Sezame
Client REST API
V 1.5

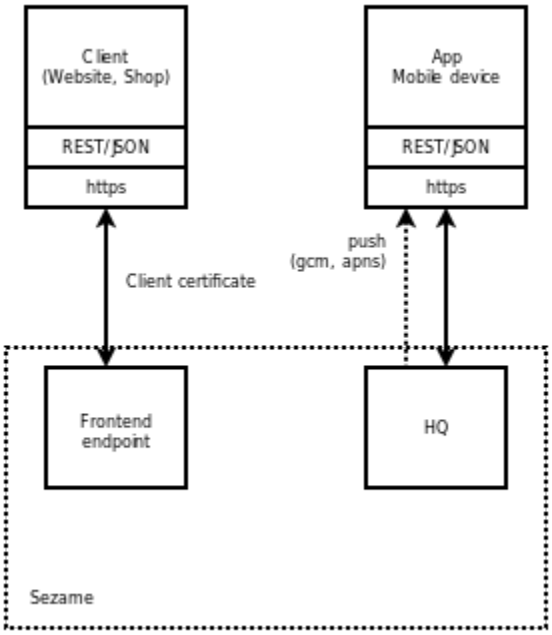
Autor: FinPin Technologies
11. August 2017

Inhaltsverzeichnis

Overview

The client communication is done using JSON/REST over HTTPS. The client must use a SSL client certificate which is signed by the HQ server.

The client endpoint is: https://hqfrontend-finprin.finprin.com



The communication is made using REST/JSON, data is transported using HTTPS only. In both cases a client certificate is needed, to be allowed to contact the HQ frontend services. The only exception are calls during the registration process.

Practically the HQ acts as certificate authority.

There are two possibilities to implement Sezame into your application:

- use one of the available SDKs for your favorite programming language
- communicate directly with the REST API by implementing this protocol

Workflow

To be able to use Sezame within your application you have to fullfill these steps:

- download and install the Sezame app from an app store
- follow the registration process in the app
- register your application/client
- obtain a SSL client certificate
- let your users pair their devices with your application
- issue authentication requests

If you dont have a supported device with fingerprint reader, you must obtain the ssl certificate by using the support channels of Sezame.

REST/API

The frontend interface use basically REST (https://en.wikipedia.org/wiki/Representational_state_transfer), the data content is encoded using JSON, HTTP headers must be set accordingly.
Error handling is done with HTTP status codes, a detailed error description is returned in the body.

Error handling

A list of used status codes

status code	description
200,204	request has been successfully processed
404	entity not found, could be a username, depends on the call
400	general or parameter error
409	entity already exists, i.e. username is already paired
403	permission denied, the call has been refused due to security constraints
500	internal server error, not a client issue

The detailed error info returned inside the body:

```
{
   "tag" : "email",
   "message" : "User not found",
   "errors" : []
}
```

property	description
tag	refers to the affected field, or contains the http status code, if it's not related to a dedicated parameter
message	the error message
errors	an array with addtional error infos, each entry contains a tag and a message property

Example register request which failed, because the supplied email address is invalid:

```
POST /client/register HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"email":"foo@bar.com","name":"xtest remove"}

HTTP/1.1 404 Not Found
Server: nginx
Date: Wed, 09 Dec 2015 11:10:02 GMT
Content-Type: application/json
Content-Length: 54
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0

{"message":"User not found","tag":"email","errors":[]}
```

Client setup

To be able to communicate with the HQ frontend server you have to use a client certificate which has been signed by the HQ server.

If you already have a valid certificate or you don't have a smartphone with fingerprint reader you can skip the register and sign calls. The register and sign calls are the only ones which are allowed without a client certificate.

It's also possible to get a certificate by contacting the Sezame support channels.

Register

HTTP method: POST

URL: /client/register

The client registration call is part of the self-registration process, you must have installed the Sezame app on your mobile device.

There are two ways to initiate the registration process:

- by recovery e-mail address:
  an authentication request is sent to the mobile device, connected to the recovery e-mail. After this call has been successfully sent, you'll get a notification on your Sezame app, which must be acknowledged.
- by QR-Code, using the mobile app:
  a QR-Code must be displayed, containing certain credentials, this QR-Code must scanned using the mobile app. If using this version, an implicit pairing may be done.

Request parameters, if using the recovery e-mail address:

parameter	**description**
email	a valid recovery e-mail address
name	descriptive name of your service

Request parameters, if using the QR-Code:

parameter	**description**
name	descriptive name of your service
username	a username which will be implicitely paired

The HQ server responds with clientcode and sharedsecret.

Response parameters:

parameter	**description**
clientcode	a unique code to identify your client
sharedsecret	a random shared secret to secure the communication
public_key	libsodium public key
id	an id, which must be included into the QR-Code (QR-Code variant only)

You have to save the response parameters somewhere in your database.

PHP sample code for building the QR code data:

```
$qrCodeData = json_encode(Array(
   'id'        => $id,
   'type'      => 'auth'
));
```

Example:

```
POST /client/register HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"email":"foo@bar.com","name":"client api doc test"}

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 12:02:36 GMT
Content-Type: application/json
Content-Length: 122
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

{"clientcode":"5668185c1aa071.35665068","sharedsecret":"09ed3abf4585
e129d3cb0ccd1cae8b47ef503e73954e3245051df791ac77c470"}
```

Checking the status

It is possible to check the status, if the user has scanned the QR Code or not.

HTTP method: GET

URL: /client/register/<id>

parameter	description
id	the id as returned by the registration call

Response parameters:

There are no parameters returned by this call, the response is simply true oder false.

Example:

```
GET /client/register/5668472be0bda685c28b456a HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 15:22:19 GMT
Content-Type: application/json
Content-Length: 54
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

true
```

Sign

HTTP method: POST

URL: /client/sign

The second step for successfully registering your client is to sign a certificate signing request. The CSR must be send to the HQ server, which signs and returns the certificate.

Request parameters:

parameter	description
csr	a certificate signing request in PEM format
sharedsecret	the sharedsecret as optained by the register call

When building the CSR, the clientcode must be used as commonName and the recovery e-mail must be put into the emailAddress field.

Response parameters:

parameter	description
cert	the certificate

After successfully completing the register and the sign call, your client is ready to communicate with the HQ server, i.e. your client must use this certificate as client certificate, if sending requests.
Remember to keep your private key on a safe place protected by a passphrase.

Example:

```
POST /client/sign HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"csr":"-----BEGIN CERTIFICATE
REQUEST-----\nMIIC2jCCAcICAQAwgZQxCzAJBgNVBAYTAkF....=\n-----END
CERTIFICATE
REQUEST-----\n","sharedsecret":"a91d95a323ce23c4c53625984371e5a33290
7cdc1c1d8eb23dc01c508f669491"}

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 12:07:01 GMT
Content-Type: application/json
Content-Length: 1496
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
```

```
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

{"cert":"-----BEGIN CERTIFICATE-----\nMIIE...=\n-----END
CERTIFICATE-----\n"}
```

Pair

HTTP method: POST

URL: /client/link

Each user which uses the Sezame as authentication to your system must be paired, i.e. the user must login with his conventional credentials (username/password), somewhere within the user account settings a QR code must be displayed, a shot of this QR code with the Sezame app must be taken. Once this has been completed, the user is able to login with Sezame.

Request parameters:

parameter	**description**
username	the username within your system/application

Response parameters:

parameter	**description**
id	a unique identifier
clientcode	client code

For build the QR code content, you have to put these parameters into an object and encode it as JSON string:

parameter	**description**
id	a unique identifier as returned by the pair call
clientcode	client code
username	the username within your system/application

PHP sample code for building the QR code data:

```
$qrCodeData = json_encode(Array(
   'id'       => $id,
   'username' => $username,
   'client'   => $clientcode
));
```

Example:

```
POST /client/link HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"username":"foo-client-user"}
```

```
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 12:24:25 GMT
Content-Type: application/json
Content-Length: 112
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

{"id":"f43b066018b2e07811b3bdc7ae39e7a650de901d109f2df8c30e5e5668c1a
197","clientcode":"554b2262a3b542.00744098"}
```

Pairing status

HTTP method: POST

URL: /client/link/status

Use this call to get the pairing status of a certain user.

Request parameters:

parameter	description
username	the username within your system/application

Response parameters:

There are no parameters returned by this call, the response is simply true oder false.

Example:

```
POST /client/link/status HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"username":"foo-client-user"}

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 12:41:18 GMT
Content-Type: application/json
Content-Length: 5
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

false
```

Remove pairing

HTTP method: DELETE

URL: /client/link

Removes a pairing, the user is not anymore able to login with Sezame into your application.

Request parameters:

parameter	description
username	the username within your system/application

Response parameters:

204, with no content on success

Example:

```
DELETE /client/link HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"username":"foo-client-user"}

HTTP/1.1 204 No Content
Server: nginx
Date: Wed, 09 Dec 2015 12:54:49 GMT
Content-Type: text/html
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;
```

Authentication

HTTP method: POST

URL: /auth/login

Sends an authentication request to the HQ frontend server, the user gets a notification on the Sezame app and can acknowledge or deny the authentication request.

The status of the authentication must be either polled by the client or the callback feature must be used.

Request parameters:

parameter	description
username	the username within your system/application

Optional request parameters:

parameter	description
message	a short message displayed on the Sezame app
timeout	the lifetime of the authentication request on the HQ server in minutes, valid range: 1 to 1440
type	auth or fraud, default is auth, a different message is displayed on the Sezame app
callback	callback url, a HTTP POST will be send to this URL once an result from the Sezame app has been received. Only ack and deny trigger this request, there is no POST send after a timeout, you have to handle timeouts yourself. Callback must contain a FQDN, no ip adresses, only https are allowed.
params	additional params send back along callback request, you could put here some session identifiers, or anything else.

Response parameters:

parameter	description
id	a unique identifier for the authentication request, needed for subsequent status requests
status	currently always initiated

Example:

```
POST /auth/login HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"username":"foo-client-user"}

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 13:25:20 GMT
Content-Type: application/json
Content-Length: 54
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

{"id":"56682bc0e0bda686c28b4568","status":"initiated"}
```

Advanced example:

```
POST /auth/login HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"username":"foo-client-user","message":"Login to
Example","timeout":30,"type":"auth","callback":"https:\/\/test.examp
le.com\/sezameauthcallback\/","params":{"foo":"bar"}}

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 13:47:19 GMT
Content-Type: application/json
Content-Length: 54
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

{"id":"566830e7e0bda686c28b4569","status":"initiated"}
```

After starting the authentication process you have to poll for the completion:

Authentication status

HTTP method: GET

URL: /auth/status/<id>

Using this GET request, the HQ frontend server delivers information about the status of an

authentication request, the id from the auth request must be appended to URL, i.e.

/auth/status/566830e7e0bda686c28b4569

Response parameters:

parameter	description
id	a unique identifier for the authentication request, needed for subsequent status requests
status	status of the authentication request: authorized: user has acknowledged the request denied: user has denied the request initiated: user has not yet responded
message	the message sent along the authentication request

If polling the authentication status, you have to handle the timeout yourself. The timeout parameter sent with the authentication request is just a lifetime parameter, it's the maximum amount of minutes an authentication request could be responded, after the lifetime has expired, the request is no longer valid.

Example:

```
GET /auth/status/566845a2e0bda686c28b456a HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 15:16:15 GMT
Content-Type: application/json
Content-Length: 91
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

{"id":"566845a2e0bda686c28b456a","status":"initiated","message":"Aut
horize my portal shop"}
```

Fraud prevention

Sezame can easily be used for fraud prevention, the client can send an authentication request of type „fraud“, if a Sezame enabled user made a login using it's password. It's a good practice to choose an higher lifetime for this request, just to be sure, that the user reads this information on the Sezame app. The message sent to the Sezame app is automatically modified, if type is fraud.

Example:

```
POST /auth/login HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"type":"fraud","username":"foo-client-user","timeout":1440}

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 15:22:19 GMT
Content-Type: application/json
Content-Length: 54
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

{"id":"5668472be0bda685c28b456a","status":"initiated"}
```

Cancel

HTTP method: POST

URL: /client/cancel

Cancels the client registration, after this call the client certificate is invalidated, no more requests are allowed.

There are no request parameters.

Response parameters:

There are no parameters returned by this call, the response is simply true.

Example:

```
POST /client/cancel HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 09 Dec 2015 16:15:44 GMT
Content-Type: application/json
Content-Length: 4
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Strict-Transport-Security: max-age=31536000; includeSubdomains;

true
```

Callback (currently not implemented)

HTTP method: PUT

URL: /client/callback

This API call registers a callback für a specific event, callbacks are POST messages sent to the given endpoint, the endpoint is sent along the registration call. Callbacks are registered per client, i.e. the https client certificate as obtained by the client registration must be used. This call my be sent multiple times, existing callbacks are overwritten.

Request parameters:

parameter	description
event	currently only: user.new
url	endpoint for sending the callback, https only
contenttype	supported types: json

There are no response parameters.

Example:

```
POST /client/callback HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"event":"user.new", "url":"https://foo.bar.com/sezame/event/",
"contenttype":"json"}

HTTP/1.1 204 No Content
Server: nginx
Date: Wed, 09 Dec 2015 12:24:25 GMT
Content-Type: application/json
Content-Length: 112
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
```

The content of the callback message is event specific:

parameter	description
event	user.new
data	an object containing event specific data

New user event:

parameter	description
event	user.new
data.username	Username of the newly registered user

Example of an event callback:

```
POST /sezame/event/ HTTP/1.1
Host:  https://foo.bar.com
Content-Type: application/json

{"event":"user.new", "data":{"username": "johndoe"}}
```

Removing callbacks

HTTP method: DELETE

URL: /client/callback

For removing registered callbacks sent a DELETE message containing the event name, if no event is given, all registered events are removed:

Example delete a specific event:

```
DELETE /client/callback HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

{"event":"user.new"}

HTTP/1.1 204 NO Content
Server: nginx
Date: Wed, 09 Dec 2015 12:24:25 GMT
Content-Type: application/json
Content-Length: 112
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
```

Example delete all registered events:

```
DELETE /client/callback HTTP/1.1
Host: https://hqfrontend-finprin.finprin.com
Content-Type: application/json

HTTP/1.1 204 No Content
```

```
Server: nginx
Date: Wed, 09 Dec 2015 12:24:25 GMT
Content-Type: application/json
Content-Length: 112
Connection: keep-alive
Expires: Tue, 10 Jul 1997 01:00:00 GMT
```