Introduction

This document defines APIs and protocols to enable integration between different web-services, servers and clients.

Open Collaboration Services v2.0 (“OCS v2.0”) consists of a set of API endpoints mainly targeted to be implemented by consumers and providers of file storage / sharing servers (“cloud”).

The goal is to enable Integration of cloud services, web services and social communities with each other, with desktop and mobile applications. This must be done in a decentralized and federated way, free and secure, privacy protected and vendor independent. OCS aims to solve these problems.

Notational Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119.

Terminology

- **“Cloud”:**
  - Referring to file storage / sharing servers.
- **“Consumer”:**
  - A desktop or mobile application or web service that access services that are provided by a server.
- **“Service provider” or “server”:**
  - A web service or server who provides a open collaboration services compatible APIs.
- **“Provider service list”:**
  - A JSON configuration endpoint which specifies which services are provided by a server or service provider.
- **“Module” or “service”:**
  - OCS is aimed to be a modular system allowing providers to only implement the required subsets. Any functionality is thus encapsulated in so called “modules” or “services” which are all optional.

Security Considerations

TRANSMISSION LAYER

All data transfer SHOULD be TLS encrypted to ensure the integrity and security of transferred data.

Consumers MUST properly validate the certificate chain and in case of an error cancel the connection and SHOULD notify the user about the occurred problem.

For enhanced security Consumers MAY follow the Public Key Pinning Extension for HTTP (RFC 7469) as well as other HTTP security best practices.

SECRETS

Any secrets used to exchange data MUST be generated using a strong random number generator, such as `/dev/urandom` or another cryptographically secure replacement.

Performance / Scalability

The service must be usable by a lot of users in parallel. Because of that it is important to build the architecture in a scalable way. Every component of the architecture must be cluster enabled, accessible in a parallel way and stateless.

COOKIES

To work together with load-balanced environments consumers SHOULD resend any cookies as defined in RFC 6265. As stated in the “Authentication” section any Basic Auth authentication header MUST be resend as the session referenced by the cookie MAY expire.

It shall be noted that OCS endpoints MUST behave properly regardless whether cookies are resent or not.

Formats / Encoding

To avoid interoperability problems this section defines formats and encoding that OCS endpoints as well as consumers MUST obey.

ENCODING

Any data sent and received MUST be UTF-8 encoded.

DATE AND TIME FORMAT

All dates and time representations MUST be in ISO 8601 format. This means all of the following values are valid:

- Date
  - 2015-06-03
- Combined date and time in UTC
  - 2015-06-03T13:21:58+00:00
  - 2015-06-03T13:21:58Z
- Week
  - 2015-W23
- Date with week number
  - 2015-W23-3
- Ordinal date
  - 2015-154

Consumers as well as endpoints SHALL NOT make assumption about the representation as long as it follows ISO 8601. The preferred variation is though the regular Date representation “YYYY-MM-DD”.

If an invalid date format has been provided consumers and endpoints MAY stop processing the data.

OCS Responses

Example of a JSON OCS response, be aware that empty fields MUST have a value of `null`:

```
{
    "ocs": {
        "meta": {
            "status": "ok",
            "statuscode": 200,
            "message": null
        },
        "data": {
            "users": [
                "Frank",
                "admin",
                "test",
                "test1"
            ]
        }
    }
}
```

Example of a XML OCS response, be aware that empty fields MUST still be existent:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data>
  <users>
   <!-- List of user names -->
   <element>Frank</element>
   <element>admin</element>
   <element>test</element>
   <element>test1</element>
  </users>
 </data>
</ocs>
```

OCS endpoints MUST be able to output data in a XML format as well as a JSON representation. This means that the returned XML output SHALL NOT contain any attributes as these cannot be mapped properly to a JSON object.

To specify the encoding a GET parameter `format` can be send by the consumer with one of the following values:

- `json`
    - Returns a JSON formatted output, the Content-Type MUST be set to `application/json; charset=utf-8`
- `xml`
    - Returns a XML formatted output, the Content-Type MUST be set to `text/xml; charset=UTF-8`

A OCS response MUST consist of the following elements:

- `ocs`: Array that contains the whole response
    - `meta`: Array that contains meta information
        - `status`: The status of the response, either “ok” or “fail”. MUST be “ok” if statuscode is set to 200, “fail” otherwise.

        - `statuscode`: The OCS status code of the response. 200 indicates a successful response.
        - `message`: An optional message that MAY contain a status message, such as a error message.
    - `data`: Array that contains the actual response, content of the array depends completely on the endpoint.

> The statuscode returned in the OCS response MAY differ by the HTTP status code. In the following module specifications the returned OCS and HTTP status codes are specified. It is encouraged to use restful status codes and use the same OCS and HTTP status code.

Authentication

```
GET /{endpoint} HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

Make sure to replace `YWRtaW46YWRtaW4=` with the encoded credentials.

Clients SHOULD directly try HTTP Basic Authentication and not try to perform a Digest Authentication based authentication. This will save the amount of requests originating to the endpoint.

Authentication is performed by sending a Basic HTTP Authentication header. Thus OCS endpoints MUST support the Basic Auth Access Authentication system as defined in RFC2617.

The credentials provided in the authentication header MUST be UTF-8 encoded. Taking `contrasea` as example the correct encoding would be `Y29udHJhc2XDsWE=`, any other encoding such as ISO-8859-1 (`Y29udHJhc2XxYQ==`) is considered invalid and MUST NOT be used.

OCS compatible APIs MAY use cookies as defined in RFC 6265 to authenticate future requests as well, if after successful authentication the endpoint sends further cookies clients SHOULD resend these for future requests. However, any request MUST then include the Basic Authentication header as well as the provided cookies as the session might expire. When resending cookies consumers MUST add the following HTTP header to all requests: `OCS-REQUEST: true`.

OCS endpoints MUST in all cases handle connections correctly regardless whether cookies are sent.

> The base64-encoded authentication header does not necessarily have to be the user credentials. The credentials could also be an unique and secret token used for authentication. (Token-based authentication)

Service Discovery

The provider service list are used by OCS for service discovery purposes to allow mapping the provided services and endpoint URIs.

Every service provider MUST provide a provider service list located as `/ocs-provider/` under the directory of the application. For example if there would be an OCS compatible application running at `https://demo.owncloud.org/` the provider service list MUST be located at `https://demo.owncloud.org/ocs-provider/`. If the application would run on `https://mycloud.com/mycloud/` the service MUST be accessible from `https://mycloud.com/mycloud/ocs-provider/`.

Serving the provider service list at this location allows providers to generate them dynamically also in environments where programming languages pose limitations on the routing, such as PHP. In the PHP scenario this would mean that a `index.php` file stored under `/ocs-provider/` would be served.

A provider service list consists of the following elements:

1. `version`: Version identifier
    - Integer indicating the current version of the supported OCS standard. (e.g. `int(2)`)
2. `services`: Array of services
    - A of services served by OCS from this instance. Services are defined as following:
    - `key`: Name of the endpoints
    - `version`: Version of the specific endpoint (may differ from the OCS version)
    - `endpoints`: Array of endpoint names and the relative URI - `key`: Name of the endpoint - `value`: Relative URI to the endpoint

A provider service definition MUST look like the following (the endpoint URI MAY be changed by providers):

```
{
    "version": 2,
    "services": {
        "FEDERATED_SHARING": {
            "version": 1,
            "endpoints": {
                "share": "/ocs/cloud/shares",
                "accept": "/ocs/cloud/shares"
            }
        },
        "PRIVATE_DATA": {
            "version": 1,
            "endpoints": {
                "getattribute": "/ocs/privatedata/getattribute",
                "setattribute": "/ocs/privatedata/setattribute",
                "deleteattribute": "/ocs/privatadata/deleteattribute"
            }
        }
    }
}
```

In this case if an instruction of the module `PRIVATE_DATA` serving at `demo.example.org/mycloud/` defines to interact on the endpoint `getattribute` the absolute URI would resolve to `demo.example.org/mycloud/ocs/privatedata/getattribute`.

Cross-Origin communication

Due to the Same-Origin-Policy it is usually not possible for browsers to access the content of files hosted on another domain. To enable browser-based clients, providers are encouraged to set proper CORS headers on the resource to allow cross-domain access to the provider service list.

If a provider service list cannot be accessed by a consumer, consumers MUST perform the OCS request and in case of a `404`

failure consider the endpoint to be not existent.

Versioning

When declaring a version within the service the service MUST stay backwards compatible on the defined URI. Providers MAY remove the endpoint but SHALL NOT make an existing endpoint incompatible.

An idiomatic approach to this would be to prepend versions to the URI, such as `/api/1/` for version 1 of an endpoint and `/api/2/` for the backwards incompatible version 2 of an API.

Modules

OCS is based on modules providing specialized functionalities. A service provider MAY only implement a smaller subset of a defined module.

Providers MUST ensure that only supported modules are listed in the provider service list. Providers MAY not include modules into the providers service list if the endpoints are considered private API.

The current specified modules are:

- **FEDERATED_SHARING:**
  - Allows different cloud service providers to share files over different instances.
- **PRIVATE_DATA:**
  - Key-Value store allowing consumers to store and retrieve values.
- **SHARING:**
  - Allows consumers to share local files.
- **PROVISIONING:**
  - Allows consumers to manage users, groups and applications on the instance.
- **ACTIVITY:**
  - Activity stream provided by the server

More modules are subject to be added to future OCS versions. Furthermore providers MAY provide their own modules if these get a vendor prefix. (e.g. “owncloud-print” for a fictional printing API of the “ownCloud” product)

Federated Sharing

A valid service definition looks as following:

```
{
    "version": 2,
    "services": {
        "FEDERATED_SHARING": {
            "version": 1,
            "endpoints": {
                "share": "/ocs/v2.php/cloud/shares",
```

```
            "webdav": "/public.php/webdav/"
        }
    }
}
}
```

The “FEDERATED_SHARING” module allows consumers to share files between service providers compatible with this module. This module handles the following tasks:

- Sending a share offer to other providers (`share`)
- Accepting a share offer from other providers (`share`)
- Denying a share offer from other providers (`share`)
- Unsharing a previously shared file (`share`)
- Accessing shared files using WebDAV (`webdav`)

Sending a share offer

This endpoint takes share offers from remote instances, once the recipient has logged-in he is expected to accept or deny the share.

HTTP REQUEST

```
POST /ocs/v2.php/cloud/shares HTTP/1.1
Host: localhost
Content-Length: 119
Content-Type: application/x-www-form-urlencoded

shareWith=RecipientName&token=MyRandomToken&name=Documents&remoteId=5&owner=ShareingUser&remote=http://localhost
```

`POST http://example.com/{share}`

QUERY PARAMETERS

Parameter	Default	Description
shareWith		User name of the receiving user.
token		Unique and secret token used to access the file.
name		Name of the file or folder.
remoteId		Unique ID to identify the file on the sender side, used for accepting and denying shares.
owner		User name of the sending user.
remote		URI of the sending instance.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
```

```
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response is, in case of success, a OCS success message. In error cases the following OCS status codes MUST be used:

OCS status code	HTTP status code	Meaning
400	400	Invalid parameters
503	503	If the server does not support Federated Sharing (i.e. disabled by administrator)

In case of an error consumers SHOULD be made aware of the error. The OCS message MUST be used by the endpoint to specify a proper error message that can be used to analyze issues.

> ℹ This request needs to be sent to the receiving instance by the sending instance. This request is not expected to be sent directly by an user.

Accept a share offer

After a share offer has been received the receiving instance should notify the user in question and give the possibility to accept or deny a share offer. Using this API call a federated share can be accepted.

Providers MAY inform the sending user if a share has been accepted.

HTTP REQUEST

Assuming a provider wants to accept the share request of the file sent in the `share` endpoint (`remoteId: 6`, `token: TjuMOB1as7iZ1c6`) the following request needs to be sent to the remote (`http://sender.example.org`):

```
POST /ocs/v2.php/cloud/shares/6/accept HTTP/1.1
Host: sender.example.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

token=TjuMOB1as7iZ1c6
```

`POST http://example.com/{share}/{remoteId}/accept`

URL PARAMETERS

Parameter	Default
remoteId	Received `remoteId` of the share.

QUERY PARAMETERS

Parameter	Default	Description
token		Unique and secret token used to access the file.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST in any case be a OCS success message even if the deny did not work. The only exception being 503 when Federated Sharing is not enabled.

OCS status code	HTTP status code	Meaning
503	503	If the server does not support Federated Sharing (i.e. disabled by administrator)

ℹ This request needs to be sent to the sending instance by the receiving instance. This request is not expected to be sent directly by an user.

Reject a share

This endpoint informs the sender that the recipient rejected the share. This endpoint is also intended to be used if the user first accepted the share and later decides to unshare it.

HTTP REQUEST

Assuming a provider wants to reject the share request of the file sent in the `share` endpoint (`remoteId: 6`, `token: TjuMOB1as7iZ1c6`) the following request needs to be sent to the remote (`http://sender.example.org`):

```
POST /ocs/v2.php/cloud/shares/6/decline HTTP/1.1
Host: sender.example.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

token=TjuMOB1as7iZ1c6
```

```
POST http://example.com/{share}/{remoteId}/accept
```

URL PARAMETERS

Parameter	Default
remoteId	Received `remoteId` of the share.

QUERY PARAMETERS

Parameter	Default	Description
token		Unique and secret token used to access the file.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST in any case be a OCS success message even if the deny did not work. The only exception being 503 when Federated Sharing is not enabled.

OCS status code	HTTP status code	Meaning
400	400	Invalid parameters
503	503	If the server does not support Federated Sharing (i.e. disabled by administrator)

ℹ This request needs to be sent to the sending instance by the receiving instance. This request is not expected to be sent directly by an user.

Unshare a file

Allows owners of shared files and folders to notify recipient of a revocation of the access permissions.

ℹ Instances MUST be able to handle not accessible remote storages gracefully even when a folder has not been unshared. The unshared endpoint is there to indicate to a remote host that a storage has finally removed and is not just temporarily unavailable.

HTTP REQUEST

Assuming a provider wants to unshare the share of the file sent to the `share` endpoint (`remoteId: 6, token:`

`TjuMOB1as7iZ1c6`) the following request needs to be sent to the receiver (`http://receiver.example.org`):

```
POST /ocs/v2.php/cloud/shares/8/unshare?format=json HTTP/1.1
Host: receiver.example.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

token=kW1gR9TRKXW9Jwk
```

`POST http://example.com/{share}/{remoteId}/unshare`

URL PARAMETERS

Parameter	Default
remoteId	Received `remoteId` of the share.

QUERY PARAMETERS

Parameter	Default	Description
token		Unique and secret token used to access the file.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST in any case be a OCS success message even if the deny did not work. The only exception being 503 when Federated Sharing is not enabled.

OCS status code	HTTP status code	Meaning
400	400	Invalid parameters
503	503	If the server does not support Federated Sharing (i.e. disabled by administrator)

ℹ This request needs to be sent to the sending instance by the receiving instance. This request is not expected to be sent directly by an user.

Accessing shared files

To access the file `sharedfile.txt` of the federated share with the token `kW1gR9TRKXW9Jwk` the following request has to be sent. Note that as password the base64 representation of `kW1gR9TRKXW9Jwk:` is used. The ending `:` is required.

```
GET /public.php/webdav/sharedfile.txt HTTP/1.1
Host: localhost
Authorization: Basic a1cxZ1I5VFJLWFc5SndrOg==
OCS-REQUEST: true
```

Shares files may be accessed using WebDAV (RFC 4918) under the specified `webdav` endpoint using HTTP Basic Authentication.

The credentials will be of the value `token:`, note the empty password field.

Sharing

A valid service definition looks as following:

```
{
    "version": 2,
    "services": {
        "SHARING": {
            "version": 1,
            "endpoints": {
                "share": "/ocs/v2.php/apps/files_sharing/api/v1/shares"
            }
        }
    }
}
```

The `SHARING` module allows to handle file sharing on the same cloud instance. It offers the following functions:

1. Get a list of shares (`share`)
2. Get a list of shared files in a folder (`share`)
3. Get information about a specific share (`share`)
4. Create a new share (`share`)
5. Delete an existing share (`share`)
6. Update an existing share (`share`)

List of shares

Get a list of all shared files for the currently logged-in user.

HTTP REQUEST

Lists all shares of the user admin with the password admin:

```
GET /ocs/v2.php/apps/files_sharing/api/v1/shares HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

`POST http://example.org/{share}`

QUERY PARAMETERS

Parameter	Default	Description
shared_with_me	`false`	Whether files shared with the user should get displayed. Defaults to `false`, `true` to only display shares that the user received.
path	'/'	Path to folder, if empty no restriction is set.
reshares	`false`	Whether reshares should get returned. (optional)
subfiles	`false`	Whether all shares within the folder should be returned. (optional)

RESPONSE

In case there are shares a successful OCS response object MUST be returned by the server, the shares MUST be embedded in the data element and at least consist of the following elements:

```
<element>
  <!-- Unique identified of the share (integer) -->
  <id>1</id>
  <!-- The item_type either "file" or "folder" (string) -->
  <item_type>file</item_type>
  <!-- '0' = user; '1' = group; '3' = public link (integer) -->
  <share_type>2</share_type>
   <!-- In case when shared with a user or a group the name (string) -->
  <share_with>Rachel</share_with>
  <!-- Path where the share is located (string) -->
  <path>/sharedFile.txt</path>
  <!-- Permissions: 1 = read; 2 = update; 4 = create; 8 = delete; 16 = share; 31 = all (default: 31, for public shares: 1) (integer) -->
  <permissions>1</permissions>
  <!-- Date when the share should expire (ISO 8601) -->
  <expiration>2015-06-12</expiration>
  <!-- Unique token that may be used to access the file in case of a public link (string) -->
  <token></token>
  <!-- Unique token that may be used to access the file in case of a public link (string) -->
  <uid_owner>Oscar</uid_owner>
  <!-- Human readable name, may be chosen by the user itself (string) -->
  <displayname_owner>Oscar Meyer</displayname_owner>
</element>
```

The expected response MUST in any case be a OCS success message containing the share data.

OCS status code	HTTP status code	Meaning
400	400	Not a folder (if the `subfile` argument was used)
404	404	User has no shared files or folder does not exist.

997	401	Authentication was not successful.

Get information about a share

Get information about a specific share using the share id.

HTTP REQUEST

Following request would request information about the share with the ID “1” under the context of the user “admin”.

```
GET /ocs/v2.php/apps/files_sharing/api/v1/shares/1 HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
POST http://example.com/{share}/{shareId}
```

URL PARAMETERS

Parameter	Default	Description
shareId		ID of the requested share.

RESPONSE

In case of success, a OCS success message with the following data structure is returned:

```
<element>
  <!-- Unique identified of the share (integer) -->
  <id>1</id>
  <!-- The item_type either "file" or "folder" (string) -->
  <item_type>file</item_type>
  <!-- '0' = user; '1' = group; '3' = public link (integer) -->
  <share_type>2</share_type>
   <!-- In case when shared with a user or a group the name (string) -->
  <share_with>Rachel</share_with>
  <!-- Path where the share is located (string) -->
  <path>/sharedFile.txt</path>
  <!-- Permissions: 1 = read; 2 = update; 4 = create; 8 = delete; 16 = share; 31 = all (default: 31, for public shares: 1) (integer) -->
  <permissions>1</permissions>
  <!-- Date when the share should expire (ISO 8601) -->
  <expiration>2015-06-12</expiration>
  <!-- Unique token that may be used to access the file in case of a public link (string) -->
  <token></token>
  <!-- Unique token that may be used to access the file in case of a public link (string) -->
  <uid_owner>Oscar</uid_owner>
  <!-- Human readable name, may be chosen by the user itself (string) -->
  <displayname_owner>Oscar Meyer</displayname_owner>
</element>
```

The expected response is, in case of success, a OCS success message containing the data structure. In error cases the following OCS status codes MUST be used:

OCS status code	HTTP status code	Meaning
404	200	Share does not exist.
997	401	Authentication was not successful.

In case of an error consumers SHOULD be made aware of the error. The OCS message MUST be used by the endpoint to specify a proper error message that can be used to analyze issues.

Create a new share

Shares a file or folder with an user on the same instance.

HTTP REQUEST

Following request will share the file `/welcome.txt` of the user `admin` with the user `test`:

```
POST /ocs/v2.php/apps/files_sharing/api/v1/shares HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 44
Content-Type: application/x-www-form-urlencoded

path=/welcome.txt&shareType=0&shareWith=test
```

```
POST http://example.com/{share}
```

QUERY PARAMETERS

Parameter	Default	Description
path		Absolute path to the file/folder which should be shared.
shareType		0 = user; 1 = group; 3 = public link
shareWith		User or group id with which the file or folder should be shared.
publicUpload	`false`	Whether to allow public upload to a public shared folder.
password		Password to protect a publicly shared file with.
permissions	31	1 = read; 2 = update; 4 = create; 8 = delete; 16 = share; 31 = all

RESPONSE

In case the share has been created a successful OCS response object MUST be returned by the server, the shares MUST be embedded in the data element and at least consist of the following elements:

```
<element>
  <!-- Unique identified of the share (integer) -->
  <id>1</id>
```

```
  <!-- The item_type either "file" or "folder" (string) -->
  <item_type>file</item_type>
  <!-- '0' = user; '1' = group; '3' = public link (integer) -->
  <share_type>2</share_type>
   <!-- In case when shared with a user or a group the name (string) -->
  <share_with>Rachel</share_with>
  <!-- Path where the share is located (string) -->
  <path>/sharedFile.txt</path>
  <!-- Permissions: 1 = read; 2 = update; 4 = create; 8 = delete; 16 = share; 31 = all (default: 31, for public shares: 1) (integer) -->
  <permissions>1</permissions>
  <!-- Date when the share should expire (ISO 8601) -->
  <expiration>2015-06-12</expiration>
  <!-- Unique token that may be used to access the file in case of a public link (string) -->
  <token></token>
  <!-- Unique token that may be used to access the file in case of a public link (string) -->
  <uid_owner>Oscar</uid_owner>
  <!-- Human readable name, may be chosen by the user itself (string) -->
  <displayname_owner>Oscar Meyer</displayname_owner>
</element>
```

The expected response is, in case of success, a OCS success message with the defined data structure. In error cases the following OCS status codes MUST be used:

OCS status code	HTTP status code	Meaning
400	400	Unknown share type.
404	404	File could not get shared.
997	401	Authentication was not successful.

Unshare an existing share

Unshares a shared file or folder.

HTTP REQUEST

Following request will delete the share with the id `1` of the user `admin`:

```
DELETE /ocs/v2.php/apps/files_sharing/api/v1/shares/1 HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
DELETE http://example.com/{share}/{shareId}
```

URL PARAMETERS

Parameter	Default	Description
shareId		ID of the share to revoke.

RESPONSE

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response is, in case of success, a OCS success message with the defined data structure. In error cases the following OCS status codes MUST be used:

OCS status code	HTTP status code	Meaning
404	200	Share could not get unshared.
997	401	Authentication was not successful.

Update an existing share

Updates an existing share such as adjusting the permissions or the password.

HTTP REQUEST

Following request will set the permissions of the share with the id `9` to `31:

```
PUT /ocs/v2.php/apps/files_sharing/api/v1/shares/9 HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 14
Content-Type: application/x-www-form-urlencoded

permissions=31
```

`PUT http://example.com/{share}/{shareId}`

URL PARAMETERS

Parameter	Default	Description
shareId		ID of the share to update.

QUERY PARAMETERS

Parameter	Default	Description
publicUpload	`false`	Whether to allow public upload to a public shared folder
password		Password to protect a publicly shared file with

permissions	1 = read; 2 = update; 4 = create; 8 = delete; 16 = share; 31 = all

RESPONSE

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response is, in case of success, a OCS success message with the defined data structure. In error cases the following OCS status codes MUST be used:

OCS status code	HTTP status code	Meaning
400	400	Invalid parameters.
403	200	Public upload disabled by the admin.
404	404	Share could not get updated.
997	401	Authentication was not successful.

Activity

A valid service definition looks as following:

```
{
    "version": 2,
    "services": {
        "ACTIVITY": {
            "version": 1,
            "endpoints": {
                "list": "/ocs/v2.php/cloud/activity"
            }
        }
    }
}
```

The “ACTIVITY” module allows consumers to show a list of actions happening on the cloud service. This can for example be a collection of recently created or deleted files. This module has only a `list` endpoint used to gather this list.

HTTP REQUEST

Following request would request the data of the user `admin`:

```
GET /ocs/v2.php/cloud/activity HTTP/1.1
```

```
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
POST http://example.com/{list}
```

QUERY PARAMETERS

Parameter	Default	Description
start	0	Optional value with which event to start.
count	30	Optional value on how many values should get returned.

RESPONSE

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data>
  <element>
   <!-- Unique identifier for the event -->
   <id>2</id>
   <!-- Subject of the mes-->
   <subject>You deleted ownCloudUserManual.pdf</subject>
   <!-- More descriptive text of the action (optional) -->
   <message></message>
   <!-- The affected file -->
   <file>/ownCloudUserManual.pdf</file>
   <!-- Link to the file or folder -->
   <link>http://example.org/index.php/apps/files?dir=%2F</link>
   <!-- Timestamp when the event happened -->
   <date>2015-06-10T09:42:58+00:00</date>
  </element>
  <element>
   <id>1</id>
   <subject>You created test.txt</subject>
   <message></message>
   <file>/test.txt</file>
   <link>http://example.org/index.php/apps/files?dir=%2F</link>
   <date>2015-06-10T09:22:38+00:00</date>
  </element>
 </data>
</ocs>
```

The response MUST be a OCS success message or a 993 forbidden statuscode if the user is not authenticated. In case of a success the response must contain the defined data blob.

In case a start or count parameter is specified endpoints MUST return the newest entries first to allow consumers to chunk the event transmission.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Provisoning [Experimental]

A valid service definition looks as following:

```
{
    "version": 2,
    "services": {
        "PROVISIONING": {
            "version": 1,
            "endpoints": {
                "user": "/ocs/v2.php/cloud/users",
                "groups": "/ocs/v2.php/cloud/groups",
                "apps": "/ocs/v2.php/cloud/apps"
            }
        }
    }
}
```

The “PROVISONING” module allows management of users and groups on a service provider. Specifically it supports the following functionalities:

1. Managing users (`user`)
    - Create users
    - Edit users
    - Delete users
    - List users
2. Managing groups (`groups`)
    - Create groups
    - Manage subadmin privileges
    - Manage group members
    - Delete groups
    - List groups
3. Managing applications (`apps`)
    - Enable applications
    - Delete applications
    - List applications

This module is likely to require administrative privileges for accessing and service providers SHOULD review their implementation carefully.

Create user

Creates a new user on the server.

HTTP REQUEST

Creates an user “newUser” with the password “newPassword”:

```
POST /ocs/v2.php/cloud/users HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 35
Content-Type: application/x-www-form-urlencoded

userid=newUser&password=newPassword
```

```
POST http://example.org/{user}
```

QUERY PARAMETERS

Parameter	Default	Description
userid		Username to create.
password		Password of the user.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message.

OCS status code	HTTP status code	Meaning
400	400	User already exists or invalid input data.
997	401	Authentication was not successful.

Get list of users

Retrieves a list of users on the server.

HTTP REQUEST

Request a list of users:

```
GET /ocs/v2.php/cloud/users HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
```

```
OCS-REQUEST: true
```

```
GET http://example.org/{user}
```

QUERY PARAMETERS

Parameter	Default	Description
search		Username to search for.
limit		How many users to list.
offset		Optional offset.

RESPONSE

In case of success, a OCS success message with the mbedded user names is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data>
  <users>
   <element>admin</element>
   <element>JonDoe</element>
   <element>JaneMeyer</element>
  </users>
 </data>
</ocs>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message containing the user names as data element.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Get a user

Retrieves information about a single user.

HTTP REQUEST

To request information about an user called “admin” the following request would be valid:

```
GET /ocs/v2.php/cloud/users/admin HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
```

```
OCS-REQUEST: true
```

```
GET http://example.com/{user}/{userId}
```

URL PARAMETERS

Parameter	Default
userId	User name of the user to lookup.

RESPONSE

In case the action was successfully a successful OCS response object MUST be returned by the server. The data element MUST contain a blob as following, values MAY be empty or more values MAY get returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data>
  <!-- Email address of the user (string) -->
  <email>user@example.org</email>
  <!-- Whether the user is enabled (boolean) -->
  <enabled>true</enabled>
  <quota>
   <!-- Free storage in byte (int) -->
   <free>80</free>
   <!-- Used storage in byte (int) -->
   <used>20</used>
   <!-- Total storage, free + used in byte (int) -->
   <total>200</total>
   <!-- Percentage of used storage (int) -->
   <relative>20</relative>
  </quota>
  <!-- Human displayable username, may be chosen by the user (string) -->
  <displayname>Hans User</displayname>
 </data>
</ocs>
```

In case the action was successfully a successful OCS response object MUST be returned by the server.

OCS status code	HTTP status code	Meaning
404	404	User not found.
997	401	Authentication was not successful.

Edit user attributes

Edits attributes related to a user. Users are able to edit email, displayname and password; admins can also edit the quota value.

HTTP REQUEST

Following request would change the password of the user “admin” to “NewUserPassword”;

```
PUT /ocs/v2.php/cloud/users/admin HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 34
Content-Type: application/x-www-form-urlencoded

key=password&value=NewUserPassword
```

```
PUT http://example.com/{user}/{uid}
```

URL PARAMETERS

Parameter	Default
userId	User name of the user to edit.

QUERY PARAMETERS

Parameter	Default	Description
key		Field to edit. One of “email”, “quota”, “displayname” or “password”.
value		New value for the field.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful or user not found.

Delete user

Deletes a user from the instance.

HTTP REQUEST

Following request would delete the user named “test”:

```
DELETE /ocs/v2.php/cloud/users/test HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
DELETE http://example.com/{user}/{userId}
```

URL PARAMETERS

Parameter	Default
userId	User ID of the user to delete.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

In case the action was successfully a successful OCS response object MUST be returned by the server.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Get group memberships

Retrieves a list of groups the specified user is member of.

HTTP REQUEST

Following request would get the membership information of the user named “test”:

```
GET /ocs/v2.php/cloud/users/test/groups HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
GET http://example.com/{user}/{userId}/groups
```

URL PARAMETERS

Parameter	Default
userId	User ID of the user to delete.

RESPONSE

In case of success, a OCS success message with the list of group memberships is returned:

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>200</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <groups>
      <!-- List of group memberships -->
      <element>admin</element>
      <element>group1</element>
    </groups>
  </data>
</ocs>
```

If the action could be performed a successful OCS response object MUST be returned by the server. The data element must contain an array of the user groups.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Add user to group

Adds the specified user to the specified group.

HTTP REQUEST

Following request adds the user “test” to the group “admin”:

```
POST /ocs/v2.php/cloud/users/test/groups HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 13
Content-Type: application/x-www-form-urlencoded

groupid=admin
```

```
POST http://example.org/{user}/{userId}/groups
```

REQUEST PARAMETERS

Parameter	Default	Description
userId		User to add to a group.

QUERY PARAMETERS

Parameter	Default	Description
groupid		ID of the group that the user should get added to.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message.

OCS status code	HTTP status code	Meaning
400	400	Failed to add user to group.
997	401	Authentication was not successful.

Remove user from group

Removes the specified user to the specified group.

HTTP REQUEST

Following request removes the user “test” from the group “admin”:

```
DELETE /ocs/v2.php/cloud/users/test/groups HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 13
Content-Type: application/x-www-form-urlencoded

groupid=admin
```

```
DELETE http://example.org/{user}/{userId}/groups
```

REQUEST PARAMETERS

Parameter	Default	Description
userId		User to remove from a group.

QUERY PARAMETERS

Parameter	Default	Description
groupid		ID of the group that the user should get removed from.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

In case the action was successfully a successful OCS response object MUST be returned by the server. According to general requirements of DELETE being an idempotent operation success is even reported in case the user does not exist. The status code in case of a successful operation is 204/No Content.

OCS status code	HTTP status code	Meaning
400	400	Failed to remove user from group.
997	401	Authentication was not successful.

Promote user to subadmin

Makes a user the subadmin of a group.

HTTP REQUEST

Following request would make the user “test” a subadmin of the group “admin”:

```
POST /ocs/v1.php/cloud/users/test/subadmins HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 13
Content-Type: application/x-www-form-urlencoded

groupid=admin
```

```
POST http://example.com/{user}/{userId}/subadmins
```

URL PARAMETERS

Parameter	Default
userId	User ID of the user to promote.

QUERY PARAMETERS

Parameter	Default
groupid	Group that the user should become subadmin of.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

If the action could be performed a successful OCS response object MUST be returned by the server.

OCS status code	HTTP status code	Meaning
101	200	User or group not found.
997	401	Authentication was not successful.

Remove subadmin privileges

Removes the subadmin rights for the user specified from the group specified.

HTTP REQUEST

Following request would remove the subadmin privileges of user “test” from “admin”:

```
DELETE /ocs/v1.php/cloud/users/test/subadmins HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 13
Content-Type: application/x-www-form-urlencoded

groupid=admin
```

```
DELETE http://example.com/{user}/{userId}/subadmins
```

URL PARAMETERS

Parameter	Default
userId	User ID of the user to demote.

QUERY PARAMETERS

Parameter	Default
groupid	Group that the user should become a regular member of of.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

If the action could be performed a successful OCS response object MUST be returned by the server. According to general requirements of DELETE being an idempotent operation success is even reported in case the user does not exist. The status code in case of a successful operation is 204/No Content.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Get subadmin privileges of user

Returns the groups in which the user is a subadmin.

HTTP REQUEST

Following request would list all groups that the user “test” is a subadmin of.

```
GET /ocs/v1.php/cloud/users/test/subadmins HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
GET http://example.com/{user}/{userId}/subadmins
```

URL PARAMETERS

Parameter	Default
userId	User ID of the user to get membership from.

RESPONSE

In case of success, a OCS success message with a data element containing a list of groups:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data>
  <element>AnotherGroup</element>
  <element>MyTestGroup</element>
 </data>
</ocs>
```

In case the action was successfully a successful OCS response object MUST be returned by the server. The `data` element MUST contain a list of groups.

OCS status code	HTTP status code	Meaning
101	200	User does not exist.
997	401	Authentication was not successful.

Get groups on the server

Retrieves a list of groups from the cloud server.

HTTP REQUEST

Following request would get a list of all groups on the server:

```
GET /ocs/v1.php/cloud/groups HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
GET http://example.com/{groups}
```

Parameter	Default	Description
search		Filter to search for.
limit		Optional limit.

offset	Optional offset.

RESPONSE

In case of success, a OCS success message with a data element containing a list of groups:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data>
  <element>AnotherGroup</element>
  <element>MyTestGroup</element>
 </data>
</ocs>
```

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Create a new group

Creates a new group.

HTTP REQUEST

Following request would create a group with the name “NewGroup”:

```
POST /ocs/v1.php/cloud/groups HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 16
Content-Type: application/x-www-form-urlencoded

groupid=NewGroup
```

```
POST http://example.com/{groups}
```

QUERY PARAMETERS

Parameter	Default	Description
groupid		Name of the group to create.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

If the action could be performed a successful OCS response object MUST be returned by the server.

OCS status code	HTTP status code	Meaning
102	200	Group already exists.
997	401	Authentication was not successful.

Get members of a group

Retrieves a list of group members.

HTTP REQUEST

Following request would get the membership information of the user named “test”:

```
GET /ocs/v1.php/cloud/groups/admin HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
GET http://example.com/{groups}/{groupId}
```

URL PARAMETERS

Parameter	Default
groupId	The ID of the group to request membership from.

RESPONSE

In case of success, a OCS success message with the list of group memberships is returned:

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>200</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <groups>
      <!-- List of members in the group -->
```

```
      <element>JohnDoe</element>
      <element>JaneMeyer</element>
    </groups>
  </data>
</ocs>
```

If the action could be performed a successful OCS response object MUST be returned by the server. The data element must contain an array of the user groups.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Get subadmins of a group

Returns subadmins of the group.

HTTP REQUEST

Following request would get the membership information of the user named “test”:

```
GET /ocs/v1.php/cloud/groups/admin/subadmins HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
GET http://example.com/{groups}/{groupId}/subadmins
```

URL PARAMETERS

Parameter	Default
groupId	The ID of the group to request a list of subadmins from.

RESPONSE

In case of success, a OCS success message with the list of subadmins is returned:

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>200</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <groups>
      <!-- List of members in the group -->
      <element>JohnDoe</element>
      <element>JaneMeyer</element>
    </groups>
  </data>
</ocs>
```

If the action could be performed a successful OCS response object MUST be returned by the server. The data element must contain an array of the subadmins.

OCS status code	HTTP status code	Meaning
404	404	Group does not exist.
997	401	Authentication was not successful.

Delete a group

Removes a group.

HTTP REQUEST

Following request would delete the group “test”:

```
DELETE /ocs/v1.php/cloud/groups/test HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
DELETE http://example.com/{groups}/{groupId}
```

URL PARAMETERS

Parameter	Default
groupId	ID of the group to delete.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

If the action could be performed a successful OCS response object MUST be returned by the server. According to general requirements of DELETE being an idempotent operation success is even reported in case the user does not exist. The status code in case of a successful operation is 204/No Content.

OCS status code	HTTP status code	Meaning
101	200	Group does not exist.

102	200	Failed to delete group.
997	401	Authentication was not successful.

Get list of installed apps

Returns a list of apps installed on the cloud server.

HTTP REQUEST

Following request would get a list of all installed apps.

```
GET /ocs/v1.php/cloud/apps HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
GET http://example.com/{apps}
```

RESPONSE

In case of success, a OCS success message with the list of installed apps is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data>
  <apps>
   <!-- List of apps on the server  -->
   <element>activity</element>
   <element>audit_monitor</element>
  </apps>
 </data>
</ocs>
```

If the action could be performed a successful OCS response object MUST be returned by the server. The data element must contain an array of the installed apps.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Get application information

Get information about the specified application.

HTTP REQUEST

Following request would get information about the app “files”:

```
GET /ocs/v2.php/cloud/apps/files HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
GET example.org/{apps}/{appId}
```

URL PARAMETERS

Parameter	Default
appId	ID of the app to get more information from.

RESPONSE

In case of success, a OCS success message with more information about the app:

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>200</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <!-- Id of the application -->
    <id>files</id>
    <!-- Name of the application -->
    <name>Files</name>
    <!-- Description of an application -->
    <description>File Management</description>
  </data>
</ocs>
```

In case the action was successfully a successful OCS response object MUST be returned by the server. The data element MUST contain a list of information about the app.

OCS status code	HTTP status code	Meaning
404	404	Application not found.
997	401	Authentication was not successful.

Enable application

Enable an app.

HTTP REQUEST

Enables the application “activity”:

```
POST /ocs/v2.php/cloud/apps/activity HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
POST example.org/{apps}/{appId}
```

REQUEST PARAMETERS

Parameter	Default	Description
appId		ID of the application to enable.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Disable application

Disables the specified app.

HTTP REQUEST

Disables the application “activity”:

```
DELETE /ocs/v2.php/cloud/apps/activity HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
DELETE example.org/{apps}/{appId}
```

REQUEST PARAMETERS

Parameter	Default	Description
appId		ID of the application to disable.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Private Data [Experimental]

A valid service definition looks as following:

```
{
    "version": 2,
    "services": {
        "PRIVATE_DATA": {
            "version": 1,
            "endpoints": {
                "store": "/ocs/v2.php/privatedata/setattribute",
                "read": "/ocs/v2.php/privatedata/getattribute",
                "delete": "/ocs/v2.php/privatedata/deleteattribute"
            }
        }
    }
}
```

The `PRIVATE_DATA` module allows consumers to store data in a key-value storage. The store supports multiple sub-stores. This module handles the following tasks:

1. Storing values in the store (“store”)
2. Reading data from the store (“read”)
3. Deleting data from the store (“delete”)

In case of a server error or connection problem consumers MUST handle this gracefully.

Set value

Sets a value in the key-value store for the currently logged-in user.

HTTP REQUEST

Stores the value “myValue” in the key “myKey” in the “myStore” store:

```
POST /ocs/v2.php/privatedata/setattribute/myStore/myKey HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
Content-Length: 13
Content-Type: application/x-www-form-urlencoded

value=myValue
```

```
POST example.org/{store}/{storeName}/{key}
```

URL PARAMETERS

Parameter	Default	Description
storeName		Store under which the value should get stored.
key		Key of the value.

REQUEST PARAMETERS

Parameter	Default	Description
value		Value of the key.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Get values of key

Get all value of a key stored within a specific store for the currently logged-in user.

HTTP REQUEST

Get the value of the the key “myKey” in the “myStore” store:

```
GET /ocs/v2.php/privatedata/getattribute/myStore/myKey HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
GET example.org/{read}/{storeName}/{key}
```

URL PARAMETERS

Parameter	Default	Description
storeName		Store under which the value should get looked-up.
key		Key of the value to look-up.

RESPONSE

In case there is an entry a successful OCS response object MUST be returned by the server, the values MUST be embedded in the data element and at least consist of the following elements:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data>
  <element>
   <!-- Key of the stored value -->
   <key>MyKey</key>
   <!-- Store in which the data is stored, internally represented as "app" -->
   <app>MyStore</app>
   <!-- Value to store -->
   <value>ValueToStore</value>
  </element>
 </data>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message containing the requested key-value entry.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.

Delete entry

Get the value of a key stored within a specific store for the currently logged-in user.

HTTP REQUEST

Deletes the key “myKey” in the “myStore” store:

```
POST /ocs/v2.php/privatedata/deleteattribute/myStore/myKey HTTP/1.1
Host: localhost
Authorization: Basic YWRtaW46YWRtaW4=
OCS-REQUEST: true
```

```
POST example.org/{delete}/{storeName}/{key}
```

URL PARAMETERS

Parameter	Default	Description
storeName		Store under which the value should get deleted.
key		Key to delete.

RESPONSE

In case of success, a OCS success message without further details is returned:

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

The expected response MUST, in case of success, be a OCS success message.

OCS status code	HTTP status code	Meaning
997	401	Authentication was not successful.