Cookie Authentication and VOSI Capabilities/securityMethods

Juan González-Núñez, J. Salgado, R. Gutiérrez-Sánchez, JC. Segovia, J. Duran, E. Racero, M. Marcos, D. Baines, A. Mora, J. Bakker, B. Merín, C. Arviset

ESAC Science Data Centre (ESDC) - ESA

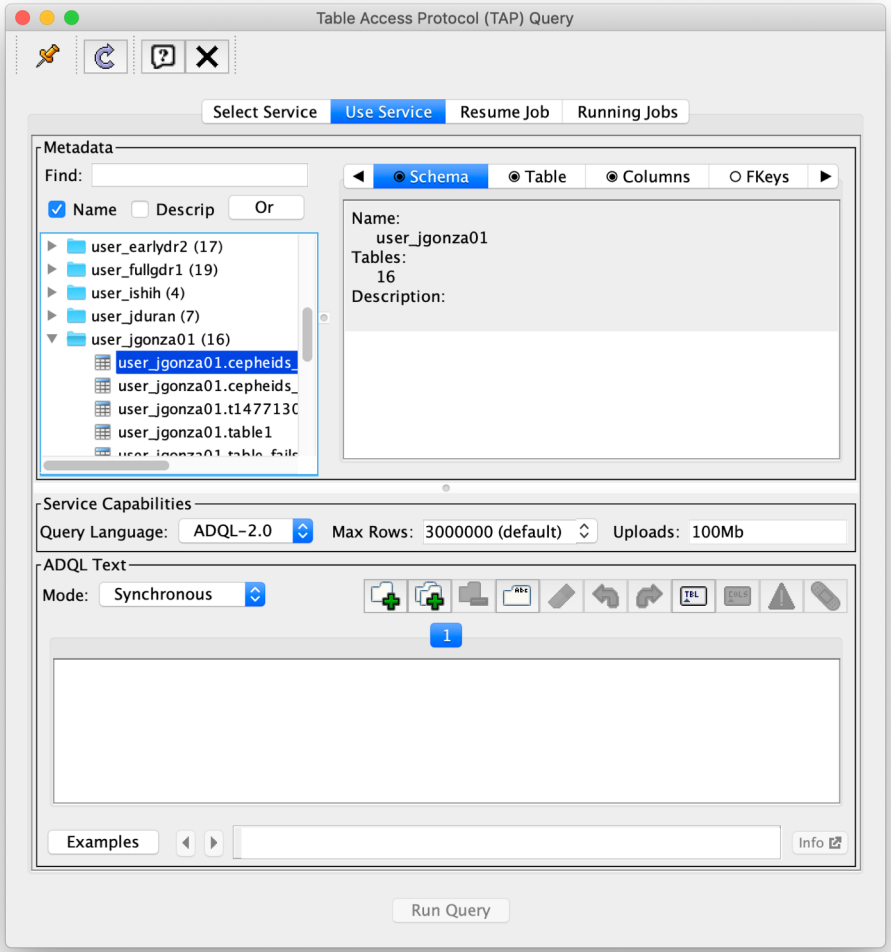
Topcat TAP11c & TAP+ auth

- Test of M. Taylor proposal in Paris 2019 Interop
- https://wiki.ivoa.net/internal/IVOA/InterOpMay2019DAL/tap11b.pdf
  - 11c TOPCAT with early implementation of Authentication
    - ftp://andromeda.star.bris.ac.uk/pub/star/topcat/pre/topcat-full_tap11c.jar
  - ESA Gaia archive with extended /capabilities response

```
−<interface role="std" xsi:type="vs:ParamHTTP">
    <accessURL use="base">https:/#URL#/tap-server/tap</accessURL>
    <securityMethod/>
    <securityMethod standardID="ivo://ivoa.net/sso#cookie"/>
  </interface>
```

Topcat TAP11c & TAP+ auth

- Retrieved archive session cookie:
  - curl -k -c cookies.txt -X POST -d username=USERNAME -d password=PASSWORD -L "https://#URL#/tap-server/login"
- Open Topcat and select 'VO->Table Acces Protocol (TAP) Query’.
- Select 'Select Service' tab, use 'https://#URL#/tap-server/tap' in 'TAP URL' textbox, select 'cookie' as authentication method.
- select 'data' and put the cookie we have retrieved using curl: JSESSIONID=xxxx



Some conclusions

- It works! Access to
  - general public archive data (DRx, external catalogues, etc)
  - User space data
  - Shared data from other users/groups
- Access process could be streamlined by extending the provision of metadata in /capabilities response
- We should indicate tools **how** to obtain the session cookie

securityMethod param extension

```
−<interface role="std" xsi:type="vs:ParamHTTP">
    <accessURL use="base">https:/#URL#/tap-server/tap</accessURL>
  −<securityMethod standardID="ivo://ivoa.net/sso#cookie">
      <param id="url" ucd="meta.ref.url" utype="Access.reference">https://#URL#/tap-server/login</param>
      <param id="method" ucd="meta.ref.method" utype="Request.method">POST</param>
      <param id="user" ucd="login.name" utype="Request.param">username</param>
      <param id="pwd" ucd="login.password" utype="Request.param">password</param>
      <param id="cookie" ucd="login.cookie" type="Response.cookie">JSESSIONID</param>
    </securityMethod>
  </interface>
```

securityMethod param extension

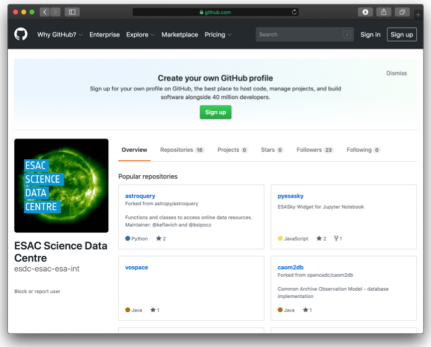
```
−<interface role="std" xsi:type="vs:ParamHTTP">
    <accessURL use="base">https:/#URL#/tap-server/tap</accessURL>
  −<securityMethod standardID="ivo://ivoa.net/sso#cookie">
    ➡ <param id="url" ucd="meta.ref.url" utype="Access.reference">https://#URL#/tap-server/login</param>
      <param id="method" ucd="meta.ref.method" utype="Request.method">POST</param>
      <param id="user" ucd="login.name" utype="Request.param">username</param>
      <param id="pwd" ucd="login.password" utype="Request.param">password</param>
      <param id="cookie" ucd="login.cookie" type="Response.cookie">JSESSIONID</param>
    </securityMethod>
  </interface>
```

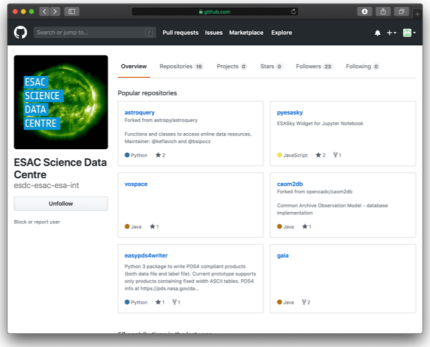
Separate login URL?

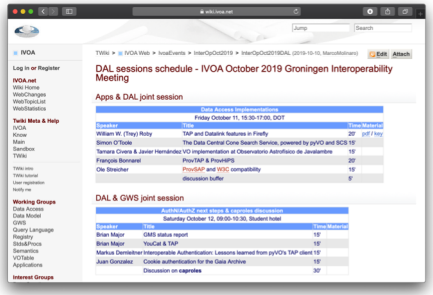
- In a general case, a single service may need to serve
  - Public data
  - Private or Proprietary data with complex sharing/grouping
  - Functionalities in the service that may change according to the user

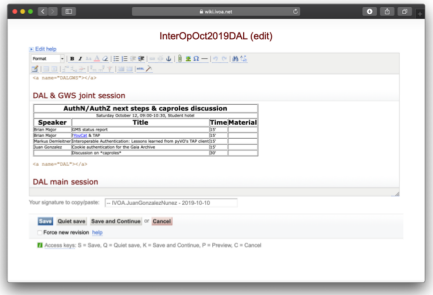
- We could force users to authenticate for any URL invocation and not serve data (even public) to unauthenticated users
  - But that is certainly not very open ☹

- What about 2 separate URLs, one for public, another enforcing authentication?









- Spec already covers the need for one single URL with anonymous + authenticated support
- Standard practice to offer public, unauthenticated services + authenticated capabilities after login without URL change
  - Eg. GitHUB or IVOA Wiki

securityMethod param extension

```
−<interface role="std" xsi:type="vs:ParamHTTP">
    <accessURL use="base">https:/#URL#/tap-server/tap</accessURL>
  −<securityMethod standardID="ivo://ivoa.net/sso#cookie">
    ➡ <param id="url" ucd="meta.ref.url" utype="Access.reference">https://#URL#/tap-server/login</param>
      <param id="method" ucd="meta.ref.method" utype="Request.method">POST</param>
      <param id="user" ucd="login.name" utype="Request.param">username</param>
      <param id="pwd" ucd="login.password" utype="Request.param">password</param>
      <param id="cookie" ucd="login.cookie" type="Response.cookie">JSESSIONID</param>
    </securityMethod>
  </interface>
```

securityMethod param extension

```
−<interface role="std" xsi:type="vs:ParamHTTP">
    <accessURL use="base">https:/#URL#/tap-server/tap</accessURL>
  −<securityMethod standardID="ivo://ivoa.net/sso#cookie">
    <param id="url" ucd="meta.ref.url" utype="Access.reference">https://#URL#/tap-server/login</param>
    <param id="method" ucd="meta.ref.method" utype="Request.method">POST</param>
    <param id="user" ucd="login.name" utype="Request.param">username</param>
    <param id="pwd" ucd="login.password" utype="Request.param">password</param>
    <param id="cookie" ucd="login.cookie" type="Response.cookie">JSESSIONID</param>
    </securityMethod>
  </interface>
```

Capabilities after authentication

- Several capabilities metadata may changes **after** authentication
  - Service limits?
    - retentionPeriod, executionDuration, outputLimit, uploadLimit
  - Additional query languages?
    - eg. posiblity to send native SQL queries for power users/admins
  - Further output formats?

- Declaration of the need to **reload** capabilities after authentication by implementing tools?

Questions / Discussion