
KEYCLOAK

Terminologie

Resource Owner

Der Benutzer. Hat Gewalt über seine Daten und kann bestimmen, was damit passieren soll und was nicht.

Client

Die Applikation, die im Namen des Benutzers etwas machen, oder auf Daten zugreifen will.

Authorization Server

Der Server, der den Benutzer bereits kennt. Der hat dort einen Account und kann sich dort auch anmelden.

Resource Server

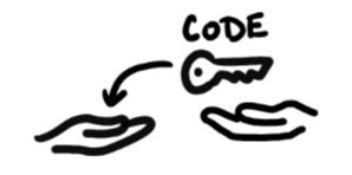
API oder Service, die der Client mit dem Account des Benutzers benutzen will.

(Oft am Authorization Server selbst)

Redirect URI

Nach dem Autorisieren wird der Benutzer an diese URI weitergeleitet. (Callback URL)

Response Type

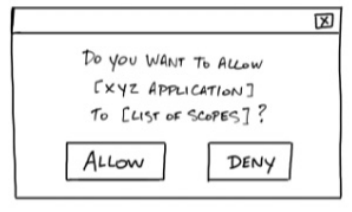


Der Response-Type, den der Client empfangen will, wenn die Autorisierung abgeschlossen ist. (meistens 'code')

Scope

Granulare Permissions, die der Client verlangt. Zugang zu Daten, Aktionen...

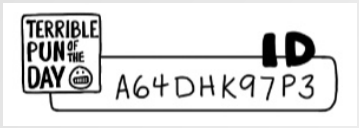
Consent



Der Benutzer wird gefragt, ob er für den Client die Liste von Scopes absegnet.

Die Antwort ist dann die Einwilligung (consent).

Client ID

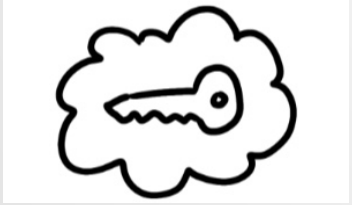


Der Authorization-Server weist jedem Client eine eindeutige ID zu.

Client Secret

Secret Password zwischen Client und Authorization-Server. Damit können die beiden Daten im Hintergrund austauschen.

Authorization Code



Nur kurze Zeit güliger Code, den der Authorization-Server an den Client schickt.

Der Client schickt dann den Code + Client Secret zum Server zurück und bekommt einen Access Token.

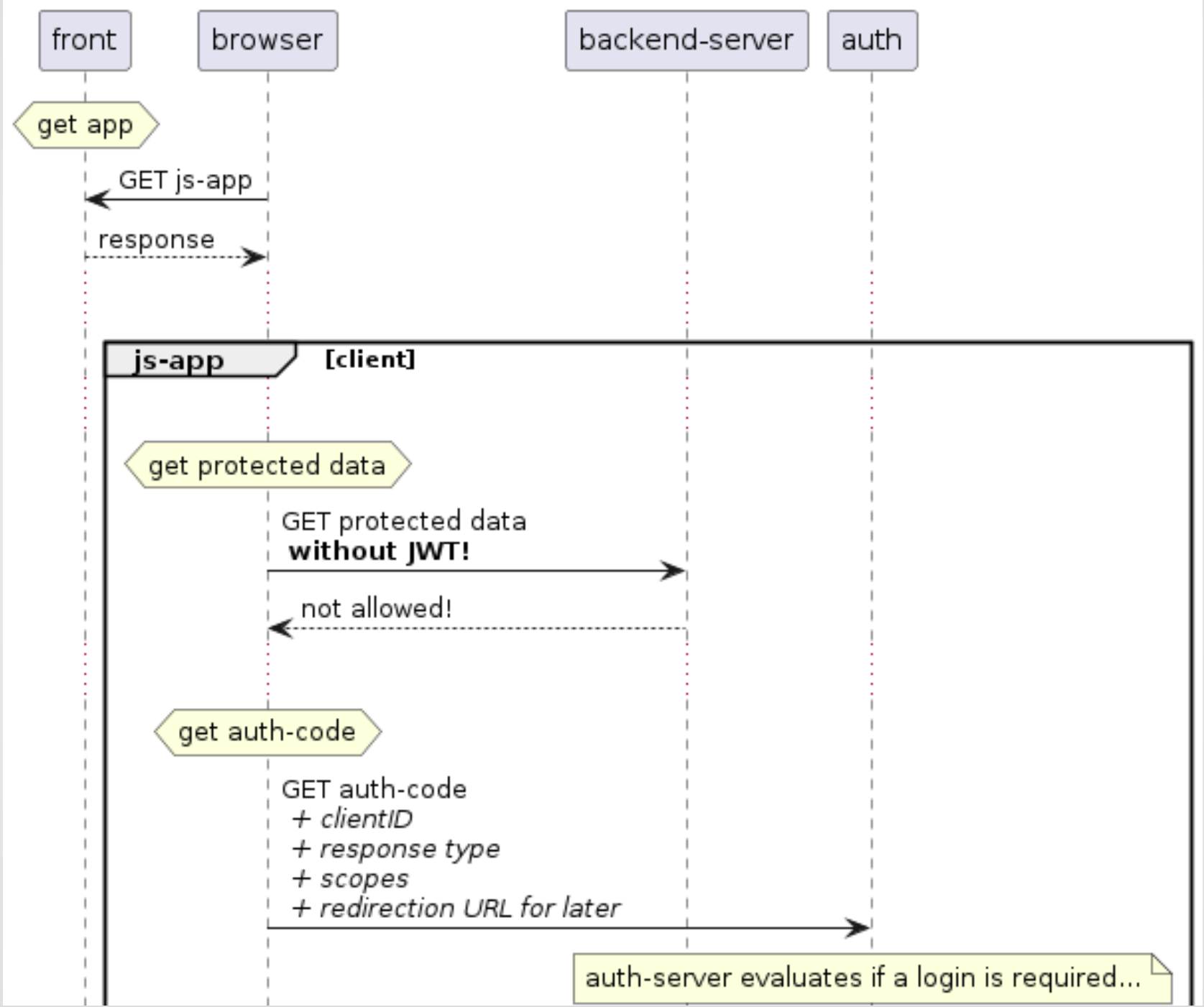
Access Token

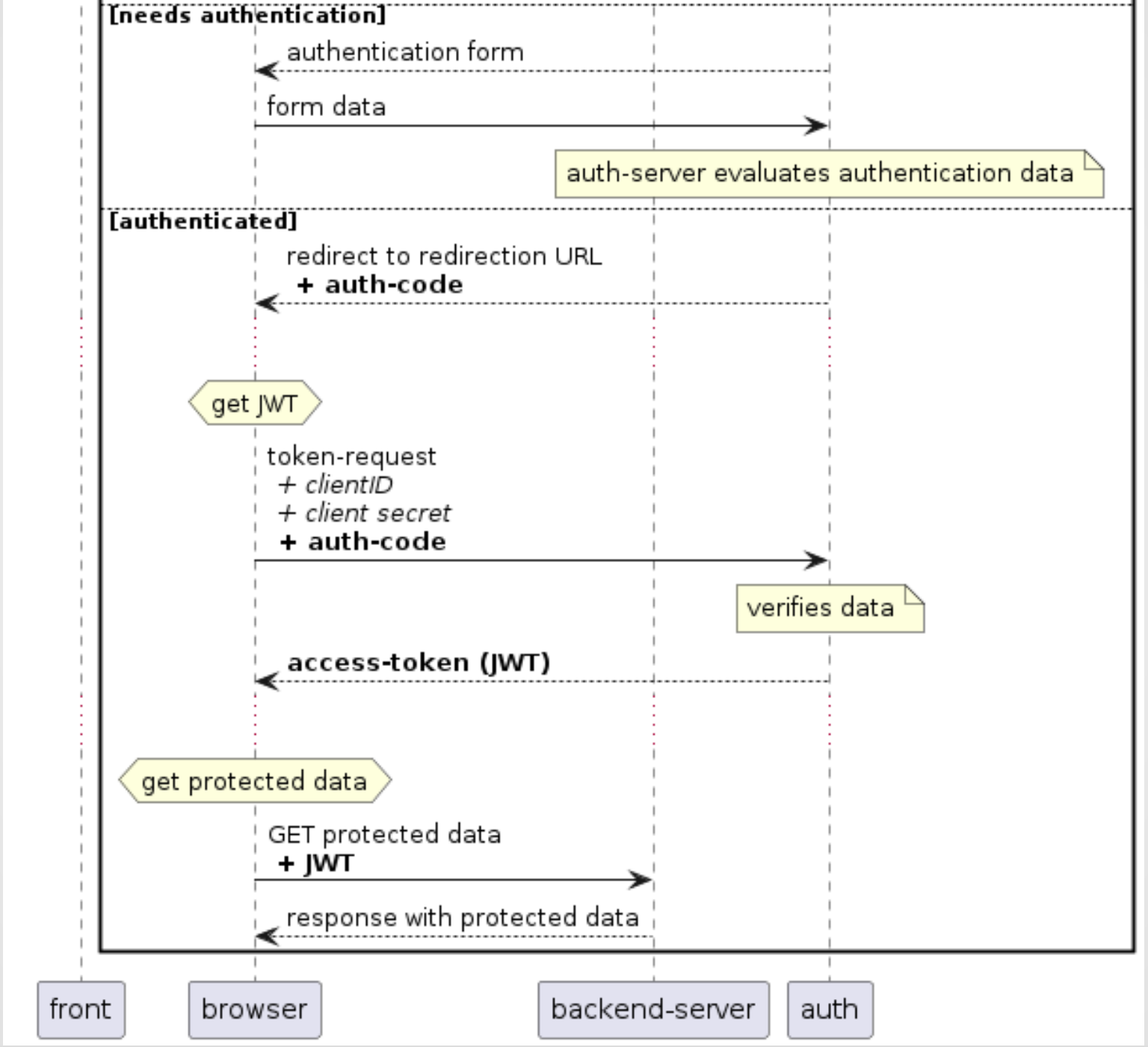
Der Schlüssel, den der Client ab jetzt dazu verwenden wird, mit dem Resource-Server zu kommunizieren.

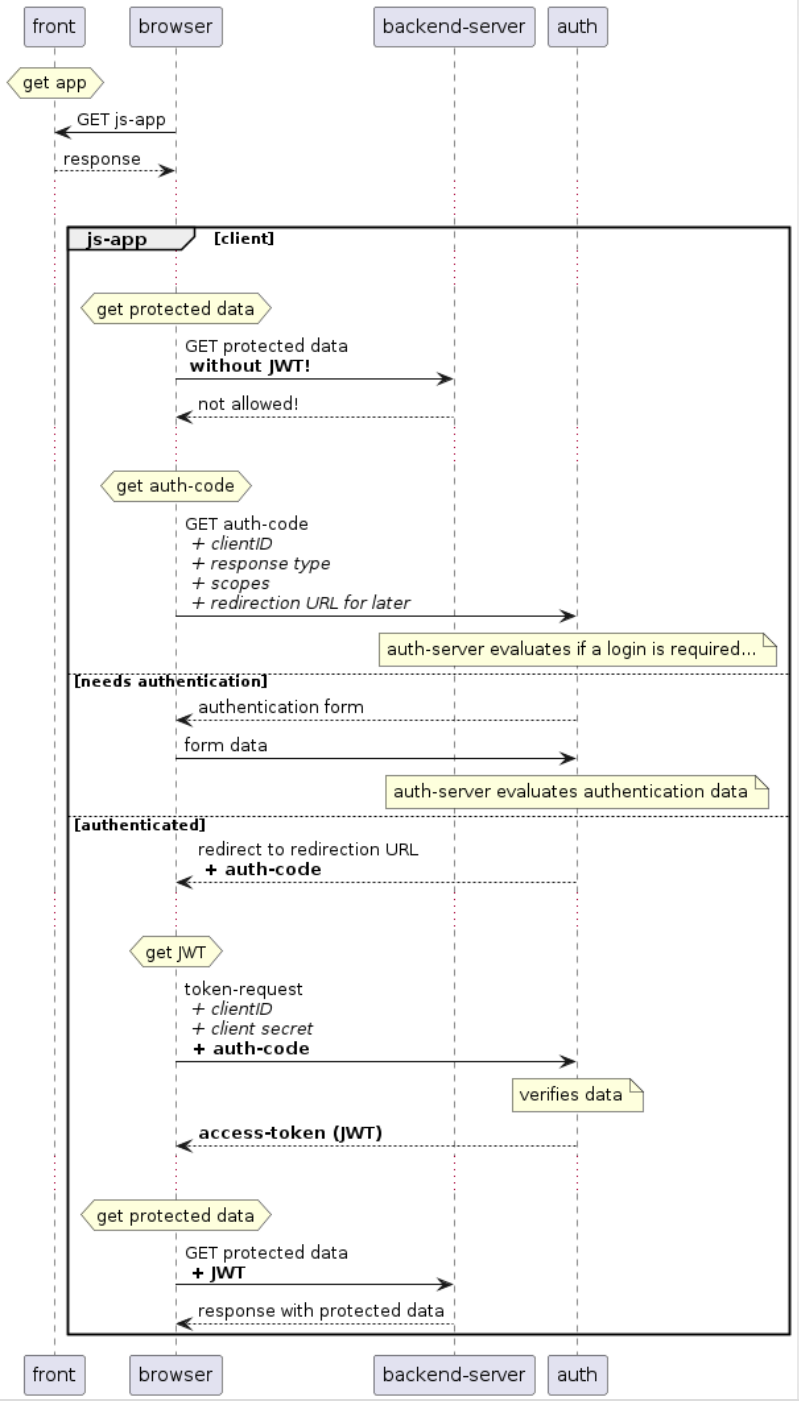
Solange der Client einen Access Token hat, der nicht abgelaufen ist, darf er auf den Resource-Server zugreifen.

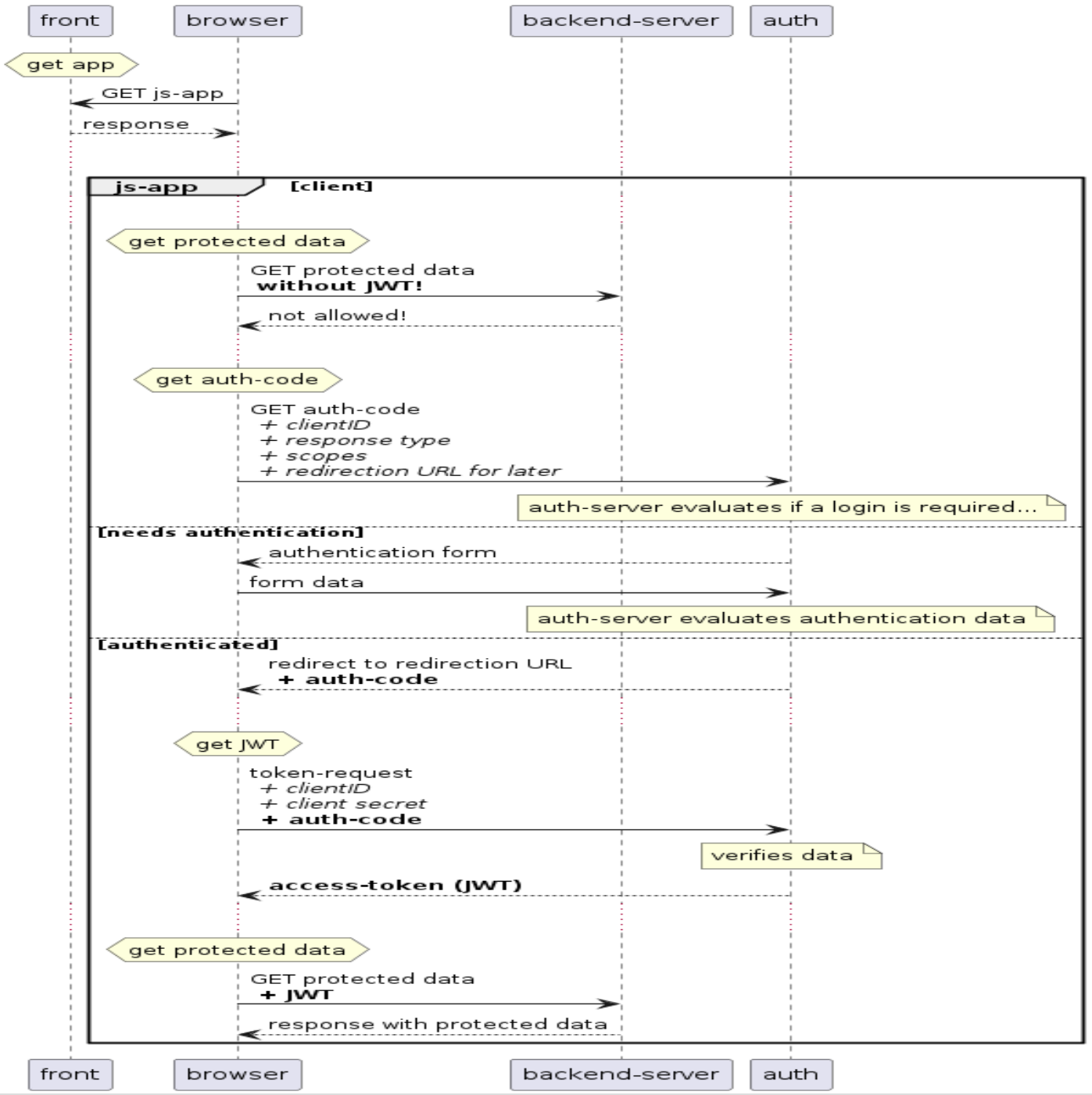
Lebenszeit eines Access Token ist meistens kurz (z.B. 3 Minuten), da er nicht invalidiert werden kann und nur beim Refresh eine Überprüfung des Zustimmung erfolgen kann.

OAuth Flow


front
browser
backend-server
auth
get app
GET js-app
response
js-app
[client]
get protected data
GET protected data
without JWT!
not allowed!
get auth-code
GET auth-code
+ clientID
+ response type
+ scopes
+ redirection URL for later
auth-server evaluates if a login is required...

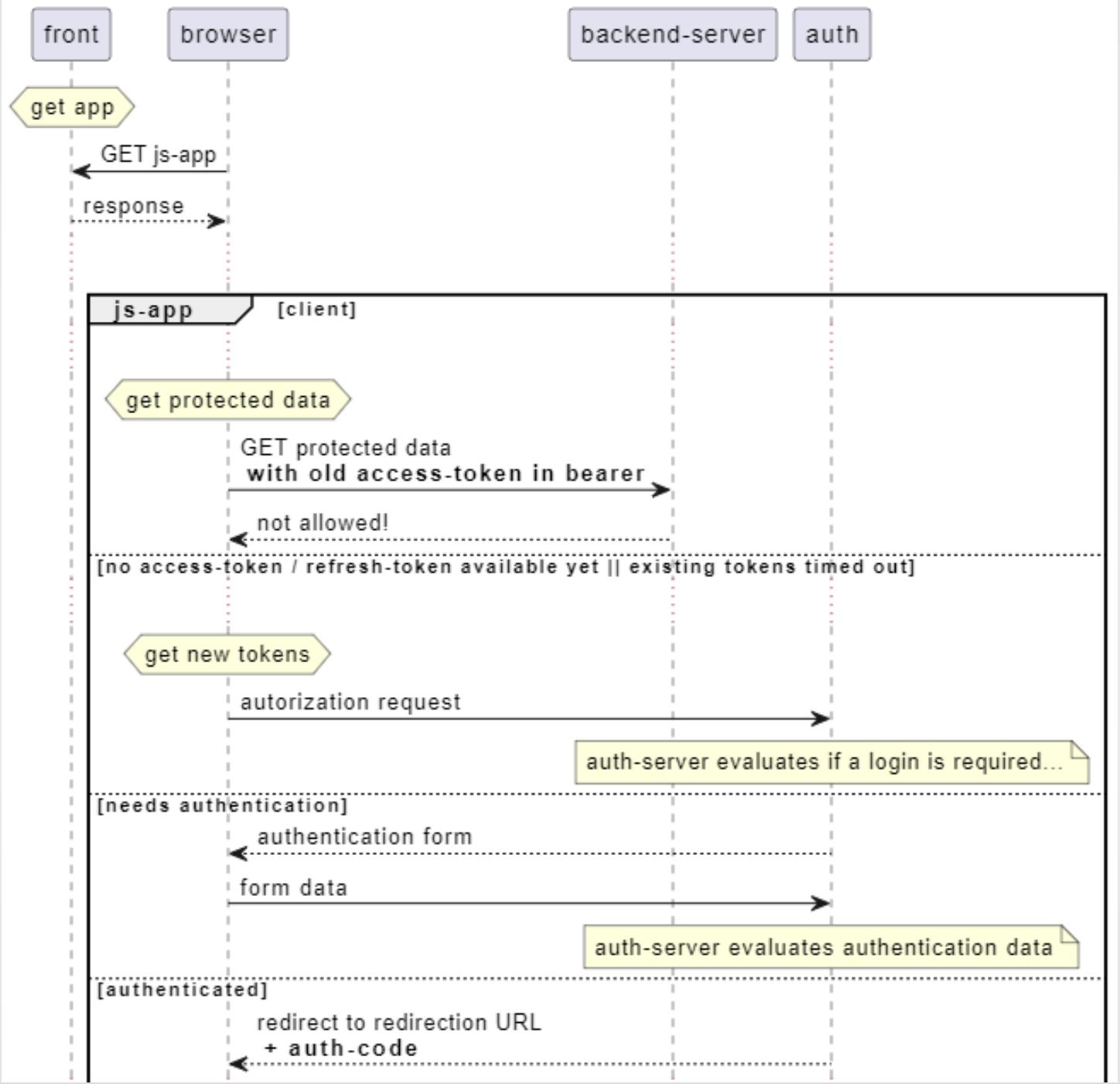

[needs authentication]
authentication form
form data
auth-server evaluates authentication data
[authenticated]
redirect to redirection URL
+ auth-code
get JWT
token-request
+ clientID
+ client secret
+ auth-code
verifies data
access-token (JWT)
get protected data
GET protected data
+ JWT
response with protected data
front
browser
backend-server
auth

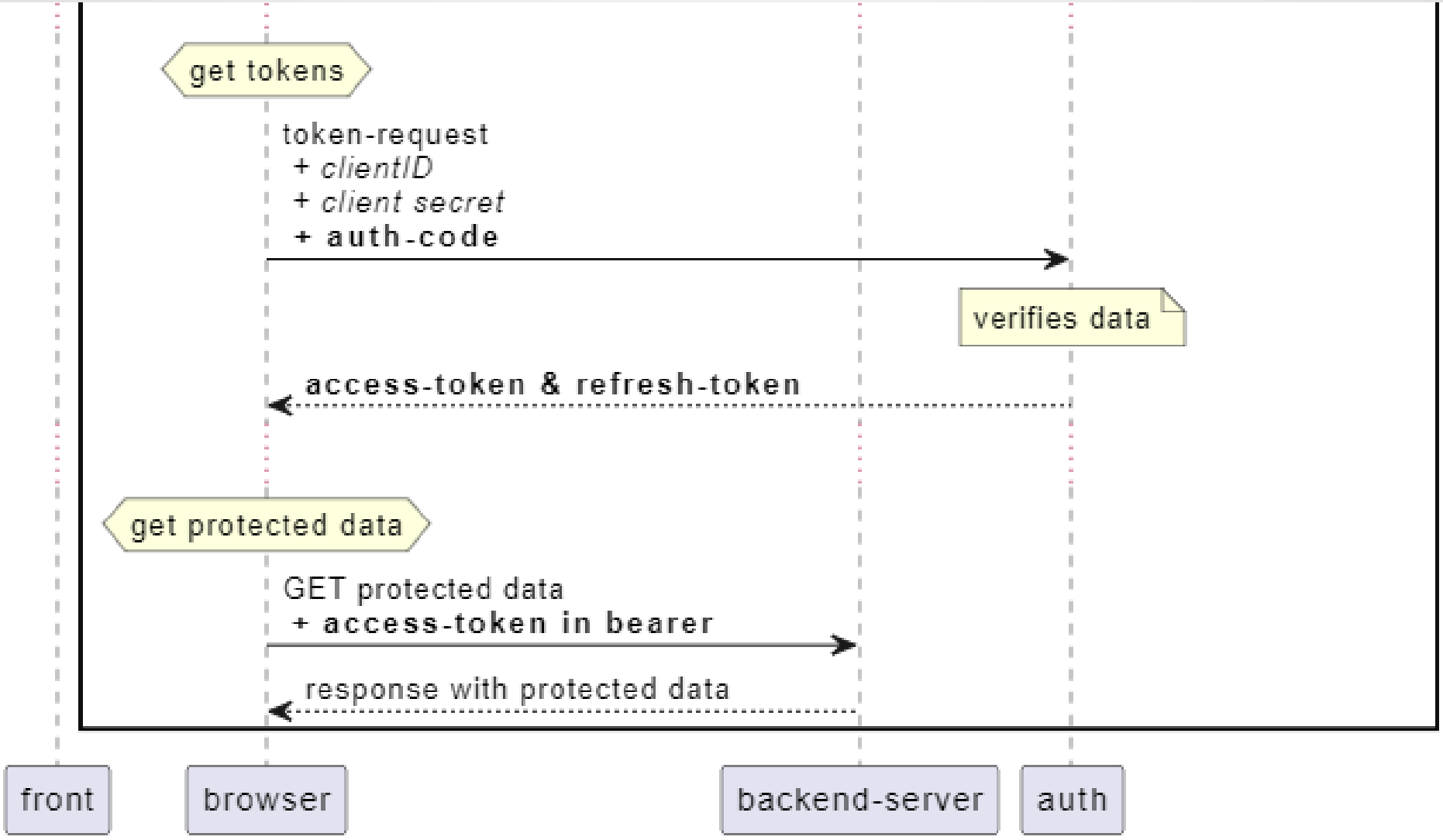

front
browser
backend-server
auth
get app
GET js-app
response
js-app
[client]
get protected data
GET protected data
without JWT!
not allowed!
get auth-code
GET auth-code
+ clientID
+ response type
+ scopes
+ redirection URL for later
auth-server evaluates if a login is required...
[needs authentication]
authentication form
form data
auth-server evaluates authentication data
[authenticated]
redirect to redirection URL
+ auth-code
get JWT
token-request
+ clientID
+ client secret
+ auth-code
verifies data
access-token (JWT)
get protected data
GET protected data
+ JWT
response with protected data
front
browser
backend-server
auth

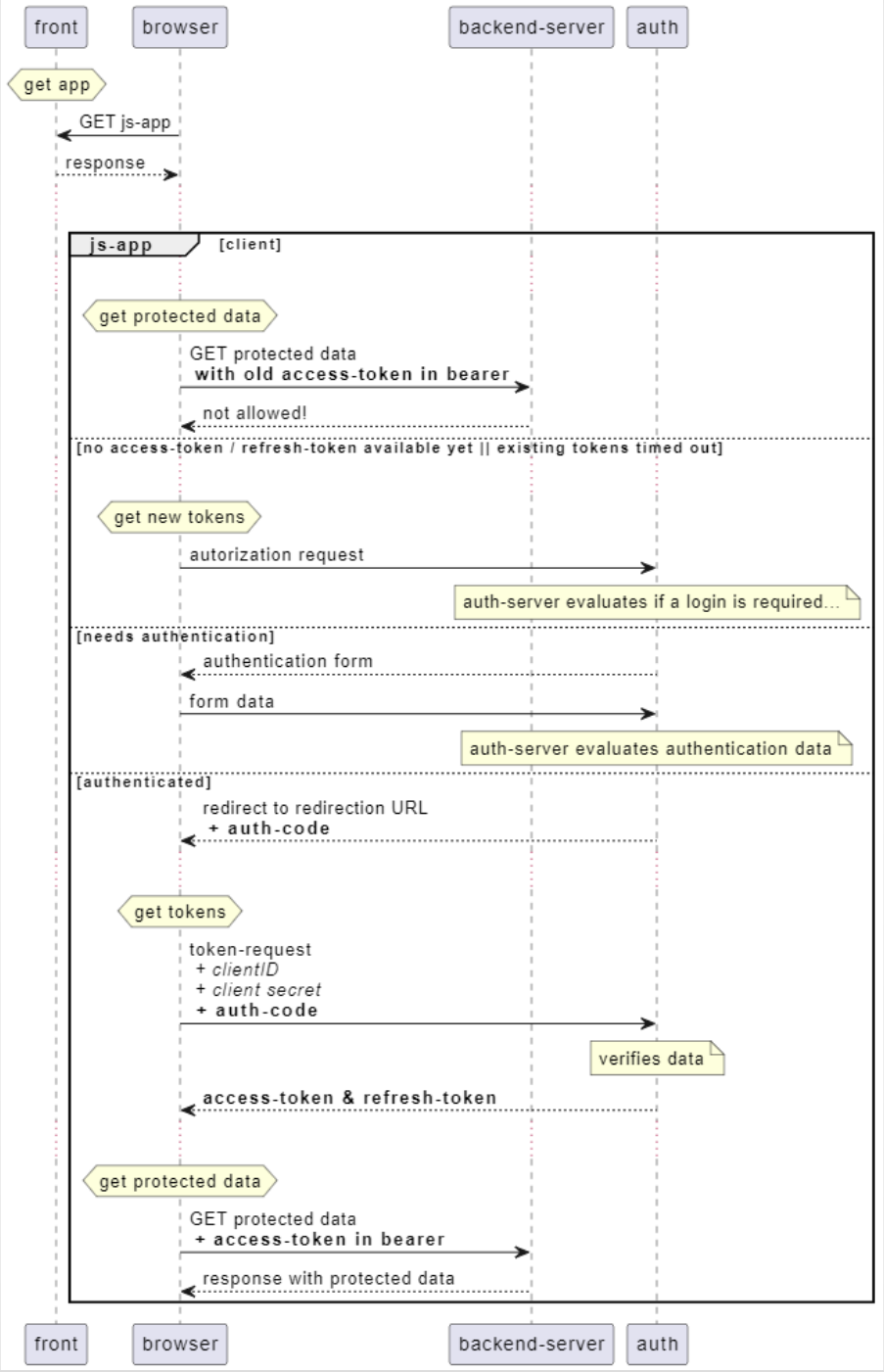

front
browser
backend-server
auth
get app
GET js-app
response
js-app
[client]
get protected data
GET protected data
without JWT!
not allowed!
get auth-code
GET auth-code
+ clientID
+ response type
+ scopes
+ redirection URL for later
auth-server evaluates if a login is required...
[needs authentication]
authentication form
form data
auth-server evaluates authentication data
[authenticated]
redirect to redirection URL
+ auth-code
get JWT
token-request
+ clientID
+ client secret
+ auth-code
verifies data
access-token (JWT)
get protected data
GET protected data
+ JWT
response with protected data
front
browser
backend-server
auth

Token Exchange

Init

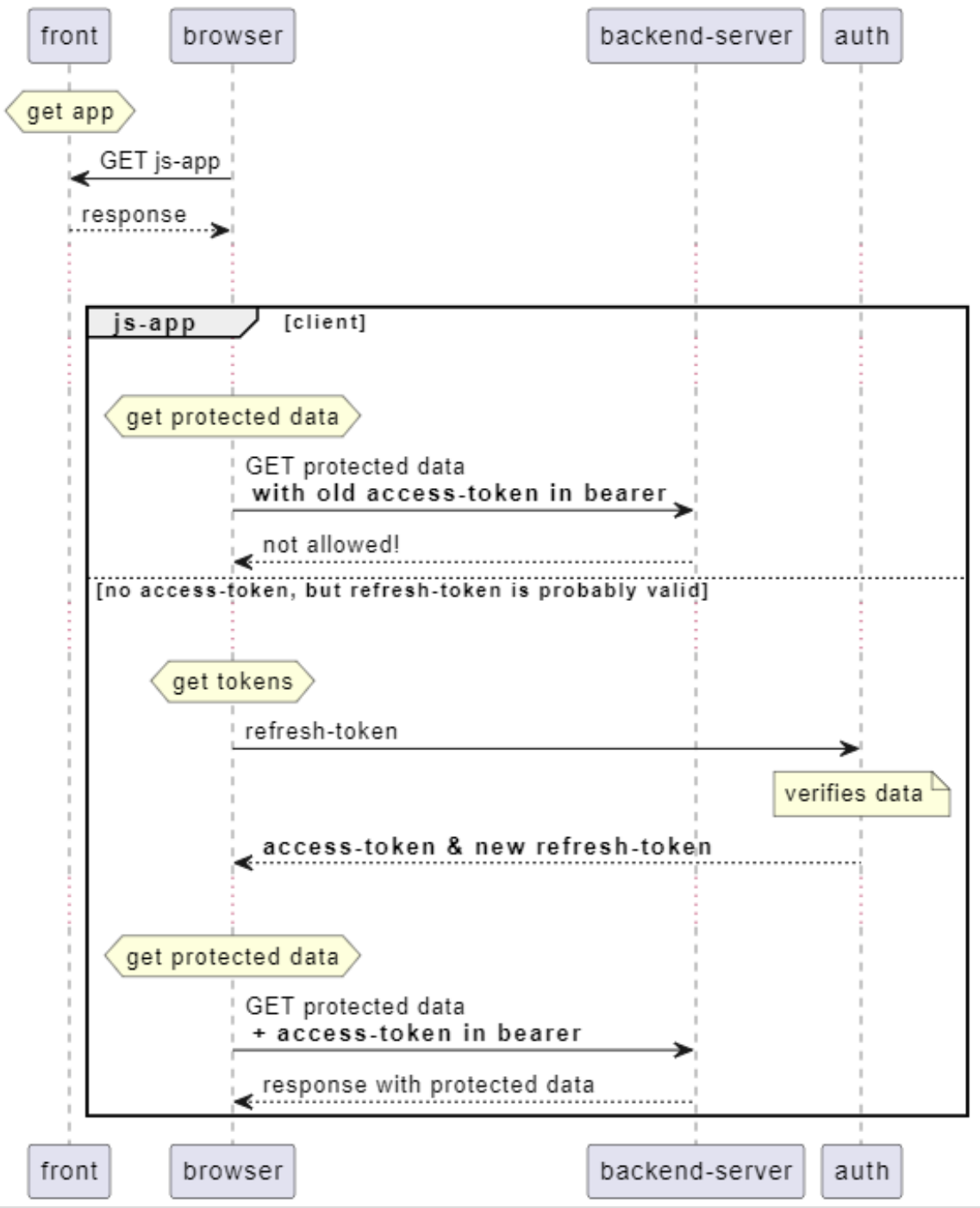

front
browser
backend-server
auth
get app
GET js-app
response
js-app
[client]
get protected data
GET protected data
with old access-token in bearer
not allowed!
[no access-token / refresh-token available yet || existing tokens timed out]
get new tokens
autorization request
auth-server evaluates if a login is required...
[needs authentication]
authentication form
form data
auth-server evaluates authentication data
[authenticated]
redirect to redirection URL
+ auth-code


get tokens
token-request
+ clientID
+ client secret
+ auth-code
verifies data
access-token & refresh-token
get protected data
GET protected data
+ access-token in bearer
response with protected data
front
browser
backend-server
auth


front
browser
backend-server
auth
get app
GET js-app
response
js-app
[client]
get protected data
GET protected data
with old access-token in bearer
not allowed!
[no access-token / refresh-token available yet || existing tokens timed out]
get new tokens
autorization request
auth-server evaluates if a login is required...
[needs authentication]
authentication form
form data
auth-server evaluates authentication data
[authenticated]
redirect to redirection URL
+ auth-code
get tokens
token-request
+ clientID
+ client secret
+ auth-code
verifies data
access-token & refresh-token
get protected data
GET protected data
+ access-token in bearer
response with protected data
front
browser
backend-server
auth

Token Exchange

Refresh


front
browser
backend-server
auth
get app
GET js-app
response
js-app
[client]
get protected data
GET protected data
with old access-token in bearer
not allowed!
[no access-token, but refresh-token is probably valid]
get tokens
refresh-token
verifies data
access-token & new refresh-token
get protected data
GET protected data
+ access-token in bearer
response with protected data
front
browser
backend-server
auth

Adapter zur Absicherung

OpenID Connect

Verschiedenste Implementierungen für verschiedenste Frameworks

Implementierung des Standards OAuth2 unter Verwendung von Json Web Token (JWT)

Java

- JBoss EAP
- WildFly
- Tomcat
- Jetty 9
- Servlet Filter
- Spring Boot
- Spring Security

JavaScript (client-side)

- JavaScript

Node.js (server-side)

- Node.js

C#

- OWIN (community)

Python

- oidc (generic)

Android

- AppAuth (generic)

iOS

- AppAuth (generic)

Apache HTTP Server

- mod_auth_openidc

SAML

Security Assertion Markup Language

Älter als OIDC

Meist in Verbindung mit App-Servern

Java

- JBoss EAP
- WildFly
- Tomcat
- Servlet filter
- Jetty

Apache HTTP Server

- mod_auth_mellon