Unabhängige Treuhandstelle

UNIVERSITÄTSMEDIZIN GREIFSWALD

Stand: Jan 2022

TTP-FHIR-Gateway mit Keycloak-basierter Authentifizierung

Installation und Einrichtung von **Keycloak**

Ab TTP-FHIR Gateway Version 2.0.0 ist eine Absicherung der TTP-FHIR-Gateway-Schnittstelle je Endpunkt (und somit je Werkzeug) vorgesehen und nach Bedarf konfigurierbar.

Installation

Wenn nicht schon geschehen, dann muss zuerst **Docker** installiert werden, z.B. **Docker Desktop**

Gemäß dieser Anleitung kann dann **Keycloak** in **Docker** installiert werden. Das Beispiel-Realm `myrealm` braucht dabei natürlich nicht angelegt werden.

ACHTUNG! Wenn das **TTP-FHIR-Gateway** und der **Keycloak**-Server auf der gleichen Maschine laufen sollen, ist zu beachten, dass beiden Anwendungen verschiedene Ports zugewiesen werden. Wenn das **TTP-FHIR-Gateway** z.B. den Port *8080* nutzt, müsste Keycloak einen anderen Port nutzen, z.B.:

```
docker run -p 8081:8080 -e KEYCLOAK_USER=admin -e KEYCLOAK_PASSWORD=
{KEYCLOAK_PASSWORD} quay.io/keycloak/keycloak:13.0.1
```

In der Standardinstallation von **Keycloak** darf der zweite Port in `-p 8081:8080` nicht verändert werden, über den ersten Port (im weiteren `{KEYCLOAK_PORT}` genannt) wird **Keycloak** auf dem Dockerhost angesprochen.

Einrichtung

Zum Einrichten von **Keycloak** startet man die Admin-Konsole unter http://`{KEYCLOAK_HOST}`:`{KEYCLOAK_PORT}`/auth/admin und meldet sich als Benutzer `admin` und mit dem `{KEYCLOAK_PASSWORD}` an.

Realm `ttp` hinzufügen

Die Admin-Konsole startet im Realm ***Master***, erkennbar links oben. Dieses ist nur zur Verwaltung von **Keycloak** selbst bestimmt. Führt man die Maus darüber, kann man ***Add Realm*** aufrufen, um ein neues Realm hinzuzufügen. Das Realm für das **TTP-FHIR-Gateway** bekommt den Namen `ttp`.

Client `ttp-fhir` anlegen und konfigurieren

Um den Client für den Zugriff auf das TTP-FHIR Gateway anzulegen, navigiert man am linken Rand zu ***Clients***, erzeugt mit ***Create*** (oben rechts) einen neuen Client und vergibt die ***Client ID*** `ttp-fhir`. Anschließend muss unter ***Settings*** für diesen Client der ***Access Type*** `confidential` gewählt werden.

Zusätzlich müssen für diesen Client die **Base Url** `http://{KEYCLOAK_HOST}:{KEYCLOAK_PORT}` und **Valid Redirect URIs** (z.B. `*`) gesetzt werden.

Unter ***Credentials*** für diesen Client findet man das ***Secret***, das im weiteren Verlauf noch benötigt wird und deshalb kopiert und gemerkt werden muss (z.B. *cd28481d-dd42-469a-b2f4-803e9476172c*).

Rollen anlegen je Endpunkt

Für den Zugriff auf die einzelnen Komponenten des **TTP-FHIR-Gateways** müssen nun jeweils Rollen angelegt werden. Dazu navigiert man am linken Rand zu ***Roles*** und erzeugt mit ***Add Role*** (oben rechts) vier neue. Als Namen vergibt man dabei

- für den **Dispatcher** `/ttp-fhir/fhir/dispatcher`,
- für **gICS** `/ttp-fhir/fhir/gics`,
- für **gPAS** `/ttp-fhir/fhir/gpas` und
- für **E-PIX** `/ttp-fhir/fhir/e-pix`.

Als Beschreibung für die Rollen bietet sich an, für **gICS** z.B. *Rolle für Zugriff auf gICS-Funktionalitäten des TTP-FHIR-Gateway* usw. zu verwenden.

User hinzufügen

Schließlich müssen noch je Komponente die Nutzer für den Zugriff auf die Funktionalitäten des **TTP-FHIR-Gateway** angelegt werden. Dazu navigiert man am linken Rand zu ***Users*** und erzeugt mit ***Add User*** (oben rechts) vier neue Nutzer. Dabei müssen die Felder ***ID*** und ***Required User Action*** unbedingt leer bleiben, lediglich der jeweilige ***Username*** wird eingetragen.

Anschließend vergibt man für die Nutzer (unter ***Credentials***) Passwörter (die man sich natürlich merkt) und ordnet ihnen (unter ***Role Mappings***) die entsprechenden Rollen zu:

Komponente	***Username***	***Assigned Role***
Dispatcher	`disp-user`	`/ttp-fhir/fhir/dispatcher`
gICS	`gics-user`	`/ttp-fhir/fhir/gics`
gPAS	`gpas-user`	`/ttp-fhir/fhir/gpas`
E-PIX	`epix-user`	`/ttp-fhir/fhir/e-pix`

Grundsätzlich können auch Nutzer, die bereits für die Keycloak-Absicherung der Frontends der THS Tools angelegt wurden, verwendet werden. In diesem Fall ist auf die Zuordnung der passenden Rollen (Assigned Role) zu achten.

Export des Realms

Es empfiehlt sich nun, die Konfiguration des Realms `ttp` mit ***Export*** (links unten in der Navigation) in eine **JSON**-Datei zu exportieren, um sie bei Bedarf wieder importieren zu können. Zu beachten ist dabei allerdings, dass die angelegten Nutzer und deren Konfigurationen dabei **nicht** mit exportiert werden.

Test und Benutzung von **Keycloak**

Für den Zugriff auf die durch **Keycloak** abgesicherten Komponenten benötigt man einen **Access-Token**. Den bekommt man mithilfe eines **POST**-Requests vom **Keycloak**-Server:

```
POST /auth/realms/ttp/protocol/openid-connect/token HTTP/1.1
Host: {KEYCLOAK_HOST}:{KEYCLOAK_PORT}
Content-Type: application/x-www-form-urlencoded
Content-Length: 128

client_id=ttp-fhir&username={USERNAME}&password=
{PASSWORD}&grant_type=password&client_secret={CLIENT_SECRET}
```

Hierbei wird, wie man sieht, auch das früher schon kopierte ***Secret*** des Clients benötigt. Die Variablen sind entsprechend zu ersetzen: beispielsweise für **gICS** wären

- der `{USERNAME}` dann `gics-user`,
- das `{PASSWORD}` das für `gics-user` vergebene Passwort,
- und `{CLIENT_SECRET}` das zuvor gemerkte ***Secret*** des Clients.

Mit `curl` könnte das etwa so aussehen:

```
curl -d 'client_id=ttp-fhir&username=gics-
user&password=fhirrocks&grant_type=password&client_secret=cd28481d-dd42-469a-b2f4-
803e9476172c' http://localhost:8081/auth/realms/ttp/protocol/openid-connect/token
```

...und die (hier etwas aufgehübschte) Antwort darauf wäre:

```
{
    "access_token" :
"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI1M3drWmNua0h...",
    "expires_in" : 300,
    "refresh_expires_in" : 1800,
    "refresh_token" :
"eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI0NGI0YTY5MC05Y...",
    "token_type" : "Bearer",
    "not-before-policy" : 0,
    "session_state" : "a343bc8f-7616-4deb-a635-293a91556b96",
    "scope" : "email profile"
}
```

Einrichtung des **TTP-FHIR-Gateways** für Keycloak-Authentifizierung

Der **Wildfly**-Server mit den **THS-Tools** und dem **TTP-FHIR-Gateway** muss mithilfe von Systemproperties für die Authentifizierung über **Keycloak** konfiguriert werden.

Diese Anbindung von TTP-FHIR Gateway an Keycloak ist derzeit über zwei Varianten möglich.

Variante 1: Direkte Anpassung der Wildfly-Konfiguration (standalone.xml)

In der Serverkonfiguration `{WILDFLY_SERVER_DIR}/standalone/configuration/standalone.xml` werden direkt hinter den Abschnitt `<extensions>...</extensions>` folgende Werte eingetragen:

```
    ...
    </extensions>
    <system-properties>
        <property name="ttp.fhir.keycloak.realm" value="ttp" />
        <property name="ttp.fhir.keycloak.clientId" value="ttp-fhir" />
        <property name="ttp.fhir.keycloak.baseUrl" value="http://{KEYCLOAK_HOST}:
{KEYCLOAK_PORT}/auth" />
        <property name="ttp.fhir.keycloak.secret" value="{CLIENT_SECRET}" />
        <property name="ttp.fhir.keycloak.enabled" value="true" />
        <!-- optional -->
        <property name="ttp.fhir.keycloak.role.gics.admin" value="
{NAME_KEYCLOAK_ROLE_GICS_ADMIN}" />
        <!-- optional -->
        <property name="ttp.fhir.keycloak.role.gpas.admin" value="
{NAME_KEYCLOAK_ROLE_GPAS_ADMIN}" />
        <!-- optional -->
        <property name="ttp.fhir.keycloak.role.epix.admin" value="
{NAME_KEYCLOAK_ROLE_EPIX_ADMIN}" />
    </system-properties>
```

Dabei sind der Host `{KEYCLOAK_HOST}`, der Port `{KEYCLOAK_PORT}` und das Client-Geheimnis `{CLIENT_SECRET}` entsprechend anzupassen.

Über das *optionale Property* z.B. im Fall von gICS `ttp.fhir.keycloak.role.gics.admin`, kann eine (werkzeug-spezifische) Admin-Rolle angebenen werden. Diese Admin-Rolle `{NAME_KEYCLOAK_ROLE_GICS_ADMIN}`, z.b. `/ttp-fhir/fhir/gics/admin` muss in Keycloak angelegt sein und dem entsprechenden Nutzer , z.b. `gics-user`, zugewiesen sein. Weitere Details zur Bedeutung der Admin-Role im Abschnitt *Bedeutung der TTP-FHIR Gateway Admin-Role*.

Variante 2: Anpassung der Wildfly-Variablen (ttp-fhir.env)

Die nötige Konfigurationsdatei `ttp-fhir.env` ist unter `compose-wildfly/ttp-fhir.env` abgelegt und muss gemäß der oben eingerichteten Keycloak-Konfiguration angepasst werden.

```
TTP_FHIR_KEYCLOAK_REALM=ttp
TTP_FHIR_KEYCLOAK_CLIENT_ID=ttp-fhir
TTP_FHIR_KEYCLOAK_SERVER_URL=http://{KEYCLOAK_HOST}:{KEYCLOAK_PORT}/auth/
TTP_FHIR_KEYCLOAK_CLIENT_SECRET={CLIENT_SECRET}
TTP_FHIR_KEYCLOAK_ENABLE=true
```

```
<!--optional-->
TTP_FHIR_KEYCLOAK_ROLE_GICS_ADMIN={NAME_KEYCLOAK_ROLE_GICS_ADMIN}
<!--optional-->
TTP_FHIR_KEYCLOAK_ROLE_GICS_ADMIN={NAME_KEYCLOAK_ROLE_GPAS_ADMIN}
<!--optional-->
TTP_FHIR_KEYCLOAK_ROLE_GICS_ADMIN={NAME_KEYCLOAK_ROLE_EPIX_ADMIN}
```

Dabei sind der Host `{KEYCLOAK_HOST}`, der Port `{KEYCLOAK_PORT}` und das Client-Geheimnis `{CLIENT_SECRET}` entsprechend anzupassen.

Per Default ist der Wert `TTP_FHIR_KEYCLOAK_ENABLE` auf `false` gesetzt.

Details zu den optionalen Properties ttp.fhir.keycloak.role.*.admin im vorstehenden Abschnitt.

Die Konfigurationsdatei `ttp-fhir.env` wird nach Bekanntmachung in der `docker-compose.yml` beim bauen der Docker-Container entsprechend vom Wildfly berücksichtigt. Dies ist bei Auslieferung des Docker-Compose Pakets bereits vorbereitet.

```
env_file:
      - ttp-fhir.env
      ...
```

Bedeutung der TTP-FHIR Gateway Admin-Role

Anmerkung: TTP-FHIR Gateway Admin-Roles sind für gICS, gPAS und E-PIX vorbereitet, werden derzeit nur von gICS verwendet

Die Angabe einer Admin-Role per Wildfly-Variable oder Wildfly-Konfiguration wird derzeit nur bei der Generierung von FHIR-ConsentPatient-Resourcen berücksichtigt. Da ein Consent durchaus mehrere SignerIds besitzen kann und diese Informationen nicht jedermann zugänglich sein sollten, kann der Umfang der im FHIR Consent Patient exportierten Identifier auf diese Weise reglementiert werden.

Variante 1: Verzicht auf Angabe einer Admin-Rolle

Ist keine Admin-Role konfiguriert oder ist die im Token angegebene Admin-Role fehlerhaft wird als Identifier der ConsentPatient-Resource **nur die SafeSignerId** verwendet. Die FHIR UUID des ConsentPatient entspricht somit der FHIR UUID der SafeSignerId (Details dazu im gics-Handbuch)

Variante 2: Angabe gültige AdminRolle

Ist in Keycloak eine Admin-Role konfiguriert und verweist das im Request verwendete Token korrekt auf diese Admin-Role, enthält die ConsentPatient-Resource **ALLE für den Consent relevanten SignerIds** als separate Identifier. Die FHIR UUID des ConsentPatient entspricht der FHIR UUID der SafeSignerId (Details dazu im gics-Handbuch).

Test und Benutzung des **TTP-FHIR-Gateways** mit Keycloak-Authentifizierung

Für die Benutzung des **TTP-FHIR-Gateway**, nun mit Authentifizierung über **Keycloak**, muss der aus dem Token-Request entnommene **Access-Token** als Authorization-Header `Authorization: Bearer {access_token}` bei jedem Request an das **TTP-FHIR-Gateway** im Header mitgegeben werden. Um die korrekte Funktion zu testen, kann man nun in einem **GET** -Request die Metadaten der FHIR-Schnittstelle (z.B. für **gICS**) anfordern:

```
GET /ttp-fhir/fhir/gics/metadata HTTP/1.1
Host: {FHIR_GATEWAY_HOST}:{FHIR_GATEWAY_PORT}
Authorization: Bearer {access_token}
```

Mit `curl` könnte das für **gICS** beispielsweise so aussehen:

```
curl -X GET http://localhost:8080/ttp-fhir/fhir/gics/metadata -H "Authorization:
Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI1M3drWmNua0h..."
```

Die erwartete (hier etwas aufgehübschte) Antwort darauf sollte etwa so aussehen:

```
{
  "resourceType": "CapabilityStatement",
  "status": "active",
  "date": "2021-06-03T20:54:47+02:00",
  "publisher": "Not provided",
  "kind": "instance",
  "software": {
    "name": "HAPI FHIR Server",
    "version": "5.0.0"
  },
  "implementation": {
    "description": "HAPI FHIR",
    "url": "http://localhost:8080/ttp-fhir/fhir/gics"
  },
  "fhirVersion": "4.0.1",
  "format": [
    "application/fhir+xml",
    "application/fhir+json"
  ],
  "rest": [
    {
      "mode": "server",
      "resource": [
        ...
      ],
      "operation": [
        {
          "name": "allConsentsForDomain",
          "definition": "http://localhost:8080/ttp-
fhir/fhir/gics/OperationDefinition/-s-allConsentsForDomain"
        },
```

```
          ...
          }
        ]
      }
    ]
  }
```

Wenn dann der **GET**-Request ohne gültigen Authorization-Header `Authorization: Bearer {access_token}`:

```
curl -X GET http://localhost:8080/ttp-fhir/fhir/gics/metadata
```

auch noch einen Fehler liefert:

```
Unauthorised access to protected resource
```

dann bedeutet das, dass die **Keycloak**-basierte Authentifizierung wie gewünscht funktioniert.

Credits 'Keycloak for TTP FHIR Gateway'

Implementation, documentation: P. Penndorf, F. M. Moser, M. Bialke, R. Schuldt

License

License: AGPLv3, https://www.gnu.org/licenses/agpl-3.0.en.html

Copyright: 2022 Trusted Third Party of the University Medicine Greifswald

Contact: https://www.ths-greifswald.de/kontakt/