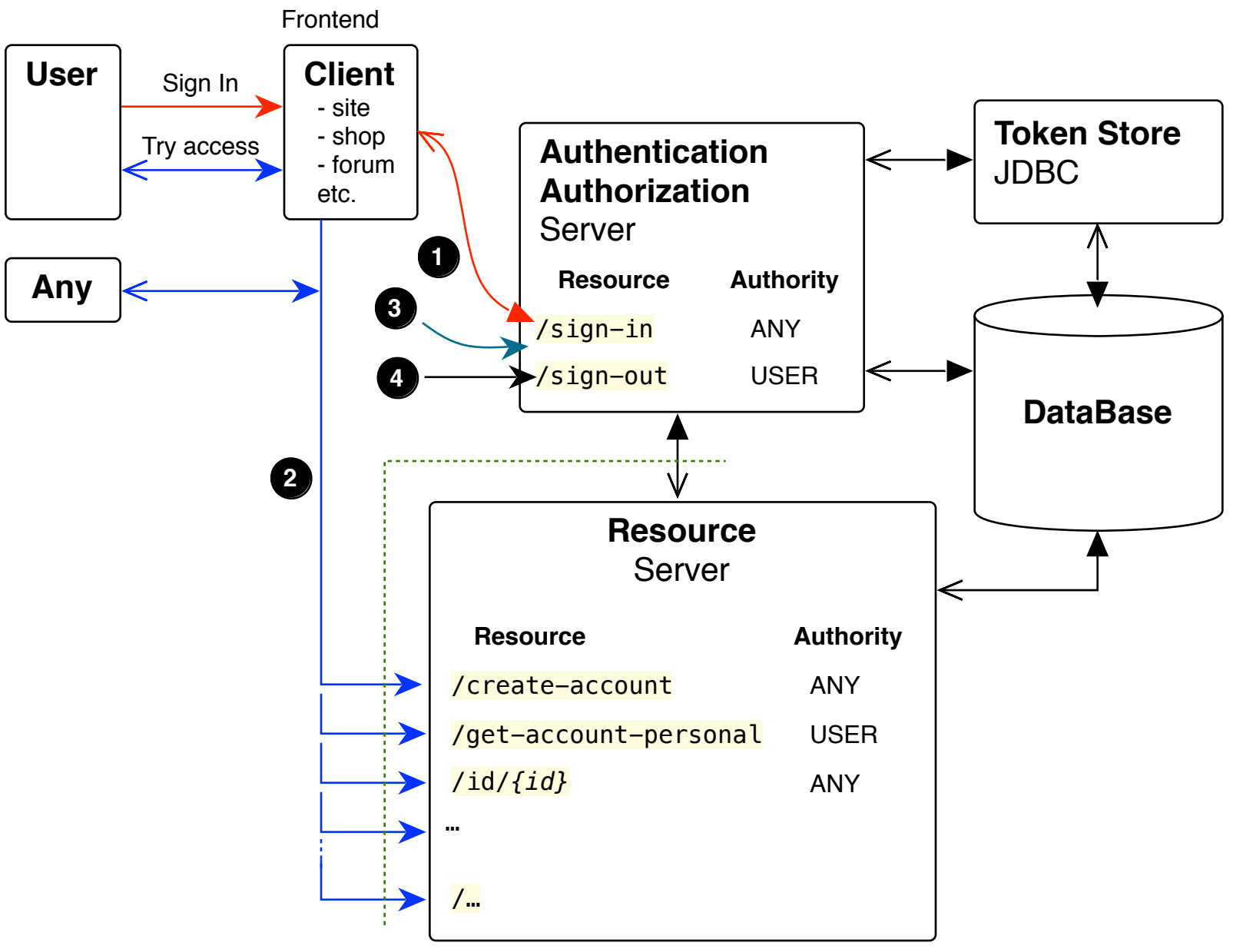
Lifecycle



1 Sign In

POST

```
Req: curl client_id : client_password @ host:port /sign-in
             -d username=username           required
             -d password=password           required
             -d grant_type=password         required
             -d usernameType=usernameType   optional
```

If you need to define a table where stored an ***username***, then, when sign-in, you need to pass an additional parameter, for example, this assumes that the parameter is named `usernameType`, its possible values are: - ***EMAIL***
- ***PHONE***
- ***NICKNAME***
- ***ID***

To avoid problems the values ***username*** and ***password*** passed in the request must contains characters from US-ASCII, and must be percent-encoded.

```
Resp: {
       access_token  : access token
       token_type    : "bearer"
       refresh_token : refresh token
       expires_in    : optional access token expiration time in seconds
       scope         : list of scopes
       data: {
         id          : account id
         username
         authorities : list of authorities: [USER, ADMIN, etc.]
         thirdParty  : null or one of: GOOGLE, FACEBOOK, etc.
         createdOn   : YYYY-MM-DDTHH:MM:SSZ
       }
     }
```

2 Resource accessing

POST or GET

```
Req: curl host:port /path/to/resource -H "Authorization: Bearer ACCESS_TOKEN" ...
```

Resp: *resource content* or *error*

3 Token refreshing

POST

```
Req: curl client_id : client_password @ host:port /sign-in
            -d refresh_token=...
            -d grant_type=refresh_token
```

```
Resp: {
       access_token  : new access token
       token_type    : "bearer"
       refresh_token : refresh token
       expires_in    : optional access token expiration time in seconds
       scope         : list of scopes
       data: {
         id          : account id
         username
         authorities : list of authorities: [USER, ADMIN, etc.]
         thirdParty  : null or one of: GOOGLE, FACEBOOK, etc.
         createdOn   : YYYY-MM-DDTHH:MM:SSZ
       }
     }
```

4 Sign Out

GET

```
Req: curl host:port /sign-out -H "Authorization: Bearer ACCESS_TOKEN"
```

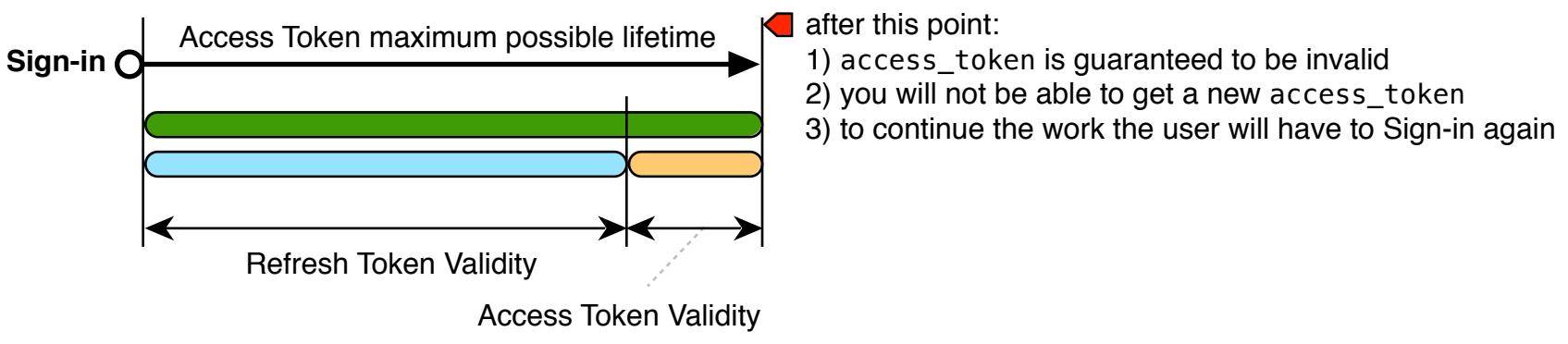
Resp: *success* or *error*

Access Token maximum possible lifetime

By default, the tokens lifetime is as follows (from Spring security oauth2 DefaultTokenServices):

```
int refreshTokenValiditySeconds = 60 * 60 * 24 * 30; // default 30 days.
int accessTokenValiditySeconds  = 60 * 60 * 12;      // default 12 hours.
```

The maximum possible lifetime can be achieved If the Client is refreshing the Access Token before its expiry:



The Nevis Backend Server is pre-configured to work with three types of Clients:

Client	Token Validity (sec.)		Access Token maximum possible lifetime
	access_token	refresh_token	
Untrusted	60*3 3 min.	1 1 sec.	~ 3 minutes
Trusted	60*60*24*20 20 days	60*60*24*340 340 days	~ 360 days
Unlimited	0 unlimite	1 1 sec.	unlimite

The User will have to **Sign-in again *every***

Thus, from the proposed options, it makes sense to periodically refresh the Access Token only for **Trusted** Client

For **Untrusted** and **Unlimited** Clients, attempting to refresh the Access Token will result in automatic Sign-out with the following reason:

```
401 (Unauthorized)
{ "error" : "invalid_token",
  "error_description" : "Invalid refresh token (expired): ..." }
```