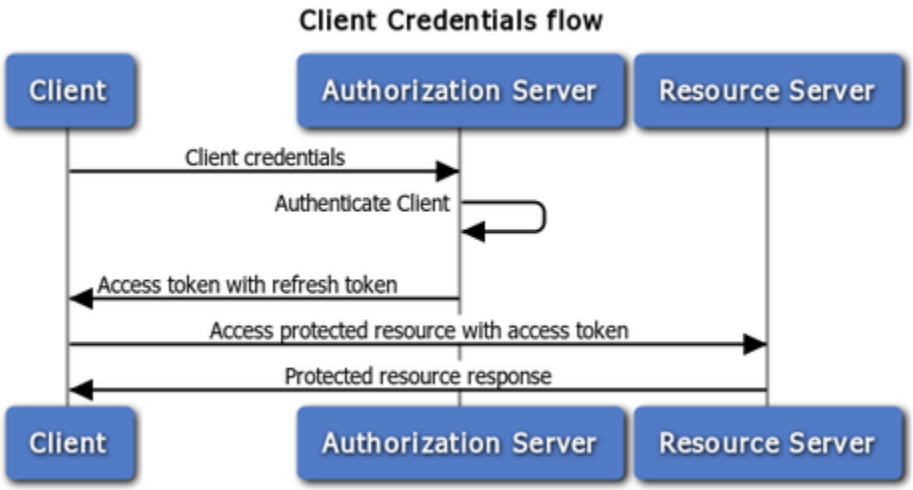
OAuth2-Zugriff auf nuPortalRS

Der Zugriff auf die nuPortalRS-Schnittstellen erfolgt über das Authentifizierungs- und Authorisierungsverfahren OAuth 2.0. Dabei kommt der *grant type* "Client Credentials" zum Einsatz. nuPortalRS folgt hier dem OAuth 2.0 Standard, näher Informationen dazu zB unter: oauth.net.

Beim *Client Credentials* Verfahren kommen eine *Client-ID* und ein *Client Secret* zum Einsatz. nuPortalRS bietet dazu eine interaktive Registrierungs- und Freischaltfunktion an. Dabei wird die *Client-ID* von nuPortalRS generiert und das *Client Secret* vom Entwickler im Registrierungsprozess eingegeben. Mit diesen beiden Informationen erhält man über einen Authentifizierungsrequest einen "Access Token". Mittels dieses "Access Tokens" können die für den Entwickler freigeschalteten Schnittstellen abgefragt werden.

Den Ablauf des gesamten Authentifizierungs- und Authorisierungsverfahren veranschaulicht folgende Grafik:



Arten der Freischaltung

1. Verbandsfreischaltung: Zugriff auf die Daten eines konkreten Verbandes (zB für Verbandshomepages)
2. Vereinsfreischaltung: Zugriff auf die Daten eines konkreten Vereins (zB für Vereinshomepages)
3. Vollständiger Zugriff: Zugriff auf alle Daten aller Verbände (für Partnerplattformen und interne Zugriffe vorgesehen)

Registrierung und Freischaltung

Ein Entwickler erhält Zugriff auf nuPortalRS, indem seitens des jeweiligen Landesverbandes ein Zugang eingerichtet wird – siehe dazu: Freischaltung Schnittstellenzugriff auf nuPortalRS.

Für die Implementierung des OAuth-Protokolls empfiehlt es sich, auf eine etablierte Programmbibliothek zurückzugreifen. Eine Sammlung an Programmbibliotheken für diverse Programmiersprachen ist über https://oauth.net/code/ verfügbar.

Endpoints

Authentifizierung - Access Token Request

URL: https://<nuPortalRSHost>/rs/auth/token

HTTP method: POST

Parameter:

- grant_type: für den initialen Request **client_credentials**
- client_id: die von nuLiga vergebene Client-ID
- client_secret: das im Registrierungsprozess angegebene secret (= Passwort)
- scope: der Geltungsbereich für den token
    - nuPortalRS_federation: Wird für die Verbandsfreischaltung benötigt
    - nuPortalRS_club: Wird für die Vereinsfreischaltung benötigt

- nuPortalRS_all: Wird für den Vollzugriff benötigt

Die Parameter können entweder als *Content-Type: application/x-www-form-urlencoded* oder als *application/json* übergeben werden.

Achtung: laut RFC 6749 müssen die Parameter im request-body übermittelt werden

Beispiel:

OAuth 2.0 authentification request

```
$ curl https://hbde-portal.liga.nu/rs/auth/token \
> -X POST \
> -H "Content-Type: application/x-www-form-urlencoded" \
> -d "grant_type=client_credentials" \
> -d "client_id=ebead868-4e50-49b1-a4a7-82a8caa0c9a3" \
> -d "client_secret=7XxZE9**********" \
> -d "scope=nuPortalRS_federation"
```

OAuth 2.0 authentication response

```
{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJudVBvcn
RhbFJTIiwic3ViIjoiZWJlYWQ4NjgtNGU1MC00OWIxLWE0YTctODJhOGNhYTBjOWEzIiwiZX
hwIjoxNTIzMzYzNjMwLCJpYXQiOjAsImFjY2Vzc2VzIjp7Im51UG9ydGFsUlNfZmVkZXJhdG
lvbiI6WyJIVk4iXX19.**************************************************
****",
"refresh_token":"fe6e949d-8f11-4f6f-83bb-***********"}
```

Das OAuth2-Protokoll erlaubt es, für einen Authentifizerungsprozess mehrere Scopes gleichzeitig anzugeben. Zu beachten ist, dass die Authentifizierung nur dann erfolgreich verläuft, wenn die abgefragten oAuth-Scopes mit den Freigaben durch den Verband übereinstimmen. Andernfalls wird die Anfrage abgelehnt.

Ist die Authentifizierung erfolgreich, enthält der Response einen *access token* und einen *refresh token*. Der *access token* ist 5 Minuten lang gültig. Nach Ablauf des *access token* kann mithilfe des *refresh tokens* ein neuer *access token* generiert werden – siehe folgenden Absatz.

Authentifizierung - Refresh Token Request

URL: https://<nuPortalRSHost>/rs/auth/token

HTTP-method: POST

Parameter:

- grant_type: für den Refresh Token Request **refresh_token**
- refresh_token: das im Authentifizierungs-response zurückgegebene *refesh token*
- client_id: die von nuLiga vergebene Client-ID
- client_secret: das im Registrierungsprozess angegebene secret (= Passwort)
- scope: der Geltungsbereich für den token
    - nuPortalRS_federation: Wird für die Verbandsfreischaltung benötigt
    - nuPortalRS_club: Wird für die Vereinsfreischaltung benötigt
    - nuPortalRS_all: Wird für den Vollzugriff benötigt

Achtung: laut RFC 6749 müssen die Parameter im request-body übermittelt werden.

Beispiel:

```
$ curl https://hbde-portal.liga.nu/rs/auth/token \
> -X POST \
> -H "Content-Type: application/x-www-form-urlencoded" \
> -d "grant_type=refresh_token" \
> -d "refresh_token=fe6e949d-8f11-4f6f-83bb-************" \
> -d "client_id=ebead868-4e50-49b1-a4a7-82a8caa0c9a3" \
> -d "client_secret=7XxZE9**********" \
> -d "scope=nuPortalRS_federation"
```

Der response ist gleich strukturiert wie oben beim *access token request*.

Zugriff auf Ressourcen

Basis-URL: https://<nuPortalRSHost>/rs/

HTTP-method: GET

Header:

- Authorization: der konstante String "Bearer " zusammengefügt mit dem über den Authentifizierungsprozess erhaltenen *access token*

Beispiel:

```
$ curl https://hbde-portal.liga.nu/rs/2014/federations \
> -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJudVBvcnRhbFJTIiwic3ViIjo
iZWJlYWQ4NjgtNGU1MC00OWIxLWE0YTctODJhOGNhYTBjOWEzIiwiZXhwIjoxNTIzMzYzNjM
wLCJpYXQiOjAsImFjY2Vzc2VzIjp7Im51UG9ydGFsUlNfZmVkZXJhdGlvbiI6WyJIVk4iXX1
9.**********************************************************"
```

Der Zugriff auf die einzelnen Ressourcen ist von dem Typ der Freischaltung abhängig und muss mittels *access token* erfolgen.

1. Verbandsfreischaltung: Es handelt sich hierbei um einen umfangreichen Zugriff auf die Daten des Verbandes, dies umfasst Meisterschaftsdaten, Vereinsdaten, Verbandsdaten, usw. - siehe - https://<nuPortalRSHost>/rs/documentation/
2. Vereinsfreischaltung: Der Zugriff ist nur auf die Vereins-Daten möglich für die eine Freischaltung vorliegt - siehe https://<nuPortalRSHost>/rs/documentation/resource_ClubsREST.html
3. Vollständiger Zugriff: Es handelt sich hierbei um einen uneingeschränkten Zugriff auf alle Daten aller Verbände, dies umfasst Meisterschaftsdaten, Vereinsdaten, Verbandsdaten, usw. - siehe - https://<nuPortalRSHost>/rs/documentation/

Expired Authorization

Das *access token* hat eine zeitlich beschränkte Gültigkeit (heute: 5 Minuten). Nach diesem Zeitraum antwortet nuPortalRS mit einem 401-Status, zB:

```
$ curl -i https://hbde-portal.liga.nu/rs/2014/federations \
> -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJudVBvcnRhbFJTIiwic3ViIjo
iZWJlYWQ4NjgtNGU1MC00OWIxLWE0YTctODJhOGNhYTBjOWEzIiwiZXhwIjoxNTIzMzYzNjM
wLCJpYXQiOjAsImFjY2Vzc2VzIjp7Im51UG9ydGFsUlNfZmVkZXJhdGlvbiI6WyJIVk4iXX1
9.**********************************************************"
HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
Content-Type: application/xml
Content-Length: 30
Date: Wed, 11 Apr 2018 08:20:20 GMT

Token is not valid
```

In diesem Fall ist zunächst mit dem refresh token ein neues access token zu holen, mit dem dann der request wiederholt werden kann:

```
$ curl https://hbde-portal.liga.nu/rs/auth/token \
> -X POST \
> -H "Content-Type: application/x-www-form-urlencoded" \
> -d "grant_type=refresh_token" \
> -d "refresh_token=e20b0c99-e88e-4591-a33a-2344447cf774" \
> -d "client_id=ebead868-4e50-49b1-a4a7-82a8caa0c9a3" \
> -d "client_secret=7XxZE9osGXCZM4Ux" \
> -d "scope=nuPortalRS_federation"
{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJudVBvcn
RhbFJTIiwic3ViIjoiZWJlYWQ4NjgtNGU1MC00OWIxLWE0YTctODJhOGNhYTBjOWEzIiwiZX
hwIjoxNTIzNDQ2NTc5LCJpYXQiOjAsImFjY2Vzc2VzIjp7Im51UG9ydGFsUlNfZmVkZXJhdG
lvbiI6WyJIVk4iXX19.**********************************************
****",
"refresh_token":"7a1c4be9-0795-4f7d-be52-9292050df570"}

$ curl -i https://hbde-portal.liga.nu/rs/2014/federations \
> -H "Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJudVBvcnRhbFJTIiwic3ViIjo
iZWJlYWQ4NjgtNGU1MC00OWIxLWE0YTctODJhOGNhYTBjOWEzIiwiZXhwIjoxNTIzNDQ2NTc
5LCJpYXQiOjAsImFjY2Vzc2VzIjp7Im51UG9ydGFsUlNfZmVkZXJhdGlvbiI6WyJIVk4iXX1
9.**********************************************************"
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml
Content-Length: 5484
Date: Wed, 11 Apr 2018 11:32:52 GMT

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<federations xmlns="http://www.liga.nu/rs/2014">
  <federationAbbr name="Deutscher Handballbund" nickname="DHB"
region="DE"
federationUri="https://hbde-portal.liga.nu/rs/2014/federations/DHB"
nuLigaState="false"/>
...
</federations>
```

Achten Sie darauf, dass der *refresh token request* neben dem neuen, frischen *access token* auch ein neues *refresh token* ausliefert, das für den nächsten *refresh token request* heran zu ziehen ist.

Implementation eines Clients in Pseudo-Code

Das folgende Codesegment dient zur Verdeutlichung des OAuth 2.0 Verfahrens. Es verwendet ein synchrones http Verfahren und eine sehr rudimentäre Fehlerbehandlung, was in der Praxis natürlich nicht so umgesetzt würde. Wir machen nochmals auf die große Auswahl von Client-Libraries aufmerksam (https://oauth.net/code/) und raten von einer eigenhändigen Umsetzung ab.

```
accessToken = null;
refreshToken = null;
clientID = "ebead868-4e50-49b1-a4a7-82a8caa0c9a3";
```

```
clientSecret = "7XxZE9**********";

init() {
 request = new PostRequest("https://hbde-portal.liga.nu/rs/auth/token");
 request.addHeader("Content-Type", "application/json");
 request.body = {
  "grant_type": "client_credentials",
  "client_id": clientID,
  "client_secret": clientSecret,
  "scope": "nuPortalRS_federation"
 };
 response = request.submit();
 if (response.statusCode() == 200) {
  accessToken = response.body.get("access_token");
  refreshToken = response.body.get("refresh_token");
 } else {
  throw new Error("error getting access token: " +
response.statusCode());
 }
}

getNuPortalResource(resource) {
 if (accessToken == null) {
  throw new Error("please call init() first.");
 }

 request = new GetRequest("https://hbde-portal.liga.nu/rs/" + resource);
 request.addHeader("Authorization", "Bearer " + accessToken);
 response = request.submit();

 if (response.statusCode() == 200) {
  return response;
 } else if (response.statusCode() == 401) {

  refreshRequest = new
PostRequest("https://hbde-portal.liga.nu/rs/auth/token");
  refreshRequest.addHeader("Content-Type", "application/json");
  refreshRequest.body = {
   "grant_type":"refresh_token",
   "refresh_token": refreshToken,
   "client_id": clientID,
   "client_secret": clientSecret,
   "scope": "nuPortalRS_federation"
  }
  response = request.submit();
  if (response.statusCode() == 200) {
   accessToken = response.body.get("access_token");
   refreshToken = response.body.get("refresh_token");
   request = new GetRequest("https://hbde-portal.liga.nu/rs/" +
resource);
```

```
   request.addHeader("Authorization", "Bearer " + accessToken);
   response = request.submit();

   if (response.statusCode() == 200) {
    return response;
   } else {
    throw new Error("error after token refresh: " +
response.statusCode());
   }
  } else {
   throw new Error("error getting refresh token: " +
response.statusCode());
  }

 } else {
  throw new Error("http error " + response.statusCode() + " occurred.");
 }
}

listAllFederations() {
 response = getNuPortalResource("2014/federations");
 println("All federations known by nuPortalRS:");
 for (fed: response.body.get("federations")) {
```

```
    println(fed.nickname, fed.name);
   }
  }
```

Implementationsdetails in nuPortalRS

Der *access token request* für die Autorisierung ist mit deutlich höherem Aufwand verbunden als die nachfolgenden *refresh token requests*. Das ist auch der Hintergrund dafür, dass es überhaupt das refresh-Verfahren in OAuth 2.0 gibt! Grundsätzlich sollte der nuPortalRS-Client mit einem einzigen *access token request* auskommen und danach nur noch *refresh token requests* machen. nuPortalRS erlaubt je Client-ID nur eine eingeschränkte Anzahl von *access token request* innerhalb einer gewissen Zeitspanne (heute: max 5 request innerhalb von 30 Min.). Und nur das refresh-token des jeweils jüngsten *access* oder *refresh token request* kann für den nächsten *refresh token request* herangezogen werden. Sollte einmal das gültige (jüngste) *refresh token* invalide sein (zB durch einen *revoke* durch nuPortalRS), dann soll ein erneuter *access token request* au sgeführt werden.

Weiterführende Links

- RFC 6749 (OAuth2): https://tools.ietf.org/html/rfc6749
- Client-Libraries für diverse Programmiersprachen: https://oauth.net/code/

Verwandte Artikel

- nuLiga OAuth2-Login
- Lizenzperioden in Tennis-Deutschland
- OAuth2-Zugriff auf nuPortalRS
- Freischaltung von Fremdsystemen für den Schnittstellenzugriff auf nuPortalRS
- TTCH Spielplanung