Authentication in Web Services and TAP-1.1 specific issues

Patrick Dowler
Canadian Astronomy Data Centre

VOSI-capabilities 101

- a web service endpoint for a self-describing service
  - e.g. http://example.net/service/capabilities
  - (contains 1+) capability standardID: what feature is this?
  - (contains 1+) interface: a single callable endpoint
  - contains 1 accessURL
  - contains 1 securityMethod* (optional)

```
<capability standardID="vos://cadc.nrc.ca~vospace/CADC/std/LOGGING#logControl-1.0">
    <interface xsi:type="vs:ParamHTTP" role="std" version="1.0">
      <accessURL use="full">
              https://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/ams/logControl
       </accessURL>
      <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate" />
     </interface>
</capability>
```

Web Service A&A @ CADC

- 23 RESTful web services in operation (CADC + CANFAR)
  - 15 are IVOA standard services
  - 8 are custom services
- all of these use VOSI-capabilities
- all of these have at least one capability which describes authenticated access (~45 capabilities)

- clients consult a runtime-registry to find the capabilities
  - optimised for resourceID → capabilities URL
- clients read the capabilities document and look for the combination of {standardID,securityMethod} that match:
  - the feature they want to invoke
  - the credentials they want to use to authenticate
- @CADC: {resourceID,standardID,securityMethod} → {accessURL} happens several times per request & millions of times per day

TAP and Authentication

- VOSI-capabilities / VOResource model is that a capability is a single feature
- In TAP-1.0, we specified relative names for the endpoints:
  - /availability
  - /capabilities
  - /tables
  - /async
  - /sync
- BUT we specified one standardID for the base URL
  - clients have to append the specified names
  - auth methods that use alternate path names not feasible
- TAP-1.0 doesn’t play nice with all securityMethod(s)
- TAP-1.0 over-specified how accessURL(s) should look (IMO)
- TAP-1.1 must support authentication and must provide a good backwards-compatible experience for older client s/w

TAP and Authentication

- prototype #1: one capability for each securityMethod

- pros:
  - none
- cons:
  - naive client that assumed one anonymous capability per standardID would fail or depend on ordering
  - lots of redundancy in VOSI-capabilities documents
  - inside-out with respect to the VOResource model where securityMethod is at the leaf
  - make an assumption that multiple capability(s) with the same standardID are the same underlying thing rather than different things…
  - … if that was specified, it would restrict how people deploy services

TAP and Authentication

- prototype #2: separate standardID for sync and async
  ivo://ivoa.net/std/TAP#sync-1.1
  ivo://ivoa.net/std/TAP#async-1.1
  SODA-1.0 defines #sync-1.0 and #async-1.0
  VOSpace-2.1 defines #transfers and #sync-2.1

- pros:
  - did not break any old clients (we had this in operational use for years)
  - matches design of VOResource
  - backwards compatible records simple
  - allows for different TAPRegExt metadata (e.g. optional features, limits) in sync and async
- cons:
  - duplicates TAPRegExt info in sync and async
  - makes example RegTAP queries return different (more) results

TAP and Authentication

- prototype #3: separate interface type for sync and async
- lookup becomes:

{resourceID,standardID,interfaceType,securityMethod} → accessURL

- pros:
  - does not break any old clients (in operational use for a few months)
  - backwards compatible records possible
- cons:
  - backwards compatible records are subtle
  - set of interface(s) mixes base (client appends resource name) and full (accessURL includes resource name)
  - makes example RegTAP queries return different (more) results that users have to grok

TAP and Authentication

- approach #1: it’s horrible and it breaks stuff

- approach #3 works, BUT: introduces subtle use of interface types and mixed interface style in a single capability

- approach #2: separate #sync-1.1 and #async-1.1
  - matches the VOResource/VOSI-capabilities design
  - works the same way as all other IVOA services
- I am convinced that:
  - we made a mistake in TAP-1.0 when we specified one standardID for two features
  - when we make mistakes we have to admit it and stop doing it in order to improve out standards
  - sometimes that makes things a little more complex