

This request will create a chat message with type `audio`. The audio data is passed in AAC encoded form (MP4 or M4A container).

The maximum encoded audio size is 10MB.

The call will return the UUID of the newly created chat message.

GET /api/chat/(*uuid*)**/message/(message-uuid)/image/(small|large)/***xxx*.**jpg** Get the contents of a chat message with type `image`

Response body example

```
Binary encoded image data in JPEG format, or an HTTP error without
response body on failure.
```

The size selector within the URL is used to select either the small (maximum dimension 500 pixels) or the large version of the image (maximum dimension of 2000 pixels).

The `xxx` portion of the URI is an arbitrarily chosen file name by the client.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the `message-uuid`. For more information on file download authentication, see *Authentication for file downloads*.

GET /api/chat/(*uuid*)**/message/(message-uuid)/audio/***xxx*.**(mp4|m4a)** Get the contents of a chat message with type audio

Parameters

- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
Binary encoded image data in AAC format, MP4/M4A container, or an HTTP
error without response body on failure.
```

The `xxx` portion of the URI is an arbitrarily chosen file name by the client.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the `message-uuid`. For more information on file download authentication, see *Authentication for file downloads*.

3.24 /api/call

GET /api/call

Get details of the current user's controlled calls within the call API.

Parameters

- **from** – optional: start time, in seconds-since-epoch (must have from <= to)
- **to** – optional: end time, in seconds-since-epoch (must have from <= to)
- **start** – optional: the UUID of the first call to return, exclusive, i.e. the call with an exact match UUID is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.
- **order** – optional: if set to `asc` (default), calls are returned in time ascending order (i.e. starting with `from` or `start` and ending with `stop`).

If set to `desc`, calls are returned in time descending order (i.e. starting with `to` or `start` and ending with `from`).

- **`count`** – optional, default `10`: limits the number of returned calls (valid range: `1...500`)
- **`uuid`** – optional: if set, the call with the given UUID is returned as a single entry (if it exists and is owned by the current user. Any other URL parameters are ignored.
- **`state`** – optional: if set to `active` (default), only active, i.e. ongoing calls will be returned. If set to `hangup`, historic calls that are already finished will be searched.

Response body example


```
{
    "call": {
        "2b3b70ce-8677-11e2-a7c1-0024e8f90cc0": {
            "incoming": false,
            "to": "+492216695777",
            "from": "+492216695888",
            "clip": "+492216695000",
            "clir": false,
            "status": "answered",
            "create_time": 1362587184,
            "answer_time": 1362587197,
            "current_control_id": "ivr-second-state",
            "last_control_id": "ivr-first-stage",
            "record_on": false
        },
        "ecbe6758-867a-11e2-a415-0024e8f90cc0": {
            "incoming": true,
            "to": "+492216695888",
            "from": "anonymous",
            "clir": true,
            "status": "ringing",
            "create_time": 1362587213,
            "last_control_id": "let-it-ring",
            "record_on": false
        },
        "ecc434b6-867b-11e2-a32f-0024e8f90cc0": {
            "incoming": true,
            "to": "+492216695888",
            "from": "+498001234567",
            "clir": false,
            "status": "answered",
            "create_time": 1362587213,
            "answer_time": 1362587215,
            "current_control_id": "play-some-stuff",
            "last_control_id": "hook-off",
            "record_on": false,
            "bridge": "1a4d67c2-867c-11e2-aadc-0024e8f90cc0"
        },
        "4644f1d4-867b-11e2-96ed-0024e8f90cc0": {
            "incoming": true,
            "to": "+492216695888",
            "from": "+492216695777",
            "clir": false,
            "status": "hangup",
            "create_time": 1362587350,
            "answer_time": 1362587358,
            "hangup_time": 1362587599,
```

```
            "last_control_id": "arrivederci",
            "record_on": false,
            "recording": "ba65e460-867b-11e2-9c3b-0024e8f90cc0",
            "hangup_cause": 16
        }
    }
}
```

Calls are listed in the call section with the call UUID as the key.

If `incoming` is `true`, the call direction is incoming (from the PSTN towards cospace), if `false`, the direction is from cospace towards the PSTN.

The `to` field specifies the called phone number (B-party).

The `from` field specifies the caller's phone number (A-party).

The `clip` field is only present if a different, user-signaled caller phone number is associated with the call.

The `clir` field signals if the presentation of the caller's phone number is restricted (CLIR feature).

The `status` field can have one of the following call states:

- `init` – only for outgoing calls, the call logic is initialized
- `calling` – the call is ongoing and has not received any confirmation or progress yet
- `progress` – the call received a confirmation and is progressing
- `ringing` – the call is progressing and has ringing indication
- `answered` – the call has been answered, media is established
- `hangup` – the call has ended

The `hangup_cause` field is only present if the call status is hangup. It carries the Q.850 termination cause code.

If `record_on` is `true`, call media is currently being recorded. If the recording process is finished and the media has been imported into the system, the `recording` field will show up and carry the UUID of a new recording object that has the media data.

The `bridge` field is only present if the call is currently bridged with another call. In this case, it carries the UUID of the other call leg.

The `create_time` field denotes the time of call creation (seconds-since-epoch).

The `answer_time` field denotes the time of answering (only present if the call was answered).

The `hangup_time` field denotes the time of call hangup (only present if the call has ended).

If a call control is currently executing on the call, the `current_control_id` field will carry the user-provided id of this call control.

If at least one call control has finished executing on the call, the `last_control_id` field will carry the user-provided id of the last call control that was finished.

GET /api/call/event

Subscribe the current session to the call API events

Response body example

```
{}
```

This API call will subscribe the current session to the flow of call events which are part of the call API.

DELETE /api/call/event

Cancel the subscription of the current session to the call API events

Response body example

```
{}
```

This API call will stop the flow of call API events to the current session.

POST /api/call/dial

Create an outgoing call within the call API

Request body example

```
{
    "to": "+492216698123",
    "from": "+492216698007",
    "clip": "+4980098989898",
    "clir": false
}
```

JSON Parameters

- **to** – `^\+[0-9]{3,20}$`

  the destination phone number

- **from** – `^\+[0-9]{3,20}$`

  the calling phone number, must be one of the current user's phone numbers

- **clip** – `^\+[0-9]{3,20}$`

  optional; an arbitrary phone number for display (CLIP – calling line identification presentation feature)

- **clir** – `true|false`

  optional, default `false`; if `true`, presentation of the calling number is restricted (CLIR feature)

Response body example

```
{
    "uuid": "2b3b70ce-8677-11e2-a7c1-0024e8f90cc0"
}
```

Status

- **wrong-phone**: The given `from` number does not match any of the user's phone numbers

POST /api/call/(*uuid*)**/control**

Execute one or more actions on the call `uuid` via the call API

Request body example

```
{
    "queue": "append",
    "controls": [
        {
            "control": "ring"
```

```
        },
        {
            "control": "silence",
            "duration": 1000
        },
        {
            "id": "hook-off",
            "control": "answer"
        }
    ]
}
```

JSON Parameters

- **queue** – `append|clear|break`

  optional; controls the behavior of the server-side control queue. If `append` (default), the listed controls are appended to the control queue and are executed after the controls that are already in the queue.

  If set to `clear`, the server-side queue is cleared and replaced with the given controls.

  `break` will behave like `clear` but in addition to clearing the server queue, it will also interrupt the currently running control on the server.

- **id** – `^.{,100}$`

  optional; an arbitrary, user-supplied identification string that will be referenced in the call state and events to identify the currently running control.

- **control** – (see below)

  a control action to execute on the call. For detailed description of available controls, see below.

Response body example

```
{}
```

Status

- **illegal-state**: the given control parameter does not match the current call state. For example, a call that is already answered will not accept a ring control.
- **wrong-call**: the given call `uuid` is unknown
- **wrong-object**: only for control play: the given announcement or recording UUID is invalid

The following call control actions:

Start ringing

```
{
    "control": "ring"
}
```

This control will let the call start ringing, which is only possible in incoming calls. The caller will typically hear a network-generated ring-tone in this state.

Signal call progress

```
{
    "control": "progress"
}
```

This control will signal call progress on an incoming call. This will establish one-way media flow, so that it is possible to play audio to the call.

Answer the call

```
{
    "control": "answer"
}
```

This control will answer the call and establish two-way media flow. This is only possible on incoming calls.

End the call

```
{
    "control": "hangup",
    "hangup_cause": 16
}
```

This control will end the call. The optional `hangup_cause` field can be supplied to specify the ITU-T Q.850 call termination cause. It defaults to 16 ("normal call clearing").

Bridge two calls

```
{
    "control": "bridge",
    "call": "0cde3f60-8683-11e2-99bd-0024e8f90cc0",
    "hangup_both": true
}
```

This control will connect the call with another call, specified by the UUID in the call field. This b-leg call must also be controlled by the same user's call API.

The optional `hangup_both` field controls how the other call leg behaves in case a call leg is hung up. If `false`, then the bridge will terminate but the other channel will stay alive. If `true` (default value if the field is not given) the other channel will automatically be hung up.

Un-bridge two calls

```
{
    "control": "split"
}
```

This control will end the bridge between two calls. To perform this operation, the call must have been previously bridged with another call using the bridge control.

Sleep on a call without audio

```
{
    "control": "sleep",
    "duration": 1000
}
```

This control will wait for the specified number of milliseconds (duration field, allowed values between 1 and 3600000). During the sleep, no audio will be played to the call. The primary use is when a call is status where

it is not accepting audio (i.e., incoming calls with status calling or outgoing calls that are not yet answered). To pause within an answered call, the use of the silence control is recommended.

This control can be interrupted with the queue break option.

Sleep on a call playing silence

```
{
    "control": "silence",
    "duration": 1000
}
```

This control will play silence to the call for the specified number of milliseconds (duration field, allowed values between 1 and 3600000).

This control can be interrupted with the queue break option.

Play a tone one a call

```
{
    "control": "tone",
    "duration": 1000,
    "frequency": 440,
    "interval": 4000,
    "loop": 10
}
```

This control will play a tone with the specified frequency (in Hz, values between 0 and 40000) for duration milliseconds (values between 1 and 3600000). If the interval field is given, it specifies a period of silence following the tone (in milliseconds, values between 1 and 3600000). If the loop field is given (range 0 to 1000), the tone (and the optional period of silence) is repeated for the specified amount of times, with 0 meaning endless repetitions.

This control can be interrupted with the queue break option.

Play audio on a call

```
{
    "control": "play",
    "object": "a93d166e-a69c-5a85-8604-4f9faa6620cc"
}
```

This control will play the audio contents of an announcement or recording object with the specified object UUID to the channel. The given UUID must point to an object of type `announcement` or `recording`.

This control can be interrupted with the queue break option.

Start call recording

```
{
    "control": "record_start",
    "limit": 30
}
```

This control will start audio recording on the call. The `limit` field specifies the maximum duration of the recording in seconds (values between 1 and 3600). The process will eventually generate a recording object and deliver its UUID with an `call_update` event when the recording is finished.

Stop call recording

```
{
    "control": "record_stop",
    "discard": true
}
```

This control will stop an ongoing audio recording on the call. If the optional `discard` field is given and has a value of `true`, the recording contents will be discarded. Otherwise, a recording object will be created and its UUID will be delivered with a `channel_update` event.

Enable / Disable DTMF collection on a call

```
{
    "control": "dtmf_collect",
    "enable": true
}
```

This control will enable or disable the recognition and delivery of DTMF digits within the call (which is disabled by default). The `enable` field specifies whether detected DTMF digits should be delivered via `call_dtmf` events.

Play DTMF digit on a call

```
{
    "control": "dtmf_send",
    "digits": "*123#",
    "duration": 250
}
```

This control will play the specified DTMF digits into the call (valid digits are "0"..."9", "#" and "*"). Instead of the default DTMF digit duration 100 milliseconds, the duration might optionally be specified by duration parameter (in milliseconds, values between 50 and 3600000).

3.25 /api/sensor

`POST /api/sensor`

Create a new sensor object and associates the sensor with a physical sensor device using the sensor device code.

Request body example

```
{
    "description": "sensor in my garden",
    "code": "ABCDE-FGHIJ-KLMNI-OPQRS"
}
```

JSON Parameters

- **`description`** – `^.{0,200}$`

  optional: a short description of this sensor

- **`code`** – `^.{10,50}$`

  the sensor device access code, which is the key to the physical device that will be attached to the new sensor

Response body example

```
{
    "uuid": "63013ee2-772a-11e1-a9b8-0024e8f90cc0",
    "sdevice": "001a45fe"
}
```

Status

- **wrong-code**: The given code does not match an available physical sensor device
- **duplicate-sensor**: The sensor device is already linked to an existing sensor object

The `uuid` field returns the UUID of the newly created sensor object as a hex encoded string. The `sdevice` field carries the hardware address of the sensor device that the sensor was linked to.

GET /api/sensor/ (*uuid*)
Get information about sensor `uuid`

Response body example

```
{
    "sensor": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
        "time": 1363333307,
        "description": "sensor in my garden",
        "sdevice": "001a45fe",
        "mbus": {
           "manufacturer" : "AMT",
           "ident_number" : "50026470",
           "version" : "2e",
           "device_type" : "04"
        },
        "model": "cospace-sensor",
        "profile": "room-sensor-thm",
        "recv_interval": 60,
        "send_interval": 10,
        "recv_after_send": true,
        "recv_time": 1363623307,
        "battery_status": 86,
        "mains_power": true,
        "tamper_detect": 1363347807,
        "fault_detect": 1363347820,
        "capabilities": {
            "data": [
                "temperature",
                "motion"
            ],
            "action": [
                "onoff",
                "text"
            ]
        },
        "state": {
            "data": [
                20.5,
                [1, 10, 50]
            ],
            "action": [
                1,
```

```
                "Hello, World!"
            ]
        },
        "box": {
            "efd04b02-8fd8-11e2-bf7e-0024e8f90cc0": {
                "time": 1320403260,
                "rssi": -80,
                "lqi": 45
            },
            "f10cb83e-8fd8-11e2-9f7c-0024e8f90cc0": {
                "time": 1320403155,
                "rssi": -38,
                "lqi": 70
            }
        },
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```


Status

- **wrong-sensor**: The given UUID does not match a sensor accessible to the current user

The `sdevice` field carries the hardware address of the sensor device associated with the sensor as a hex encoded string. The `model` field contains the model identification string of the associated sensor device. If the sensor device belongs to a standardized profile class, this is shown in the `profile` field.

The optional `mbus` field carries information retrieved from the M-Bus Device Id if the sensor is an M-Bus device. If so, the sensor uses the M-Bus protocol for communication in the local sensor network. The subfield `dev_id` provides the complete hex string of the M-Bus device id as transported on the wire (or on the ether if the sensor is an wireless M-Bus device). The `manufacturer` subfield informs about the device manufacturer in the format given by ASCII code of EN 62056-21 (three uppercase letters). The flag association, UK (http://www.dlms.com/) administers these three letter manufacturers ID of EN 62056-21. The `ident_number` is a fixed fabrication number that runs from 00000000 to 99999999. The subfield `version` specifies the generation or version of the device and depends on the manufacturer. It can be used to make sure, that within each ''version" number the `ident_number` is unique. The subfield field `dev_type` mirrors the device type of the sensor in hex code format as specified in EN 113757-3.

The `recv_interval` field is only present if the sensor is supposed to send data in regular time intervals. It specifies the maximum time interval between sensor transmissions in seconds.

The `send_interval` field is only present if the sensor participates in the beacon protocol. It specifies the maximum beacon interval for the sensor in seconds.

If the `recv_after_send` field is present, the sensor will receive data only after having transmitted a packet itself. This is typically used in battery powered applications.

The `recv_time` field contains the timestamp of the last received data from the sensor. It is only present if the sensor has delivered data at least once in its lifetime.

The `battery_ok` field (type `boolean`) will be present if the sensor is battery-powered and is capable of signaling a simple "battery ok" (`true`) or "battery low" (`false`) condition. If the sensor is capable of reporting detailed battery statistics, the `battery_status` field will reflect the battery level in percent. The `mains_power` field might be present if a battery-powered device is currently running on mains power. If the device has the ability to detect a tampering attempt, the `tamper_detect` field will be present and will show

the timestamp of the first detection. Likewise, the presence of the `fault_detect` field signals a sensor fault condition timestamp.

The box section will list the UUIDs of the box objects that have had contact with the sensor in the last time. Each box will have some meta-information about the connection to the sensor. Possible fields in this section include `time` (time of the last sensor contact in seconds-since-epoch), `rssi` (received signal strength indication in dBm) and `lqi` (link quality indicator, higher value means better signal).

The `capabilities` section lists the feature attributes of the sensor device.

In the capabilities `data` subsection, sensor capabilities are displayed as an ordered array of features (thus with an implicit index). The sensor device in the above example has two sensor features, a temperature sensor at index 0 which will yield a temperature in degrees Celsius), and a motion sensor at index 1 which will yield a motion detection duration time in seconds.

In the capabilities `action` subsection, actor capabilities are displayed as an ordered array of features (thus with an implicit index). The sensor device in the above example has two actor features, a onoff actor at index 0 that can be switched on (1) or off (0) and a text display at index 1.

The following capabilities are defined with their corresponding data format (see *GET /api/sensor/(uuid)/data*):

- `temperature`: A temperature sensor that has a single JSON number value as data, measured in degree Celsius (°C).
- `light`: An illuminance sensor (typically an ambient light sensor) that has a single JSON number value as data, measured in lux (lx; 1 lx = 1 lm/m^2 = 1 cd sr/m^2).
- `humidity`: A sensor for relative humidity that has a single JSON number value as data, measured in percent.
- `open`: A sensor that monitors the state of an opening such as a door, window, vent or a similar object. The value is a single JSON number that reads 1 if the state is "open", or 0 if the state is "closed".
- `open_percent`: A sensor that monitors the state of an opening such as a door, window, vent or a similar object. The value is a percentage, expressed as a single JSON number. The reading is 100 for "open" and 0 if the state is "closed". Numbers in between signal a half-closed half-open condition.
- `motion`: A motion detection sensor which has a JSON array containing exactly three JSON numbers as data. The first number represents the initial state of the motion sensor (`0` for no motion detect, `1` for motion detect), the second number represents a time interval in seconds for this state. The third number represents a time interval for the opposite state. As an example, the data `[1, 3, 5]` would mean that the initial state of the sensor was "motion detect", this state lasted for 3 seconds, and afterwards the sensor remained 5 seconds in the state "no motion detect". The data `[0, 100, 1]` means that the sensor did not detect motion for 100 seconds, but then a motion detect happened for 1 second.
- `energy`: An energy meter that has a single JSON number value as data, measured in Watt-hours (Wh; 1 Wh = 0.001 kWh = 3600 J [Joule]).
- `voltage`: An electrical voltage sensor that has a single JSON number value as data, measured in Volts (V).
- `current`: An electrical current sensor that has a single JSON number value as data, measured in Amperes (A).
- `power`: An electrical power sensor that has a single JSON number value as data, measured in Watts (W).
- `power_factor`: An electrical power factor. Data is a single JSON number in the range -1..1.
- `frequency`: A frequency counter. The data is represented as a JSON number, measured in Hz (1 Hertz = 1 cycle per second).

- `onoff`: A switch-type sensor that has a single JSON number as a representation. Valid values are 1 (representing the "on" state) and 0 ("off" state).
- `text`: A device that has some sort of text display. The representation is a JSON string which holds the text that is to be displayed with a maximum length of 255 characters.
- `button`: A button-type sensor that has a single JSON number as a representation. There is only one valid value, 1, which represents that the button was pressed.
- `color_rgb`: A color sensor. The data is represented as a JSON array with exactly three JSON numbers, one for each color component: red, green and blue. Values for the individual components range from 0 to 255. As an example, the data [255, 255, 0] represents a bright yellow.
- `interval`: A time interval expressed as a single JSON number, representing the time period in seconds (s).
- `datetime`: An absolute point in time, expressed as a single JSON number that holds the seconds elapsed since the epoch of Jan 1st, 1970.
- `dimmer`: A dimmer switch in percent, represented by a single JSON number with values between 0 (completely off) and 100 (completely on).
- `distance`: A physical distance (length), expressed by a single JSON number, representing the distance in meters (m).
- `mass`: A mass, expressed as a single JSON number, representing the mass in kilograms (kg).
- `mass_flow`: A flow rate of mass, expressed as a single JSON number, representing the flow in kilograms per second (kg/s).
- `volume`: A space volume, expressed as a single JSON number, representing the volume in cubic meters (m^3).
- `volume_flow`: A flow rate of volume, expressed as a single JSON number, representing the flow in cubic meters per second (m^3/s).
- `fuel_use`: A mileage (fuel usage), expressed as a single JSON number, representing the value in liters per 100 km (l/100km).
- `velocity`: A velocity, expressed as a single JSON number, representing the speed in meters per second (m/s).
- `acceleration`: An acceleration, expressed as a single JSON number, representing the speed gain in meters per square second (m/s^2).
- `resistance`: An electrical resistance, expressed as a single JSON number, representing the resistance in ohms (Ω).
- `pressure`: A pressure, expressed as a single JSON number, representing the pressure in Pascal (Pa; 1 Pa = 1 N/m^2).
- `force`: A force, expressed as a single JSON number, representing the force in Newton (N).
- `torque`: A circular force (torque), expressed as a single JSON number, representing the torque in Newton meters (Nm).
- `angle`: An angle, expressed as a single JSON number, representing the angle in degrees (°, full circle is 360°).
- `compass`: A compass reading, expressed as a single JSON number, in degrees °, clockwise from the north direction (0°) to east (90°), south (180°) and west (270°) back to north (360°).

- `location`: A geographical position. The data is represented as a JSON array with exactly two JSON numbers. The first number represents the longitude, the second number the latitude of the position. Both values are in degrees (°) ranging from -180° to 180°.
- `concentration`: A concentration (ratio of mixture between two components), expressed as a single JSON number, representing the concentration in parts-per-million (ppm).
- `ph`: A pH value, expressed as a single JSON number, representing the pH (no unit, typical values between 1 and 14).
- `radiation`: An ionizing radiation dose. The data is expressed as a single JSON number, representing the dose in Sievert (Sv).
- `sound_pressure`: A sound pressure, expressed as a single JSON number, representing acoustic pressure in dezibels (dB).
- `level`: An otherwise unspecified logarithmic level, expressed as a single JSON number, representing the level in dezibels (dB).
- `alarm`: An alarm sensor. The data is expressed as a single JSON number. Valid values are 0 ("no alarm" state) and 1 ("alarm" state).
- `gauge`: An otherwise unspecified absolute value, expressed as a single JSON number.
- `counter`: An otherwise unspecified counter value, expressed as a single JSON number.
- `load`: An load percentage value, expressed as a single JSON number, representing the load in percent (%).
- `cycles`: A rotary speed value, expressed as a single JSON number, representing the rotary speed in cycles per second (1/s).
- `binary_8bit`: An otherwise unspecified value, expressed as a single JSON integer number with a range between 0 and 255.
- `binary_16bit`: An otherwise unspecified value, expressed as a single JSON integer number with a range between 0 and 65535.
- `binary_32bit`: An otherwise unspecified value, expressed as a single JSON integer number with a range between 0 and 4294967295.
- `octets`: An otherwise unspecified array of octets (binary string), expressed as a single JSON string in Base64 encoding.
- `mbus_raw`: A raw M-Bus (long-)frame, expressed as a single JSON string in Base64 encoding. After its end marker, the frame may be padded with 0x00.

The `state` section contains the most recent up-to-date status of the `data` and `action` elements. Its structure matches the `capabilities` section. While the *`GET /api/sensor/(uuid)/data`* and *`GET /api/sensor/(uuid)/action`* calls can be used to get the precise time series of sensor data and actions, the `state` section represents the current state of each of the `data` and `action` items. For the `data` items, this is the last sensor data update which was not null, and the the `action` items, this is that last action command which was not null.

`POST /api/sensor/`(*uuid*)

Modifies sensor `uuid`, including the possibility to change the underlying physical sensor device.

Request body example

```
{
    "description": "new sensor in garden",
    "code": "XYZKL-FGHIJ-KLMNI-OPQRS"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this sensor

- **code** – `^.{10,50}$`

  optional: the sensor device access code, which is the key to the new physical sensor device that will be attached to the sensor

- **tamper_detect** – `^0$`

  optional: if set to the fixed value of `0`, a tamper detect condition of the sensor is cleared

- **fault_detect** – `^0$`

  optional: if set to the fixed value of `0`, a fault condition of the sensor is cleared

Response body example

```
{
    "sdevice": "014a5fe7"
}
```

Status

- **wrong-sensor**: The given UUID does not match a sensor accessible to the current user
- **duplicate-sensor**: The given code is already linked to an existing sensor object
- **access-denied**: The user does not have sufficient privileges on the sensor (at least write privileges are required)
- **wrong-code**: The given code does not match an available physical sensor device

The `sdevice` field carries the hardware address of the new sensor device that the sensor was linked to as a hex encoded string and is only present if the device association was changed during this request.

DELETE /api/sensor/(*uuid*)

Delete sensor object `uuid`

Response body example

```
{}
```

Status

- **wrong-sensor**: The given UUID does not match a sensor accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the sensor (at least write privileges are required)

The sensor object is is permanently deleted (no `trash` option) and the association with the physical sensor device is removed. All persistent data associated with the sensor is deleted.

GET /api/sensor/(*uuid*)**/data**

Get the sensor data for sensor object `uuid`

Query Parameters

- **from** – optional: start time, in seconds-since-epoch (must have from <= to)
- **to** – optional: end time, in seconds-since-epoch (must have from <= to)

- **start** – optional: the start key of the first data object to return, exclusive, i.e. the result with an exact match data key (definition see below) is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.

- **order** – optional: If set to `asc` (default), data objects are returned in time ascending order (i.e. starting with `from` or `start` and ending with `stop`).

  If set to `desc`, data objects are returned in time descending order (i.e. starting with `to` or `start` and ending with `from`).

- **count** – optional, default `10`: limits the number of returned data objects (valid range: `1...50000`)

Response body example

```
{
    "data": {
        "1363623247000": [
            20.5,
            [1, 10, 50]
        ],
        "1363623307001": [
            null,
            [0, 500, 0]
        ],
        "1363623367000": [
            21,
            null
        ]
    }
}
```

Status

- **wrong-sensor**: The given UUID does not match a sensor object accessible to the current user

Within the `data` section, the requested data objects are listed in form of a JSON object.

The key within the `data` section consists of the data sampling time in seconds-since-epoch, multiplied by 1000, and increased by an arbitrary ID between 0 and 999 to make the number unique. The key should not be interpreted as a time in milliseconds. The time of the data record has only second precision and can be obtained by converting the key to an integer number and dividing the value by 1000.

The value within the `data` section is a JSON array. The entries in this array represent the data capabilities of the sensor (see *GET /api/sensor/(uuid)*). The number of entries in each JSON value array must exactly match the amount of data capabilities of the sensor. The type of values within the array depend on the specific data capabilities.

In this example, since the sensor has a temperature sensor at index `0` and a motion sensor at index `1`, each data value consists of two entries, the first one being a single JSON number (temperature in °C), the second one an array of motion detection count and detection time. Each entry might be null, to signal that the sensor did not deliver any data for the respective index.

POST /api/sensor/(*uuid*)**/data**

Store sensor data for sensor object `uuid`

Request body example

```
{
    "data": {
        "1363623247000": [
            20.5,
            [1, 10, 50]
        ],
        "1363623307001": [
            null,
            [0, 500, 0]
        ],
        "1363623367000": [
            21,
            null
        ]
    }
}
```

Response body example

```
{}
```

Status

- **wrong-sensor**: The given UUID does not match a sensor object accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the sensor (at least write privileges are required)
- **malformed-encoding**: The Content-Encoding is set to gzip but the payload can not be decompressed.

The body of the request might be gzip encoded which must then be indicated by setting the `Content-Encoding` HTTP header to `gzip`.

The data in the request body is given in form of a JSON object.

The keys of the JSON object consists of the data sampling time in milliseconds-since-epoch or of the data sampling time in seconds-since-epoch, multiplied by 1000, and increased by an arbitrary ID between 0 and 999 to make the number unique.

The values of the JSON object are JSON arrays. The entries in this array represent the data capabilities of the sensor (see *GET /api/sensor/(uuid)*). The number of entries in each JSON value array must exactly match the amount of data capabilities of the sensor. The type of values within the array depend on the specific data capabilities.

In this example, since the sensor has a temperature sensor at index `0` and a motion sensor at index `1`, each data value consists of two entries, the first one being a single JSON number (temperature in °C), the second one an array of motion detection count and detection time. Each entry might be `null`, to signal that the sensor did not deliver any data for the respective index.

GET /api/sensor/(*uuid*)**/action**

Get the sensor actions for sensor object `uuid`

Query Parameters

- **from** – optional: start time, in seconds-since-epoch (must have `from` <= `to`)
- **to** – optional: end time, in seconds-since-epoch (must have `from` <= `to`)

- **start** – optional: the start key of the first action object to return, exclusive, i.e. the result with an exact match action key (definition see below) is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.

- **order** – optional: If set to `asc` (default), action objects are returned in time ascending order (i.e. starting with `from` or `start` and ending with `stop`).

  If set to `desc`, action objects are returned in time descending order (i.e. starting with `to` or `start` and ending with `from`).

- **count** – optional, default `10`: limits the number of returned action objects (valid range: `1...50000`)

Response body example

```
{
    "action": {
        "1363623253000": [
            0,
            "Hello, World!"
        ],
        "1363623378001": [
            1,
            null
        ]
    }
}
```

Status

- **wrong-sensor**: The given UUID does not match a sensor object accessible to the current user

This call lists the time series of sensor action commands (i.e. actions that were sent to the sensor with the *POST /api/sensor/(uuid)/action* call)

Within the `action` section, the requested data objects are listed in form of a JSON object.

The key within the `action` section consists of the data sampling time in seconds-since-epoch, multiplied by 1000, and increased by an arbitrary ID between 0 and 999 to make the number unique. The key should not be interpreted as a time in milliseconds. The time of the data record has only second precision and can be obtained by converting the key to an integer number and dividing the value by 1000.

The value within the `action` section is a JSON array. The entries in this array represent the action capabilities of the sensor (see *GET /api/sensor/(uuid)*). The number of entries in each JSON value array must exactly match the amount of action capabilities of the sensor. The type of values within the array depend on the specific action capabilities.

GET /api/sensor/(*uuid*)**/data/**

xxx`.csv` Get the sensor data for sensor object `uuid` in CSV format

Query Parameters

- **from** – optional: start time, in seconds-since-epoch (must have `from` <= `to`)
- **to** – optional: end time, in seconds-since-epoch (must have `from` <= `to`)
- **inline** – optional: if present, the response will include an HTTP `Content-Disposition` header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
time,U/V,U/V,U/V,t/s,I/A,I/A,I/A,t/s
2014-06-04 13:00:00,230,225,233,300,1.2,1.98,1.34,300
2014-06-04 13:05:00,231,223,233,300,1.24,1.98,1.32,300
2014-06-04 13:10:00,230,222,233,300,1.22,1.98,1.34,300
2014-06-04 13:15:00,231,227,233,300,1.21,1.98,1.31,300
```

This API call allows for downloading of sensor data results in comma-separated-values format. The output is streaming, so very large result sets can be handled.

The fields in the CSV output correspond to the fields in the *GET /api/sensor/(uuid)/data* call depending on the data capabilities the sensor supports. If a sensor supports a multidimensional capability, the data is flattened in the downloaded CSV format. This means that every dimension of such a capability has its own header field in the CSV file and the corresponding data separated by commas. For more information on supported capabilities please see documentation of API call *GET /api/sensor/(uuid)*.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the sensor `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

GET /api/sensor/(*uuid*)**/action/***xxx***.csv** Get the sensor action for sensor object `uuid` in CSV format

Query Parameters

- **from** – optional: start time, in seconds-since-epoch (must have `from` <= `to`)
- **to** – optional: end time, in seconds-since-epoch (must have `from` <= `to`)
- **inline** – optional: if present, the response will include an HTTP Content-Disposition header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
time,state
2014-06-04 13:00:00,0
2014-06-04 13:05:00,1
2014-06-04 13:10:00,1
2014-06-04 13:15:00,0
```

This API call allows for downloading of sensor action data in comma-separated-values format. The output is streaming, so very large result sets can be handled.

The fields in the CSV output correspond to the fields in the *GET /api/sensor/(uuid)/action* call depending on the action capabilities the sensor supports. If a sensor supports a multidimensional capability, the data is flattened in the downloaded CSV format. This means that every dimension of such a capability has its own header field in the CSV file and the corresponding data separated by commas. For more information on supported capabilities please see documentation of API call *GET /api/sensor/(uuid)*.

This API is eligible for use with the session's download id (did). In this case, the object UUID is the sensor `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

POST /api/sensor/(*uuid*)**/action**
Triggers an action on the sensor `uuid`.

Request body example

```
{
    "action": [
        0,
```

```
        "Hello, World!"
    ]
}
```

- or -

```
{
    "action": [
        1,
        null
    ]
}
```

Response body example

```
{
    "action-uuid": "97a1f558-9167-11e2-892a-0024e8f90cc0"
}
```

Status

- **wrong-sensor**: The given UUID does not match a sensor accessible to the current user
- **sensor-offline**: The sensor is currently not visible through any devices, so the action cannot be delivered

The entries in the JSON array `action` must exactly match the `action` capabilities of the sensor (see *GET /api/sensor/(uuid)*). The type of values in the array depend on the specific `action` capabilities. In the first example, the sensor has an `onoff` capability at index `0`, which is set to the "off" state (value `0`) with this call, and a text capability at index `1` which is instructed to display the string "Hello, World!". In the second example, the onoff-switch is set to "on", while the text display remains unchanged.

Note that even though the number of elements in the `action` array must exactly match the number of `action` capabilities of the given sensor, a value of `null` might be given for a specific element to signal that no change is desired for this actor.

If the action was successfully queued to the sensor, the call will return an action UUID. A `sensor_action` event will be delivered to the user after an acknowledgement message has been received by the sensor to confirm that the action was successfully carried out.

GET /api/sensor/(*uuid*)**/event**

Subscribe the current session to the event flow of sensor `uuid`

Query Parameters

- **timeout** – optional: expiration timeout of the event flow in seconds. Default: 1 hour (3600 seconds), valid range 1..36000 seconds (10 hours)

Response body example

```
{}
```

Status

- **wrong-sensor**: The given UUID does not match a sensor object accessible to the current user

This API call will enable the flow of `sensor_data` and `sensor_action` events to the current session for the specified sensor object. The flow of events will automatically stop after the timeout period. If this behavior is not desired, this API call should be called again before expiration to re-new the timeout period.

DELETE /api/sensor/(*uuid*)**/event**
Cancel the subscription of the current session to the data events of sensor `uuid`

Response body example

```
{}
```

Status

- **wrong-sensor**: The given UUID does not match a sensor object accessible to the current user

This API call will stop the flow of sensor data events to the current session for the given sensor.

3.26 /api/xobject

POST /api/xobject
Create a new object of type xobject.

Request body example

```
{
    "description": "some custom object"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this xobject

Response body example

```
{
    "uuid": "be4b6102-e3b6-11e2-b2d6-0024e8f90cc0"
}
```

The `uuid` field returns the UUID of the newly created xobject.

The conceptual idea behind xobjects is to support a form of structured object storage in the cospace system that has no semantic representation in the platform itself. By using objects of type `xobject`, applications can leverage generic cospace features like attaching tags, comments and linking objects, storing metadata and passing events based on metadata without cospace having to understand the meaning of the application itself. In this way, generic use cases can be implemented on top of the cospace framework.

GET /api/xobject/(*uuid*)
Get information about the xobject `uuid`

Response body example

```
{
    "xobject": {
        "owner": "550e8400-e29b-41d4-a716-446655440000",
```

```
        "time": 1372839009,
        "description": "some custom object",
        "tag": [
            "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
            "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
        ]
    }
}
```

Status

- **wrong-xobject**: The given UUID does not match a xobject accessible to the current user

This API call is provided mainly to maintain consistency with other object calls. Since objects of type `xobject` don't have any real content besides the description field, applications might want to store custom information using metadata (see *POST /api/object/(uuid)/metadata*).

POST /api/xobject/(*uuid*)

Modifies xobject `uuid`

Request body example

```
{
    "description": "some other custom object"
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: a short description of this xobject

Response body example

```
{}
```

Status

- **wrong-xobject**: The given UUID does not match a xobject accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the xobject (at least write privileges are required)

Since objects of type `xobject` do not have any real content besides the description field, applications might want to store custom information using metadata (see *POST /api/object/(uuid)/metadata*).

DELETE /api/xobject/(*uuid*)

Delete xobject `uuid`

Parameters

- **purge** – optional: if present, delete this xobject permanently (do not move to trash)

Response body example

```
{}
```

Status

- **wrong-xobject**: The given UUID does not match a xobject accessible to the current user
- **access-denied**: The user does not have sufficient privileges on the xobject (at least write privileges are required)

3.27 /api/event

GET /api/event
Returns events pending for the session (event system)

Parameters

- **timeout** – optional: a timeout in seconds after which the call returns even if no event was triggered (valid range: `0...300`, default: `55`)
- **next** – optional: The next event that the client wants to see. For subsequent calls to this function, the value of the `next` field of the response (see below) can be used here (default: `0`)

Response body example

```
{
    "event": [
        {
            "id": 0,
            "event": "object_new",
            "time": 2182782732,
            "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
            "type": "contact",
            "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
            "description": "Hans Mustermann",
            "tag": [
                "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0"
            ]
        },
        {
            "id": 1,
            "event": "fax_report",
            "time": 2182782758,
            "fax": "94f1bb1a-5931-11e0-8db5-0024e8f90cc0",
            "report": {
                "7b149600-5921-11e0-b85f-0024e8f90cc0": {
                    "incoming": true,
                    "from": "+492216689711",
                    "to": "+492216689712",
                    "time_start": 2128383234,
                    "time_end": 2128384351,
                    "status": "ok"
                }
            }
        }
    ],
    "next": 2
}
```

JSON Parameters

- **timeout** – The specified (or default) timeout has occurred before an event was triggered

Events are ordered by ascending time (seconds-since-epoch). Please see chapter Generic Event JSON Considerations for a detailed description of the events and their JSON representation.

The `next` field specifies the next event (i.e., the one expected after the events returned with this call). It can be used for subsequent calls in the `next` URL parameter.

Note that the amount of events that are stored in the server side might be limited. So if events are not requested by a client for a long period of time, old events might be lost. The client can detect this condition by looking at the `id` fields: since ids are contiguous, a gap in the `id` fields indicates that events have been dropped on the server side.

`POST /api/event/filter`

Attaches a filter to the session's stream of events

Request body example

```
{
    "op": "or",
    "args": [
        {
            "op": "and",
            "args": [
                {
                    "op": "equal",
                    "key": "owner",
                    "value": "b4e38072-dcd1-11e2-a597-0024e8f90cc0"
                },
                {
                    "op": "rematch",
                    "key": "type",
                    "value": "sensor|box"
                }
            ]
        },
        {
            "op": "filter",
            "key": "object",
            "arg": {
                "op": "equal",
                "key": "owner",
                "value": "b4e38072-dcd1-11e2-a597-0024e8f90cc0"
            }
        }
    ]
}
```

Response body example

```
{}
```

With this API call, the client can attach a filter to the stream of events for the current session, effectively reducing the amount of events delivered to the client via the *`GET /api/event`* call. Every event that is received for the current session is matched against the filter, and only if it passes the filter it is queued for delivery to the client. Note that the filter works on a per-session level. A user might have several sessions with different filters. If a new session is created, it does not have a filter associated with it.

A filter consists of the operator (`op` field) and one or more additional fields specific for the selected operator. The following operators are supported:

- **equal**

Passes the event if it has a field with the name defined in the filter's `key` field that exactly matches the content given in the filter's `value` field. Any JSON type might be used for the `value` field, as the type is also part of the matching process.

•rematch

Passes the event if it has a field with the name defined in the filter's `key` field that is of JSON string type and matches the regular expression given in the filter's `value` field.

•greater

Passes the event if it has a field with the name defined in the filter's `key` field that is of JSON number type and has a value greater than the JSON number in the filter's `value` field.

•lesser

Passes the event if it has a field with the name defined in the filter's `key` field that is of JSON number type and has a value lesser than the JSON number in the filter's `value` field.

•contain

Passes the event if it has a field with the name defined in the filter's `key` field that is of JSON array type and this array contains an element that exactly matches the content given in the filter's `value` field. Any JSON type might be used for the `value` field, as the type is also part of the matching process.

•filter

Passes the event if it has a field with the name defined in the filter's `key` field that is of JSON object type and this object passes the filter given in the filter's value field. With this operator, matching of fields that are nested in deeper JSON structures is possible, since the filter in the `value` field is now applied one level deeper than the original filter.

•and

This filter has an additional `args` field that is an array of JSON objects, each in turn being a new filter that is applied to the event. If all of these new filters match, then the event is passed.

•or

This filter has an additional `args` field that is an array of JSON objects, each in turn being a new filter that is applied to the event. If at least one of these new filters matches, then the event is passed.

•not

This filter has an additional `arg` field of type JSON object, which in turn is a new filter that is applied to the event. The result of this new filter is negated, i.e. the outer filter passes the event if the inner filter would reject it and the other way around.

The filter given in the example above will pass the event via the *`GET /api/event`* call if either condition is true:

- the event has an `owner` field with value `b4e38072-dcd1-11e2-a597-0024e8f90cc0` and the event has a `type` field with value `sensor` or `box` (regular expression match).
- the event has an `object` field which in turn is a JSON object that has an `owner` field with value `b4e38072-dcd1-11e2-a597-0024e8f90cc0`.

If the event filtering fails because there is an error in the filter syntax that could not be detected when the filter was attached, the event will pass and will be delivered with the *`GET /api/event`* call. In this case, a `filter_error` field will be present in the event so that the filter failure can be detected by the client.

`DELETE /api/event/filter`

Removes a session filter from the event system

Response body example

```
{}
```

This API call will remove a filter previously applied to the session's event stream.

CHAPTER

FOUR

DIALPLAN API

The dialplan handles the routing of incoming calls.

4.1 /api/dialplan/v2

The dialplan v2 is a replacement for the old dialplan routing implementation.

GET /api/dialplan/v2
Get list of dialplan objects

Response body example

```
{
    "dialplan" : [
        {
            "uuid": "a4a33462-c88a-42f7-be35-ae06775fde6e",
            "description": "A description text"
        },
        {
            "uuid": "b6774686-2bc5-4af0-9dbf-33f4427adbd9",
            "description": "Holidays"
        },
        {
            "uuid": "84126d2c-b2d7-4e39-8455-fc4788515928",
            "description": "Busy season"
        }
    ]
}
```

POST /api/dialplan/v2
Creates a new dialplan.

Request body example

```
{
    "dialplan" : {
        "description": "A description text",
        "ringtone": {
            "announcement": "91e85f5c-b9eb-11e1-8b7e-0024e8f90cc0",
            "mix": true
        },
        "profiles": [
            {
                "filter": {
```

```
                    "caller": "\+49221.*|\+4830123456|anonymous",
                    "time": {
                        "weekdays": ["MON","TUE","WED","THU","FRI","SAT","SUN"],
                        "time_start": "07:30",
                        "time_end": "13:00"
                    }
                },
                "linear": {
                    "timeout" : 7,
                    "targets": [
                        {
                            "target": "joe_05",
                            "timeout": 5
                        },
                        {
                            "target": "+491771234567",
                            "timeout": 4
                        }
                    ]
                }
            },
            {
                "filter": {
                    "time": {
                        "date": "11.06.2015",
                        "time_start": "07:30",
                        "time_end": "13:00"
                    }
                },
                "parallel": {
                    "timeout" : 10,
                    "targets": [
                        "joe_04",
                        "+491771234567"
                    ]
                }
            },
            {
                "cyclic": {
                    "timeout": 7,
                    "targets": [
                        {
                            "target": "joe_08",
                            "timeout": 4
                        },
                        {
                            "target": "+491771234567",
                            "timeout": 5
                        }
                    ]
                }
            },
            {
                "record": {
                    "announcement": "b6774686-2bc5-4af0-9dbf-33f4427adbd9"
                }
            }
        ]
```

```
    }
}
```

JSON Parameters

- **description** – ^.{0,200}$

  optional: a short description of this dialplan

- **ringtone** – optional: The section to configure a "custom ringback tone"

- **ringtone.announcement** – UUID

  optional: announcement to be played during the connection process

- **ringtone.mix** – true|false

  optional: if true, a ringing tone is mixed with the ringtone.announcement

- **profiles** – optional: first to last element in this list will be evaluated. The elements of profiles contain a filter and an action object. The action object is mandatory it will be executed. Only one action object in a profile element is allowed.

  Possible action's:

  - reject: will hang up the call.
  - record: will start mailbox recording.
  - parallel: will originate calls to all targets in parallel.
  - linear: will originate calls to all targets in a sequential way. if a target is busy or timeouts the next will be connected. The starting point is the first list element.
  - cyclic: originate calls to all targets in a sequential way. if a target is busy or timeouts the next will be connected. The starting point will be choosen by Round-robin scheduling.

- **filter** – optional: if the filter object isn't present or is present and the all criteria matches the action object will be executed

- **caller** – regular expression (max. 500 characters)

  optional: if the regex doesn't match the next rule in the profiles will be evaluated. Examples: "anonymous" matches calls with clir enabled. "+49.*" matches calls for Germany. "+4922112345" exact match

- **time** – optional: You can define a time range and/or a date and/or date range and/or weekdays. if the field date is present, the fields weekdays, date_start and date_end should not be used.

- **date** – dd.MM.yyyy

  optional: if the current day doesn't match the given date the next rule in the profiles will be evaluated.

- **date_start** – dd.MM.yyyy

  optional: if the current day doesn't follow the given date the next rule in the profiles will be evaluated.

- **date_end** – dd.MM.yyyy

  optional: if the current day doesn't precede the given date the next rule in the profiles will be evaluated. date_end should not precede date_start

- **weekdays** – optional: if the current day doesn't match the given list of `weekdays` the next rule in the `profiles` will be evaluated.
- **time_start** – `hh:mm`

  optional: the start time of this time filter. The server timezone is UTC.
- **time_end"** – `hh:mm`

  optional: the end time of this time filter. If `time_start` is not earlier than the `time_end`, then the `time_end` refers to the following day.
- **(parallel|linear|cyclic).timeout** – optional: the timeout in seconds of this action. The default value is 0. if set to lower than 1 this action won't stop until network disconnect.
- **parallel.targets** – mandatory: must be present with at least one `target`.
- **(linear|cyclic).targets** – mandatory: must be present with at least one element. Elements are objects with a `target` and a `timeout`.
- **target** – `^\+[0-9]{3,20}$|^[a-z0-9]{2,20}_[0-9]{2}$`

  optional: the phone number or the SIP account target. The solucon SIP accounts must be owned by this account.
- **(linear|cyclic).targets.timeout** – optional: the timeout in seconds of the ringing action. The default value is 0. if set to lower than 1 the ringing won't stop until network disconnect.

Response body example

```
{
    "uuid": "6e33d20f-2052-4e1e-9109-868c06bc5a17"
}
```

The UUID of the newly created dialplan is returned.

Status

- **wrong-announcement**: This announcement is not a valid announcement accessible from the user

GET /api/dialplan/v2/ (*uuid*)
Get dialplan `uuid`

Response body example

```
{
    "dialplan" : {
        "description": "Busy season",
        "profiles": [
            {
                "reject": {}
            }
        ]
    }
}
```

For a description of the JSON fields see *POST /api/dialplan/v2*

Status

- **wrong-announcement**: This announcement is not a valid announcement accessible from the user
- **wrong-dialplan**: The given UUID does not match to a dialplan accessible by the current user

POST /api/dialplan/v2/ (*uuid*)

Modifies dialplan `uuid`

Request body example

```
{
    "dialplan" : {
        "description": "Holidays",
        "profiles": [
            {
                "record": {
                    "announcement" : "b6774686-2bc5-4af0-9dbf-33f4427adbd9"
                }
            }
        ]
    }
}
```

Response body example

```
{}
```

For a description of the JSON fields see *POST /api/dialplan/v2*

Status

- **wrong-dialplan**: The given UUID does not match to a dialplan accessible by the current user
- **wrong-announcement**: This announcement is not a valid announcement accessible from the user

DELETE /api/dialplan/v2/ (*uuid*)

Delete the dialplan `uuid`

Response body example

```
{}
```

Status

- **wrong-dialplan**: The given UUID does not match to a dialplan accessible by the current user

When the dialplan is active in :http:get:'/api/phone' it will be removed there as well. This will end in a unintended number configuration.

4.2 /api/dialplan

GET /api/dialplan

Get the configuration of the user's dialplan.

Response body example

```
{
    "announcement": "1de1227a-4f34-11e0-ab6e-0024e8f90cc1",
    "mix_ringtone": false,
    "clip_number": "+492216698980",
    "dial_prefix": "+492216698",
    "clir_enable": false,
    "cti_via": "+492216698777",
    "action": "none",
    "phone": "+492216698999",
    "sip": "joe_01",
    "phone_pool": {
        "+492216698991": true,
        "+492216698992": false
    },
    "sip_pool": [
        "joe_08",
        "joe_13"
    ],
    "callreverse_enable": false,
    "timeout": {
        "timeout": 20,
        "action": "record"
    },
    "unavailable": {
        "action": "redirect",
        "phone": "+492216698998",
        "sip": "joe_01"
    },
    "busy": {
        "action": "sip",
        "sip": "joe_07"
    }
}
```

The `sip` and `phone` fields can exist independently of the selection action. E.g. if the action is `sip`, a phone number can be configured for the respective session, however it will not be used. Each entry in the `phone_pool` section is either active (value `true`, the number will be dialled if the dialplan gets called) or inactive (value `false`).

`POST /api/dialplan`

Modify the user's dialplan configuration

Request body example

```
{
    "announcement": "91e85f5c-b9eb-11e1-8b7e-0024e8f90cc0",
    "mix_ringtone": true,
    "clip_number": "+492216698888",
    "dial_prefix": "+492216698",
    "clir_enable": false,
    "cti_via": "+492216698777",
    "action": "none",
    "phone": "+492216698001",
    "sip": "joe_01",
    "phone_pool": {
        "add": {
            "+492216698990": false,
            "+492216698995": true
```

```
        },
        "delete": [
            "+492216698992"
        ]
    },
    "sip_pool": {
        "add": [
            "joe_01",
            "joe_03"
        ],
        "delete": [
            "joe_02"
        ]
    },
    "callreverse_enable": false,
    "timeout": {
        "timeout": 20,
        "action": "record"
    },
    "unavailable": {
        "action": "redirect",
        "phone": "+492216698993"
    },
    "busy": {
        "action": "sip",
        "sip": "joe_01"
    }
}
```

JSON Parameters

- **announcement** – `UUID`

  optional: announcement to be played during the connection process

- **mix_ringtone** – `true|false`

  optional: if true, a ringing tone is mixed with the announcement ("custom ringback tone")

- **action** – `none|redirect|reject|sip|record`

  optional: the action to proceed with if the condition is reached

- **phone** – `^\+[0-9]{3,20}$`

  optional: the phone number target for action redirect

- **sip** – `^[a-z0-9]{2,20}_[0-9]{2}$`

  optional: the SIP account target for action sip

- **dial_prefix** – `^\+[0-9]{3,20}$`

  optional: prefix for quick dialing

- **clip_number** – `^\+[0-9]{3,20}$`

  optional: custom CLIP number

- **clir_enable** – `true|false`

  optional: to enable / disable CLIR feature

- **cti_via** – `^\+[0-9]{3,20}$`

  optional: the user's phone number to receive callback calls on

- **callreverse_enable** – `true|false`

  optional: if `true`, enables the call-reverse function

Response body example

```
{}
```

Status

- **wrong-announcement**: The given announcement does not match an announcement accessible by the current user
- **wrong-sip**: The given SIP account is not one of the current user's SIP accounts

All fields and structures in this request are optional.

If entries listed in the `add` section of the `phone_pool` refer to existing phone numbers, they are overwritten.

To delete the `announcement`, `phone`, `dial_prefix`, `clip_number`, `cti_via` or `sip` fields, an empty string value can be given for these fields.

In the case of `timeout` or `unavailable`, the action `none` is not possible.

If present, the `dial_prefix` specifies a phone number that will be prepended to the original dialed number if the user (SIP account) dials a short number (quick dialing).

If present, the `clip_number` will specify the phone number that is transmitted as the calling line identification for outgoing calls made by a SIP account (PSTN CLIP feature).

If `clir_enable` is `true`, the transmission of the calling line identification is suppressed (PSTN CLIR feature).

If `callreverse_enable` is `true`, a call to the user's dialplan will activate the call-reverse function as an additional target just like the targets in the `phone_pool` and `sip_pool`.

POST /api/dialplan/callthrough

Initiate a call-through function

Request body example

```
{
    "phone": "+492216698001",
    "filter": "+492118271982"
}
```

JSON Parameters

- **phone** – `^([a-zA-Z0-9]{2,20})|(\+[0-9]{3,20})$`

  optional: the target (e.g. a phone number) to call in the call-through function.

- **filter** – `^\+[0-9]{3,20}$`

  optional: a phone number that has to match to trigger the call-through function.

Response body example

```
{}
```

All fields and structures in this request are optional.

When this API method is called, a new call-through process is started. The system saves the given `phone` and optional `filter` numbers. When the user's dialplan is called from an external number that matches the `filter`, or when the short-dialling code `*98` is dialled from one of the user's SIP accounts, the call will be connected to the destination phone.

There can only be one active call-through process per user. The process state is reset when the call-through is initiated or with any subsequent call to this API method. If this method is called without the phone field, an outstanding call-through request is cleared. Outstanding call-through requests will also time out after an appropriate time interval (typically 1-2 minutes).

POST /api/dialplan/callback

Initiate a callback call

Request body example

```
{
    "from": "+492118271982",
    "to": "+492216698711"
}
```

JSON Parameters

- **from** – `^\+[0-9]{3,20}$`

  the originating phone number

- **to** – `^([a-zA-Z0-9]{2,20})|(\+[0-9]{3,20})$`

  the destination target (e.g. a phone number)

Response body example

```
{}
```

Status

- **wrong-phone**: The given `from` number does not match any of the user's phone numbers or the `dialplan_enable` flag is not set on the phone number
- **no-via**: No `cti_via` field is set in the user's dial plan
- **access-denied**: The user has no permissions to use the callback feature

This API call initiates a call back procedure. First, a call is iniated towards the user's `cti_via` phone number (see *GET /api/dialplan* and *POST /api/dialplan*). When the user answers the call, the given to number will be called and bridged to the first call. The from number will used as an originating caller number on both call legs.

The callback feature will utilize the `clip_number`, `clir_number` and `cti_via` values of the user's dialplan.

POST /api/dialplan/callreverse

Set the filter for an ongoing call-reverse function

Request body example

```
{
    "filter": "+492118271982"
}
```

JSON Parameters

- **filter** – `^\+[0-9]{3,20}$`

  a phone number that has to match to trigger the call-reverse function.

Response body example

```
{}
```

Status

- **no-callreverse**: There is no active call-reverse process

This API method should be called if an ongoing call-reverse process has previously been signalled to the user with a `dialplan_callreverse` event. A phone number can be set as a `filter`, and if this phone number calls the user's dialplan number, the incoming call-reverse call will be connected to this call.

CHAPTER

FIVE

INTERACTIVE VOICE RESPONSE API

The following section describes the API calls required to create, modify, list and delete IVRs. As with other API calls, the IVR API also requires a valid and authenticated user session.

An IVR is defined by the actions to execute once a DTMF signal is received. IVRs can be cascaded to allow multiple levels.

5.1 API Call Reference

5.1.1 /api/ivr

POST /api/ivr

Creates a new IVR and returns the `uuid` of the newly created IVR.

Request body example

```
{
    "description" : "Company Welcome",
    "announcement" : "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
    "1" : "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
    "2" : "ce19884c-8fed-11e2-93fe-0024e8f90cc3",
    "4" : "+4922116696000",
    "*" : "ce19884c-8fed-11e2-93fe-0024e8f90cc5",
    "other" : "repeat",
    "timeout" : {
        "seconds": 10,
        "action": "play",
        "announcement": "ce19884c-8fed-11e2-93fe-0024e8f90cc2"
    }
}
```

JSON Parameters

- **description** – `^.{1,200}$`

  mandatory: A user-given description for the IVR.

- **announcement** – `UUID`

  mandatory: The initial announcement to be played at the beginning.

- **[0-9*#]** – `UUID | ^\\+[0-9]{3,20}$`

mandatory: at least one field with a valid DTMF digit must be given. A valid DTMF must be one of `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, #, *]`. Each field defines the action to proceed if this DTMF is received:

- an *UUID* - a new IVR whose UUID is given will run.
- a *phone number* - the call will be redirected to the given phone number.

- **`other`** – `repeat | ^\\+[0-9]{3,20}$`

  optional: define the action to execute if an unspecified DTMF signal is received:

  - `repeat` - the initial announcement will be repeated.
  - a *phone number* - the call will be redirected to the given phone number

- **`timeout`** – optional: this block defines what happens if no DTMF signal was received for a certain amount of time.
- **`timeout.seconds`** – `integer > 0`

  mandatory: the timeout duration in seconds.

- **`timeout.action`** – `repeat | call | play`

  mandatory: define the action to proceed after a timeout from the end of the announcement without receiving a DTMF already defined in this IVR.

  - `repeat` - the IVR will be repeated.
  - `call` - the call will be redirected.
  - `play` - an other announcement will be played.

- **`timeout.phone`** – `^\\+[0-9]{3,20}$`

  mandatory if `action` is `call`: the phone number to call.

- **`timeout.announcement`** – `UUID`

  mandatory if `action` is `play`: the announcement to be played.

Response body example

```
{
    "uuid": "ce19884c-8fed-11e2-93fe-0024e8f90cc0"
}
```

`POST /api/ivr/`(*uuid*)

Updates an existing IVR.

Request body example

For the description of the request body see *`POST /api/ivr`*

Response body example

```
{}
```

Status

- **wrong-ivr**: The IVR with ID `uuid` does not exist or does not belong to the user.

`GET /api/ivr`

Lists the current user's IVRs.

Query Parameters

- **`from`** – optional: start time, in seconds-since-epoch (must have from <= to)
- **`to`** – optional: end time, in seconds-since-epoch (must have from <= to)
- **`start`** – optional: the UUID of the first IVR to return, exclusive, i.e. the IVR with an exact match UUID is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.
- **`order`** – optional: If set to `asc` (default), IVRs are returned in time ascending order (i.e. starting with from or start and ending with stop).

  If set to `desc`, IVRs are returned in time descending order (i.e. starting with to or start and ending with from).
- **`count`** – optional, default `30`: limits the number of returned IVRs (valid range: `1...500`)

Response body example

```
{
    "ivr" : [
        {
            "description" : "Company Welcome",
            "uuid" : "db19884c-8fed-11e2-93fe-0024e8f90aa3"
        },
        {
            "description" : "Company Support",
            "uuid" : "cd19882b-8fed-11e2-93fe-0024e8f90ab2"
        }
    ]
}
```

`GET` `/api/ivr/` (*uuid*)
Gets the details of the specified IVR.

Response body example

```
{
    "ivr" : {
        "description" : "Company Welcome",
        "announcement" : "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
        "1" : "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
        "2" : "ce19884c-8fed-11e2-93fe-0024e8f90cc3",
        "4" : "+4922116696000",
        "*" : "ce19884c-8fed-11e2-93fe-0024e8f90cc5",
        "other" : "repeat",
        "timeout" : {
            "seconds": 10,
            "action": "play",
            "announcement": "ce19884c-8fed-11e2-93fe-0024e8f90cc2"
        }
    }
}
```

Status

- **wrong-ivr**: The IVR with ID `uuid` does not exist or does not belong to the user.

DELETE /api/ivr/(*uuid*)
Deletes the specified IVR.

Response body example

```
{}
```

Status

- **wrong-ivr**: The IVR with ID `uuid` does not exist or does not belong to the user.

CHAPTER

SIX

GENERIC EVENT JSON CONSIDERATIONS

Each API event (as returned with the *GET /api/event* call) has a specific JSON representation that is defined in the following sections. Besides the specific elements, there is also a common structure, i.e. some fields that are present in all events. The generic JSON representation of any event („event skeleton") looks like this:

```
{
    "id": 42,
    "event": "object_new",
    "time": 2182782732
}
```

The `id` field is a unique identifier of the event *in the context of the current session*. The first event of a newly created session has id `0`, and the id is increased by `1` with every new event. This enables a client to detect whether the stream of events is contiguous, and it enables multiple clients to share the event stream of one server-side session (because every client knows what events it has already seen). The `time` field lists the time of event generation in seconds-since-epoch.

CHAPTER

SEVEN

EVENT REFERENCE

7.1 User events

7.1.1 Event user_link_new

When a new link is established between two users, this event is sent to both users

```
{
    "user": "a1edadee-9ccd-11e0-ba16-0024e8f90cc0"
}
```

The `user` field contains the UUID of the other party.

7.1.2 Event user_link_delete

When a link between two users is deleted, this event is sent to both users

```
{
    "user": "a1edadee-9ccd-11e0-ba16-0024e8f90cc0"
}
```

The `user` field contains the UUID of the other party.

7.1.3 Event user_modify

When the user settings of a user are modified, this event is sent to the user

```
{}
```

7.2 Invitation events

7.2.1 Event invitation_new

When an internal invitation is created, this event is sent both to the originator and the recipient of the invitation

```
{
    "owner": "2a578836-e7b3-11e1-8f2c-0024e8f90cc0",
    "user": "a1edadee-9ccd-11e0-ba16-0024e8f90cc0"
}
```

The `owner` field contains the UUID of the invitation originator.

The `user` field contains the UUID of the other party, the invited person.

7.2.2 Event invitation_delete

When an internal invitation is deleted by either party of the invitation, this event is sent both to the originator and the recipient of the invitation

```
{
    "owner": "2a578836-e7b3-11e1-8f2c-0024e8f90cc0",
    "user": "a1edadee-9ccd-11e0-ba16-0024e8f90cc0"
}
```

The `owner` field contains the UUID of the invitation originator.

The `user` field contains the UUID of the other party, the invited person.

7.3 Phone events

7.3.1 Event phone_modify

This event is sent to a user if her phone configuration is modified (usually via the *POST /api/phone* call). The event carries no additional payload, it just indicates the modification. Modified data can be retrieved using the *GET /api/phone* call.

```
{}
```

7.4 Tag events

7.4.1 Event tag_new

When a new tag is created, this event is sent to all users that have any access level within the tag (including the tag owner):

```
{
    "tag": "7b3785b8-974f-11e0-a900-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "label": "myTag",
    "foreign_use": true,
    "policy": {
        "1de7257a-4f34-11e0-ab6e-0024e8f90cc1": 4,
        "c5dbbdd4-0ad9-11e1-bd0d-0024e8f90cc0": 2,
        "c63ff81c-0ad9-11e1-bb94-0024e8f90cc0": 3
    }
}
```

The `tag` field contains the UUID of the new tag.

The `owner` field contains the UUID of user that owns the tag.

The `policy` section lists the access users of the tag with their respective access privileges (values are `1` = read, `2` = write, `3` = propagate, `4` = owner).

7.4.2 Event tag_modify

When an existing tag is modified (e.g., the label or the access policy is changed), this event is sent to all users that have at least read access before *or* after the modification.

```
{
    "tag": "7b3785b8-974f-11e0-a900-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "label": "myTag",
    "foreign_use": true,
    "policy": {
        "1de7257a-4f34-11e0-ab6e-0024e8f90cc1": 4,
        "c5dbbdd4-0ad9-11e1-bd0d-0024e8f90cc0": 3,
        "c63ff81c-0ad9-11e1-bb94-0024e8f90cc0": 0
    }
}
```

The `tag` field contains the UUID of the tag.

The `owner` field contains the UUID of user that owns the tag.

The `policy` section lists the access users of the tag with their respective access privileges (values are `0` = no access, `1` = read, `2` = write, `3` = propagate, `4` = owner). Note that in this case, a value of `0` is sent if the respective user has lost access the tag because of the modification.

7.4.3 Event tag_delete

When an existing tag is deleted, this event is sent to all users that had at least read access before the operation.

```
{
    "tag": "7b3785b8-974f-11e0-a900-0024e8f90cc0"
}
```

The `tag` field contains the UUID of the deleted tag.

7.5 Trash events

7.5.1 Event trash_purge

This event is sent to the user if her trash is purged, i.e. all objects in the trash are permanently deleted.

```
{}
```

7.6 Generic object events

7.6.1 Event object_new

When a new object is created, this event is sent to all users that access to the object (i.e. in most cases solely the owner).

```
{
    "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "type": "contact",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "description": "Hans Mustermann",
    "tag": [
        "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0",
        "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
        "97d3a606-a961-11e0-9a43-0024e8f90cc0"
    ]
}
```

The `object` field contains the UUID of the object.

The `type` field contains the object type, e.g. `fax`, `contact`, `recording`, ...

The `owner` field contains the UUID of user who owns the object.

The `description` field, if present, contains a short, human-readable description of the object. This may be the actual description field of the object (if the object type has one), or the name of a contact or something similar.

The `tag` section contains a list of all tags attached to the object.

7.6.2 Event object_modify

When an existing object is modified (i.e. its fields are modified), this event is sent to all users that have at least read access before *and* after the modification. This event is *not* sent if only comments or attached tags are modified: specific events exist for these cases.

```
{
    "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "type": "contact",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user": "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
    "description": "Hans Mustermann"
}
```

The `object` field contains the UUID of the object.

The `type` field contains the object type, e.g. `fax`, `contact`, `recording`, ...

The `owner` field contains the UUID of user who owns the object.

The `user` field contains the UUID of user who submitted the modification that triggered this event, if this action was initiated by a user. If not, this field will not be present.

The `description` field, if present, contains a short, human-readable description of the object. This may be the actual description field of the object (if the object type has one), or the name of a contact or something similar.

7.6.3 Event object_delete

When an existing object is deleted, this event is sent to all users that had at least read access before the delete operation.

```
{
    "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "type": "contact",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user": "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
    "description": "Hans Mustermann",
    "purge": true
}
```

The `object` field contains the UUID of the object.

The `type` field contains the object type, e.g. `fax`, `contact`, `recording`, ...

The `owner` field contains the UUID of user who owned the object.

The `user` field contains the UUID of user who deleted the object, if this action was initiated by a user. If not, this field will not be present.

The `description` field, if present, contains a short, human-readable description of the object. This may be the actual description field of the object (if the object type has one), or the name of a contact or something similar.

The `purge` field, if present, indicates that this is a permanent delete ("delete-from-trash").

7.6.4 Event object_share

When an object is shared with a new user for the first time by attaching a tag that grants access to this user, an event of this type is sent towards the user.

```
{
    "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "type": "contact",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user": "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
    "description": "Hans Mustermann",
    "tag": [
        "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0",
        "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
        "97d3a606-a961-11e0-9a43-0024e8f90cc0"
    ]
}
```

The `object` field contains the UUID of the object.

The `type` field contains the object type, e.g. `fax`, `contact`, `recording`, ...

The `owner` field contains the UUID of user who owned the object.

The `user` field contains the UUID of user who shared the object, if this action was initiated by a user. If not, this field will not be present.

The `description` field, if present, contains a short, human-readable description of the object. This may be the actual description field of the object (if the object type has one), or the name of a contact or something similar.

The `tag` section contains a list of all tags currently attached to the object.

7.6.5 Event object_tag

When tags are attached to or detached from an existing object, this event is sent to all users that have at least read access before *and* after the modification, and that have at least access to one of the tags added or deleted to the object.

```
{
    "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "type": "contact",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user": "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
    "description": "Hans Mustermann",
    "tag": [
        "be00e8aa-5c3d-11e0-b0cf-0024e8f90cc0",
        "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
        "97d3a606-a961-11e0-9a43-0024e8f90cc0"
    ],
    "tag_add": [
        "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
        "97d3a606-a961-11e0-9a43-0024e8f90cc0"
    ],
    "tag_delete": [
        "0f3441b4-1141-11e1-8829-0024e8f90cc0"
    ]
}
```

The `object` field contains the UUID of the object.

The `type` field contains the object type, e.g. `fax`, `contact`, `recording`, ...

The `owner` field contains the UUID of user who owned the object.

The `user` field contains the UUID of user who modified the tag attachments on the object, if this action was initiated by a user. If not, this field will not be present.

The `description` field, if present, contains a short, human-readable description of the object. This may be the actual description field of the object (if the object type has one), or the name of a contact or something similar.

The `tag` section contains a list of all tags currently attached to the object.

The `tag_add` section contains a list of all tags added to the object during this operation.

The `tag_delete` section contains a list of all tags deleted from the object during this operation.

7.6.6 Event object_unshare

When an object gets invisible to some user because the tag that grants access is removed, this user gets an event of this type.

```
{
    "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "type": "contact",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user": "ae30e5b0-5c3d-11e0-817e-0024e8f90cc0",
    "description": "Hans Mustermann"
}
```

The `object` field contains the UUID of the object.

The `type` field contains the object type, e.g. `fax`, `contact`, `recording`, ...

The `owner` field contains the UUID of user who owned the object.

The `user` field contains the UUID of user who removed the tag attachments on the object, if this action was initiated by a user. If not, this field will not be present.

The `description` field, if present, contains a short, human-readable description of the object. This may be the actual description field of the object (if the object type has one), or the name of a contact or something similar.

7.6.7 Event object_link_new

When a new link is established between two objects, this event is sent to all users that have at least read privileges on both objects

```
{
    "a": {
        "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
        "type": "contact",
        "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
        "description": "Hans Mustermann"
    },
    "b": {
        "object": "3b4ed57c-79da-11e2-8e15-0024e8f90cc0",
        "type": "fax",
        "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
        "description": "Coffee Order"
    }
}
```

The `a` and `b` sections contain information about both objects that are now connected. Note that the description field may not be present.

7.6.8 Event object_link_delete

When a link between two objects is deleted, this event is sent to all users that have at least read privileges on both objects. The event is not sent in case that one of the objects is deleted. In this case, the object connection ends implicitely.

```
{
    "a": {
        "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
        "type": "contact",
        "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
        "description": "Hans Mustermann"
    },
    "b": {
        "object": "3b4ed57c-79da-11e2-8e15-0024e8f90cc0",
        "type": "fax",
        "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
        "description": "Coffee Order"
    }
}
```

The `a` and `b` sections contain information about both objects that were previously connected. Note that the description field may not be present.

7.7 Comment events

7.7.1 Event comment_new

When a new comment is added to an object, this event is sent to all users that access to the object.

```
{
    "comment": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "text": "This is a new comment",
    "object": {
        "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
        "type": "contact",
        "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
        "description": "Hans Mustermann"
    }
}
```

The `comment` field contains the UUID of the comment.

The `owner` field contains the UUID of the user who owns the comment.

The `text` field contains the full text content of the comment.

The `object` section contains the object's UUID (`object`), `type`, `owner` and a textual description of the object (`description`).

7.7.2 Event comment_modify

When an existing comment is modified, this event is sent to all users that access to the object.

```
{
    "comment": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "text": "I changed the text of this comment",
    "object": {
        "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
        "type": "contact",
        "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
        "description": "Hans Mustermann"
    }
}
```

The `comment` field contains the UUID of the comment.

The `owner` field contains the UUID of the user who owns the comment.

The `text` field contains the new full text content of the comment.

The `object` section contains the object's UUID (`object`), `type`, `owner` and a textual description of the object (`description`).

7.7.3 Event comment_delete

When an existing comment is deleted, this event is sent to all users that access to the object.

```
{
    "comment": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "object": {
        "object": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
        "type": "contact",
        "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
        "description": "Hans Mustermann"
    }
}
```

The `comment` field contains the UUID of the comment.

The `owner` field contains the UUID of the user who owned the comment.

The `object` section contains the object's UUID (`object`), `type`, `owner` and a textual description of the object (`description`).

7.8 Metadata events

7.8.1 Event user_metadata

When the user's metadata is changed, this event is sent to all sessions of that user

```
{
    "update": {
        "com.otherdomain.client.nice": true,
        "com.otherdomain.client.timestamp": 1324899645
    },
    "delete": [
        "com.otherdomain.client.something"
    ]
}
```

The `update` sections contain the metadata elements (key and value) that have been added or modified.

The `delete` array contains the metadata keys that have been deleted.

7.8.2 Event object_metadata

When an object's metadata is changed, this event is sent to all users that have at least read access to the object.

```
{
    "object": "94f1bb1a-5931-11e0-8db5-0024e8f90cc0",
    "user": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "update": {
        "com.otherdomain.client.nice": true,
        "com.otherdomain.client.timestamp": 1324899645
    },
    "delete": [
        "com.otherdomain.client.something"
    ]
}
```

The `object` field contains the UUID of the object for which metadata elements were changed.

The `user` field contains the UUID of the user that did the change.

The `update` sections contain the metadata elements (key and value) that have been added or modified.

The `delete` array contains the metadata keys that have been deleted.

7.9 Fax events

7.9.1 Event fax_convert

When the fax contents of an existing fax are updated (after an upload and the following format conversion are finished), this event is sent to all users that have at least read access.

```
{
    "fax": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "page_count": 12
}
```

The `fax` field contains the UUID of the fax object.

In case of successful format conversion, the `page_count` field is present and contains the number of pages.

7.9.2 Event fax_report

This event is sent to all users that have at least read access to a fax in case a send/receive status change ("fax report") is updated for this fax.

```
{
    "fax": "94f1bb1a-5931-11e0-8db5-0024e8f90cc0",
    "report": {
        "7b149600-5921-11e0-b85f-0024e8f90cc0": {
            "incoming": true,
            "from": "+492216689711",
            "to": "+492216689712",
            "time_start": 2128383234,
            "time_end": 2128384351,
            "status": "ok",
            "sip_status_code": 200,
            "fax_error_code": 0
        }
    }
}
```

The `fax` field contains the UUID of the fax object.

The fax report itself is encapsulated in the `report` section of the event. See the API documentation of the *GET /api/fax/(uuid)* API call for details about the fax report.

7.10 Announcement events

7.10.1 Event announcement_convert

When the announcement contents of an existing announcement are updated (after an upload and the following format conversion are finished), this event is sent to all users that have at least read access.

```
{
    "announcement": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "success": true
}
```

The `announcement` field contains the UUID of the announcement object.

In case of successful conversion, the `success` field will be present with a value of `true`.

7.11 Conference events

7.11.1 Event conference_start

When the first participant enters an active conference, this event is sent to all users that have at least read access to the conference. It serves as a signal that this conference is now „live".

```
{
    "conference": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "description": "My Conference Room"
}
```

The `conference` field contains the UUID of the conference object.

The `owner` field contains the UUID of user who owned the object.

The `description` field, if present, contains a short, human-readable description of the conference.

7.11.2 Event conference_stop

When the last participant exits an active conference, this event is sent to all users that have at least read access to the conference.

```
{
    "conference": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "description": "My Conference Room"
}
```

The `conference` field contains the UUID of the conference object.

The `owner` field contains the UUID of user who owned the object.

The `description` field, if present, contains a short, human-readable description of the conference.

7.11.3 Event conference_status

To receive this event, the user needs to subscribe to the detailed event stream of the conference using the *GET /api/conference/(uuid)/event* call.

The server will send this event once after receiving the event stream subscription, and may send this event again in regular intervals as long as the subscription exists.

```
{
    "conference": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "member": {
        "9fe1cc2a-8ab6-11e0-b368-0024e8f90cc0": {
            "phone": "+492216698000",
            "user": "2d07f7be-8abc-11e0-8166-0024e8f90cc0",
            "mute": false,
            "deaf": false,
            "talk": true,
            "floor": true,
            "volume_in": 0,
            "volume_out": 0
        },
        "b1439c46-8ab6-11e0-acdc-0024e8f90cc0": {
            "phone": "+49241441010",
            "mute": false,
            "deaf": false,
            "talk": true,
            "floor": false,
            "volume_in": 0,
            "volume_out": 0
        }
    }
}
```

The `conference` field contains the UUID of the conference object.

The `member` section lists all participants of the conference, indexed by a UUID that is used to uniquely identify them.

The `phone` field is only present if the participant transmitted her originating phone number with the call.

The `user` field, connects the conference participant to a user account of a system. This field is only present if the participant was successfully connected to a user UUID via the specific API call.

Volume levels range from `-4` to `4`, where `0` is the default volume level.

7.11.4 Event conference_update

To receive this event, the user needs to subscribe to the detailed event stream of the conference using the *GET /api/conference/(uuid)/event* call.

The server will send this event if the current status of one or more conference members has changed.

```
{
    "conference": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "member": {
        "9fe1cc2a-8ab6-11e0-b368-0024e8f90cc0": {
            "talk": false,
            "floor": false
        },
        "b1439c46-8ab6-11e0-acdc-0024e8f90cc0": {
```

```
            "mute": true,
            "talk": false
        }
    }
}
```

The `member` sections only lists those members that actually have changes, and within the member only those fields that actually represent changes to the previous `conference_status` / `conference_update` event are present.

The fields in the event are identical to the definition in the `conference_status` event.

7.11.5 Event conference_join

To receive this event, the user needs to subscribe to the detailed event stream of the conference using the *GET /api/conference/(uuid)/event* call.

The server will send this event if one or more new conference participants join the conference.

```
{
    "conference": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "member": {
        "b20116b8-8acf-11e0-b38e-0024e8f90cc0": {
            "phone": "+492216698123",
            "mute": false,
            "deaf": false,
            "talk": true,
            "floor": true,
            "volume_in": 0,
            "volume_out": 0
        }
    }
}
```

The fields in the event are identical to the definition in the `conference_status` event.

7.11.6 Event conference_leave

To receive this event, the user needs to subscribe to the detailed event stream of the conference using the *GET /api/conference/(uuid)/event* call.

The server will send this event if one or more new conference participants leave the conference.

```
{
    "conference": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "member": [
        "b20116b8-8acf-11e0-b38e-0024e8f90cc0",
        "359e049a-8ad0-11e0-8a33-0024e8f90cc0"
    ]
}
```

The `member` sections lists the participant UUIDs that left the conference.

7.12 File events

7.12.1 Event file_new

When a file or folder is created within a volume object, this event is sent to all users that have at least read access to the volume.


```
{
    "commit": "383a5354-362f-11e3-837a-0024e8f90cc0",
    "file": "8aaf6000-474f-11e1-b0ba-0024e8f90cc0",
    "volume": "6a8432ec-474f-11e1-8495-0024e8f90cc0",
    "type": "file",
    "name": "letter.pdf",
    "parent": "9e6bde98-474f-11e1-9a62-0024e8f90cc0",
    "ctime": 20232142212,
    "mtime": 21232145373,
    "user": "bc93f42a-45d6-11e1-b856-0024e8f90cc0"
}
```


The `commit` field contains the commit UUID that this event relates to.

The `file` field contains the UUID of the file or folder (the name of the field is always file).

The `volume` field contains the UUID of the volume which contains the file.

The `type` field has a value of `file` if this event is about a file, or a value of `folder` if it is about a folder.

`name` is the file name of the file or folder.

The `parent` field is only present if the file or folder was not created on the root-level of the volume; it contains the parent folder UUID that the file or folder was created in.

The `ctime` and `mtime` fields contain the creation an modification timestamps of the file (seconds-since-epoch). Usually these reflect the current time if they were not set to a different value during file creation.

The `user` field contains the UUID of the user that created the file or folder.

7.12.2 Event file_modify

When a file or folder is modified, this event is sent to all users that have at least read access to the volume.

Note that there are no "hierarchical" events, that is, if a file is created, modified or deleted, no event will be sent for the folder that contains the file.


```
{
    "commit": "400f9be8-362f-11e3-b868-0024e8f90cc0",
    "file": "8aaf6000-474f-11e1-b0ba-0024e8f90cc0",
    "volume": "6a8432ec-474f-11e1-8495-0024e8f90cc0",
    "type": "file",
    "name": "letter2.pdf",
    "parent": "d5edfe04-4750-11e1-be06-0024e8f90cc0",
    "ctime": 20232142212,
    "mtime": 21232145373,
    "user": "bc93f42a-45d6-11e1-b856-0024e8f90cc0",
    "size": 386231,
    "mime_type": "application/pdf",
    "md5": "92f2e0728cd03376ed13a191734cc065"
}
```

The `commit` field contains the commit UUID that this event relates to.

The `file` field contains the UUID of the file or folder (the name of the field is always file).

The `volume` field contains the UUID of the volume which contains the file.

The `type` field has a value of `file` if this event is about a file, or a value of `folder` if it is about a folder.

The `name` field is only present if the file name of the file or folder was changed; it contains the new file name of the file or folder.

The `parent` field is only present if the file or folder was re-located (moved) within the volume.; it contains the new parent folder UUID that the file or folder was moved to.

The `ctime` and `mtime` fields contain the creation an modification timestamps of the file (seconds-since-epoch). Either field is only present if the value has changed.

The `user` field contains the UUID of the user that modified the file or folder.

The `size`, `mime_type` and `md5` fields contain the file size (in octets), the MIME type (for example "image/png") and the md5 checksum of the file contents. These fields are only present if new content was uploaded to the file.

7.12.3 Event file_delete

When a file or folder is deleted, this event is sent to all users that have at least read access to the volume.

Note that if a folder is deleted, all the contents of this folder is recursively deleted without sending file_delete events for every element that was deleted in the recursion.

```
{
    "commit": "46bf0e74-362f-11e3-a637-0024e8f90cc0",
    "file": "8aaf6000-474f-11e1-b0ba-0024e8f90cc0",
    "volume": "6a8432ec-474f-11e1-8495-0024e8f90cc0"
}
```

The `commit` field contains the commit UUID that this event relates to.

The `file` field contains the UUID of the file or folder (the name of the field is always file). As a special case, the `file` field may also contain the UUID of the volume itself to signal that all of the volume's content were deleted (root-level delete).

The `volume` field contains the UUID of the volume which contains the file.

7.13 Box events

7.13.1 Event box_online

When a device comes online, this event is sent to all users that have access to the associated `box` object.

```
{
    "box": "964ae742-77fe-11e1-89b7-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "description": "box in office basement",
    "ip_addr": "192.168.0.13",
    "mac_addr": "00:24:d6:83:c1:54",
    "sw_version": "1.0",
    "hw_version": "X86"
}
```

The `box` field contains the UUID of the box object.

The `owner` field contains the UUID of user who ownes the object.

The `description` field, if present, contains a short, human-readable description of the box object.

The `ip_addr` field contains the current IP address of the device.

The `mac_addr` field contains the current MAC address of the device.

The `sw_version` field contains the running software version of the device.

The `hw_version` field contains the hardware model of the device.

7.13.2 Event box_offline

When a device goes offline, this event is sent to all users that have access to the associated box object.

```
{
    "box": "964ae742-77fe-11e1-89b7-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "description": "box in office basement"
}
```

The `box` field contains the UUID of the box object.

The `owner` field contains the UUID of user who ownes the object.

The `description` field, if present, contains a short, human-readable description of the box object.

7.14 Dialplan events

7.14.1 Event dialplan_modify

When a user's dialplan is modified, this event it sent to the user.

```
{}
```

7.14.2 Event dialplan_callreverse_start

When an incoming phone call to the user's dialplan activates the call-reverse function, this event is sent to the user.

```
{
    "phone_from": "+4989123248278",
    "phone_to": "+492216698000"
}
```

The `phone_from` field contains the calling phone number.

The `phone_to` field contains the called phone number.

7.14.3 Event dialplan_callreverse_stop

When a call-reverse function ends, this event is sent to the user.

```
{
    "phone_from": "+4989123248278",
    "phone_to": "+492216698000"
}
```

The `phone_from` field contains the calling phone number.

The `phone_to` field contains the called phone number.

7.15 SIP events

7.15.1 Event sip_new

When a SIP account is created, this event is sent to the user that owns the account.

```
{
    "user": "joe_01",
    "domain": "cospace.de",
    "description": "office phone"
}
```

The `user` field contains the user part of the SIP account.

The `domain` field contains the domain part of the SIP account.

The `description` field will only be present if a description was set when creating the account.

7.15.2 Event sip_modify

When a SIP account is modified, this event is sent to the user that owns the account.

```
{
    "user": "joe_01",
    "domain": "cospace.de",
    "description": "softclient smartphone"
}
```

The `user` field contains the user part of the SIP account.

The `domain` field contains the domain part of the SIP account.

The `description` field will only be present if the description of the account was actually changed.

7.15.3 Event sip_delete

When a SIP account is deleted, this event is sent to the user that owned the account.

```
{
    "user": "joe_01",
    "domain": "cospace.de"
}
```

The `user` field contains the user part of the SIP account.

The `domain` field contains the domain part of the SIP account.

7.15.4 Event sip_register

When a SIP end-device registers with a SIP account, this event is sent to the user that owns the account.

```
{
    "user": "joe_01",
    "domain": "cospace.de",
    "contact": "212.202.0.32:5060",
    "expire": 1340107084
}
```

The `user` field contains the user part of the SIP account.

The `domain` field contains the domain part of the SIP account.

The `contact` field contains the IP address (and optional a port number) of the endpoint, the so-called contact information of the SIP registration.

The `expire` field contains the absolute time of registration expiration in seconds-since-epoch.

7.15.5 Event sip_unregister

When a SIP account is modified, this event is sent to the user that owns the account.

```
{
    "user": "joe_01",
    "domain": "cospace.de",
    "contact": "212.202.0.32:5060"
}
```

The `user` field contains the user part of the SIP account.

The `domain` field contains the domain part of the SIP account.

The `contact` field contains the IP address (and optional a port number) of the endpoint, the so-called contact information of the SIP registration.

7.16 Presentation events

7.16.1 Event presentation_convert

When the page contents of an existing presentation are updated (after an upload and the following format conversion is finished), this event is sent to all users that have at least read access.

```
{
    "presentation": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "page_count": 12
}
```

The presentation field contains the UUID of the presentation object.

In case of successful format conversion, the `page_count` field is present and contains the number of pages.

7.16.2 Event presentation_start

When a presentation is started, this event is sent to all users that have at least read access to the conference. It serves as a signal that this presentation is now „live".

```
{
    "presentation": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user": "cc8a411a-13ab-11e2-bf75-0024e8f90cc0",
    "description": "Why The Sky Is Black At Nighttime"
}
```

The `presentation` field contains the UUID of the presentation object.

The `owner` field contains the UUID of user who owns the object.

The `user` field contains the UUID of user who started the presentation and is now the first member and moderator of the conference.

The `description` field, if present, contains a short, human-readable description of the presentation.

7.16.3 Event presentation_stop

When a presentation is ended, this event is sent to all users that have at least read access to the presentation.

```
{
    "presentation": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "owner": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user": "cc8a411a-13ab-11e2-bf75-0024e8f90cc0",
    "description": "Why The Sky Is Black At Nighttime"
}
```

The `presentation` field contains the UUID of the presentation object.

The `owner` field contains the UUID of user who owns the object.

The `user` field contains the UUID of user who ended the presentation (and who was the moderator). This field might be missing if the presentation is not actively ended by the moderator, but through other means (e.g., a timeout or failure condition).

The `description` field, if present, contains a short, human-readable description of the presentation.

7.16.4 Event presentation_join

To receive this event, the user needs to subscribe to the detailed event stream of the presentation using the *`GET /api/presentation/(uuid)/event`* call.

This event is sent when a new member joins the presentation. It will not be sent for the first conference member (the first member included in the `presentation_start` event).

```
{
    "presentation": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "user": "cc8a411a-13ab-11e2-bf75-0024e8f90cc0"
}
```

The `presentation` field contains the UUID of the presentation object.

The `user` field contains the UUID of user who joined the presentation.

7.16.5 Event presentation_leave

To receive this event, the user needs to subscribe to the detailed event stream of the presentation using the *GET /api/presentation/(uuid)/event* call.

This event is sent when a member leaves the presentation. It will not be sent if the conference is stopped completely (in this case, the `conference_stop` should be considered as an implicit leave for all conference members).

```
{
    "presentation": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "user": "cc8a411a-13ab-11e2-bf75-0024e8f90cc0"
}
```

The `presentation` field contains the UUID of the presentation object.

The `user` field contains the UUID of user who left the presentation.

7.16.6 Event presentation_status

To receive this event, the user needs to subscribe to the detailed event stream of the presentation using the *GET /api/presentation/(uuid)/event* call.

The server will send this event once after receiving the event stream subscription, and may send this event again in regular intervals as long as the subscription exists.

```
{
    "presentation": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "page": 2,
    "x": 0.45,
    "y": 0.8,
    "visible": true,
    "moderator": "2d515d81-a459-471b-917b-bcf6424575b1",
    "member": [
        "3068ef3f-eba4-421a-90e6-02f443ac52ff",
        "2d515d81-a459-471b-917b-bcf6424575b1",
        "e1eaa085-58b7-4e56-8fcb-581393bd7bc6"
    ]
}
```

The `presentation` field contains the UUID of the presentation object.

The `page` field contains number of the current page.

The `x` field contains horizontal position of the pointer between 0.0 (left) and 1.0 (right).

The `y` field contains vertical position of the pointer between 0.0 (top) and 1.0 (bottom).

The `visible` field denotes whether the pointer is visible (`true`) or not (`false`).

The `moderator` field contains the UUID of the moderator.

The `member` array lists all participants of the presentation.

7.16.7 Event presentation_update

To receive this event, the user needs to subscribe to the detailed event stream of the presentation using the *GET /api/presentation/(uuid)/event* call.

The server will send this event if one or more properties of the presentation have changed.

```
{
    "presentation": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "page": 3,
    "x": 0.53,
    "y": 0.8,
    "visible": true,
    "moderator": "2d515d81-a459-471b-917b-bcf6424575b1"
}
```

The fields in the event are a subset of the fields in the `presentation_status` event (see there).

All fields all optional, usually the server will only send fields that have changed values since the last event.

7.16.8 Event presentation_keepalive

This event is sent periodically to all members who have joined a presentation. Clients are requested to answer with a keepalive control message (*`POST /api/presentation/(uuid)/control`*) to refresh the join state at the presentation controller. Clients that receive this event and that are not joined to the presentation should *not* send a leave control message, because other clients for the same user might be joining the presentation.

```
{
    "presentation": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0"
}
```

The presentation field contains the UUID of the presentation object.

7.17 Chat events

7.17.1 Event chat_new

When a chat is created, this event is sent to the user who created the chat (directly after creating a chat, this user is the only one in the chat).

```
{
    "chat": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "create_time": 1362587184
}
```

The `chat` field contains the UUID of the chat.

The `create_time` field contains the creation time of the chat in seconds-since-epoch.

7.17.2 Event chat_message

When a message is sent within a chat, this event is sent to all users participating in the chat.

```
{
    "chat": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "message": {
        "c1aa1566-84d7-11e2-a7a8-0024e8f90cc0": {
            "user": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
            "time": 1362407297,
            "type": "text",
```

```
            "text": "what's for lunch today?"
        }
    }
}
```

The `chat` field contains the UUID of the chat.

The `message` section corresponds to the format of the *GET /api/chat/(uuid)/message* call. For a description about possible message types, refer to the *POST /api/chat/(uuid)/message* call.

If `type` is `add`, this event will be the first for this chat that is delivered to the user who entered the chat. The `member` field will show the user that was added to the chat. In this case, a `create_time` field will be present in the main section of this event that contains the creation time of the chat in seconds-since-epoch and a `description` field will be present if the chat has a description.

If `type` is `leave`, this event will be the last for this chat that is delivered to the user who left the chat. Since a user can only leave the chat by herself, the `user` field will also tell who left the chat.

7.17.3 Event chat_user

When the expiration time of a transient user state within a chat changes, this event is sent to all users participating in the chat.

```
{
    "chat": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "user": {
        "1de7257a-4f34-11e0-ab6e-0024e8f90cc1": {
            "typing": 10
        }
    }
}
```

The `chat` field contains the UUID of the chat.

The `user` section corresponds to the format of the *GET /api/chat/(uuid)/user* call. This event will not deliver a complete list of user states, but will only contain states that have a changed expiration time. The `message_read` field will be present if the message read pointer was modified for the corresponding user.

Note that the reception of a `chat_user` event does not in every case mean that the user state actually changed, but is used only to deliver a modified expiration time for the transient change to other participant. Actual state changes will only happen if an expire value > 0 is given for a state that is currently inactive in the client, or if a timer set by this event expires. As a special case, this event might carry a timer of 0, which will immediately set the state to inactive.

7.18 Call events

Please note that all events described in this chapter are part of the call API and only delivered in case that the user enables call API events with the *GET /api/call/event* method.

7.18.1 Event call_update

This event signals changes in the call parameters for the call given in the `call` section (the key is the call UUID). The fields match those defined in the *GET /api/call* method. Only parameters that are actually changed will be included in the event.

```
{
    "call": {
        "2b3b70ce-8677-11e2-a7c1-0024e8f90cc0": {
            "answer_time": 1362587197,
            "status": "answered"
        }
    }
}
```

In the special case of the `current_control` parameter, an empty `value` string signals that no control is currently operating on the call (in the *GET /api/call* method, the `current_control` field will be absent in this case). An empty `bridge` field signals that the call is no longer bridged to another call.

7.18.2 Event call_dtmf

If DTMF detection and delivery is enabled on a call, incoming DTMF digits will be delivered via this event.

```
{
    "call": "2b3b70ce-8677-11e2-a7c1-0024e8f90cc0",
    "digit": "2",
    "duration": 143
}
```

The `digit` field contains the DTMF digit that was detected.

The `duration` field contains duration of the digit.

7.19 Sensor events

7.19.1 Event sensor_data

When a sensor delivers data to the cloud, this event is delivered to all user sessions that requested the delivery of sensor data events with *GET /api/sensor/(uuid)/event*.

```
{
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc0",
    "data": {
        "1363623247000": [
            20.5,
            [
                0,
                255
            ]
        ]
    }
}
```

The `sensor` field holds the UUID of the sensor object.

The `data` section lists the new sensor data, typically a single entry. The format is the same as in the *GET /api/sensor/(uuid)/data* call.

7.19.2 Event sensor_action

When an action was carried out on a sensor, this event is delivered to all user sessions that requested the delivery of sensor data events with *GET /api/sensor/(uuid)/event*.


```
{
    "status": "ok",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc0",
    "action": {
        "1363623253000": [
            1,
            null
        ]
    },
    "action-uuid": "97a1f558-9167-11e2-892a-0024e8f90cc0"
}
```


The `status` field contains `ok` if the action was carried out. It might contain `replaced` if the action was superseded by another action while waiting for the sensor to become available for radio transmittion.

The `action` section lists the action that was carried out, typically a single entry. The format is the same as in the *GET /api/sensor/(uuid)/action* call.

The `action-uuid` field holds the action UUID that corresponds to the `action-uuid` field in the response of the *POST /api/sensor/(uuid)/action* call.

CHAPTER

EIGHT

APPLE PUSH NOTIFICATIONS & GOOGLE CLOUD MESSAGING

8.1 API Call Reference

GET /api/push

Get the current user's configuration for Apple Push Notification service (APNs) and Google Cloud Messaging for Android (GCM).

Response body example

```
{
    "gcm": {
        "sender_id": "4815162342",
        "registration_ids": [
            "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...",
            "Iausdj89apcoaSDdojfAsidfAs8d84-5621i..."
        ]
    },
    "apns": {
        "device_tokens": {
            "a6461732069737d42065496e...f39ad": {
                "application_id": "39JSDKGJ38.de.cospace.app",
                "sandbox": true
            },
            "a6281e62e55932df9ec72829...5e2d3": {
                "application_id": "39JSDKGJ38.de.cospace.app",
                "sandbox": false
            }
        }
    }
}
```

The `gcm` section lists the items necessary to use Google Cloud Messaging for Android (GCM). The `sender_id` is needed by the App to apply for a registration ID that identifies the combination of App and Device as a receiver for cloud messages (push notifications). The `registration_ids` array lists all registration IDs that are stored for the current user.

The `apns` section lists the items necessary to use Apple Push Notification service (APNs). In the `device_tokens` section, all active device tokens are listed as keys, with their respective application ID in the value (the application ID consists of the application's bundle ID prefixed with a ten-character code generated by Apple). Device tokens are 32-byte octet strings and represented here as 64-character hexadecimal strings.

POST /api/push

Add or delete tokens for Apple Push Notification service (APNs) or registration IDs for Google Cloud Messaging for Android (GCM).

Request body example

```
{
    "system": "gcm",
    "action": "add",
    "registration_id": "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx..."
}
```

- or -

```
{
    "system": "gcm",
    "action": "delete",
    "registration_id": "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx..."
}
```

- or -

```
{
    "system": "apns",
    "action": "add",
    "device_token": "a6461732069737d42065496e...f39ad",
    "sandbox": true,
    "application_id": "39JSDKGJ38.de.cospace.app"
}
```

- or -

```
{
    "system": "apns",
    "action": "delete",
    "device_token": "a6461732069737d42065496e...f39ad"
}
```

JSON Parameters

- **system** – `gcm|apns`

  selects operation for Apple (APNs) or Google (GCM)

- **action** – `add|delete`
- **registration_id** – `^.{10,250}$`

  only for GCM: the registration ID

- **device_token** – `^[0-9a-fA-F]{64}$`

  only for APNs: the device token

- **sandbox** – `true|false`

  optional; only for APNs and action add: selects the sandbox APNS

- **application_id** – `^.{10,200}$`

  only for APNs and action add: the application ID

Response body example

```
{}
```

JSON Parameters

- **status** –
  - **unknown-appid**: The application ID is not known to the system (see note below)

The `system` field selects Apple Push Notification service (`apns`) or Google Cloud Messaging for Android (`gcm`) as the basis of the operation. The following fields depend on this choice.

The `action` field defines whether this API call will result in a new registration ID / device token being registered with the system (`add`) or an existing entry to be removed from the system (`delete`).

For GCM, the `registration_id` must be specified with both the add or delete operation.

For APNs, both the `device_token` and the `application_id` need to be specified with the `add` operation. For the `delete` operation, it is sufficient to specify only the `device_token`. The `sandbox` field specifies whether notifications for this device token should be send to the sandbox APNS.

Please note that for APNs, since Apple requires custom certificates for each application ID, it is necessary to contact the cospace team if you want to integrate a new app. No such restriction exists for GCM.

8.2 Push Event Reference

8.2.1 Push Event fax_new

This push event is sent to all registered devices when a new fax has been received by the system.

GCM content:

```
{
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "fax_new",
    "fax": "af729cc0-3251-11e2-81c1-0800200c9a66",
    "phone": "+492215587412",
    "page_count": "2"
}
```

APNs content:

```
{
    "aps": {
        "content-available": 1
    },
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "fax_new",
    "fax": "af729cc0-3251-11e2-81c1-0800200c9a66",
    "phone": "+492215587412",
    "page_count": 2
}
```

The `owner` field contains the UUID of the user who received the fax.

The `fax` field contains the UUID of the new fax, the `phone` field contains the phone number of the sender and the `page_count` field contains the number of pages in the fax.

Note that for GCM, `page_count` is of type string because GCM suggests to only use strings as values in the notification payload.

8.2.2 Push Event recording_new

This push event is sent to all registered devices when a new recording has been received by the system.

GCM content:

```
{
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "recording_new",
    "recording": "af729cc0-3251-11e2-81c1-0800200c9a66",
    "phone": "+492215587412"
}
```

APNs content:

```
{
    "aps": {
        "content-available": 1
    },
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "recording_new",
    "recording": "af729cc0-3251-11e2-81c1-0800200c9a66",
    "phone": "+492215587412"
}
```

The `owner` field contains the UUID of the user who received the recording.

The `recording` field contains the UUID of the new recording and the phone field contains the phone number of the caller.

8.2.3 Push Event dialplan_callreverse_start

This push event is sent to all registered devices when a call-reverse function has been started.

GCM content:

```
{
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "dialplan_callreverse_start",
    "phone_from": "+4989123248278",
    "phone_to": "+492216698000"
}
```

APNs content:

```
{
    "aps": {
        "content-available": 1
    },
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "dialplan_callreverse_start",
    "phone_from": "+4989123248278",
    "phone_to": "+492216698000"
}
```

The `owner` field contains the UUID of the user who started the call-reverse function.

The `phone_from` field contains the caller's phone number (origination number).

The `phone_to` field contains the called phone number (destination number).

8.2.4 Push Event dialplan_callreverse_stop

This push event is sent to all registered devices when a call-reverse function has been stopped.

GCM content:

```
{
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "dialplan_callreverse_stop",
    "phone_from": "+4989123248278",
    "phone_to": "+492216698000"
}
```

APNs content:

```
{
    "aps": {
        "content-available": 1
    },
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "dialplan_callreverse_stop",
    "phone_from": "+4989123248278",
    "phone_to": "+492216698000"
}
```

The `owner` field contains the UUID of the user who started the call-reverse function.

The `phone_from` field contains the caller's phone number (origination number).

The `phone_to` field contains the called phone number (destination number).

8.2.5 Push Event chat_message

This push event is sent to all registered devices when a chat message is delivered to a user.

GCM content:

```
{
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "chat_message",
    "chat": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "uuid": "c1aa1566-84d7-11e2-a7a8-0024e8f90cc0",
    "time": "1362407297",
    "user": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user_name": "John Doe",
    "type": "text",
    "text": "what's for lunch today?"
}
```

APNs content:

```
{
    "aps": {
        "content-available": 1
    },
```

```
    "owner": "9aed5e94-3803-11e3-8475-0024e8f90cc0",
    "message": "chat_message",
    "chat": "52a9fd3c-8aad-11e0-9138-0024e8f90cc0",
    "uuid": "c1aa1566-84d7-11e2-a7a8-0024e8f90cc0",
    "time": 1362407297,
    "user": "1de7257a-4f34-11e0-ab6e-0024e8f90cc1",
    "user_name": "John Doe",
    "type": "text",
    "text": "what's for lunch today?"
}
```

The `owner` field contains the UUID of the user who got this chat message.

The `chat` field contains the UUID of the chat, the `uuid` field contains the UUID of the message.

The push event contains the fields of the chat message as described in the API call *`GET /api/chat/(uuid)/message`* (in this case `user`, `type` and `text`). The additional `user_name` field contains the full name of the user.

Note that for GCM, `time` is of type string because GCM suggests to only use strings as values in the notification payload.

CHAPTER

NINE

PARTNER API

9.1 API Call Reference

9.1.1 /api/partner/user

GET /api/partner/user/(user-uuid)
Get user-related information of the user with UUID `user-uuid` that was created under the current partner.

Response body example


```
{
    "user": {
        "username": "johndoe47",
        "firstname": "John",
        "lastname": "Doe",
        "display_name": "Johnny Doe",
        "country_code": "+49",
        "ignore_ip_whitelisting": true,
        "usergroup": "Accounting",
        "email": "john@doe.com",
        "language": "de",
    },
    "phone": {
         "+492216698712": {
             "conference_enable": false,
             "fax_enable": true,
             "recording_enable": true,
             "notification_email": true,
             "notification_attachment": true
         },
         "+491234567890": {
             "conference_enable": true,
             "fax_enable": false,
             "recording_enable": false,
             "notification_email": false,
             "notification_attachment": false,
             "p_asserted_identity": "+492216698712"
         }
     },
     "feature": {
         "pack": [
             "20GB+",
             "COSPACE-BOX"
         ]
```

```
    }
}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner

This API call requires a session that is authenticated with a partner. The `phone` section will only be included if the partner controls phone number assignments for its users.

The `feature` section, `pack` subsection lists the feature packs assigned to the user.

GET /api/partner/user

Lists all partner related users.

Query Parameters

- **from** –

  optional: start time, in seconds-since-epoch when the user was created (must have from <= to)

  query to optional: end time, in seconds-since-epoch when the user was created (must have from <= to)

  query start optional: the UUID of the first user to return, exclusive, i.e. the user with an exact match UUID is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.

  query order optional: If set to `asc` (default), users are returned in time ascending order (i.e. starting with from or start and ending with stop).

  If set to `desc`, users are returned in time descending order (i.e. starting with to or start and ending with from).

  query count optional, default `50`: limits the number of returned users (valid range: `1...500`)

Response body example

```
{
    "users": [
        "3d6371b0-b9d8-11e5-b5f6-f8a96351ccd6",
        "4418f930-b9d8-11e5-b5f6-f8a96351ccd6",
        "4475e640-b9d8-11e5-b5f6-f8a96351ccd6"
    ]
}
```

This API call requires a session that is authenticated with a partner.

POST /api/partner/user/(user-uuid)

Allows a partner to temporarily disable or permanently delete the user UUID `user-uuid` that was created under the current partner.

Request body example

```
{
    "country_code": "+49",
    "disabled": true,
    "feature": {
        "pack": {
            "add": [
                "20GB+",
                "COSPACE-BOX"
            ],
            "delete": [
                "10GB+"
            ]
        }
    }
}
```

JSON Parameters

- **country_code** – `^\+[0-9]{1,3}$`

  Optional; the country code of the user.

- **ignore_ip_whitelisting** – `true|false`

  Optional; the user's ip address is not taken into account for nomadic use restrictions (and therefore disables this feature for this user)

- **disabled** – `true|false`

  Optional; `true` means the user account will be temporarily disabled, `false` means the user account will be enabled again.

- **deleted** – `true`

  Optional; if set to `true`, the user account will be permanently deleted. *This action cannot be undone.*

Response body example

```
{}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner

This API call requires a session that is authenticated with a partner.

Within the `feature` and `pack` section, feature packs can be associated with the user by listing them in the `add` subsection, or they can be deleted from the user having them in the `delete` subsection.

POST /api/partner/recovery/(user-uuid)

Start password recovery procedure and create a password recovery code for the user UUID `user-uuid` that was created under the current partner.

Request body example

```
{}
```

Response body example

```
{
    "code": "iausDia8sdfasjb9aaobasdfDF"
}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner

This API call requires a session that is authenticated with a partner.

9.1.2 /api/partner/phone

`GET /api/partner/phone`
Lists all partner related phones and the associated users.

Query Parameters

- **`start`** – optional: the first phone number to return, exclusive, i.e. the phone number will not be returned, if it is an exact match.
- **`count`** – optional, default `50`: limits the number of returned users (valid range: `1...500`)

Response body example

```
{
    "phones": {
        "+4922166966910": "48999930-d3b3-11e4-b265-f8a96351ccd6",
        "+4922166966920": "4418f930-b9d8-11e5-b5f6-f8a96351ccd6",
        "+4922166966930": "4418f930-b9d8-11e5-b5f6-f8a96351ccd6",
        "+4922166966940": "4418f930-b9d8-11e5-b5f6-f8a96351ccd6"
    }
}
```

This API call requires a session that is authenticated with a partner.

`POST /api/partner/phone/(user-uuid)`
Modify phone configuration for the user UUID `user-uuid` that was created under the current partner.

Request body example

```
{
    "create": {
        "+492216698123": {
            "dialplan_enable": true,
            "call_enable": false,
            "recording_enable": false,
            "fax_enable": false,
            "conference_enable": false,
            "play_announcement_only": false,
            "notification_email": true
        }
    },
    "update": {
        "+492216698712": {
            "conference_enable": false,
            "notification_email": true,
            "notification_attachment": true
```

```
        },
        "+491796548354": {
            "fax_enable": true,
            "conference_enable": false,
            "notification_attachment": false
        },
        "+49221669871200": {
            "p_asserted_identity": "+492216698712"
        }
    },
    "delete": [
        "+492211110004",
        "+492211110006"
    ]
}
```

JSON Parameters

- **phonenumber** – `^\+[0-9]{3,20}$`
- **p_asserted_identity** – `^\+[0-9]{3,20}$` Optional; The given number will used for billing and law issues. It must be a correct number. Setting the field to `null` will remove the field.

Response body example

```
{}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner
- **wrong-phone**: A phone number listed in the `update` or `delete` section does not belong to this user
- **phone-duplicate**: A phone number listed in the `update` section is already assigned to some user
- **phone-forbidden**: The current partner does not have the right to control phone number assignments for its users

This call requires a session that is authenticated with a partner. Additionally, the partner must have the right to assign phone numbers to its users.

The numbers in the `create` section are added to the user's list of phone numbers, the numbers in the `update` section of the request body are modified in the current user's phone data and the numbers in the `delete` section are deleted from the user's phone data.

The `dialplan_enable`, `call_enable`, `fax_enable`, `conference_enable` and `recording_enable` elements in the `create` and `update` sections are all optional (`dialplan_enable` and `call_enable` default to `false`, all other switches default to `true`).

Any combination of the `*_enable` switches is possible; however `dialplan_enable` takes priority, i.e. if `dialplan_enable` is `true`, the setting of `call_enable`, `fax_enable`, `conference_enable` and `recording_enable` is ignored for that number. The call API (`call_enable`) will only work if `call_enable` is the only switch that is `true`.

The `play_announcement_only` field is optional and defaults to `false`. If set to `true`, the system will not record a voice message after the announcement has been played in voice recorder mode.

The `notification_email` and `notification_attachment` elements in the `create` and `update` sections are optional (default `false`).

See *`POST /api/phone`* for further information. The field's `announcement` and `on_hold_music` can't be set with this call.

9.1.3 /api/partner/callfilter

`POST /api/partner/callfilter/(user-uuid)`
Modify a user's call filter.

Request body example

```
{
    "callfilter": [
        {
            "operator": "replace",
            "key": "^00",
            "value": "+"
        },
        {
            "operator": "replace",
            "key": "^0",
            "value": "+49"
        },
        {
            "operator": "replace",
            "key": "^+4911",
            "value": "11"
        },
        [
            {
                "operator": "service",
                "key": "sip",
                "reverse": true,
                "break": true
            },
            {
                "operator": "match",
                "key": "^\\+492216698711$",
                "pass": false
            },
            {
                "operator": "match",
                "key": "^\\+492216698499$",
                "pass": false
            },
            {
                "operator": "match",
                "key": "^(\\+49[2-8][0-9]{3,16})|(\\+499[1-9][0-9]{3,15}
↪)|(\\+4990[1-9][0-9]{3,14})$",
                "pass": true
            },
            {
                "operator": "match",
```

```
                "key": "^[1]{2}[0-7][0-9]{0,3}$",
                "pass": true
            }
        ],
        [
            {
                "operator": "service",
                "key": "fax",
                "reverse": true,
                "break": true
            },
            {
                "operator": "match",
                "key": "^(\\+49[2-8][0-9]{3,16})|(\\+499[1-9][0-9]{3,15}
↪)|(\\+4990[1-9][0-9]{3,14})$",
                "pass": true
            }
        ],
        {
            "operator": "match",
            "key": ".+",
            "pass": false
        }
    ]
}
```

Response body example

```
{}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner
- **too-large**: The given call filter exceeds the limit of 250 operators

This API call requires a session that is authenticated with a partner.

The partner can attach a call filter to user accounts that were created under the partner. Every time a user tries to make an outgoing phone call (e.g. send a fax, or place a call using SIP), the call filter will be checked in turn.

A filter consists of the operator and one or more more additional fields specific for the selected operator.

Operators can be nested using JSON arrays. The linear evaluation of such an array can be canceled by using the `break` field in certain operators. An operator can also directly allow or deny the call (skipping all following checks), using the `pass` field in certain operators. If nothing matches or the filter is empty, the call will be allowed. Every filter must contain the opening string call filter on the top level, containing at least one operator.

The following operators are supported:

•replace

This operator can alter the dialed destination number. If the regular expression given in the `key` field matches the destination number, each occurence of the matched regular expression will be replaced by the `value` field. Both `key` and `value` fields must be JSON strings.

`Break` or `pass` fields may not be used in this operator.

As an example, this operator may be used to enforce country specific international numbers format (e.g. replace a leading „0" by „+49" for calls in Germany.

•match

This operator matches the regular expression given in the `key` field against the destination number. If the expression matches, it can either allow or deny the call depending on the contents of the pass field. Alternatively, the evaluation of the current operator array can be stopped by setting the `break` field to `true`. Exactly one of the `break` or `pass` fields must be present.

If a additional `reverse` field is contained with a value of `true`, it will negate the check.

As an example, this operator can be used to allow (or deny) certain number areas like cell phone numbers or to implement outgoing call black lists.

•service

This operator matches the type of outgoing call against the JSON string given in the `key` field. If the call type matches, it can either allow or deny the call depending on the contents of the pass field. Alternatively, the evaluation of the current operator array can be stopped by setting the `break` field to `true`. Exactly one of the `break` or `pass` fields must be present.

If a additional `reverse` field is contained with a value of `true`, it will negate the check.

Valid service types are `fax`, `sip`, `dialplan`, `call_through`, `callback`, `ivr`, `announcement_call` and `call_api`.

The call filter given in the example will work like this:

- The destination number will be altered to match the German international phone number format, excluding the special number prefix 11 (used for emergency calls) .
- If the service type is not of the type `sip`, the inner JSON array will be canceled. Otherwise, the destination number will be checked. `+492216698711` and `+492216698499` are explicitly denied while German fixed network numbers, service numbers that are free of charge and emergency calls are allowed.
- If the call is not allowed or denied so far, because no operator containing a pass has matched, the filter continues with the second inner JSON array.
- If the service type is not of the type `fax`, this JSON array will be canceled. Otherwise the destination number will be checked again. German fixed network numbers and service numbers that are free of charge are allowed.
- At the end of the filter, all calls are denied.

`GET /api/partner/callfilter/(user-uuid)`
Get a user's call filter, if any.

Response body example

```
{
    "callfilter": {
        "operator": "match",
        "key": ".+",
        "pass": false
    }
}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner

- **empty-filter**: The given user does not have a call filter

This API call requires a session that is authenticated with a partner.

DELETE /api/partner/callfilter/(user-uuid)
Delete a user's call filter.

Response body example

```
{}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner

This API call requires a session that is authenticated with a partner.

9.1.4 /api/partner/cdr

GET /api/partner/cdr/ (*phone*)
Get the list of call detail records (CDR) related to this phone number.

Query Parameters

- **from** – optional: start time, in seconds-since-epoch (must have `from` <= `to`)
- **to** – optional: end time, in seconds-since-epoch (must have `from` <= `to`)
- **start** – optional: the id of the first CDR to return, exclusive, i.e. the result with an exact match id is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.
- **order** – optional: If set to `asc` (default), objects are returned in time ascending order (i.e. starting with `from` or `start` and ending with `stop`).

  If set to `desc`, objects are returned in time descending order (i.e. starting with `to` or `start` and ending with `from`).
- **count** – optional, default `10`: limits the number of returned objects (valid range: `1...5000`)

Response body example

```
{
    "cdr": [
        {
            "id": "2b5e1560-124f-11e4-b449-f8a96351ccd6",
            "date": 1406109196726
            "service": "call",
            "b_number": "+492216698711",
            "t_pre": 4,
            "t_post: 107,
        },
        {
            "id": "b31e29b0-3813-11e4-bae1-f8a96351ccd6",
            "date": 1406109203444
            "service": "call",
```

```
            "a_number": "+492216698499",
            "parent": "2b5e1560-124f-11e4-b449-f8a96351ccd6",
            "root": "2b5e1560-124f-11e4-b449-f8a96351ccd6",
            "t_pre": 7,
            "t_post: 33,
        },
        {
            "id": "981c8030-41cb-11e6-bc0b-f8a96351cc93",
            "b_number": "+492216698711",
            "root": "97ef2ea0-41cb-11e6-bc0b-f8a96351cc93",
            "parent": "97ef2ea0-41cb-11e6-bc0b-f8a96351cc93",
            "date": 1467625336745,
            "service": "call",
            "t_post": 7,
            "t_pre": 3
        },
        {
            "id": "97ef2ea0-41cb-11e6-bc0b-f8a96351cc93",
            "a_number": "csc_00",
            "date": 1467625336365,
            "phone": "+4922166966914",
            "t_post": 7,
            "t_pre": 3
        },
    ]
}
```


Status

- **wrong-phone**: The given phone number was not assigned by the partner or does not exist

This API call requires a session that is authenticated with a partner.

The partner can fetch CDRs according to the given `phone`. All CDRs have the following fields:

- `date` (the current time in ms since epoch when the call was initiated)
- `service` (one of `call`, `conference`, `fax`, `blf_pickup`, `call_through`, `call_reverse` and `call_back`)
- `fax_id` (Optional uuid of fax)
- `recording_id` (Optional uuid of recording)
- `conference_id` (Optional uuid of conference)
- `a_number` (for incoming calls only) Contains the phone number or sip account.
- `b_number` (for outgoing calls only) Contains the phone number or sip account.
- `parent` (Optional uuid of the parent cdr)
- `root` (Optional uuid of the first cdr in this group)
- `t_pre` (pre_answer duration)
- `t_post` (post_answer duration)

Certain service types have additional fields:

Conference service calls have a `conference_id` field, containing the conference id.

Fax service calls have a `fax_id`, containing the fax id. The numbers of pages sent / received will be in the `page_count` field.

Mailbox service calls have a `recording_id` field, containing the recording id.

GET /api/partner/cdr/event
Subscribe the current session to the event flow of CDRs associated to the partner.

Query Parameters

- **timeout** – optional: expiration timeout of the event flow in seconds. Default: 1 hour (3600 seconds), valid range 1..36000 seconds (10 hours)

Response body example

```
{}
```

This API call will enable the flow of CDR events to the current session. The flow of events will automatically stop after the timeout period. If this behavior is not desired, this API call should be called again before expiration to re-new the timeout period. If there are more than one sessions of the partner subscribing, then the stream will be split between the subscribing sessions and if one of the sessions die, then the corresponding stream is just diverted to other sessions.

DELETE /api/partner/cdr/event
Cancel the subscription of the current session to the data events of CDRs

Response body example

```
{}
```

This API call will stop the flow of CDR events to the current session.

9.1.5 /api/partner/emergency

GET /api/partner/emergency/(user-uuid)
Get emergency-call related information.

Response body example

```
{
    "uui_header": "001D73306138F193FFFF4A6F6C696F742D43757269652D5374722E",
    "emergency_police": "221CC00",
    "emergency_rescue": "221CC02"
}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner

This API call requires a session that is authenticated with a partner.

POST /api/partner/emergency/(user-uuid)
Update emergency-call related information.

Request body example

```
{
    "uui_header": "001D73306138F193FFFF4A6F6C696F742D43757269652D5374722E",
    "emergency_police": "221CC00",
    "emergency_rescue": "221CC02"
}
```

JSON Parameters

- **uui_header** – ^.{0,200}$

  Optional; the user-to-user-information for this user.

- **emergency_police** – ^.{0,10}$

  Optional; the emergency number of the responsible police control center in german emergency encoding.

- **emergency_rescue** – ^.{0,10}$

  Optional; the emergency number of the responsible rescue control center in german emergency encoding.

Response body example

```
{}
```

Status

- **wrong-user**: The given `user-uuid` does not exist or the user does not belong to the current partner

This API call requires a session that is authenticated with a partner.

If a user makes an emergency call (110 or 112) and has an `emergency_police` or `emergency_rescue` entry, the call will be routed to the responsible control center. An empty value for any of the parameters will delete them.

9.1.6 /api/partner/usergroup

POST /api/partner/usergroup

Creates an usergroup.

Request body example

```
{
    "description" : "Controlling",
    "prefix" : 7
}
```

JSON Parameters

- **description** – ^.{0,200}$

  optional: the description for this usergroup

- **prefix** – 0..9

  optional: the prefix for this usergroup

Response body example

```
{
    "group_uuid": "dfc3e9c4-4e81-11e1-9fe3-0024e8f90cc0"
}
```

This API call requires a session that is authenticated with a partner.

A new usergroup will be created for the partner. The call will return the UUID of the group. In order to active the usergroups prefix, a partner must enable the `group_prefix_enable` flag for the surrounding enterprise.

POST /api/partner/usergroup/ (*uuid*)
Updates an usergroup.

Request body example

```
{
    "description" : "Accounting",
    "prefix" : 7
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: the description for this usergroup

- **prefix** – `^0..9$`

  optional: the prefix for this usergroup

Response body example

```
{}
```

This API call requires a session that is authenticated with a partner.

`0` is no valid `prefix`. The transmission of `0` will remove the current `prefix`.

DELETE /api/partner/usergroup/ (*uuid*)
Deletes an usergroup

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner
- **not-empty**: The given usergroup `uuid` is not empty.

This API call requires a session that is authenticated with a partner.

An usergroup can only be deleted if there are no associated users.

POST /api/partner/usergroup/ (*uuid*) **/user**
Modifies the list of users in an usergroup.

Request body example

```
{
    "users": {
        "add": [
            "6f756b60-b672-11e3-9948-0026b9c1d1a6",
            "3759ea04-4005-11e4-ba4e-0314e1b2a9ec",
            "37e25268-4005-11e4-8832-9781ed589c3f"
        ],
        "delete": [
            "383a87d0-4005-11e4-a8b2-1f1752f935a9"
        ]
    }
}
```

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner
- **wrong-user**: At least one of the users listed does not exist or does not belong to the partner
- **already-grouped**: At least one of the users listed in the add section is already listed in another usergroup

This API call requires a session that is authenticated with a partner.

With this API call, users can be grouped into usergroups. Note that each user may only join one usergroup.

GET /api/partner/usergroup
Lists all usergroups that belong to the current partner.

Response body example

```
{
    "usergroups": {
        "d322dfe0-9730-11e4-8ff7-f8a96351ccd6": {
            "description": "Accounting",
            "prefix" : 7
        },
        "e8ee4030-9730-11e4-8ff7-f8a96351ccd6": {
            "description": "Controlling"
        }
    }
}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

GET /api/partner/usergroup/ (*uuid*) **/user**
Get the list of users in an usergroup.

Response body example

```
{
    "usergroup" : {
        "users" : [
            "6f756b60-b672-11e3-9948-0026b9c1d1a6",
            "3759ea04-4005-11e4-ba4e-0314e1b2a9ec",
            "37e25268-4005-11e4-8832-9781ed589c3f"
        ]
    }
}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

GET /api/partner/usergroup/ (*uuid*) **/extension**
Get information about the usergroup `uuid` internal extensions

Response body example

```
{
    "extensions" : {
        "10": "77a93f80-7f96-11e4-bfab-f8a96351ccd6",
        "20": "c759a6b0-7f9a-11e4-bfab-f8a96351ccd6"
    }
}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

POST /api/partner/usergroup/ (*uuid*) **/extension**
Updates the usergroup `uuid` internal extensions

Request body example

```
{
    "add": {
        "100" : "5704a3d8-7b9e-11e4-af85-1725ff033d08",
        "200" : "57054036-7b9e-11e4-af28-17b08d2664a8",
        "300" : "5705df5a-7b9e-11e4-be13-8f63fafaa45c",
        "400" : "5707524a-7b9e-11e4-a24c-8bf86e2591ea",
        "500" : "57092ab6-7b9e-11e4-9633-9ba60e6dbcd6"
    },
    "delete" : [
        "10",
        "20",
        "30",
        "40",
        "50"
    ]
}
```

JSON Parameters

- **extension** – ^[0-9]{1,5}$

  the users internal extension

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner
- **wrong-user**: At least one user does not exist, does not belong to the current partner or is not a member of the given `uuid`

This API call requires a session that is authenticated with a partner.

A partner can set internal extensions to an usergroup. Every user that is grouped into that usergroup and makes an outgoing call can dial a valid extension and will be connected to the given user's dialplan.

GET /api/partner/usergroup/(*uuid*)**/limit**
Gets the concurrent call limit for an usergroup.

Response body example

```
{
    "limit": 2
}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner
- **no-limit**: The given usergroup does not have a concurrent call limit

This API call requires a session that is authenticated with a partner.

POST /api/partner/usergroup/(*uuid*)**/limit**
Sets a concurrent call limit for an usergroup.

Request body example

```
{
    "limit": 2
}
```

JSON Parameters

- **limit** – ^[0-9]{1,3}$

  The new concurrent call limit for this usergroup

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

The concurrent call limit restricts the number of concurrent phone calls for all users in an usergroup. A limit of `0` effectively prevents the users in the group from establishing any calls.

DELETE /api/partner/usergroup/ (*uuid*) **/limit**
Deletes the concurrent call limit for an usergroup.

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

POST /api/partner/usergroup/ (*uuid*) **/credit**
Modifies the credit count of an usergroup.

Request body example

```
{
    "add": 10000
}
```

JSON Parameters

- **add** – `Integer`

  optional; the amount of credits to add

- **subtract** – `Integer`

  optional; the amount of credits to subtract

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

At least one of the `add` or `subtract` fields must be present in the request.

GET /api/partner/usergroup/ (*uuid*) **/credit**
Gets the credits left for an usergroup.

Response body example

```
{
    "credit": 4820
}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner
- **no-credit**: The given usergroup `uuid` does not have credits configured

This API call requires a session that is authenticated with a partner.

GET /api/partner/usergroup/credit
Lists all credits for the selected usergroups.

Query Parameters

- **min** – optional, integer: only list usergroups with at least `min` credits (must be >= 0; if both present, `min` must be less than `max`)
- **max** – optional, integer: only list usergroups with a maximum of `max` credits (must be >= 0; if both present, `max` must be greather than `min`)

Response body example

```
{
    "credits": {
        "47795c20-d3b3-11e4-b265-f8a96351ccd6": 10000,
        "d322dfe0-9730-11e4-8ff7-f8a96351ccd6": 12000,
        "e8ee4030-9730-11e4-8ff7-f8a96351ccd6": 14000
    }
}
```

This API call requires a session that is authenticated with a partner.

If neither `min` nor `max` is present, all usergroups with credits will be returned.

DELETE /api/partner/usergroup/(*uuid*)**/credit**
Delete credit limit on an usergroup.

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

POST /api/partner/usergroup/(*uuid*)**/zonefilter**
Modifies the zone filter of an usergroup.

Request body example

```
{
    "zonefilter": [
        {
```

```
            "description": "national",
            "prefix": ["+49"],
            "cost": 139
        },
        {
            "description": "euro1",
            "prefix": ["+32", "+39", "+44"],
            "cost": 189
        },
        {
            "description": "rest",
            "prefix": [""],
            "cost": 599
        }
    ]
}
```

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner
- **too-large**: The given zone filter exceeds the limit of 250 entries

This API call requires a session that is authenticated with a partner.

A partner can define a zone filter for an usergroup. Every time a user associated with that usergroup initiates an outgoing call, this zone filter is checked and the corresponding cost is applied and subtracted from the usergroup's credits.

A zone filter consists of a JSON object as shown in the example. It is an array of up to 250 rules, each of which have a description, a number of prefixes that make the rule match if the one of the prefix matches the outgoing phone number, and a cost (in credits per minute).

If a prefix matches, the corresponding cost will be removed from the usergroup's credits as soon as the call is answered, and will from then on be removed every 60 seconds until the call ends. If the usergroup does not have enough credits left, the call is terminated.

The cost does not have a specific currency associated with it. It might be defined in cents, 1/100 cents or any other metric, as long as the definitions within the usergroup's credit and the zone filter cost match.

Usually, a partner would periodically add to the credits in a usergroup, to ensure that the users in that group can continue to make outgoing calls.

If a destination number does not match any of the entires, a cost value of 0 is assumed and therefore no credits are used for the particular call.

GET /api/partner/usergroup/ (*uuid*) **/zonefilter**
Get the zone filter of an usergroup.

Response body example

```
{
    "zonefilter": [
        {
            "description": "national",
```

```
            "prefix": ["+49"],
            "cost": 139
        },
        {
            "description": "euro1",
            "prefix": ["+32", "+39", "+44"],
            "cost": 189
        },
        {
            "description": "rest",
            "prefix": [""],
            "cost": 599
        }
    ]
}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner
- **empty-filter**: The given usergroup does not have a zone filter

This API call requires a session that is authenticated with a partner.

DELETE /api/partner/usergroup/(*uuid*)**/zonefilter**

Delete the zone filter of an usergroup.

Response body example

```
{}
```

Status

- **wrong-usergroup**: The given usergroup `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

9.1.7 /api/partner/enterprise

POST /api/partner/enterprise

Creates an enterprise.

Request body example

```
{
    "description" : "Company"
    "group_prefix_enable" : false
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: the description for this enterprise

- **group_prefix_enable** – `true|false`

  optional: enable the usergroups prefix for internal extensions. (default: false)

Response body example

```
{
    "enterprise_uuid": "9c362cb0-d3cd-11e4-82ed-f8a96351ccd6"
}
```

This API call requires a session that is authenticated with a partner.

A new enterprise will be created for the partner. The call will return the UUID of the enterprise. If the `group_prefix_enable` flag is set to `true`, users have to dial the usergroup's prefix (*POST /api/partner/usergroup/(uuid)*) in order to reach users in other usergroups via internal extensions.

GET /api/partner/enterprise

Lists all enterprises that belong to the current partner.

Response body example

```
{
    "enterprises": {
        "93f59fc0-fd4b-11e4-8571-f8a96351ccd6": {
            "description": "",
            "group_prefix_enable": false
        },
        "9c362cb0-d3cd-11e4-82ed-f8a96351ccd6": {
            "description": "Company",
            "group_prefix_enable": false
        }
    }
}
```

This API call requires a session that is authenticated with a partner.

POST /api/partner/enterprise/ (*uuid*)

Updates an enterprise.

Request body example

```
{
    "description" : "Accounting",
    "group_prefix_enable" : true
}
```

JSON Parameters

- **description** – `^.{0,200}$`

  optional: the description for this enterprise

- **group_prefix_enable** – `true|false`

  optional: enable the usergroups prefix for internal extensions.

Response body example

```
{}
```

Status

- **wrong-enterprise**: The given enterprise `uuid` does not exist or does not belong to the partner

This API call requires a session that is authenticated with a partner.

DELETE /api/partner/enterprise/ (*uuid*)

Deletes an enterprise

Response body example

```
{}
```

Status

- **wrong-enterprise**: The given enterprise `uuid` does not exist or does not belong to the partner
- **not-empty**: The given enterprise `uuid` is not empty.

This API call requires a session that is authenticated with a partner.

An enterprise can only be deleted if there are no associated usergroups.

POST /api/partner/enterprise/ (*uuid*) **/usergroup**

Modifies the list of usergroups in an enterprise.

Request body example

```
{
    "usergroups": {
        "add": [
            "d322dfe0-9730-11e4-8ff7-f8a96351ccd6",
            "e8ee4030-9730-11e4-8ff7-f8a96351ccd6"
        ],
        "delete": [
            "95336280-ab9f-11e4-bdfb-f8a96351ccd6"
        ]
    }
}
```

Response body example

```
{}
```

Status

- **wrong-enterprise**: The given enterprise `uuid` does not exist or does not belong to the partner
- **wrong-usergroup**: At least one of the usergroups listed does not exist or does not belong to the partner
- **already-grouped**: At least one of the usergroups listed in the add section is already listed in another enterprise

This API call requires a session that is authenticated with a partner.

With this API call, usergroups can be grouped into enterprises. Note that each usergroup may only join one enterprise.

9.1.8 /api/partner/sensor

GET /api/partner/sensor/event
Subscribe the current session to the event flow of sensors associated to the partner.

Query Parameters

- **timeout** – optional: expiration timeout of the event flow in seconds. Default: 1 hour (3600 seconds), valid range 1..36000 seconds (10 hours)

Response body example

```
{}
```

This API call will enable the flow of `partner_sensor_data` and `partner_sensor_action` events to the current session for the specified sensor object. The flow of events will automatically stop after the timeout period. If this behavior is not desired, this API call should be called again before expiration to re-new the timeout period. If there are more than one sessions of the partner subscribing, then the stream will be split between the subscribing sessions and if one of the sessions die, then the corresponding stream is just diverted to other sessions.

DELETE /api/partner/sensor/event
Cancel the subscription of the current session to the data events of sensors

Response body example

```
{}
```

This API call will stop the flow of sensor data events to the current session.

POST /api/partner/sensor/(*sdevice*)**/action**
Triggers an action on the sensor `sdevice`.

The `sdevice` field carries the hardware address of the sensor device associated with the sensor as a hex encoded string. The `model` field contains the model identification string of the associated sensor device. If the sensor device belongs to a standardized profile class, this is shown in the `profile` field.

Request body example

```
{
    "action": [
        0,
        "Hello, World!"
    ]
}
```

- or -

```
{
    "action": [
        1,
        null
    ]
}
```

Response body example

```
{
    "action-uuid": "97a1f558-9167-11e2-892a-0024e8f90cc0"
}
```

Status

- **wrong-sensor**: The given sdevice does not match a sensor accessible to the current partner
- **sensor-offline**: The sensor is currently not visible through any devices, so the action cannot be delivered

The entries in the JSON array `action` must exactly match the `action` capabilities of the sensor (see *GET /api/partner/sensor/(sdevice)*). The type of values in the array depend on the specific action capabilities. In the first example, the sensor has an `onoff` capability at index `0`, which is set to the "off" state (value `0`) with this call, and a `text` capability at index `1` which is instructed to display the string "Hello, World!". In the second example, the onoff-switch is set to "on", while the text display remains unchanged.

Note that even though the number of elements in the action array must exactly match the number of action capabilities of the given sensor, a value of null might be given for a specific element to signal that no change is desired for this actor.

If the action was successfully queued to the sensor, the call will return an action UUID. A `sensor_action` event will be delivered to the current partner session after an acknowledgement message has been received by the sensor to confirm that the action was successfully carried out.

GET /api/partner/sensor/ (*sdevice*)
Get information about sensor `sdevice`

The `sdevice` field carries the hardware address of the sensor device associated with the sensor as a hex encoded string. The `model` field contains the model identification string of the associated sensor device. If the sensor device belongs to a standardized profile class, this is shown in the `profile` field.

Response body example


```
{
    "sensor": {
        "model": "cospace-sensor",
        "profile": "room-sensor-thm",
        "recv_interval": 60,
        "recv_after_send": true,
        "recv_time": 1363623307,
        "battery_status": 86,
        "mains_power": true,
        "tamper_detect": 1363347807,
        "fault_detect": 1363347820,
        "capabilities": {
            "data": [
                "temperature",
                "motion"
            ],
            "action": [
                "onoff",
                "text"
            ]
        },
        "state": {
            "data": [
                20.5,
                [1, 10, 50]
            ],
            "action": [
                1,
                "Hello, World!"
            ]
```

```
        },
        "box": {
            "efd04b02-8fd8-11e2-bf7e-0024e8f90cc0": {
                "time": 1320403260,
                "rssi": -80,
                "lqi": 45
            },
            "f10cb83e-8fd8-11e2-9f7c-0024e8f90cc0": {
                "time": 1320403155,
                "rssi": -38,
                "lqi": 70
            }
        },
    }
}
```

Status

- **wrong-sensor**: The given sdevice does not match a sensor accessible to the current partner

The `recv_interval` field is only present if the sensor is supposed to send data in regular time intervals. It specifies the maximum time interval between sensor transmissions in seconds.

If the `recv_after_send` field is present, the sensor will receive data only after having transmitted a packet itself. This is typically used in battery powered applications.

The `recv_time` field contains the timestamp of the last received data from the sensor. It is only present if the sensor has delivered data at least once in its lifetime.

The `battery_ok` field (type `boolean`) will be present if the sensor is battery-powered and is capable of signaling a simple "battery ok" (`true`) or "battery low" (`false`) condition. If the sensor is capable of reporting detailed battery statistics, the `battery_status` field will reflect the battery level in percent. The `mains_power` field might be present if a battery-powered device is currently running on mains power. If the device has the ability to detect a tampering attempt, the `tamper_detect` field will be present and will show the timestamp of the first detection. Likewise, the presence of the `fault_detect` field signals a sensor fault condition timestamp.

The box section will list the UUIDs of the box objects that have had contact with the sensor in the last time. Each box will have some meta-information about the connection to the sensor. Possible fields in this section include `time` (time of the last sensor contact in seconds-since-epoch), `rssi` (received signal strength indication in dBm) and `lqi` (link quality indicator, higher value means better signal).

The `capabilities` section lists the feature attributes of the sensor device.

In the capabilities `data` subsection, sensor capabilities are displayed as an ordered array of features (thus with an implicit index). The sensor device in the above example has two sensor features, a temperature sensor at index 0 which will yield a temperature in degrees Celsius), and a motion sensor at index 1 which will yield a motion detection duration time in seconds.

In the capabilities `action` subsection, actor capabilities are displayed as an ordered array of features (thus with an implicit index). The sensor device in the above example has two actor features, a onoff actor at index 0 that can be switched on (1) or off (0) and a text display at index 1.

The following capabilities are defined with their corresponding data format (see *GET /api/sensor/(uuid)/data*):

- `temperature`: A temperature sensor that has a single JSON number value as data, measured in degree Celsius (°C).

- `light`: An illuminance sensor (typically an ambient light sensor) that has a single JSON number value as data, measured in lux (lx; 1 lx = 1 lm/m^2 = 1 $cd\ sr/m^2$).
- `humidity`: A sensor for relative humidity that has a single JSON number value as data, measured in percent.
- `open`: A sensor that monitors the state of an opening such as a door, window, vent or a similar object. The value is a single JSON number that reads 1 if the state is "open", or 0 if the state is "closed".
- `open_percent`: A sensor that monitors the state of an opening such as a door, window, vent or a similar object. The value is a percentage, expressed as a single JSON number. The reading is 100 for "open" and 0 if the state is "closed". Numbers in between signal a half-closed half-open condition.
- `motion`: A motion detection sensor which has a JSON array containing exactly three JSON numbers as data. The first number represents the initial state of the motion sensor (0 for no motion detect, 1 for motion detect), the second number represents a time interval in seconds for this state. The third number represents a time interval for the opposite state. As an example, the data [1, 3, 5] would mean that the initial state of the sensor was "motion detect", this state lasted for 3 seconds, and afterwards the sensor remained 5 seconds in the state "no motion detect". The data [0, 100, 1] means that the sensor did not detect motion for 100 seconds, but then a motion detect happened for 1 second.
- `energy`: An energy meter that has a single JSON number value as data, measured in Watt-hours (Wh; 1 Wh = 0.001 kWh = 3600 J [Joule]).
- `voltage`: An electrical voltage sensor that has a single JSON number value as data, measured in Volts (V).
- `current`: An electrical current sensor that has a single JSON number value as data, measured in Amperes (A).
- `power`: An electrical power sensor that has a single JSON number value as data, measured in Watts (W).
- `power_factor`: An electrical power factor. Data is a single JSON number in the range -1..1.
- `frequency`: A frequency counter. The data is represented as a JSON number, measured in Hz (1 Hertz = 1 cycle per second).
- `onoff`: A switch-type sensor that has a single JSON number as a representation. Valid values are 1 (representing the "on" state) and 0 ("off" state).
- `text`: A device that has some sort of text display. The representation is a JSON string which holds the text that is to be displayed with a maximum length of 255 characters.
- `button`: A button-type sensor that has a single JSON number as a representation. There is only one valid value, 1, which represents that the button was pressed.
- `color_rgb`: A color sensor. The data is represented as a JSON array with exactly three JSON numbers, one for each color component: red, green and blue. Values for the individual components range from 0 to 255. As an example, the data [255, 255, 0] represents a bright yellow.
- `interval`: A time interval expressed as a single JSON number, representing the time period in seconds (s).
- `datetime`: An absolute point in time, expressed as a single JSON number that holds the seconds elapsed since the epoch of Jan 1st, 1970.
- `dimmer`: A dimmer switch in percent, represented by a single JSON number with values between 0 (completely off) and 100 (completely on).
- `distance`: A physical distance (length), expressed by a single JSON number, representing the distance in meters (m).
- `mass`: A mass, expressed as a single JSON number, representing the mass in kilograms (kg).

- `mass_flow`: A flow rate of mass, expressed as a single JSON number, representing the flow in kilograms per second (kg/s).
- `volume`: A space volume, expressed as a single JSON number, representing the volume in cubic meters (m^3).
- `volume_flow`: A flow rate of volume, expressed as a single JSON number, representing the flow in cubic meters per second (m^3/s).
- `fuel_use`: A mileage (fuel usage), expressed as a single JSON number, representing the value in liters per 100 km (l/100km).
- `velocity`: A velocity, expressed as a single JSON number, representing the speed in meters per second (m/s).
- `acceleration`: An acceleration, expressed as a single JSON number, representing the speed gain in meters per square second (m/s^2).
- `resistance`: An electrical resistance, expressed as a single JSON number, representing the resistance in ohms (Ω).
- `pressure`: A pressure, expressed as a single JSON number, representing the pressure in Pascal (Pa; 1 Pa = 1 N/m^2).
- `force`: A force, expressed as a single JSON number, representing the force in Newton (N).
- `torque`: A circular force (torque), expressed as a single JSON number, representing the torque in Newton meters (Nm).
- `angle`: An angle, expressed as a single JSON number, representing the angle in degrees (°, full circle is 360°).
- `compass`: A compass reading, expressed as a single JSON number, in degrees °, clockwise from the north direction (0°) to east (90°), south (180°) and west (270°) back to north (360°).
- `location`: A geographical position. The data is represented as a JSON array with exactly two JSON numbers. The first number represents the longitude, the second number the latitude of the position. Both values are in degrees (°) ranging from -180° to 180°.
- `concentration`: A concentration (ratio of mixture between two components), expressed as a single JSON number, representing the concentration in parts-per-million (ppm).
- `ph`: A pH value, expressed as a single JSON number, representing the pH (no unit, typical values between 1 and 14).
- `radiation`: An ionizing radiation dose. The data is expressed as a single JSON number, representing the dose in Sievert (Sv).
- `sound_pressure`: A sound pressure, expressed as a single JSON number, representing acoustic pressure in dezibels (dB).
- `level`: An otherwise unspecified logarithmic level, expressed as a single JSON number, representing the level in dezibels (dB).
- `alarm`: An alarm sensor. The data is expressed as a single JSON number. Valid values are 0 (“no alarm” state) and 1 (“alarm” state).
- `gauge`: An otherwise unspecified absolute value, expressed as a single JSON number.
- `counter`: An otherwise unspecified counter value, expressed as a single JSON number.
- `load`: An load percentage value, expressed as a single JSON number, representing the load in percent (%).

- `cycles`: A rotary speed value, expressed as a single JSON number, representing the rotary speed in cycles per second (1/s).
- `binary_8bit`: An otherwise unspecified value, expressed as a single JSON integer number with a range between 0 and 255.
- `binary_16bit`: An otherwise unspecified value, expressed as a single JSON integer number with a range between 0 and 65535.
- `binary_32bit`: An otherwise unspecified value, expressed as a single JSON integer number with a range between 0 and 4294967295.
- `octets`: An otherwise unspecified array of octets (binary string), expressed as a single JSON string in Base64 encoding.

The state section contains the most recent up-to-date status of the sensor and action elements. Its structure matches the capabilities section. While the *GET /api/sensor/(uuid)/data* and *GET /api/sensor/(uuid)/action* calls can be used to get the precise time series of sensor data and actions, the state section represents the current state of each of the data and action items. For the data items, this is the last sensor data update which was not null, and the the action items, this is that last action command which was not null.

9.2 Partner events

9.2.1 Event cdr

When a user from a partner generates a cdr event, this event is delivered to the elected partner sessions that requested the delivery of events with *GET /api/partner/cdr/event*.

```
{
    "cdr": {
        "phone": "+492216698712",
        "id": "2b5e1560-124f-11e4-b449-f8a96351ccd6",
        "date": 1406109196726
        "service": "call",
        "b_number": "+492216698711",
        "t_pre": 4,
        "t_post: 107
    }
}
```

9.2.2 Event partner_sensor_data

When a sensor from a partner delivers data to the cloud, this event is delivered to the elected partner sessions that requested the delivery of sensor data events with *GET /api/partner/sensor/event*.

```
{
    "sensor": "ce19884c",
    "data": {
        "1363623247000": [
            20.5,
            [
                0,
                255
            ]
```

```
        ]
    }
}
```

The `sensor` field holds the `sdevice` of the sensor object.

The `data` section lists the new sensor data, typically a single entry.

9.2.3 Event partner_sensor_action

When an action was carried out on a sensor, this event is delivered to the partner session that invoked *GET /api/partner/sensor/event*.

```
{
    "status": "ok",
    "sensor": "ce19884c",
    "action": {
        "1363623253000": [
            1,
            null
        ]
    },
    "action-uuid": "97a1f558-9167-11e2-892a-0024e8f90cc0"
}
```

The `status` field contains `ok` if the action was carried out. It might contain `replaced` if the action was superseded by another action while waiting for the sensor to become available for radio transmittion.

The `action` section lists the action that was carried out, typically a single entry.

The `action-uuid` field holds the action UUID that corresponds to the `action-uuid` field in the response of the *POST /api/partner/sensor/(sdevice)/action* call.

CHAPTER

TEN

CALLING NAME DELIVERY

Calling Name Delivery (CNAM) is a service that converts E.164 telephone numbers into names that can be shown on compatible SIP phones. The following API can be used to view and modify the database of CNAM entries for users, usergroups and enterprises.

10.1 /api/user

GET /api/user/cnam
Read CNAM entries of the current user.

Query Parameters

- **next** – optional: phone number of the first CNAM entry to return
- **count** – optional: max. number of CNAM entries to return (range: 1...500, default: 50)

Response body example

```
{
    "cnam": [{
        "phone": "+492216698712",
        "name": "John Doe"
    }, {
        "phone": "+492216698713",
        "name": "Jane Doe"
    }],

    "next": "+492216698714"
}
```

JSON Parameters

- **cname** – array of CNAM entries
- **next** – phone number of the next CNAM entry or `null` if there are no more CNAM entries after the current page

POST /api/user/cnam/ (*phone*)
Create or modify a CNAM entry of the current user.

Parameters

- **phone** – phone number in E.164 format

Request body example

```
{
    "name": "John Doe"
}
```

JSON Parameters

- **name** – Resolved name for the given `phone` number. Valid characters are letters from A-Z and a-z, digits, `&'()*+,-./:;@[]_` and the space character.

DELETE /api/user/cnam/ (*phone*)
Delete an existing CNAM entry of the current user.

Parameters

- **phone** – phone number in E.164 format

Response body example

```
{}
```

Status

- **wrong-cnam**: The given CNAM entry does not exist

10.2 /api/partner/usergroup

GET /api/partner/usergroup/ (*usergroup*) **/cnam**
Read CNAM entries of the selected usergroup.

Query Parameters

- **next** – optional: phone number of the first CNAM entry to return
- **count** – optional: max. number of CNAM entries to return (range: 1...500, default: 50)

Response body example

```
{
    "cnam": [{
        "phone": "+492216698712",
        "name": "John Doe"
    }, {
        "phone": "+492216698713",
        "name": "Jane Doe"
    }],

    "next": "+492216698714"
}
```

JSON Parameters

- **cname** – array of CNAM entries
- **next** – phone number of the next CNAM entry or `null` if there are no more CNAM entries after the current page

POST /api/partner/usergroup/ (*usergroup*) **/cnam/**
phone Create or modify a CNAM entry of the selected usergroup.

Parameters

- **phone** – phone number in E.164 format

Request body example

```
{
    "name": "John Doe"
}
```

JSON Parameters

- **name** – Resolved name for the given `phone` number. Valid characters are letters from A-Z and a-z, digits, `&'()*+,-./:;@[]_` and the space character.

DELETE /api/partner/usergroup/ (*usergroup*) **/cnam/**
phone Delete an existing CNAM entry of the selected usergroup.

Parameters

- **phone** – phone number in E.164 format

Response body example

```
{}
```

Status

- **wrong-cnam**: The given CNAM entry does not exist

10.3 /api/partner/enterprise

GET /api/partner/enterprise/ (*enterprise*) **/cnam**
Read CNAM entries of the selected enterprise.

Query Parameters

- **next** – optional: phone number of the first CNAM entry to return
- **count** – optional: max. number of CNAM entries to return (range: 1...500, default: 50)

Response body example

```
{
    "cnam": [{
        "phone": "+492216698712",
        "name": "John Doe"
    }, {
        "phone": "+492216698713",
        "name": "Jane Doe"
    }],

    "next": "+492216698714"
}
```

JSON Parameters

- **cname** – array of CNAM entries
- **next** – phone number of the next CNAM entry or `null` if there are no more CNAM entries after the current page

POST /api/partner/enterprise/ (*enterprise*) **/cnam/**

phone Create or modify a CNAM entry of the selected enterprise.

Parameters

- **phone** – phone number in E.164 format

Request body example

```
{
    "name": "John Doe"
}
```

JSON Parameters

- **name** – Resolved name for the given `phone` number. Valid characters are letters from A-Z and a-z, digits, `&'()*+,-./:;@[]_` and the space character.

DELETE /api/partner/enterprise/ (*enterprise*) **/cnam/**

phone Delete an existing CNAM entry of the selected enterprise.

Parameters

- **phone** – phone number in E.164 format

Response body example

```
{}
```

Status

- **wrong-cnam**: The given CNAM entry does not exist

CHAPTER

ELEVEN

RULES API

Warning: This API is still experimental and might be changed at any time!

The following describes the API calls required to create, modify, activate and deactivate rules. It can also be used to query the current state of an existing rule. As with other API calls, the rule API also requires a valid and an authenticated user session.

A rule is a collection of states and edges. An edge (transition) is defined as a step from a preceding state to the succeeding state when the given conditions are fulfilled and is characterized by the actions to be performed while undergoing the transition. A description of rules, its states, edges and transitions will be called as a `rule definition` throughout this chapter.

11.1 API Call Reference

11.1.1 /api/rule

`POST /api/rule`

Creates a new rule and returns the `rule-id` of the newly created rule.

Request body example

```
{
    "name": "Livingroom lights",
    "initial_state": "lights_off",
    "edges": [{
        "from": "lights_off",
        "to": "lights_on",
        "conditions": [{
            "type": "sensor_data",
            "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
            "value": [1]
        }],
        "actions": [{
            "type": "sensor_action",
            "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
            "value": [1]
        }]
    }, {
        "from": "lights_on",
        "to": "lights_off",
        "conditions": [{
```

```
            "type": "sensor_data",
            "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
            "value": [0]
        }],
        "actions": [{
            "type": "sensor_action",
            "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
            "value": [0]
        }]
    }],
    "active": true,
    "state": "lights_on",
    "metadata": {
          "place": "Living Room",
          "properties": {
                "color": "green"
          }
    }
}
```

JSON Parameters

- **name** – `^.{1,200}$`

  mandatory: A user-given name for the rule, does not need to be unique.

- **initial_state** – `^.{1,200}$`

  mandatory: The initial state from which the execution of rule begins, when no further state is specified.

- **edges** – mandatory: A collections of transitions. `edges` field has to be a json array and can't be empty.

- **from** – `^.{1,200}$`

  mandatory: The state from which the specific transition is defined.

- **to** – `^.{1,200}$`

  mandatory: The state to which the specific transition terminates after performing the action.

- **conditions** – mandatory: A set of conditions necessary to begin a transition, i.e., follow an edge. The `conditions` field contains an array of conditions. A condition can be described based on the mandatory `type` field. For example, a condition of type `sensor_data` is fulfilled when the measurement from the sensor is equal to the one specified in `value` field.

- **actions** – mandatory: A set of actions to be performed during a transition. The `actions` field contains an array of actions. An action can be described based on the mandatory `type` field. For example, a action of type `sensor_action` is performed by changing the sensor value to the one specified in the `value` field. In case, no actions are to be performed, `actions` can be an empty array.

- **active** – `true|false`

  optional: A JSON boolean value, that specifies if the rule will be activated after the creation or not. The default value is `true`.

- **state** – `^.{1,200}$`

optional: A JSON string value, that specifies the state, from which the rule will start its execution. The rule will start from the `initial_state` by default. Note that, on setting the `active` field to false the `starting_state` field will be ignored. Upon next activation, the rule will start from the initial state when no different state is specified.

- **`metadata`** – optional: A JSON object with the metadata to be stored with the rule. e.g. { "gui_profile": "client-id"}. The metadata object can't be larger than 5 KB. It is in the responsibility of the client to not corrupt any pre-existing metadata, i.e., an overwrite to the metadata must not break its interpretation by other clients or delete any entries added by other clients.

A more detailed description on the types of conditions and actions and how to define them can be found at *Conditions in a rule* and *Actions by a rule*.

Response body example

```
{
    "status": "ok",
    "uuid": "ce19884c-8fed-11e2-93fe-0024e8f90cc0"
}
```

JSON Parameters

- **`status`** –
  - **wrong-sensor**: The user does not have the necessary permissions for one or more of the sensors mentioned in the rule definition.
  - **wrong-state**: The rule definition has a field `state` that does not exist in the collection of `edges`.
  - **deactivated-rule**: The rule definition has a field *state* while *active* is false.
  - **too-large**: The `metadata` field is bigger than 5kB.
  - **access-denied**: The `edges` contain action(s), that the user is not authorized to execute.

`GET /api/rule`

Fetches rules in the system that belong to the user.

Parameters

- **`from`** – `timestamp`

  optional: `start time` in seconds-since-epoch (must have from <= to)

- **`to`** – `timestamp`

  optional: `end time` in seconds-since-epoch (must have from <= to)

- **`start`** – `uuid`

  optional: The UUID of the first rule to be returned. The rule with an exact match UUID is not returned. If present, it overrides the *from* parameter

- **`count`** – `1...500`

  optional: limits the number of returned rules. The default value is *20*.

Response body example

```
{
    "status": "ok",
    "rule": {
        "ce19884c-8fed-11e2-93fe-0024e8f90cc0" : {
            "name": "Livingroom lights",
            "active": true,
            "state": "lights_on",
            "metadata":  {}
        },
        "ce19884c-8fed-11e2-93fe-0024e8f90cc7" : {
            "name": "Bedroom lights",
            "active": false,
            "state": "lights_off",
            "metadata":  {}
        }
    }
}
```

JSON Parameters

- **name** – The name of the rule as specified in the rule definition.
- **active** – The execution state of the rule, if it is active or not.
- **state** – The current `state` at which the rule is waiting for the set of conditions to be satisfied. If the rule is inactive, the last saved state before the deactivation is returned.
- **metadata** – A JSON object containing the latest version of `metadata`.

DELETE /api/rule/(rule-id)

Deletes an existing rule definition with the given `rule-id`. The rule will be deactivated automatically before it is deleted.

Response body example

```
{
    "status": "ok"
}
```

JSON Parameters

- **status** –
  - **wrong-rule**: The rule with `rule-id` does not exist or does not belong to the user.

GET /api/rule/(rule-id)

Returns all information about the rule with the given `rule-id`.

Response body example

```
{
    "status":"ok",
    "rule":{
        "active":true,
        "state":"lights_on",
        "metadata": {},
        "name":"Livingroom lights",
        "initial_state":"lights_off",
        "edges": [{
```

```
            "from": "lights_off",
            "to": "lights_on",
            "conditions": [{
                "type": "sensor_data",
                "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
                "value": [1]
            }],
            "actions": [{
                "type": "sensor_action",
                "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
                "value": [1]
            }]
        }, {
            "from": "lights_on",
            "to": "lights_off",
            "conditions": [{
                "type": "sensor_data",
                "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
                "value": [0]
            }],
            "actions": [{
                "type": "sensor_action",
                "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
                "value": [0]
            }]
        }]
    }
}
```

JSON Parameters

- **status** –
  - **wrong-rule**: The rule with `rule-id` does not exist or does not belong to the user.

POST /api/rule/(rule-id)

Updates the rule with the given `rule-id`.

Request body example

```
{
    "name": "Livingroom lights",
    "metadata": {},
    "state": "lights_on",
    "active": true,
    "initial_state": "lights_off",
    "edges": [{
        "from": "lights_off",
        "to": "lights_on",
        "conditions": [{
            "type": "sensor_data",
            "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
            "value": [1]
        }],
        "actions": [{
            "type": "sensor_action",
            "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
            "value": [1]
```

```
        }]
    }, {
        "from": "lights_on",
        "to": "lights_off",
        "conditions": [{
            "type": "sensor_data",
            "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
            "value": [0]
        }],
        "actions": [{
            "type": "sensor_action",
            "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
            "value": [0]
        }]
    }]
}
```

JSON Parameters

- **name** – `^.{0,200}$`

  optional: New name of the rule.

- **metadata** – optional: A JSON object with the metadata to be stored with the rule. The metadata object can not be larger than 5 KB. For more information, check *POST /api/rule*

- **state** – `^.{0,200}$`

  optional: The state to which the rule will jump without following intermittent edges, before continuing the execution. In other words, no actions will be executed during this change since none of the edges were followed. More importantly, the state can only be changed, when the rule is active and the `state` exists. As a consequence, setting the `active` field to false and changing the `state` field has no effect.

- **active** – `true|false`

  optional: A boolean JSON value, that specifies if the rule shall be activated or deactivated. The rule will be activated if the `active` field is `true` and the rule is currently inactive.

- **initial_state** – `^.{1,200}$`

  optional: The initial state from which the execution of rule shall begin after the next start/restart. Note, that the state has to exist.

- **edges** – optional: A set of transitions. If the rule is active and the `active` field is not `false`, the rule will be restarted with the updated definition. However, if the rule is inactive and the `active` field is `true`, the rule will activated and the state will be set in accordance with `state` field.

At least, one of the fields must be present in the request body.

Response body example

```
{
    "status": "ok"
}
```

JSON Parameters

- **status** –
  - **wrong-rule**: The rule with `rule-id` does not exist or does not belong to the user.
  - **deactivated-rule**: The state could not be changed because the rule is inactive.
  - **wrong-state**: The rule definition has a field `state` that does not exist in the `edges`.
  - **too-large**: The `metadata` is larger than 5kB.
  - **access-denied**: The `edges` contain action(s), that the user is not authorized to execute.

If any one of the fields in the API request is invalid or causes an error, then none of the specified changes will be executed.

11.2 Conditions in a rule

Conditions are checks that are performed before stepping from one state to another via an *edge*. This section describes all currently supported types of conditions.

Conditions are usually transformed to *triggers*, and a state check and transition is only performed when triggered.

11.2.1 Sensor Data

This condition checks one or more measured data points of a sensor for equality and whether the value is inside a given range.

Equality is evaluated by checking for all the capabilities provided in the `value` property.

Ranges can be checked by a combination of `min` and `max`. Note that, all the entries in the `min` property require a corresponding entry in the `max` property, that is greater than the `min` property. In order to specify a condition outside of the range specified by `min` and `max`, use the boolean `not` operator as specified in *Boolean conditions*. Also, `min` and `max` mean *more or equal* and *less or equal* respectively. `Null` values will be skipped in all checks. This also applies to multi sensor values, which will be defined as value-arrays.

Examples

If capability with index `0` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc1` equals `1`:

```
{
    "type": "sensor_data",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
    "value": [1]
}
```

If capability with index `0` and `1` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc1` equals `1` and `0` respectively:

```
{
    "type": "sensor_data",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
    "value": [1, 0]
}
```

If capability with index `1` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc2` is at least `5.3`:

```
{
    "type": "sensor_data",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
    "min": [null, 5.3]
}
```

If capability with index `0` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc2` is less than or equal to `5.3`:

```
{
    "type": "sensor_data",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
    "max": [5.3]
}
```

If capability with index `0` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc3` is in the range of `10` and `234`, with `10` and `234` included:

```
{
    "type": "sensor_data",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc3",
    "min": [10],
    "max": [234]
}
```

Trigger

This condition creates a trigger that is fired every time a new sensor data is received. It does not necessarily mean that the data has changed. The trigger can also fire when the condition was already satisfied by the last seen value of the sensor, which will be checked when the condition is enabled. In case this behavior is not wanted, have a look at the `sensor_event` condition type.

11.2.2 Sensor Event

The `sensor_event` condition skips the initial check with the last value, otherwise works in the same way as `sensor_data` conditions. This condition supports all of the operators as supported by the `sensor_data` conditions. The creation of triggers is slightly different, as it will only react to sensor events that enter the cloud when the condition is enabled.

Examples

If capability with index `0` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc1` equals `1`:

```
{
    "type": "sensor_event",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
    "value": [1]
}
```

Trigger

This condition creates a trigger that is fired every time a new sensor data is entering the cloud. It does not necessarily mean that the data has changed.

11.2.3 Time

The time condition checks if it is a certain point in time or whether the time is in a certain range. Time must be represented in the format `hh:mm:ss` with `hh` ranging from 0 to 23, and `mm` and `ss` ranging from 0 to 60.

Examples

When it is 4:30 pm:

```
{
    "type": "time",
    "value": "16:30:00"
}
```

When the event occurs between 10 pm and 5 am:

```
{
    "type": "time",
    "min": "22:00:00",
    "max": "05:00:00"
}
```

Trigger

This condition will create triggers based on the type of the condition (value vs. range):

- A time-condition with value will be triggered only at the specified time.
- A time-range-condition with min and max will be triggered at the the specified time in the min property.

Time conditions can often be combined with additional conditions specifying repetitions and recurrences, as we specify below.

11.2.4 Weekday

The `weekday` condition checks if it is a certain day of the week. The value field is an array containing all the days, in abbreviated form, on which the condition will be true.

```
{
    "type": "weekday",
    "value": [ "MON", "TUE", "WED", "THU", "FRI", "SAT", "SUN" ]
}
```

Trigger

This condition will create triggers at the start (00:00:00) of the specified day(s):

11.2.5 Date

The user can also specify exact dates optionally. The `date` condition may contain the specific date or a range of two dates in "DD.MM" format.

When it is 22nd of March (00:00:00),

```
{
    "type": "date",
    "value": "22.03"
}
```

or a range from 22nd of March (00:00:00) to 27th of April (23:59:59)

```
{
    "type": "date",
    "min": "22.03",
    "max": "27.04"
}
```

Trigger

This condition will create triggers based on the type of the condition (value vs. range):

- A date-condition with value will be triggered only at the start (00:00:00) of the specified date.
- In case, one has specified the 29th of February as a date-condition, the condition will only be triggered in a leap year.
- A date-range-condition with min and max will be triggered at the beginning of the specified date in the min property.
- In case, one has specified the 29th of February as a bound of a date-range-condition, it will be interpreted as the 28th of February if the current year is not a leap year.

11.2.6 Timeout

The timeout condition checks how long is the rule waiting at the current state. The timeout is reset, if on following an edge the rule comes back to the same state from which the edge starts.

The duration of the timeout is determined by the `value` field which must be given in seconds.

Examples

After 5 minutes:

```
{
    "type": "timeout",
    "value": 300
}
```

Trigger

This condition will be triggered when the timeout expires.

11.2.7 Boolean conditions

The `and`, `or` and `not` conditions can be used to create boolean combinations of other conditions. A `not` operator on a timeout-condition and daytime-condition alone is invalid.

Examples

If both conditions are required to be satisfied before following an edge:

```
{
    "type": "and",
    "conditions": [{
        "type": "sensor_data",
        "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
        "value": [1, 0]
    }, {
        "type": "date",
        "value": "22.03"
    }]
}
```

If one of the conditions is to be satisfied:

```
{
    "type": "or",
    "conditions": [{
        "type": "sensor_data",
        "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
        "value": [1, 0]
    }, {
        "type": "sensor_data",
        "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
        "min": [null, 5.3]
    }]
}
```

If the negation of a condition is to be satisfied:

```
{
    "type": "not",
    "condition": {
        "type": "sensor_data",
        "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc3",
        "min": [10],
        "max": [234]
    }
}
```

Trigger

The triggers are created for the conditions connected by the boolean conjuncts.

11.3 Actions by a rule

Actions are performed when stepping from one state to another via an *edge*. This sections describes all currently supported types of actions.

11.3.1 Sensor Action

This action sets one or more values on a sensor device. Each sensor action holds information about the sensor identifier (uuid) and the values to be set. A particular value for a specified capability can also be an array (example below).

Examples

Set capability with index `1` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc1` to `1`:

```
{
    "type": "sensor_action",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
    "value": [null, 1]
}
```

Set capability with index `0` and `1` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc1` to `0` and `1` resp.:

```
{
    "type": "sensor_action",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc1",
    "value": [0, 1]
}
```

Set capability with index `0` of sensor `ce19884c-8fed-11e2-93fe-0024e8f90cc2` to green (0 red, 255 green, 0 blue):

```
{
    "type": "sensor_action",
    "sensor": "ce19884c-8fed-11e2-93fe-0024e8f90cc2",
    "value": [[0, 255, 0]]
}
```

11.3.2 Notification E-Mail

This action will send an email to one or more recipients.

Examples

Send an email to `user@domain.com` and `user_2@domain.com` with the specified subject and text:

```
{
    "type": "email",
    "recipients": ["user@domain.com", "user_2@domain.com"],
    "subject": "Movement at ${NAME} detected",
    "text": "You might want to check this ${METADATA.properties.color} ${METADATA.
↪place} ...!"
}
```

Both recipients will be added to the `to-field` in the email header. The `${NAME}` template will be replaced by the rule name. The `${METADATA.place}` template will be replace by the metadata field `place`. The `${METADATA.properties.color}` template will be replace by the metadata field `properties.color`.

Predefined templates:

- `${NAME}` rule name

- `${STATE}` rule state
- `${ISO_DATE}` trigger date in ISO format
- `${ISO_DATE_TIME}` trigger date & time in ISO format
- `${ISO_TIME}` trigger time in ISO format

11.3.3 Set credit counter of usergroup

This action will set the credit counter of the specified usergroup to the specified credit value. This action can only be used by authorized users!

Examples

Sets the credit of usergroup: cc17884c-8fed-11e2-93fe-0024e8f90cc3 to a value of 100:

```
{
    "type": "set_credit",
    "usergroup": "cc17884c-8fed-11e2-93fe-0024e8f90cc3",
    "credit": 100
}
```

The `credit` field has to be a positive integer. In case the user loses the authorization to execute the action during the runtime of the corresponding rule, it will be deactivated once the action will be executed for the next time, but it will nevertheless follow the edge to the followup state.

11.3.4 Push-Notification Android and iOS

This action will send push notification messages to all registered devices of the rule creator.

Examples

```
{
    "type": "push_notification",
    "content": "Movement at home detected"
}
```

The `content` field can be an arbitrary JSON value. The corresponding payload the receiving devices can expect looks like the following:

```
{
    "message": "rule_engine_notification",
    "rule": "af729cc0-3251-11e2-81c1-0800200c9a66",
    "content": "Movement at ${NAME} detected"
}
```

The `message` field identifies the received notification as payload produced by the rule engine. The `rule` field tells the receiving application, which rule has executed the `push_notification` action and the `content` field is carrying the specified content in the action definition. The overall size of resulting push notification messages is limited to 2048 Byte. As the `message` and `rule` fields are static, the `content` field has to be chosen properly to not exceed this limit. The `content` field can contain the same templates like the email notification.

11.3.5 Call-Notification

This action will call the one or more given numbers.

Examples

```
{
    "type": "call_notification",

    "from": "+492216698701",
    "from_name": "John Doe",
    "clip": "+492216698702",
    "recipients": ["+492216698706", "+492216698707"],
    "commands": [
        { "silence":500 },
        { "announcement": "550625e0-fb34-11e5-aabf-2fe9eca72bbf" }
    ]
}
```

JSON Parameters

- **from** – `^\+[0-9]{3,20}$`

  sets the user phone number.

- **from_name** – `^\+[0-9]{3,20}$`

  optional; sets the from name for this call setup.

- **clip** – `^\+[0-9]{3,20}$`

  optional; sets the clip number for this call setup.

- **recipients** – JSON array

  contains all targets(`^\+[0-9]{3,20}$`).

- **commands** – JSON array

  contains the commands for playback.

In the example above the object which contains silence will generate 500 ms of silence in the call. The announcement container holds the UUID of the object for playback.

11.4 Rule events

11.4.1 Event rule_new

This event is sent to all session of the current user when a new rule is created. It also contains the `metadata` corresponding to the rule if added by the rule creator.

```
{
    "rule" : "ce19884c-8fed-11e2-93fe-0024e8f90cc0",
    "name": "Livingroom lights",
    "initial_state": "lights_off",
    "active": false,
    "metadata" : {}
}
```

11.4.2 Event rule_delete

This event is sent to all session of the current user when a rule is deleted.

```
{
    "rule" : "ce19884c-8fed-11e2-93fe-0024e8f90cc0"
}
```

11.4.3 Event rule_update

This event is sent to all sessions of the current user when any property of the rule is updated or modified.

```
{
    "rule" : "ce19884c-8fed-11e2-93fe-0024e8f90cc0",
    "name": "Livingroom lights",
    "initial_state": "lights_off",
    "edges_updated": true,
    "active" : true,
    "state" : "init_state",
    "metadata": {}
}
```

The event will only carry the properties that have changed. In case, the edges were updated, the *edges_updated* field is set to *true*.

CHAPTER

TWELVE

DEVICE API

12.1 API Call Reference

POST /api/device/session

Create and authenticate a session object with the specified device.

Request body example

```
{
    "device": "dc7d3012-7736-11e1-8a6d-0024e8f90cc0",
    "key": "dijaDia8Aiofjb9aaobasdfDF",
    "hw_version": "cospace-box",
    "sw_version": "001",
    "ip_addr": "212.202.0.10",
    "mac_addr": "00:24:d6:83:c1:54",
    "timeout": 15
}
```

JSON Parameters

- **device** – `UUID`

  The UUID of the physical device

- **key** – `^.{5,50}$`

  The authentication key of the device (plain-text)

- **hw_version** – `^.{1,20}$`

  The hardware version / model of the device

- **sw_version** – `^.{1,20}$`

  The currently running software version on the device

- **ip_addr** – `IP address`

  The current IP address of the device

- **mac_addr** – `^[0-9a-f]{2}(:[0-9a-f]{2}){5}$`

  The current MAC address of the device

- **timeout** – 15-1000

  Optional: the timeout for the *GET /api/device/task* call

Response body example

```
{
    "sid": "boajw93bawjckbja2ZdbfdGa84Afib",
    "server": "https://api43.service.de:1234",
    "timeout": 15
}
```

Status

- **wrong-auth**: device or key are invalid.

This command creates a new device session. Device sessions are used to communicate between physical devices ("cospace box") and the cospace cloud. If a new device session is established, an existing session towards this device that might exist is closed.

Further API calls that reference this session can only be made towards the server specified in the `server` field, as the session state is only maintained on this server.

The `timeout` field informs the client about the response timeout for the *`GET /api/device/task`* call.

`DELETE /api/device/session`

Closes a device session and invalidates the corresponding session id.

Response body example

```
{}
```

After closing the session, the session id is deleted on the server and can not be used for any further requests.

`POST /api/device/csp`

Inform the system of a cospace sensor protocol (cSP) payload packet

Request body example

```
{
    "packet": "wqc4bHc3ZW5kOC85NGlha0Nvc0VxK2YjKjQ5MiYyIQo=",
    "time": 1320403260,
    "rssi": -80,
    "lqi": 45
}
```

JSON Parameters

- **`packet`** – `^.{10,500}$`

  The complete cSP packet (excluding preamble, sync word and CRC field), as a base64 encoded string

- **`time`** – The time of packet reception in seconds-since-epoch
- **`rssi`** – The received signal strength indication in dBm
- **`lqi`** – The link quality indication, higher value means better signal

Response body example

```
{}
```

If a devices receives a valid (i.e., the CRC is correct) cospace sensor protocol message, this API call is used to inform the system about this packet.

GET /api/device/task
Requests the list of open tasks for the device

Query Parameters

- **next** – optional: the id of the first task to be returned (default: 0)

Response body example

```
{
    "task": [
        {
            "id": 1,
            "task": "ipquality_source",
            "time": 2182782732,
            "remote": "212.202.3.45",
            "port": 4711,
            "duration": 10
        },
        {
            "id": 2,
            "task": "ipquality_destination",
            "time": 2182782733,
            "duration": 10
        }
    ]
}
```

This API call is the way tasks are delivered to devices. A device is expected to repeatedly call this function in order to receive all tasks scheduled for it. See *Generic Task JSON Considerations* for details about the individual tasks.

A call to this API may block for some time, if there are no tasks in the queue with id equal or greater than the given next parameter. The maximum wait time is announced to the client in the *POST /api/device/session* call.

The list of tasks may be empty if no tasks were scheduled and the (server-defined) blocking time is reached.

POST /api/device/task/(task-id)
Deliver a response to the requested task `task-id`

Query Parameters

- **intermediate** – optional: if present, signals an intermediate response. In this case, a final response (without the intermediate flag) will follow later.

Response body example

```
{}
```

Status

- **wrong-task**: The task with the specified `task-id` does not exist in the server's task list

The content of the request body is task-specific and is defined with the respective task in chapter Task definitions on page 252.

12.2 Generic Task JSON Considerations

Each task (as returned with the *GET /api/device/task* call) has a specific JSON representation that is defined in the following sections. Besides the specific elements, there is also a common structure, i.e. some fields that are present in all tasks. The generic JSON representation of any task („task skeleton") looks like this:

```
{
    "id": 42,
    "task": "ipquality_source",
    "time": 2182782732
}
```

The `id` field is a unique identifier of the task *in the context of the current device session*. The first task of a newly created session has id 0, and the id is increased by 1 with every new task. This enables a client to detect whether the stream of tasks in contiguous. The `time` field lists the time of event generation in seconds-since-epoch.

12.3 Task definitions

12.3.1 Task reload

Instructs the device to perform a reboot. If the `force` option exists and is set to `true`, the device should reboot immediately, without going through the various system shut-down routines and sending a response to this task (e.g. by directly calling the reboot(2) system call).

Task example (JSON)

```
{
    "force": false
}
```

Final response example (JSON)

```
{
    "status": "ok"
}
```

If a forced reboot is requested, the device will not send a response to this task. Otherwise, it will send the response before starting the reboot sequence.

If a response is sent and the `status` field does not contain `ok`, a `message` field should also be present describing the specific failure.

12.3.2 Task upgrade

Instructs the device to perform a software upgrade.

The device must downloaded all software components from the given volume. When a device gets the `upgrade` task, it is permitted to access the given volume read-only. The volume should contain a number of first-level sub-folders, called `batch01`, `batch02`, and so on. For each sub-folder, the device will

- create a local, temporary directory
- download all files from the given sub-folder to the temporary directory
- execute `/bin/sh ./run.sh` within the temporary directory

- remove the temporary directory and all files within it

This mechanism allows software upgrades to be performed in multiple batches that will be run in a defined order (e.g., to cope with memory restrictings on the device).

Task example (JSON)

```
{
    "volume": "7fc63784-b39e-11e1-80cc-0024e8f90cc0"
}
```

Intermediate response example (JSON)

```
{
    "status": "ok",
    "batch": 1
}
```

Final response example (JSON)

```
{
    "status": "ok"
}
```

For each batch that was executed, the device will send an intermediate response containing the batch number.

If a response or intermediate response is sent and the `status` field does not contain `ok`, a `message` field should also be present describing the specific failure.

However, it depends on the upgrade procedure whether the device will send intermediate or final responses at all. For example, if the upgrade process reboots the device, responses might not be reliable.

12.3.3 Task ipquality_destination

Instructs the device to start the passive (receiving) side of an IP quality measurement.

The `duration` field contains the duration of the measurement.

The device will send an intermediate response as soon as the receiving side is ready, and include both the device-local IP-address (`remote`) and `port`. This information is used by the server to setup the active (sending) side of the IP quality measurement afterwards.

Task example (JSON)

```
{
    "duration": 10,
    "dscp": 46,
    "ptime": 20,
    "psize": 200
}
```

Intermediate response example (JSON)

```
{
    "remote": "212.202.32.5",
    "port": 4711
}
```

Final response example (JSON)

```
{
    "p_s": 500,
    "p_r": 480,
    "t_n": 475,
    "t_min": 13,
    "t_max": 56,
    "t_sum": 14970,
    "t_sq": 311875,
    "j_n": 465,
    "j_max": 34,
    "j_sum": 2413,
    "j_sq": 11625
}
```

Explanation of fields in the response:

`p_s`: number of packets sent

`p_r`: number of packets received

`t_n`: number of packets valid for RTT (round-trip time) calculation

`t_min`: minimum RTT (ms)

`t_max`: maximum RTT (ms)

`t_sum`: sum of all RTT values of valid packets (ms)

`t_sq`: sum of all squared RTT values of valid packets (ms^2)

`j_n`: number of packets valid for jitter calculation

`j_max`: maximum jitter (ms)

`j_sum`: sum of all jitter values of valid packets (ms)

`j_sq`: sum of all squared jitter value of valid packets (ms^2)

12.3.4 Task ipquality_source

Instructs the device to start the active (sending) side of an IP quality measurement.

The `duration` field contains the duration of the measurement. The measurement should start immediately against the passive side (IP address `remote` and `port`) after the device receives this task.

Task example (JSON)

```
{
    "remote": "212.202.32.5",
    "port": 4711,
    "duration": 10,
    "dscp": 46,
    "ptime": 20,
    "psize": 200
}
```

Final response example (JSON)

```
{
    "p_s": 500,
    "p_r": 480,
```

```
    "t_n": 475,
    "t_min": 13,
    "t_max": 56,
    "t_sum": 14970,
    "t_sq": 311875,
    "j_n": 465,
    "j_max": 34,
    "j_sum": 2413,
    "j_sq": 11625
}
```

Explanation of fields in the response:

`p_s`: number of packets sent

`p_r`: number of packets received

`t_n`: number of packets valid for RTT (round-trip time) calculation

`t_min`: minimum RTT (ms)

`t_max`: maximum RTT (ms)

`t_sum`: sum of all RTT values of valid packets (ms)

`t_sq`: sum of all squared RTT values of valid packets (ms^2)

`j_n`: number of packets valid for jitter calculation

`j_max`: maximum jitter (ms)

`j_sum`: sum of all jitter values of valid packets (ms)

`j_sq`: sum of all squared jitter value of valid packets (ms^2)

12.3.5 Task license

Informs the device about the license status of the associated box object.

If there is a license, the task carries the license key in the `license` field.

If the license is present and valid, the `valid` field will have the value `true`, and `false` if not.

Task example (JSON)

```
{
    "license": "EXAMP-LELIC-ENSEK-EY007",
    "valid": true
}
```

Final response example (JSON)

```
{}
```

12.3.6 Task csp

Informs the device to send out a message in the cospace sensor protocol (cSP).

The `csp_xxx` fields represent the respective fields within the cSP. The `csp_payload` field contains the payload section of the cSP packet as a base64 encoded string. If `wait_recv` is present and true, it informs the device to first

wait for a transmission from the sensor before sending out the packet. If in addition `wait_timeout` is present, it is a hint for the device to set the timeout waiting for a transmission from the sensor.

Task example (JSON)

```
{
    "wait_recv": true,
    "wait_timeout": 30,
    "csp_proto_type": 0,
    "csp_address": "013a5e7f",
    "csp_packet_id": 7,
    "csp_time": 1363790008,
    "csp_payload": "wqc4bHc3ZW5kaWE0OTImMiEK"
}
```

Final response example (JSON)

```
{
    "status": "ok",
    "rssi": -80,
    "lqi": 45
}
```

The device will send a response when the sensor has confirmed the reception of the message (for message types that require acknowledgement) or when the message has been sent out (for message types that do not require acknowledgement).

If `status` contains `replaced`, this indicates that the cSP message has been superseded by another message while still waiting for the sensor to transmit.

If the `status` field does not contain `ok` or `replaced`, a `message` field should also be present describing the specific failure.

If the sensor confirmed the reception of the packet, the final response will contain the `rssi` and `lqi` fields to signal the received signal strength indication (in dBm) and the link quality indication (higher value means better signal) of the acknowledgement packet.

CHAPTER

THIRTEEN

INTERNAL API

13.1 API Call Reference

13.1.1 /api/jeye

GET /api/jeye/ (*host*)
Get Jeye statistic data for cluster host `host`

Query Parameters

- **from** – optional: start time, in seconds-since-epoch (must have `from` <= `to`)
- **to** – optional: end time, in seconds-since-epoch (must have `from` <= `to`)
- **start** – optional: the UUID of the first result to return, exclusive, i.e. the result with an exact match UUID is not returned.

  If given, overrides the `from` parameter when `asc` order is selected or overrides the `to` parameter when `desc` order is selected.
- **order** – optional: If set to `asc` (default), objects are returned in time ascending order (i.e. starting with `from` or `start` and ending with `stop`).

  If set to `desc`, objects are returned in time descending order (i.e. starting with `to` or `start` and ending with `from`).
- **count** – optional, default `10`: limits the number of returned objects (valid range: `1...5000`)

Response body example

```
{
    "jeye": {
        "c7c1e972-7747-11e1-85ea-0024e8f90cc0": {
            (...jeye data object...)
        },
        "d8c42a00-7747-11e1-a938-0024e8f90cc0": {
            (...jeye data object...)
        }
    }
}
```

Status

- **access-denied**: The current session's user does not have jeye access permissions
- **empty-data**: No statistics items were found

13.1.2 /api/mwi

GET /api/mwi/ (*phone*)
Get the current message waiting indication URI (MWI) mapping for the specified `phone`.

Response body example

```
{
    "mwi_uri": "sip:04066889977@bmcag.com"
}
```

Status

- **access-denied**: The user does not have special permissions required for this API call
- **wrong-phone**: The given phone number is not known to the system

To use this API call, the user needs to have special permissions in the cospace system.

If there is no MWI URI mapping, the `mwi_uri` field will not be present in the response.

POST /api/mwi
Modify the message waiting indication URI (MWI) mapping for one or more phone numbers.

Request body example

```
{
    "+4940382738473": "sip:04066889977@bmcag.com",
    "+4940123212328": "sip:04066889988@bmcag.com"
}
```

JSON Parameters

- **number** (*phone*) – `^\+[0-9]{3,20}$`

  the phone number to modify the MWI for
- **URI** (*MWI*) – `sip:<user>@<host>`

  the MWI URI (target for SIP NOTIFY messages)

```
{
    "phone-ok": [
        "+4940382738473"
    ],
    "phone-fail": [
        "+4940123212328"
    ]
}
```

Status

- **access-denied**: The user does not have special permissions required for this API call

To use this API call, the user needs to have special permissions in the cospace system.

The `phone-ok` array in the response contains phone numbers whose MWI mapping were successfully updated. The phone-fail array contains phone numbers that could not be updated because they are unknown to the system.

DELETE /api/mwi/ (*phone*)
Delete the current message waiting indication URI (MWI) mapping for the specified `phone`.

Response body example

```
{}
```

Status

- **access-denied**: The user does not have special permissions required for this API call
- **wrong-phone**: The given phone number is not known to the system

To use this API call, the user needs to have special permissions in the cospace system.

13.1.3 /api/statistics

GET /api/statistics/ (*date*)
Get the system statistics for the given `date`, in format `YYY-MM-DD`

Response body example

```
{
    "result": {
        (... statistics data object...)
    }
}
```

Status

- **access-denied**: The current session's user does not have jeye access permissions
- **empty-data**: No statistics items were found

13.1.4 /api/usage

GET /api/usage/ (*month*)
Get the monthly usage statistics, `month` in format `YYY-MM`

Response body example

```
{
    "result": {
        (... statistics data object...)
    }
}
```

Status

- **access-denied**: The current session's user does not have jeye access permissions
- **empty-data**: No statistics items were found

13.1.5 /api/idcardcheck

GET /api/idcardcheck
Get a list with validation jobs.

Response body example

```
{
    "jobs": [
        "c179ba70-e9e1-11e5-9546-c73f4b1584b3",
        "d76cc78c-e9e1-11e5-b37f-2f91a0480be5"
    ]
}
```

Status

- **access-denied**: The current session's user does not have the permission for idcardcheck

GET /api/idcardcheck/ (*uuid*) **/address**
Get the address data of the validation job `uuid`

Response body example

```
{
    "user": {
        "firstname": "John",
        "lastname": "Doe",
        "zip_code": "10557",
        "town": "Berlin",
        "street": "Willy-Brandt-Stra\u00dfe",
        "house_number": "15",
        "country": "de"
    }
}
```

Status

- **wrong-job**: The current UUID does not match a validation job
- **access-denied**: The current session's user does not have the permission for idcardcheck

GET /api/idcardcheck/ (*uuid*) **/idcard/**
name **.pdf** Get the PDF of the validation job `uuid`

Query Parameters

- **inline** – optional: if present, the response will include an HTTP `Content-Disposition` header (see RFC 2183) with a value of inline. Otherwise, the content disposition will be attachment.

Response body example

```
The binary data of PDF, or HTTP 404 without response body on failure.
```

`name` is a file name arbitrarily chosen by the user agent.

```
The content of the id card in PDF format on success, or
HTTP 404 without response body on failure.
```

This API is eligible for use with the session's download id (did). In this case, the object UUID is the job `uuid`. For more information on file download authentication, see *Authentication for file downloads*.

`POST /api/idcardcheck/` (*uuid*)

Send the result of the validation process.

Request body example

```
{
    "validation" : "ok"
}
```

JSON Parameters

- **`validation`** – `^ok|missmatch|baddocument|fail$`

  The result of the validation process

  - `ok` the address data match
  - `missmatch` the address data missmatch
  - `baddocument` the document dosen't match the requirements.
  - `fail` the document isn't readable

Response body example

```
{}
```

Status

- **wrong-job**: The current UUID does not match a validation job
- **access-denied**: The current session's user does not have the permission for idcardcheck

CHAPTER

FOURTEEN

LEGAL STATEMENT

Use of the cospace API (referred to in this document as the "API"), its documentation and any QSC services accessible via the API (collectively referred to as "Services") is subject to Terms of Use which represent a legal agreement between you and QSC AG ("QSC"). The Terms of Use can be found at http://cospace.de/developer.

In order to use the Services you must agree to these Terms of Use. By using the Services you understand and agree that QSC will treat your use of the Services as acceptance of these Terms of Use from that point onwards.

CHAPTER

FIFTEEN

CHANGE LOG

15.1 Version 1.14.0

- added call-notification to Rule API

15.2 Version 1.13.0

- added *POST /api/announcement/(uuid)/text2speech*

15.3 Version 1.12.0

- added `blf_pickup` to `service` field for CDR's in partner API
- Rule email & push notifications can contain templates

15.4 Version 1.11.0

- added call filter service ivr in *POST /api/partner/callfilter/(user-uuid)*
- added push-notification for android and iOS to Rule API
- added examples for CDR's in partner API

15.5 Version 1.10.0

- added ignore_ip_whitelisting flag to *POST /api/partner/user/(user-uuid)* and *GET /api/partner/user/(user-uuid)*

15.6 Version 1.9.0

- added baddocument to /api/idcardcheck & validation field to *GET /api/user*

15.7 Version 1.8.0

- added `Authorization: Bearer <sid>` header support
- added /api/idcardcheck section in *Internal API*.
- added *POST /api/user/idcard*.
- added validation field to *GET /api/user*

15.8 Version 1.7.0

- increased the max. count of sensor data/action items per request in *GET /api/sensor/(uuid)/data*
- added conference-offline error to *POST /api/conference/(uuid)/member*

15.9 Version 1.6.0

- added paging functionality to *GET /api/partner/user*
- added paging functionality to *GET /api/partner/phone*

15.10 Version 1.5.0

- added *GET /api/partner/user*
- added *GET /api/partner/phone*
- added *GET /api/partner/usergroup/credit*

15.11 Version 1.4.0

- added custom notification feature: *POST /api/sip/(sip-user)/notify*
- added date_start and date_end fields to *POST /api/dialplan/v2*

15.12 Version 1.3.0

- added parent & root field to *GET /api/partner/cdr/(phone)*
- change *POST /api/dialplan/v2* time field logic

15.13 Version 1.2.0

- added P_Asserted_Identity field to *POST /api/partner/phone/(user-uuid)* and *GET /api/partner/user/(user-uuid)*

- added BLF (Busy Lamp Field) Feature for SIP-Accounts *GET /api/sip/(sip-user)/blf* and *POST /api/sip/(sip-user)/blf*

15.14 Version 1.1.1

- JSON structure fix in :http:get:/api/partner/user/(user-uuid)

15.15 Version 1.1.0

- added *Interactive Voice Response API* API
- moved dialplan calls to a new section *Dialplan API*
- added dialplan v2 calls in *Dialplan API*
- added *GET /api/partner/emergency/(user-uuid)*
- added *POST /api/partner/emergency/(user-uuid)*
- added `on_hold_music` field to *GET /api/phone* and *POST /api/phone*

15.16 Version 1.0.39

- added *Calling Name Delivery* API

15.17 Version 1.0.38

- added *POST /api/partner/enterprise*
- added *GET /api/partner/enterprise*.
- added *POST /api/partner/enterprise/(uuid)*.
- added *DELETE /api/partner/enterprise/(uuid)*.
- added prefix flag to Usergroups
- added CDR event in partner api
- added *GET /api/partner/cdr/event*.
- added *DELETE /api/partner/cdr/event*.

15.18 Version 1.0.37

- added timeout property to *POST /api/device/session*

15.19 Version 1.0.36

- several formatting fixes
- adjusted Rules API

15.20 Version 1.0.35

- added email field to *POST /api/user/delete*
- added partner section for sensor /api/partner/sensor
- added new api event partner_sensor_data & partner_sensor_action

15.21 Version 1.0.34

- update *GET /api/rule/(rule-id)* to also return the declaration of a rule
- remove `GET /api/rule/(rule-id)/declaration`
- remove `POST /api/rule/(rule-id)/activate`
- remove `POST /api/rule/(rule-id)/deactivate`

15.22 Version 1.0.33

- add CDR (call detail record) support to partner related phones
- add usergroup feature
- add `usergroup` field to *GET /api/user* and *GET /api/partner/user*
- add hot billing support (zone filter)
- add callback feature, *POST /api/dialplan/callback*
- add `cti_via` field to *GET /api/dialplan* and *POST /api/dialplan*
- add callback as a call filter service operator
- add API for rule engine
  - add *POST /api/rule* to create a new rule
  - add *GET /api/rule* to fetch all rules
  - add *DELETE /api/rule/(rule-id)* to delete a specific rule
  - add *GET /api/rule/(rule-id)* to fetch a specific rule
  - add *POST /api/rule/(rule-id)* to update the state or name of a rule
- add user events for rule engine
  - add `rule_new` event in case a new rule was created
  - add `rule_delete` event in case a rule was deleted
  - add `rule_update` event in case a rule has changed

15.23 Version 1.0.32

- add sensor capability `octets`

15.24 Version 1.0.31

- add partner proxy authentication (*POST /api/session*)

15.25 Version 1.0.30

- add `wait_timeout` field to task csp.
- add `replaced` status to task csp response and to `sensor_action` event.
- add `info` field to *GET /api/signup/verification/(verification)* and *POST /api/signup/verification*.
- add `chat`, `uuid` and `time` fields to iOS `chat_message` push notification.

15.26 Version 1.0.29

- add `model`, `profile` fields to *GET /api/tag/(uuid)/object* for object of types sensor to enable distinguishing sensor types in the overview
- add `profile`, `recv_after_send` fields to *GET /api/sensor/(uuid)*.
- add `tamper_detect`, `fault_detect` fields to *GET /api/sensor/(uuid)*. These fields can be cleared using *POST /api/sensor/(uuid)*
- read access to a sensor is now sufficient to initiate sensor actions (*POST /api/sensor/(uuid)/action*)
- modify the content format of sensor capabilities energy, voltage, current and power
- renamed sensor capability door to open (see *GET /api/sensor/(uuid)*)
- several new sensor capabilities (see *GET /api/sensor/(uuid)*)

15.27 Version 1.0.28

- add streaming download of sensor data and sensor action data in CSV format (*GET /api/sensor/(uuid)/data/(xxx).csv* and *GET /api/sensor/(uuid)/action/(xxx).csv*)
- add bulk upload of sensor data via *POST /api/sensor/(uuid)/data*

15.28 Version 1.0.27

- add chat media messages, types image and audio
- add call filter (*GET /api/partner/callfilter/(user-uuid)*, *POST /api/partner/callfilter/(user-uuid)* and *DELETE /api/partner/callfilter/(user-uuid)*)

- add concurrent call limit
- add `country_code` to *POST /api/signup/user*, *GET /api/user*, *POST /api/user*, *POST /api/partner/user/(user-uuid)*
- add `call_enable`, `dialplan_enable` and `play_announcement_only` to *POST /api/partner/phone/(user-uuid)*

15.29 Version 1.0.26

- added `battery_ok`, `battery_status` and `mains_power` fields to *GET /api/sensor/(uuid)*
- added `color_rgb` sensor and actor capabilities

15.30 Version 1.0.25

- modify Apple push notifications to match "silent notification" style
- add `recv_interval` field to *GET /api/sensor/(uuid)*

15.31 Version 1.0.24

- rename sensor action capability flip to onoff. Add sensor data capabilities onoff and text. Add both sensor data and actor capability button
- add new internal statistic calls

15.32 Version 1.0.23

- add `recv_time` field to *GET /api/sensor/(uuid)*

15.33 Version 1.0.22

- Support gzip encoding for various API calls (via HTTP `Accept-Encoding` header)
- remove binary parameter for *GET /api/jeye/(host)*

15.34 Version 1.0.21

- new sensor data capabilities: light, humidity, door

15.35 Version 1.0.20

- add order parameter to *GET /api/volume/(uuid)/commit*
- added clarification section on the use of UUIDs

15.36 Version 1.0.19

- add volume commit log feature: *GET /api/volume/(uuid)/commit*, commit field in file events `file_new`, `file_modify`, `file_delete` and `commit` field in several `POST /api/volume/(uuid)/...` and `DELETE /api/volume/(uuid)/...` calls
- add owner field to all GCM notifications

15.37 Version 1.0.18

- add user profile pictures (*POST /api/user/picture* and *DELETE /api/user/picture*, *GET /api/user/(user-id)/picture/(size)/(xxx).jpg*, extensions to *GET /api/user*, *GET /api/user/(user-uuid)*, *GET /api/user/link*)
- add `display_name` property to support user-controlled name display (*GET /api/user* and *POST /api/user*, *GET /api/user/(user-uuid)*, *GET /api/user/link*, *POST /api/signup/user*, *GET /api/invitation/(invitation)*, *GET /api/partner/user/(user-uuid)*)
- add *GET /api/language* call to inform the clients about languages supported by the system
- add concurrent-access failure status for *POST /api/phone*
- add user section to *GET /api/chat*

15.38 Version 1.0.17

- add event system filters (*POST /api/event/filter* and *DELETE /api/event/filter*)
- add new sensor capabilities `energy`, `voltage`, `current`, `power`, `frequency`
- add generic object type xobject (*GET /api/xobject/(uuid)* and *POST /api/xobject*)
- add metadata events `object_metadata` and `user_metadata`

15.39 Version 1.0.16

- add metadata information to objects and users

15.40 Version 1.0.15

- add `time` fields to *GET /api/chat/(uuid)/message*, event `chat_message` and push notification
- lower limit for parameter psize is now 80 for various ipquality calls

15.41 Version 1.0.14

- add event `user_modify`
- add `dscp`, `psize`, `ptime` fields to *GET /api/ipquality/(uuid)* and *POST /api/ipquality*

- add `description` field to *GET /api/sip* and *POST /api/sip*, *POST /api/sip/(sip-user)*, `sip_new`, `sip_modifiy` events
- add `fax_ident`, `fax_header` fields to *GET /api/user* and *POST /api/user*, *POST /api/fax/(uuid)/send*
- add object links and corresponding events `object_link_new` `object_link_delete`
- add physical location fields to *GET /api/user* and *POST /api/user*
- change process of obtaining and unlocking a phone number (*GET /api/phone* and *POST /api/phone*), remove phone offer process
- call-through feature, *POST /api/dialplan/callthrough*
- call-reverse feature, *POST /api/dialplan/callreverse*, `callreverse_enable` flag in *GET /api/dialplan* and *POST /api/dialplan*. Replace `incoming_call` event with `dialplan_callreverse_start` and `dialplan_callreverse_stop` events. New push event `dialplan_callreverse_start`, `dialplan_callreverse_stop`.
- chat API and events
- call API and events, `call_enable` flag for *GET /api/phone* and *POST /api/phone*
- add license section / field to *GET /api/box/(uuid)* and *POST /api/box/(uuid)*, task license in device API
- sensor API and events
- add "share tags" to allow for sharing with other individual users: fields `share_read`, `share_write` and `share_propagate` in *GET /api/user*, *GET /api/user/link*
- enable use of download id (did) authentication for *GET /api/fax/(uuid)/(page)/(xxx).png*
- possibility to download ipquality results in CSV format: *GET /api/ipquality/(uuid)/result/(xxx).csv*

15.42 Version 1.0.13

- presentation API and events
- added email field to *POST /api/password*
- add deleted field to *GET /api/user/link*, *GET /api/user/(user-uuid)*
- internal API for MWI (message waiting indication) support
- support for Apple Push (APS), Google Cloud Messaging (GCM)
- additional status already-invited for *POST /api/user/invitation*
- add `dial_prefix`, `clip_number` and `clir_enable` fields to *GET /api/dialplan* and *POST /api/dialplan*
- read privileges are now sufficient to create a comment with *POST /api/object/(uuid)/comment*
- added `sip_status_code`, `fax_error_code` fields to *GET /api/fax/(uuid)* and to `fax_report` event
- add `callreverse_enable` field to *GET /api/dialplan* and *POST /api/dialplan*
- added `l_n`, `rfac` results to *GET /api/ipquality/(uuid)/result*

15.43 Version 1.0.12

- session authentication (*POST /api/session*) and password recovery (*POST /api/password/recovery*) is now possible with the username *or* the e-mail address
- added `already-exist` failure condition to *POST /api/signup/verification*, *POST /api/signup/user*, *POST /api/user* since e-mail addresses now have to be unique
- added internal invitations (i.e. invitations directed specifically to another known user) to *GET /api/user/invitation* and *POST /api/user/invitation*
- added accept section to *POST /api/user/link*
- added events `invitation_new` and `invitation_delete` for internal invitations
- added know section to *GET /api/user/link* (this enables link suggestions for the current user)
- added `play_announcement_only` flag to *GET /api/phone* and *POST /api/phone*
- added feature section to *GET /api/user*, *GET /api/partner/user/(user-uuid)* and *POST /api/partner/user/(user-uuid)*
- added *DELETE /api/signup/verification/(verification)*
- added `mute_default` parameter to *GET /api/conference/(uuid)* and *POST /api/conference/(uuid)*
- added *GET /api/user/(user-uuid)* to get information about other users
- added *GET /api/fax/(fax-uuid)/report/(report-uuid)/(xxx).pdf* (fax sender report in PDF format)

15.44 Version 1.0.11

- modified ipquality duration to only accept values >= 10
- `dialplan` and `sip` API and events
- `dialplan_enable` flag for *GET /api/phone* and *POST /api/phone*

15.45 Version 1.0.10

- added `hw_version` and `sw_version` fields to *POST /api/device/session* and *GET /api/box/(uuid)*, and to `box_online` event
- added reload an upgrade tasks to device API

15.46 Version 1.0.9

- box and ipquality API
- device API
- new events `box_online`, `box_offline`
- new call *POST /api/user/delete* to permanently delete a user account

- added possibility to permanently delete a user to *POST /api/partner/user/(user-uuid)*
- internal system statistics via *GET /api/jeye/(host)*
- add purge parameter to various `DELETE` calls that delete objects
- add *DELETE /api/trash* call to purge the objects in trash tag
- add (optional) purge field to `object_delete` event
- add `trash_purge` event
- add `newsletter` field to *GET /api/user*, *POST /api/user*, *POST /api/signup/user*

15.47 Version 1.0.8

- added `pin` parameter to change phone PIN to *POST /api/user*
- added `inline` parameter to binary download methods to control HTTP `Content-Disposition` header

15.48 Version 1.0.7

- added download id (did) authentication to session an various API calls.

15.49 Version 1.0.6

- added *GET /api/volume/(uuid)/folder/(folder-uuid)/(xxx)* to retrieve folder contents as a compressed ZIP file

15.50 Version 1.0.5

- added *POST /api/partner/user/(user-uuid)* to allow partner to disable/enable user accounts

15.51 Version 1.0.4

- added partner authentication to *POST /api/session*
- added partner-controlled sign-up process, involves changes in: *POST /api/signup/verification* *GET /api/signup/verification/(verification)* *POST /api/signup/user* *GET /api/session* (partner field) *POST /api/session* (partner field) *POST /api/password* (partner field) *GET /api/user* (partner field) *GET /api/phone* (partner field)
- added partner API calls: *GET /api/partner/user/(user-uuid)* *POST /api/partner/recovery/(user-uuid)* *POST /api/partner/phone/(user-uuid)*
- added phone-forbidden status message to `GET /api/phone/gateway/(country)` `GET /api/phone/offer/(country)/(gateway)`
- added mode parameter to *POST /api/fax/(uuid)/(xxx).pdf*

15.52 Version 1.0.3

- added object type volume and `/api/volume/...` API calls
- added forbidden-to response to *POST /api/fax/(uuid)/send*
- added orientation parameter to *POST /api/fax/(uuid)/(xxx).pdf*

15.53 Version 1.0.2

- added quota-exceeded and no-content responses to *POST /api/fax/(uuid)/send*
- added status field to *GET /api/fax/(uuid)*, more types in status field in *GET /api/recording/(uuid)*, *GET /api/announcement/(uuid)*
- allow undelete of objects by removing the `tag_trash` with *POST /api/object/(uuid)/tag*

15.54 Version 1.0.1

- support for sending e-mail notifications on incoming fax/announcement/recording:

  added `notification_email`, `notification_attachment` fields to *GET /api/phone*

  added `notification_email`, `notification_attachment` fields to *POST /api/phone*
- server-side multi-language / localization support:

  added language parameter to *GET /api/session*

  added language field to *POST /api/signup/user*

  added language field to *GET /api/user*

  added language field to *POST /api/user*
- allow user to modify email address: added email and code fields to *POST /api/user*
- possibility to page through object comments: added start, to, from, count and order parameters to *GET /api/object/(uuid)/comment*
- merged `{fax, announcement, recording, contact, conference}_{new,modify, delete, share, unshare}` events into generic `object_{new, modify, delete, share, unshare}` events
- new event `object_tag` sent if tags attached to an object are changed (`object_modify` is not sent in this case any more)
- new events `comment_{new, modify, delete}` to explicitely signal comment actions (`object_modify` is not sent in this case any more)
- add `owner` and `description` fields to `conference_{start, stop}` events

15.55 Version 1.0.0

- added tag, search parameters to *GET /api/tag/(uuid)/object*

/api