Syncto Documentation

Release 2.0.0

Mozilla Services — Da French Team

October 06, 2015

Contents

Syncto is a server allowing you to store and retrieve Firefox Sync user data attached to your Firefox account using the Kinto API in order to be able to use Kinto.js for that task.

- Online documentation
- Issue tracker

CHAPTER 1

Table of content

1.1 Installation

1.1.1 Run locally

Sycnto is based on top of the cliquet project, and as such, please refer to cliquet's documentation for more details.

For development

By default, *Sycnto* persists internal cache in Redis.

```
git clone https://github.com/mozilla-services/syncto
cd syncto
make serve
```

note OSX users are warned that supplementary steps are needed to ensure proper installation of cryptographic dependencies is properly done; see *dedicated note*.

If you already installed Syncto earlier and you want to recreate a full environment (because of errors when running `make serve`), please run:

```
make maintainer-clean serve
```

Authentication

By default, *Sycnto* relies on Firefox Account OAuth2 Bearer tokens to authenticate users.

See cliquet documentation to configure authentication options.

Note that you will also need to pass through a BrowserID assertion in order for Syncto to read the Firefox Sync server.

1.1.2 Install and setup PostgreSQL

(*requires PostgreSQL 9.3 or higher*).

Using Docker

```
docker run -e POSTGRES_PASSWORD=postgres -p 5434:5432 postgres
```

Linux

On debian / ubuntu based systems:

```
apt-get install postgresql postgresql-contrib
```

By default, the `postgres` user has no password and can hence only connect if ran by the `postgres` system user. The following command will assign it:

```
sudo -u postgres psql -c "ALTER USER postgres PASSWORD 'postgres';"
```

OS X

Assuming brew is installed:

```
brew update
brew install postgresql
```

Create the initial database:

```
initdb /usr/local/var/postgres
```

1.1.3 Cryptography libraries

Linux

On Debian / Ubuntu based systems:

```
apt-get install libffi-dev libssl-dev
```

On RHEL-derivatives:

```
apt-get install libffi-devel openssl-devel
```

OS X

Assuming brew is installed:

```
brew install libffi openssl pkg-config
```

Warning: Apple having dropped support for OpenSSL and moving to their own library recently, you have to force its usage to properly install cryptography-related dependencies:

```
$ env LDFLAGS="-L$(brew --prefix openssl)/lib" \
      CFLAGS="-I$(brew --prefix openssl)/include" \
          .venv/bin/pip install cryptography
$ make serve
```

1.1.4 Running in production

Recommended settings

Most default setting values in the application code base are suitable for production.

However, the set of settings mentionned below might deserve some review or adjustments:

```
cliquet.http_scheme = https
cliquet.paginate_by = 100
cliquet.batch_max_requests = 25
cliquet.delete_collection_enabled = false
cliquet.storage_pool_maxconn = 50
cliquet.cache_pool_maxconn = 50
fxa-oauth.cache_ttl_seconds = 3600
```

note For an exhaustive list of available settings and their default values, refer to cliquet source code.

Enable write access

By default, collections are read-only. In order to enable write operations on remote Sync collections, add some settings in the configuration with the collection name:

```
syncto.record_tabs_put_enabled = true
syncto.record_tabs_delete_enabled = true
syncto.record_passwords_put_enabled = true
syncto.record_passwords_delete_enabled = true
syncto.record_bookmarks_put_enabled = true
syncto.record_bookmarks_delete_enabled = true
syncto.record_history_put_enabled = true
syncto.record_history_delete_enabled = true
```

Monitoring

```
# Heka
cliquet.logging_renderer = cliquet.logs.MozillaHekaRenderer

# StatsD
cliquet.statsd_url = udp://carbon.server:8125
```

Application output should go to `stdout`, and message format should have no prefix string:

```
[handler_console]
class = StreamHandler
args = (sys.stdout,)
level = INFO
formater = heka

[formatter_heka]
format = %(message)s
```

Adapt the logging configuration in order to plug Sentry:

```
[loggers]
keys = root, sentry

[handlers]
```

```
keys = console, sentry

[formatters]
keys = generic

[logger_root]
level = INFO
handlers = console, sentry

[logger_sentry]
level = WARN
handlers = console
qualname = sentry.errors
propagate = 0

[handler_console]
class = StreamHandler
args = (sys.stdout,)
level = INFO
formater = heka

[formatter_heka]
format = %(message)s

[handler_sentry]
class = raven.handlers.logging.SentryHandler
args = ('http://public:secret@example.com/1',)
level = WARNING
formatter = generic

[formatter_generic]
format = %(asctime)s,%(msecs)03d %(levelname)-5.5s [%(name)s] %(message)s
datefmt = %H:%M:%S
```

PostgreSQL setup

In production, it is wise to run the application with a dedicated database and user.

```
postgres=# CREATE USER produser;
postgres=# CREATE DATABASE proddb OWNER produser;
CREATE DATABASE
```

The tables needs to be created with the *cliquet* tool.

```
$ cliquet --ini config/syncto.ini migrate
```

note Alternatively the SQL initialization files can be found in the *cliquet* source code (`cliquet/cache/postgresql/schema1.sql` and `cliquet/storage/postgresql/schema1.sql`).

Running with uWsgi

To run the application using uWsgi, an **app.wsgi** file is provided. This command can be used to run it:

```
uwsgi --ini config/syncto.ini
```

uWsgi configuration can be tweaked in the ini file in the dedicated **[uwsgi]** section.

Here's an example:

```
[uwsgi]
wsgi-file = app.wsgi
enable-threads = true
http-socket = 127.0.0.1:8000
processes =  3
master = true
module = syncto
harakiri = 30
uid = syncto
gid = syncto
virtualenv = .
lazy = true
lazy-apps = true
```

To use a different ini file, the `SYNCTO_INI` environment variable should be present with a path to it.

1.2 API Endpoints

1.2.1 Authentication

Firefox Account OAuth Bearer token

Use the OAuth token with this header:

```
Authorization: Bearer <oauth_token>
```

Obtain the token

The `GET /v0/fxa-oauth/params` endpoint can be use to get the configuration in order to trade the Firefox Account BrowserID with a Bearer Token. See Firefox Account documentation about this behavior

```
$ http GET http://localhost:8000/v0/fxa-oauth/params -v

GET /v0/fxa-oauth/params HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Host: localhost:8000
User-Agent: HTTPie/0.8.0


HTTP/1.1 200 OK
Content-Length: 103
Content-Type: application/json; charset=UTF-8
Date: Thu, 19 Feb 2015 09:28:37 GMT
Server: waitress

{
    "client_id": "89513028159972bc",
    "oauth_uri": "https://oauth-stable.dev.lcip.org",
    "scope": "sync"
}
```

BrowserID assertion

Note that you should also pass the BrowserID assertion to be used with the Firefox Sync Server in the `Backend-Authorization` header.

1.2.2 Resource endpoints

In this section, the request example provided are performed using httpie .

GET /history

Requires authentication

Returns all records of the current user for this resource.

The returned value is a JSON mapping containing:

- `items`: the list of records, with exhaustive attributes

A `Total-Records` header is sent back to indicate the estimated total number of records included in the response.

A header `ETag` will provide the current timestamp of the collection (*see Server timestamps section*). It is likely to be used by client to provide `If-Modified-Since` or `If-Unmodified-Since` headers in subsequent requests.

1.2.3 Batch operations

POST /batch

Requires an FxA OAuth authentication

The POST body is a mapping, with the following attributes:

- `requests`: the list of requests (*limited to 25 by default*)
- `defaults`: (*optional*) default requests values in common for all requests

  Each request is a JSON mapping, with the following attribute:

- `method`: HTTP verb
- `path`: URI
- `body`: a mapping
- `headers`: (*optional*), otherwise take those of batch request

```
{
  "defaults": {
    "method" : "POST",
    "path" : "/articles",
    "headers" : {
      ...
    }
  },
  "requests": [
    {
      "body" : {
        "title": "MoFo",
        "url" : "http://mozilla.org",
```

```
        "added_by": "FxOS",
      }
    },
    {
      "body" : {
        "title": "MoCo",
        "url" : "http://mozilla.com"
        "added_by": "FxOS",
      }
    },
    {
      "method" : "PATCH",
      "path" : "/history/409",
      "body" : {
        "read_position" : 3477
      }
    }
  ]
]
```

The response body is a list of all responses:

```
{
  "responses": [
    {
      "path" : "/history/409",
      "status": 200,
      "body" : {
        "id": 409,
        "url": "...",
        ...
        "read_position" : 3477
      },
      "headers": {
        ...
      }
    },
    {
      "status": 201,
      "path" : "/history",
      "body" : {
        "id": 411,
        "title": "MoFo",
        "url" : "http://mozilla.org",
        ...
      },
    },
    {
      "status": 201,
      "path" : "/history",
      "body" : {
        "id": 412,
        "title": "MoCo",
        "url" : "http://mozilla.com",
        ...
      },
    },
  ]
]
```

warning Since the requests bodies are necessarily mappings, posting arbitrary data (*like raw text or binary*)is not supported.

note Responses are provided in the same order than requests.

note A form of payload optimization for massive operations is planned.

1.2.4 Utility endpoints for OPS and Devs

GET /

The returned value is a JSON mapping containing:

- `hello`: the name of the service (e.g. `"reading list"`)
- `version`: complete version (`"X.Y.Z"`)
- `commit`: the HEAD git revision number when run from a git repository.
- `url`: absolute URI (without a trailing slash) of the API (*can be used by client to build URIs*)
- `eos`: date of end of support in ISO 8601 format (`"yyyy-mm-dd"`, undefined if unknown)
- `documentation`: The url to the service documentation. (this document!)
- **`settings`: a mapping with the values of relevant public settings for clients**
  - `cliquet.batch_max_requests`: Number of requests that can be made in a batch request.
- `userid`: The connected perso user id. The field is not present when no Authorization header is provided.

GET /__heartbeat__

Return the status of each service your application depends on. The returned value is a JSON mapping containing:

- `cache` true if operational
- `sync` true if operational

Return `200` if the connection with each service is working properly and `503` if something doesn't work.

1.2.5 Server timestamps

In order to avoid race conditions, all timestamps manipulated by the server are not true HTTP date values, nor milliseconds EPOCH timestamps.

They are milliseconds EPOCH timestamps with the guarantee of a change per timestamp update. If two changes happen at the same millisecond, they will have two differents timestamps.

The `ETag` header with the last timestamp of the collection for a given user will be given on collection and record GET endpoints.

```
ETag: "1422375916186"
```

All timestamp of the app will be set in milliseconds.

1.2.6 API versioning

Versionning

The API versioning is based on the application version deployed. It follows semver.

During development the server will be 0.X.X, the server endpoint will be prefixed by `/v0`.

Each non retro-compatible API change will imply the major version number to be incremented.

Everything will be made to avoid retro incompatible changes.

The `/` endpoint will redirect to the last API version.

Deprecation

A track of the client version will be kept to know after which date each old version can be shutdown.

The date of the end of support is provided in the API root URL (e.g. `/v0`)

Using the `Alert` response header, the server can communicate any potential warning messages, information, or other alerts.

The value is JSON mapping with the following attributes:

- `code`: one of the strings `"soft-eol"` or `"hard-eol"`;
- `message`: a human-readable message (optional);
- `url`: a URL at which more information is available (optional).

A `410 Gone` error response can be returned if the client version is too old, or the service had been remplaced with a new and better service using a new protocol version.

1.2.7 Backoff indicators

Backoff header on heavy load

A `Backoff` header will be added to the success responses (>=200 and <400) when the server is under heavy load. It provides the client with a number of seconds during which it should avoid doing unnecessary requests.

```
Backoff: 30
```

note The back-off time is configurable on the server.

note In other implementations at Mozilla, there was `X-Weave-Backoff` and `X-Backoff` but the `X-` prefix for header has been deprecated since.

Retry-After indicators

A `Retry-After` header will be added to error responses (>=500), telling the client how many seconds it should wait before trying again.

```
Retry-After: 30
```

1.2.8 Error responses

Protocol description

Every response is JSON.

If the HTTP status is not OK (<200 or >=400), the response contains a JSON mapping, with the following attributes:

- `code`: matches the HTTP status code (e.g `400`)
- `errno`: stable application-level error number (e.g. `109`)
- `error`: string description of error type (e.g. `"Bad request"`)
- `message`: context information (e.g. `"Invalid request parameters"`)
- `info`: additional details (e.g. URL to error details)

Example response

```
{
    "code": 400,
    "errno": 109,
    "error": "Bad Request",
    "message": "Invalid posted data",
    "info": "https://server/docs/api.html#errors"
}
```

Error codes

status code	errno	description
401	104	Missing Authorization Token
401	105	Invalid Authorization Token
400	106	request body was not valid JSON
400	107	invalid request parameter
400	108	missing request parameter
400	109	invalid posted data
404	110	Invalid Token / id
404	111	Missing Token / id
411	112	Content-Length header was not provided
413	113	Request body too large
412	114	Resource was modified meanwhile
405	115	Method not allowed on this end point
429	117	Client has sent too many requests
403	121	Resource's access forbidden for this user
409	122	Another resource violates constraint
500	999	Internal Server Error
503	201	Service Temporary unavailable due to high load
410	202	Service deprecated

Validation errors

In case multiple validation errors occur on a request, they will be returned one at a time.

1.3 Syncto specific headers

Syncto also expose Firefox Sync information such as:

- `Quota-Remaining`: Remaining KB for the user collection if Quota are enabled on the user sync server.

1.4 Contributing

Thank you for considering to contribute to *Syncto*!

note No contribution is too small; please submit as many fixes for typos and grammar bloopers as you can!

note Open a pull-request even if your contribution is not ready yet! It can be discussed and improved collaboratively!

1.4.1 Run tests

```
make tests
```

1.4.2 Run load tests

From the `loadtests` folder:

```
make test SERVER_URL=http://localhost:8000
```

1.4.3 IRC channel

Join `#fxos-sync` on `irc.mozilla.org`!

1.5 CHANGELOG

This document describes changes between each past release.

1.5.1 1.1.0 (unreleased)

- Nothing changed yet.

1.5.2 1.0.0 (2015-10-06)

- First implementation of Syncto server.
- Connection with Token server and Sync servers.
- Encrypted credentials caching (#30, #31)
- Collections are Read-only by default
- Write permission on collection can be configured.

- Statsd monitoring for backends calls.
- Convert Syncto requests headers to Firefox Sync ones.
- Convert Firefox Sync headers to Syncto ones.