kinto webpush plugin Documentation

Release 0.1.0

Mansimar Kaur

March 01, 2017

Contents:

CHAPTER 1

Overview

What is WebPush?

WebPush is the web standard to allow a server to notify its client that something changed.

In your webapp you ask the browser for a subscription, you start a service-worker that will handle notifications and you send your subscription to the server so that it can send you notifications.

For more about WebPush, do not hesitate to read this article

What is this WebPush Channel Broadcasting service about?

tl;dr Broadcasting notifications using WebPush made simple.

Handling sending notifications to a large number of users for a web service might be a problem.

- It can slow down the request response time by a lot, or you need to spawn tasks workers (i.e celery, rq))
- You need to store and retrieve the subscription and have endpoint for your clients to manage them.
- You need to handle errors, retry, etc.

The WebPush Channel broadcasting service follow the micro-services philosophy and handle all these tasks for you:

- It will allow you to notify all your users with one simple HTTP POST request that will be accepted in milliseconds.
- It will automatically handle the load of sending thousands of notifications, encrypt them specifically for each subscription.
- It will allow your users to manage their subscriptions and the list of channels they are listening to.
- It will handle errors and retry.

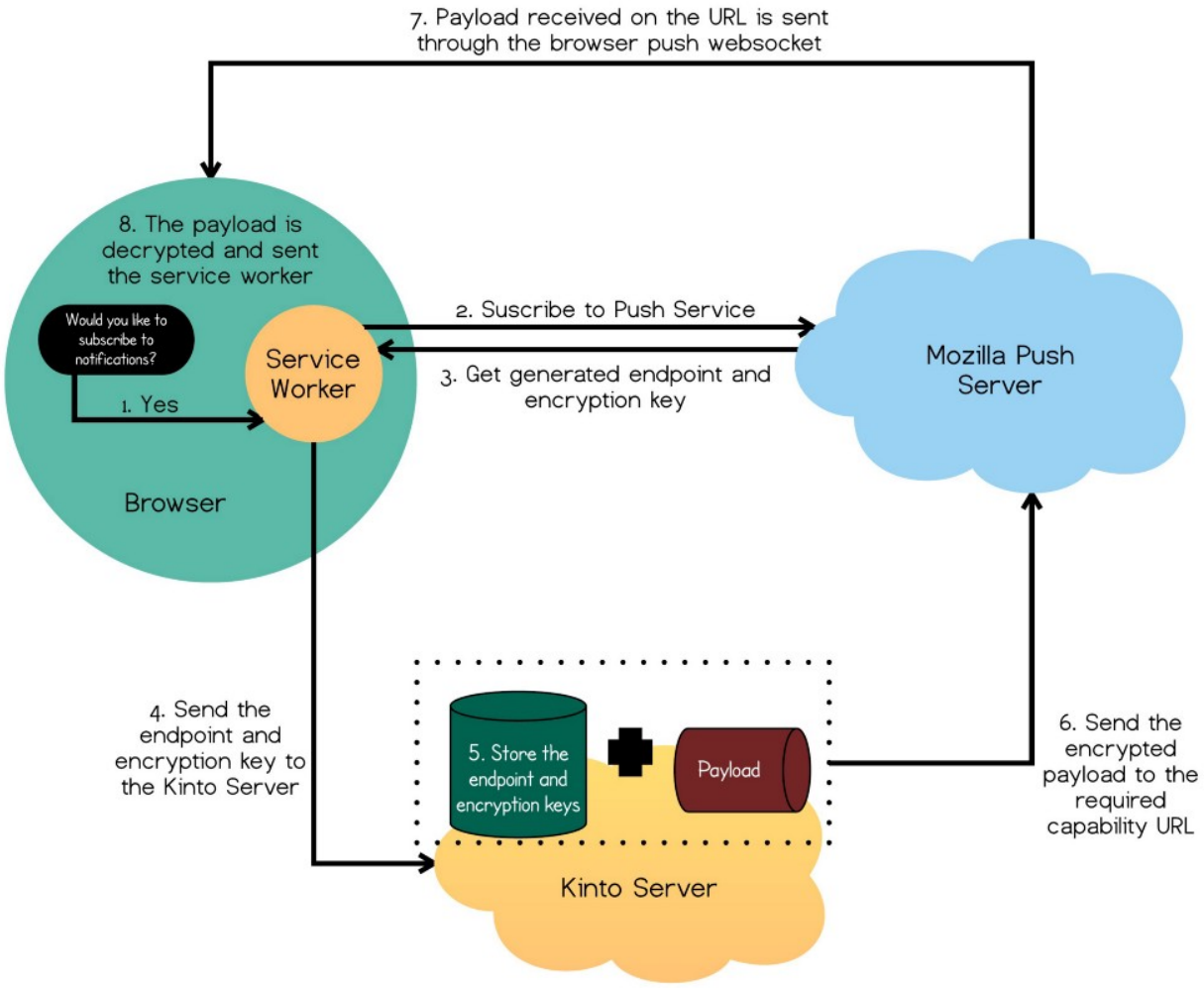

Fig. 1.1: Example of how we could use WebPush with Kinto.

Full reference

Full detailed API documentation:

Authentication

A word about users

First of all, WebPush Channels **doesn't provide users management**.

There is no such thing as user sign-up, password modification, etc.

However, users are uniquely identified.

How is that possible?

WebPush Channels uses the request headers to authenticate the current user.

Depending on the authentication methods enabled in configuration, the HTTP method to authenticate requests may differ.

WebPush Channels can rely on a third-party called «Identity provider» to authenticate the request and assign a user id.

There are many identity providers solutions in the wild. The most common are OAuth, JWT, SAML, x509, Hawk sessions...

A policy based on *OAuth2 bearer tokens* is recommended, but not mandatory.

Multiple policies

It is possible to enable several authentication methods.

In the current implementation, when multiple policies are configured, the first one in the list that succeeds is picked.

User identifiers are prefixed with the policy name being used.

OAuth Bearer token

If the configured authentication policy uses *OAuth2 bearer tokens*, authentication shall be done using this header:

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

The policy will verify the provided *OAuth2 bearer token* on a remote server.

notes If the token is not valid, this will result in a `401 Unauthorized` error response.

Portier

In order to enable authentication with Portier, install and configure Kinto/kinto-portier.

Firefox Accounts

In order to enable authentication with Firefox Accounts, install and configure mozilla-services/kinto-fxa.

Channels

Basically the idea being channels is the same as TV channels where one broadcast a message and others can listen to it.

The idea is that people can listen to channels even if they don't exists.

The service configuration as well as the permission backend define a list of users that can publish on given channels.

Register to a channel

PUT /channels/ (*channel_id*) **/registration**

synopsis Subscribe the user to the channel

Requires authentication

Example Request

```
$ http PUT http://localhost:9999/v0/channels/formbuilder-collections-update/registration Aut
```

```
PUT /v0/channels/formbuilder-collections-update/registration HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Portier dccd8ac07f3e45c9907da638e994ff98
Connection: keep-alive
Host: localhost:9999
User-Agent: HTTPie/0.9.2
```

Example Response

```
HTTP/1.1 202 Accepted
Access-Control-Expose-Headers: Backoff, Retry-After, Alert
Date: Thu, 18 Jun 2015 17:02:23 GMT
Server: waitress


{"code": 202, "message": "Accepted"}
```

Unsubscribing from a channel

DELETE /channels/ (*channel_id*) **/registration**

synopsis Unsubscribe the user from the channel

Requires authentication

Example Request

```
$ http delete http://localhost:9999/v0/channels/formbuilder-collections-write/registration A
```

```
DELETE /v0/channels/formbuilder-collections-update/registration HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Portier dccd8ac07f3e45c9907da638e994ff98
Connection: keep-alive
Host: localhost:9999
User-Agent: HTTPie/0.9.2
```

Example Response

```
HTTP/1.1 202 Accepted
Access-Control-Expose-Headers: Backoff, Retry-After, Alert
Date: Thu, 18 Jun 2015 17:02:23 GMT
Server: waitress


{"code": 202, "message": "Accepted"}
```

Getting channels informations

GET /channels/ (*channel_id*)

Synopsis Retrieve channel informations

Example Request

```
$ http get http://localhost:9999/v0/channels/formbuilder-collection-write Authorization:"Portier
```

```
GET /v0/channels/formbuilder-collection-write HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Basic Ym9iOg==
Connection: keep-alive
Host: localhost:9999
User-Agent: HTTPie/0.9.2
```

Example Response

```
HTTP/1.1 200 OK
Access-Control-Expose-Headers: Backoff, Retry-After, Alert, Last-Modified, ETag
Content-Length: 211
Content-Type: application/json; charset=UTF-8
Date: Thu, 18 Jun 2015 17:29:59 GMT
Etag: "1434648599199"
Last-Modified: Thu, 18 Jun 2015 17:29:59 GMT
Server: waitress

{
```

```
        "data": {
            "id": "formbuilder-collection-write",
            "registrations": 1,
            "push": 0
        }
    }
```

- **registration** contains the number of users that subscribed to the channel.
- **push** contains the number of push that were sent to the channel.

Broadcasting a push notification

For the first version, only users configured in the service configuration can broadcast notifications.

However in the future we aim at adding a permissions management feature to the channel.

POST /channels/(*channel_id*)

synopsis Push a notification

Requires authentication

Example Request

```
$ http post http://localhost:9999/v0/channels/formbuilder-collections-write Authorization:"B
```

```
POST /v0/channels/formbuilder-collections-update HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
Authorization: Basic Ym9iOg==
Connection: keep-alive
Content-Length: 25
Content-Type: application/json
Host: localhost:9999
User-Agent: HTTPie/0.9.2

{
  "data": {
      "last_modified": 1434647996969
  }
}
```

Example Response

```
HTTP/1.1 202 Accepted
Access-Control-Expose-Headers: Backoff, Retry-After, Alert
Date: Thu, 18 Jun 2015 17:02:23 GMT
Server: waitress


{"code": 202, "message": "Accepted"}
```

The `data` payload will be encrypted for each subscriptions and sent authenticated through the endpoint.

Subscriptions

Subscriptions belongs to an user.

A user can have multiple subscriptions (one per browser session or device).

A subscription consist on the information that you can get from a requesting a Push subscription on the browser.

```
{
    "data": {
        "endpoint": "https://updates.push.services.mozilla.com/wpush/v1/gAAAAABYZNChoTLTAeA9vv-_zeqGu
        "keys": {
            "auth": "pnipzxpMvKBNYZAcxc-MAA",
            "p256dh": "BEVoH6cOlNPuvYR0aVJo4GVv84nbymzpXxNff7hpKYjVIFcuIEtqiLtIe4rLOXF_A2w3KWRJoCYJE
        }
    }
}
```

Add a new user subscription

POST /subscriptions

synopsis Store a subscription. The ID will be assigned automatically.

Requires authentication

Example Request

```
$ echo '{"data": {"endpoint": "URL", "keys": {}}}' | http post http://localhost:9999/v0/subs
```

```
POST /v0/subscriptions HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
Authorization: Portier dccd8ac07f3e45c9907da638e994ff98
Connection: keep-alive
Content-Length: 25
Content-Type: application/json
Host: localhost:9999
User-Agent: HTTPie/0.9.2

{
    "data": {
        "endpoint": "https://updates.push.services.mozilla.com/wpush/v1/gAAAAABYZNChoTLTAeA9
        "keys": {
            "auth": "pnipzxpMvKBNYZAcxc-MAA",
            "p256dh": "BEVoH6cOlNPuvYR0aVJo4GVv84nbymzpXxNff7hpKYjVIFcuIEtqiLtIe4rLOXF_A2w3K
        }
    }
}
```

Example Response

```
HTTP/1.1 201 Created
Access-Control-Expose-Headers: Backoff, Retry-After, Alert
Content-Length: 199
Content-Type: application/json; charset=UTF-8
Date: Thu, 18 Jun 2015 17:02:23 GMT
Server: waitress

{
    "data": {
        "endpoint": "https://updates.push.services.mozilla.com/wpush/v1/gAAAAABYZNChoTLTAeA9vv-_
        "keys": {
```

```
                "auth": "pnipzxpMvKBNYZAcxc-MAA",
                "p256dh": "BEVoH6cOlNPuvYR0aVJo4GVv84nbymzpXxNff7hpKYjVIFcuIEtqiLtIe4rLOXF_A2w3KWRJc
            }
        }
    }
```

Validation

If the posted values are invalid (e.g. *field value is not an integer*) an error response is returned with `400 Bad Request`.

See *details on error responses*.

HTTP Status Codes

- `200 OK`: This object already exists, the one stored on the database is returned
- `201 Created`: The object was created
- `400 Bad Request`: The request body is invalid
- `401 Unauthorized`: The request is missing authentication headers
- `403 Forbidden`: The user is not allowed to perform the operation, or the resource is not accessible
- `406 Not Acceptable`: The client doesn't accept supported responses Content-Type
- `412 Precondition Failed`: List has changed since value in `If-Match` header
- `415 Unsupported Media Type`: The client request was not sent with a correct Content-Type

Retrieving user's subscriptions

`GET /subscriptions`

Synopsis Retrieve all the subscriptions for the user.

Requires authentication

Example Request

```
$ http get http://localhost:9999/v0/subscriptions Authorization:"Portier dccd8ac07f3e45c9907da63
```

```
GET /v0/subscriptions HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Portier dccd8ac07f3e45c9907da638e994ff98
Connection: keep-alive
Host: localhost:9999
User-Agent: HTTPie/0.9.2
```

```
HTTP/1.1 200 OK
Access-Control-Expose-Headers: Backoff, Retry-After, Alert, Next-Page, Total-Records, Last-Modif
Content-Length: 110
Content-Type: application/json; charset=UTF-8
Date: Thu, 18 Jun 2015 17:24:38 GMT
Etag: "1434648278603"
Last-Modified: Thu, 18 Jun 2015 17:24:38 GMT
```

```
    Server: waitress
    Total-Records: 1

    {
        "data": [
            {
                "endpoint": "https://updates.push.services.mozilla.com/wpush/v1/gAAAAABYZNChoTLTAeA9
                "keys": {
                    "auth": "pnipzxpMvKBNYZAcxc-MAA",
                    "p256dh": "BEVoH6cOlNPuvYR0aVJo4GVv84nbymzpXxNff7hpKYjVIFcuIEtqiLtIe4rLOXF_A2w3K
                },
                "id": "89881454-e4e9-4ef0-99a9-404d95900352",
                "last_modified": 1434647996969
            }
        ]
    }
```

Delete user's subscriptions

DELETE /subscriptions

Synopsis Delete all the user's subscriptions

Requires authentication

Example Request

```
    $ http delete http://localhost:9999/v0/subscriptions Authorization:"Portier dccd8ac07f3e45c9907d
```

```
    DELETE /v0/subscriptions HTTP/1.1
    Accept: */*
    Accept-Encoding: gzip, deflate
    Authorization: Portier dccd8ac07f3e45c9907da638e994ff98
    Connection: keep-alive
    Host: localhost:9999
    User-Agent: HTTPie/0.9.2
```

Example Response

```
    HTTP/1.1 200 OK
    Access-Control-Expose-Headers: Backoff, Retry-After, Alert, Last-Modified, ETag
    Content-Length: 211
    Content-Type: application/json; charset=UTF-8
    Date: Thu, 18 Jun 2015 17:29:59 GMT
    Etag: "1434648599199"
    Last-Modified: Thu, 18 Jun 2015 17:29:59 GMT
    Server: waitress

    {
        "data": [{
            "deleted": true,
            "id": "89881454-e4e9-4ef0-99a9-404d95900352",
            "last_modified": 1434648749173
        }]
    }
```

Deleting a single subscription

DELETE /subscriptions/ (*subscription_id*)

Synopsis Delete a subscription by its ID.

Example Request

```
$ http delete http://localhost:9999/v0/subscriptions/89881454-e4e9-4ef0-99a9-404d95900352  Authc
```

```
DELETE /v0/subscriptions/89881454-e4e9-4ef0-99a9-404d95900352 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Authorization: Portier dccd8ac07f3e45c9907da638e994ff98
Connection: keep-alive
Content-Length: 0
Host: localhost:9999
User-Agent: HTTPie/0.9.2
```

Example Response

```
HTTP/1.1 200 OK
Access-Control-Expose-Headers: Backoff, Retry-After, Alert
Content-Length: 99
Content-Type: application/json; charset=UTF-8
Date: Thu, 18 Jun 2015 17:32:29 GMT
Server: waitress

{
    "data": {
        "deleted": true,
        "id": "89881454-e4e9-4ef0-99a9-404d95900352",
        "last_modified": 1434648749173
    }
}
```

Utility endpoints for OPS and Devs

GET /

The returned value is a JSON mapping containing:

Changed in version 3.0.

- `project_name`: the name of the service (e.g. `"reading list"`)
- `project_docs`: The URL to the service documentation. (this document!)
- `project_version`: complete application/project version (`"3.14.116"`)
- `http_api_version`: the MAJOR.MINOR version of the exposed HTTP API (`"1.1"`) defined in the project.
- `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)
- `settings`: a mapping with the values of relevant public settings for clients
  - `batch_max_requests`: Number of requests that can be made in a batch request.

  - `readonly`: Only requests with read operations are allowed.
- `capabilities`: a mapping used by clients to detect optional features of the API.
  - Example:

```
{
  "fxa": {
    "description": "Firefox Account authentication",
    "url": "http://github.com/mozilla-services/kinto-fxa"
  }
}
```

Optional

- `user`: A mapping with an `id` field and a list of `principals` for the currently connected user id. The field is not present when no Authorization header is provided.

Note: The `project_version` contains the source code version, whereas the `http_api_version` contains the exposed HTTP API version.

The source code of the service can suffer changes and have its *project version* incremented, without impacting the publicly exposed HTTP API.

GET /__heartbeat__

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

- `storage` true if storage backend is operational
- `cache` true if cache backend operational

If `kinto-fxa` is installed, an additional key is present:

- `oauth` true if FxA authentication is operational

If `kinto-portier` is installed, an additional key is present:

- `portier` true if portier authentication is operational

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

GET /__lbheartbeat__

Always return `200 OK` with empty body.

Unlike the `__heartbeat__` health check endpoint, which return an error when backends and other upstream services are unavailable, this should always return `200 OK`.

This endpoint is suitable for a load balancer membership test. It the load balancer cannot obtain a response from this endpoint, it will stop sending traffic to the instance and replace it.

GET /contribute.json

The returned value is a JSON mapping containing open source contribution information as advocated by https://www.contributejson.org

GET /__version__

Return a JSON mapping containing information about what distribution has been deployed by OPs.

```
{
  "name":"webpush-channels",
  "version":"1.0.1",
  "commit":"ab8db089ee63dc8e14f4bcfc427a86f311dd7e52",
  "source":"https://github.com/mansimarkaur/webpush-channels.git"
}
```

The content of this view comes from a file, whose location is specified via the `webpush_channels.version_json_path` setting or `WEBPUSH_CHANNELS_VERSION_JSON_PATH` environment variable (*default location is* `version.json` *in current working directory*).

Return `404 Not Found` if no `version.json` file is found.

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 if response is an error (>=500). See more details about *error responses*.

Error responses

API 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`: online resource (e.g. URL to error details)

- `details`: additional details (e.g. list of validation errors)

Example response

```
{
    "code": 412,
    "errno": 114,
    "error": "Precondition Failed",
    "message": "Resource was modified meanwhile",
    "info": "https://server/docs/api.html#errors",
}
```

Refer yourself to the set of errors codes.

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
```

Validation errors

When multiple validation errors occur on a request, the first one is presented in the message.

The full list of validation errors is provided in the `details` field.

```
{
    "code": 400,
    "errno": 109,
    "error": "Bad Request",
    "message": "Invalid posted data",
    "info": "https://server/docs/api.html#errors",
    "details": [
        {
            "description": "42 is not a string: {'name': ''}",
            "location": "body",
            "name": "name"
        }
    ]
}
```

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. `/v1`)

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 API version.

See details in configuration to activate deprecation.

Cheatsheet

Method	URI	Description
GET	*/*	*Information about the running instance*
GET	*/__heartbeat__*	*Return the status of dependent services*
Channels		
PUT	*/channels/(channel_id)/registration*	*Subscribe to a channel*
DELETE	*/channels/(channel_id)/registration*	*Unsubscribe from a channel*
GET	*/channels/(channel_id)*	*Get channel details*
POST	*/channels/(channel_id)*	*Broadcast push notification*
Subscriptions		
POST	*/subscriptions*	*Add a new user subscription*
GET	*/subscriptions*	*Get the list of user's subscriptions*
DELETE	*/subscriptions*	*Delete user's subscriptions*
DELETE	*/subscriptions/(subscription_id)*	*Delete an user subscription*

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

/channels

```
GET /channels/(channel_id), 5
POST /channels/(channel_id), 6
PUT /channels/(channel_id)/registration,
       4
DELETE /channels/(channel_id)/registration,
       5
```

/subscriptions

```
GET /subscriptions, 8
POST /subscriptions, 7
DELETE /subscriptions, 9
DELETE /subscriptions/(subscription_id),
       10
```