TAP 1.1 Authentication: Implementation in TOPCAT/STILTS

Mark Taylor (Bristol)

DAL/Apps/GWS/Reg session
IVOA Interop
Paris

15 May 2019

$Id: tap11b.tex,v 1.10 2019/05/10 13:08:03 mbt Exp $

Outline

- History
- TAP 1.1 Authentication: Capabilities refresher
- TOPCAT/STILTS implementation details
- Conclusions/Recommendations

History

College Park, Nov 2018:

- PR-TAP-1.1-20181024
- TOPCAT/STILTS experimental version topcat-full_tap11.jar
- Interrogates `/capabilities`, offers user choice of securityMethod-specific endpoint bundle
- User supplies credentials interactively (BasicAA) or by system property (tls-with-certificate)
- Application code accesses URLs without auth-specific measures
- Actual authentication handled at JRE level
  (BasicAA in J2SE, tls-with-certificate using Brian's SSL libraries)
- Service-specific credentials only possible for BasicAA
- Tested against CADC services in late 2018 (no others available)
- Also some other TAP1.1 functionality, including `taplint` updates
- TAP concerns (Section 2.4 "VOSI-Capabilities"):
  - hard to extract TAP service bundle from `capabilities` document
  - there is no TAP "base URL"
  - `mirrorURL`s very difficult to handle
  - relies on draft UWSRegExt Note
- Details reported in College Park DAL Session

Present

Paris, May 2019:

- PR-TAP-1.1-20190420
- TOPCAT/STILTS experimental version topcat-full_tap11b.jar
- Interrogates `/capabilities`, offers user choice of securityMethod-specific endpoint bundle
- SecurityMethod+Service-specific credentials sought when user chooses securityMethod
- User supplies credentials interactively on request or by system property
- All TAP URLs accessed using securityMethod-specific HTTP requests
- Tested against CADC services in Apr/May 2019 (no others available)
- No other TAP1.1 functionality (no `taplint` updates)
- `mirrorURL`s *should* be OK, but not implemented yet
- TAP concerns (Section 2.4 "VOSI-Capabilities"):
  - Marginalises BasicAA, but other securityMethods not usable yet; hard for services to request simple user/password entry
  - Known and maybe unknown issues with different securityMethods coexisting on a single URL

TAP 1.1 Capabilities

PR-TAP-1.1-20190420 Sec 2.4: *"One URL to rule them all"*

- Capabilities declares a TAP *Base URL*, with associated *securityMethod*s:

```
<vosi:capabilities>

  <!-- base URL for clients that apppend /async, /sync, etc -->
  <capability standardID="ivo://ivoa.net/std/TAP">
    <interface xsi:type="vs:ParamHTTP" role="std" version="1.1">
      <accessURL use="base">https://example.net/srv</accessURL>
      <!-- anonymous or cookie or client certificate -->
      <securityMethod/>
      <securityMethod standardID="ivo://ivoa.net/sso#cookie"/>
      <securityMethod standardID="ivo://ivoa.net/sso#tls-with-certificate"/>
      <!-- useful TAPRegExt stuff -->
    </interface>
  </capability>

  <!-- blah blah blah -->
</vosi:capabilities>
```

- TOPCAT ignores everything else!
  - ▷ There is additional relevant information in capabilities, e.g. presence/absence and securityMethods for optional endpoints (`/tables`, `/examples`), but I just try them and check for a 404 anyway
- It works.

TAP 1.1 Capabilities

PR-TAP-1.1-20190420 Sec 2.4: *"One URL to rule them all"*

- Pro:
  - Much easier than PR-TAP-1.1-20181024 UWSRegExt-based solution
    - ... for client code to work out what endpoints to use
    - ... for users to specify a service
- Concerns:
  - Different securityMethods on the same URL may interfere with each other
    - Basic/Bearer *vs.* anonymous:
      - can't easily coexist, since an anonymous access will provoke a 401 challenge (looks like the pre-challenge phase of an RFC7235 authenticated access)
      - You could have challenge-less BasicAA, and TOPCAT would work with it, but it might(?) be hard to configure in web servers.
    - Cookie *vs.* anonymous:
      - if you accidentally send the wrong cookie you may think you're authenticated but you aren't — there's no way to tell
    - Others??
  - Currently hard for services to provide simple Username/Password authentication
    - BasicAA marginalised, but other user/pass securityMethods not yet usable

TOPCAT Behaviour

TOPCAT TAP service interaction overview:

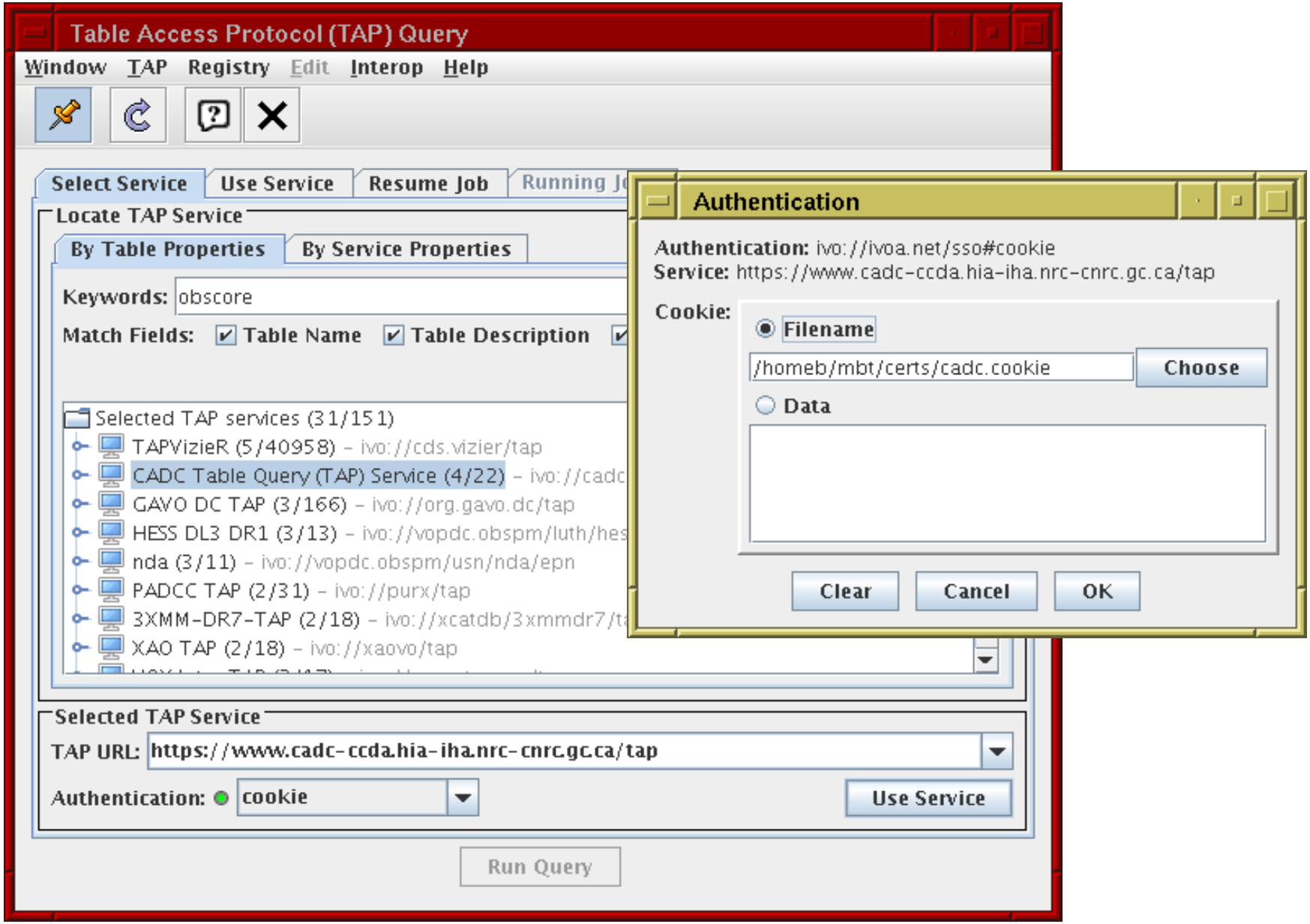
- User selects TAP service
  - ▷ usually from registry-supplied list
  - ▷ but can specify custom URL

  ⟶ baseURL defined
- TOPCAT asynchronously loads {Base-URL}`/capabilities`
- Identifies securityMethod options listed in capabilities document
- Lists all securityMethods in **Authentication** selection box (unauth is default if available)
- User may select a non-default securityMethod
- User hits **Use Service** button, defining currently-selected securityMethod

  ⟶ securityMethod defined
- TOPCAT acquires and caches credentials for (securityMethod, BaseURL) pair
  - ▷ from system properties if configured (currently per-JRE not per service)
  - ▷ from GUI/CLI user interaction otherwise (per service)

  ⟶ credentials defined
- TOPCAT tries to test credentials (short sync query, reject on 401/403/SSLException)
- All subsequent interactions with that service use credentials (this touches a lot of code)

TOPCAT Example



STILTS Behaviour

STILTS `tapquery` and other TAP clients:

- User optionally sets credentials using system properties:
  - BasicAA: `star.basicaa.user`, `star.basicaa.password`
  - Cookie: `star.auth.cookie` (literal or file)
  - TLS-with-certificate: `star.cert.pem` (file)
- User sets securityMethod using `interface` parameter

  `interface=tap1.0`
  no authentication, TAP 1.0 protocol *(default)*

  `interface=tap1.1`
  no authentication, TAP 1.1 protocol

  `interface=cap:xxx`
  read `/capabilities` document and use securityMethod/@standardID (approximately) matching "`xxx`"; lists available options if not found

  `interface=auth:xxx`
  user securityMethod matching "`xxx`" without reading capabilities
- STILTS acquires credentials using system properties if present; else prompts on console

STILTS Examples

Existing securityMethod, supply credentials interactively:

```
% stilts tapquery
         interface=cap:cookie
         tapurl=https://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap
         adql='SELECT TOP 3 * FROM ivoa.obscore'
Authenticate for cookie:https://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap
Enter Cookie literal value or @<filename>
Cookie: @/tmp/cadc.cookie
...
```

Non-existent securityMethod:

```
% stilts tapquery
         interface=cap:OAuth
         tapurl=https://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap
         adql='SELECT TOP 3 * FROM ivoa.obscore'
Error: Bad value for parameter interface: SecurityMethod-id "OAuth" not available;
service offers <anonymous>, "cookie", "tls-with-certificate"
Usage: interface=tap1.0|tap1.1|auth:<security-method>|cap:<security-method>
```

Known securityMethod, supply credentials by system properties, curl logging:

```
% stilts -verbose -verbose
         -Dstar.basicaa.user=tap -Dstar.basicaa.password=tap
         tapquery
         interface=auth:BasicAA
         tapurl=http://www.g-vo.org:8080/tap
         adql='SELECT COUNT(*) FROM tap_schema.tables' sync=true
...
CONFIG: Doing something like: curl --url http://www.g-vo.org:8080/tap/sync --location
       --user tap:<PASSWORD> --compress --form REQUEST=doQuery --form LANG=ADQL
       --form QUERY='SELECT COUNT(*) FROM tap_schema.tables'
...
```

Implementation Status

Done:

- Authenticated access to TAP 1.1 services
- SecurityMethods: BasicAA, tls-with-certificate, Cookie *(can add more)*
- Tested/working with CADC
- Bonus feature: `-verbose -verbose` shows equivalent `curl(1)` command

Not done yet:

- Persisted per-service credential storage (`.netrc` or similar?)
- Smart(?) per-realm/per-domain credential caching
- SecurityMethods: OAuth (+others?)
- `mirrorURL`s
- Authentication for non-TAP (and pre-1.1 TAP) services
  - ▷ BasicAA used to work but may be broken now — needs testing
- TAP 1.1 support apart from authentication
  - ▷ `taplint` support was done but tied to implementation of previous PR
- Some other things...

No idea how to do:

- Advise user where to get credentials
- Make a good guess for default/suggested auth method

Implementation Notes

Design decisions

- Capabilities doc downloaded asynchronously
  - to avoid negative impact on (majority) unauthenticated services
  - ... but user might select service before auth methods are listed
  - mitigated by graphical indicator
- Use same credentials for all endpoints of service
  (e.g. even `/tables`, even if declared without securityMethod)
- As implemented, not hard to add new SecurityMethods
  - ... but I don't know where to start with OAuth

Gripes

- Custom authentication
  - JRE contains standard mechanisms for e.g. BasicAA, cookies, default SSL certificates
  - Service-specific authentication requires turning them off and handling it all explicitly
  - ... which is a lot of code to modify
  - ... unmodified connections no longer benefit from e.g. standard BasicAA handling
- No obvious way to check whether credentials are correct or not
  - Would like to inform the user at an early stage:
    Am I authenticated? (as who?) Am I authorized? (for what?)

Recommendations

Missing components in standards suite:

- Some way to find out where/how to acquire credentials
  - BasicAA requires Username/Password — obvious what to do
  - All other securityMethods need some external input to authenticate
  - Required: additional securityMethod-specific information in capabilities
    e.g. link to external SSO login service
  - Current workaround: ask Brian how to get a cookie etc
- Some way to check submitted credentials
  - Would like to inform user at an early stage whether login was successful
    - Am I authenticated? With what identity?
    - Am I authorised? To do what?
  - Otherwise user finds out at at an unhelpful stage
    - ... after typing in and submitting a ADQL query
    - ... never, but unauthenticated access gives them a different result than if they were authenticated
  - Required: new AuthTest service *or capability or interface or endpoint ...*
  - Current workaround: make small sync TAP query with available credentials, but that could be slow, and doesn't test silent degradation to coexisting anonymous service

Software Availability

Experimental!

- Still under development
- No intention to include in public release before TAP 1.1 is more stable
- If you have TAP 1.1 authenticated services, I'd like to test them

`ftp://andromeda.star.bris.ac.uk/pub/star/topcat/pre/topcat-full_tap11b.jar`

Conclusions

- Return of TAP Base URL is a Good Thing
- Multiple securityMethods on single URL may cause problems
  - some discovered, some maybe not
- Missing standards components:
  - Additional securityMethod metadata
  - Authentication test service
- Implementation appears to work
  - But this is only one client talking to one service, tested by one person.
  - More implementation experience is required