Authentication: A Client [G]UI Perspective

Mark Taylor (Bristol)

A&A F2F Meeting
ASTERICS (DADI/OBELICS)
Trieste

January 2019

`$Id: clientauth.tex,v 1.8 2019/01/25 14:42:57 mbt Exp $`

Outline

- Client requirements for authentication
- Prototype TAP implementation in TOPCAT & STILTS
- Summary/Issues

Requirements

Client Authentication Requirements:

- Determine what service interface to use
- Acquire suitable credentials
- Access authenticated service with credentials

General Principles

Central problem of interactive software:

- The software is there to help the user achieve his/her aims
- *The software don't know in advance what the user is trying to do*
- Maybe the user doesn't either

UI design principles:

- *Users are not (in general) experts.* Give them all the help you can.
- Require minimal user effort
- Make it obvious to the user what options there are, what actions are required
- Try not to offer too many options
- Don't make the (unauthenticating) many suffer to support the needs of the (authenticating) few

Priorities:

- Complicating implementation to support A&A isn't too bad
- Complicating the UI (especially for unauthenticated access) to support A&A is bad

Service Identification

Determine what service interface to use

- "service interface" = data-service + authentication-option (+ mirror-specification? + ...)
- Typically corresponds to a *bundle* of endpoints, not just one URL
  - ▷ e.g. for TAP: `/sync`, `/async`, `/capabilities`, `/tables`, `/examples`
- Interactive/GUI client:
  - ▷ General approach:
    - ○ Identify a list of options
    - ○ Present them to the user
    - ○ Get the user to choose one
  - ▷ Probably makes sense to break it down hierarchically:
    1. select data service (query e.g. registry)
    2. select authentication option (query e.g. `/capabilities` document)
    3. select mirror
  - ▷ User must also be able to specify *custom* (e.g. unregistered) service interfaces
- Command-line/batch client:
  - ▷ There must be some way to specify service interface using a string or strings, something like a base URL, cut'n'pasteable
    (also required for *custom* specification in interactive client, see above)

Service Identification: User Interaction

Give the user as much help as possible:

- Only offer appropriate options
  - ▷ Try to avoid services/interfaces the user *cannot* use?
  - ▷ List should not be too big
- Offer a default if possible:
  - ▷ Unauthenticated?
  - ▷ Most commonly used?
  - ▷ Suitable for less-expert users?
- Prioritise most appropriate ones:
  - ▷ for which credentials are already known?
  - ▷ for which required authentication is known possible?
- Use comprehensible language
  - ▷ e.g. choice "`BasicAA`" vs. "`tls-with-certificate`" is not ideal
- Don't ask the same question twice

Credential Acquisition

Acquire Credentials

- Know what credentials are required, if any
  - ▷ Name + password? Certificate file?
  - ▷ Has the user already supplied them during an earlier interaction?
    May not be easy to determine (which services use same credentials?)
  - ▷ Not always possible/necessary at application level;
    may be service-specific or only configurable outside application (e.g. JVM flags)
- Make it easy for the user to specify them
  - ▷ (G)UI design
  - ▷ Client application documentation
  - ▷ Service documentation (+ pointers from client?)
  - ▷ Persistence
    - ◦ Remember supplied credentials per-service for next time
    - ◦ Allow pre-emptive user setup: command-line args, config file
    - ◦ Standards for cross-application well-known config (`~/.netrc` file?)
- Inform user of authentication success/failure
  - ▷ Not always easy:
    - ◦ may only know if successful when service interaction is attempted
    - ◦ may be difficult to distinguish authentication failures from unrelated errors

Service Interaction

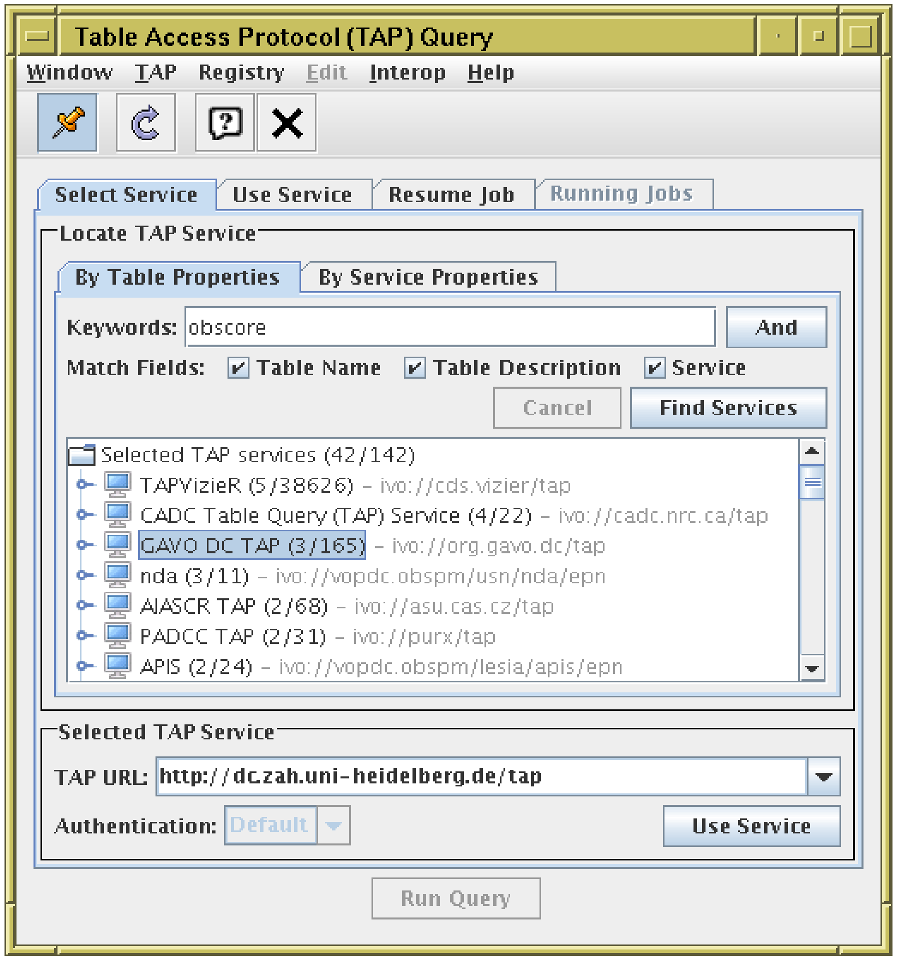
Access authenticated services with credentials

- May need to rewrite all HTTP interaction with authentication support...
- ... or may be managed at language/library level
  - ▷ JRE supports JVM-level configuration for e.g. BasicAA, TLS-with-certificate
- Standard library code might be useful/appropriate
  - ▷ More experience required to understand what would be worth providing
- SSO defines 7 securityMethod options; which ones should clients support?
  - ▷ Current/expected use by services: `BasicAA`, `tls-with-certificate`, `OAuth`?

TOPCAT TAP UI

TOPCAT TAP client supports authenticated use

- New **Authentication** selector below TAP URL selector
- Populated asynchronously when TAP URL is selected (or entered by hand)
- Select a non-default value if you like
- SecurityMethod-specific endpoint bundle is selected accordingly



Keywords: obscore

Match Fields: ✓ Table Name ✓ Table Description ✓ Service

```
Selected TAP services (42/142)
  TAPVizieR (5/38626) – ivo://cds.vizier/tap
  CADC Table Query (TAP) Service (4/22) – ivo://cadc.nrc.ca/tap
  GAVO DC TAP (3/165) – ivo://org.gavo.dc/tap
  nda (3/11) – ivo://vopdc.obspm/usn/nda/epn
  AIASCR TAP (2/68) – ivo://asu.cas.cz/tap
  PADCC TAP (2/31) – ivo://purx/tap
  APIS (2/24) – ivo://vopdc.obspm/lesia/apis/epn
```

TAP URL: http://dc.zah.uni-heidelberg.de/tap

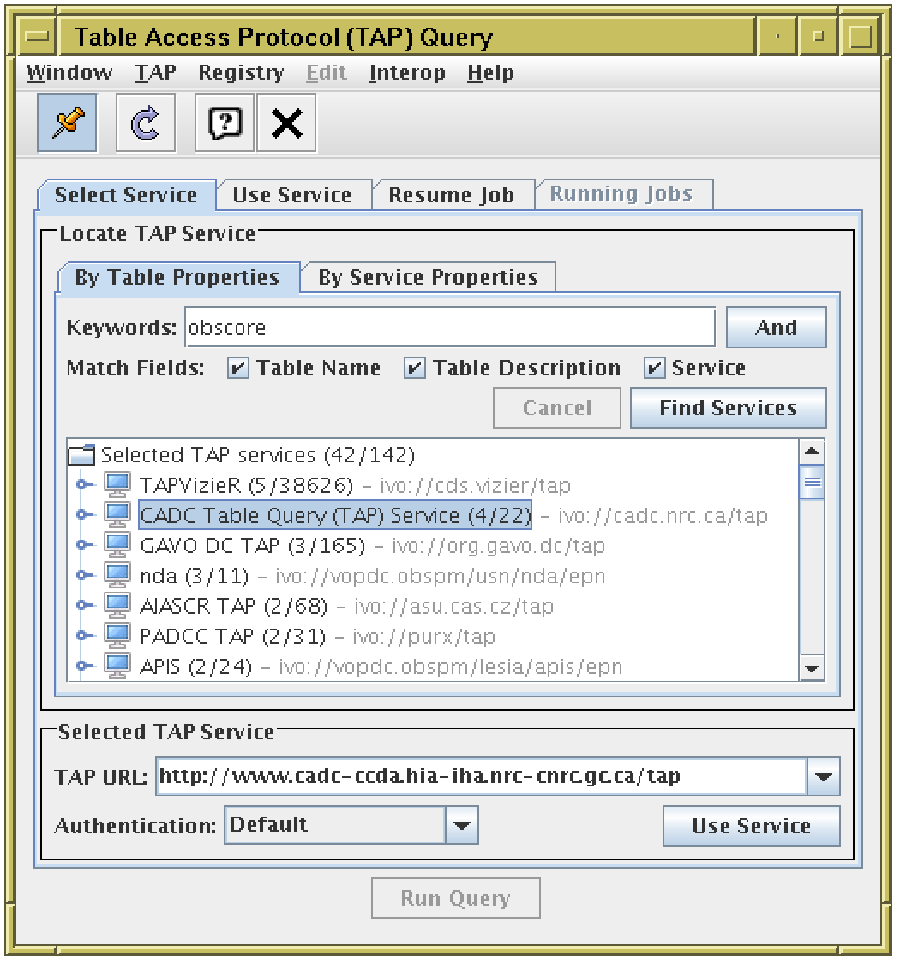
Authentication: Default

ftp://andromeda.star.bris.ac.uk/pub/star/topcat/pre/topcat-full_tap11.jar

TOPCAT TAP UI

TOPCAT TAP client supports authenticated use

- New **Authentication** selector below TAP URL selector
- Populated asynchronously when TAP URL is selected (or entered by hand)
- Select a non-default value if you like
- SecurityMethod-specific endpoint bundle is selected accordingly



Keywords: obscore

Match Fields: ✓ Table Name ✓ Table Description ✓ Service

Selected TAP services (42/142)
- TAPVizieR (5/38626) – ivo://cds.vizier/tap
- CADC Table Query (TAP) Service (4/22) – ivo://cadc.nrc.ca/tap
- GAVO DC TAP (3/165) – ivo://org.gavo.dc/tap
- nda (3/11) – ivo://vopdc.obspm/usn/nda/epn
- AIASCR TAP (2/68) – ivo://asu.cas.cz/tap
- PADCC TAP (2/31) – ivo://purx/tap
- APIS (2/24) – ivo://vopdc.obspm/lesia/apis/epn

TAP URL: http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap

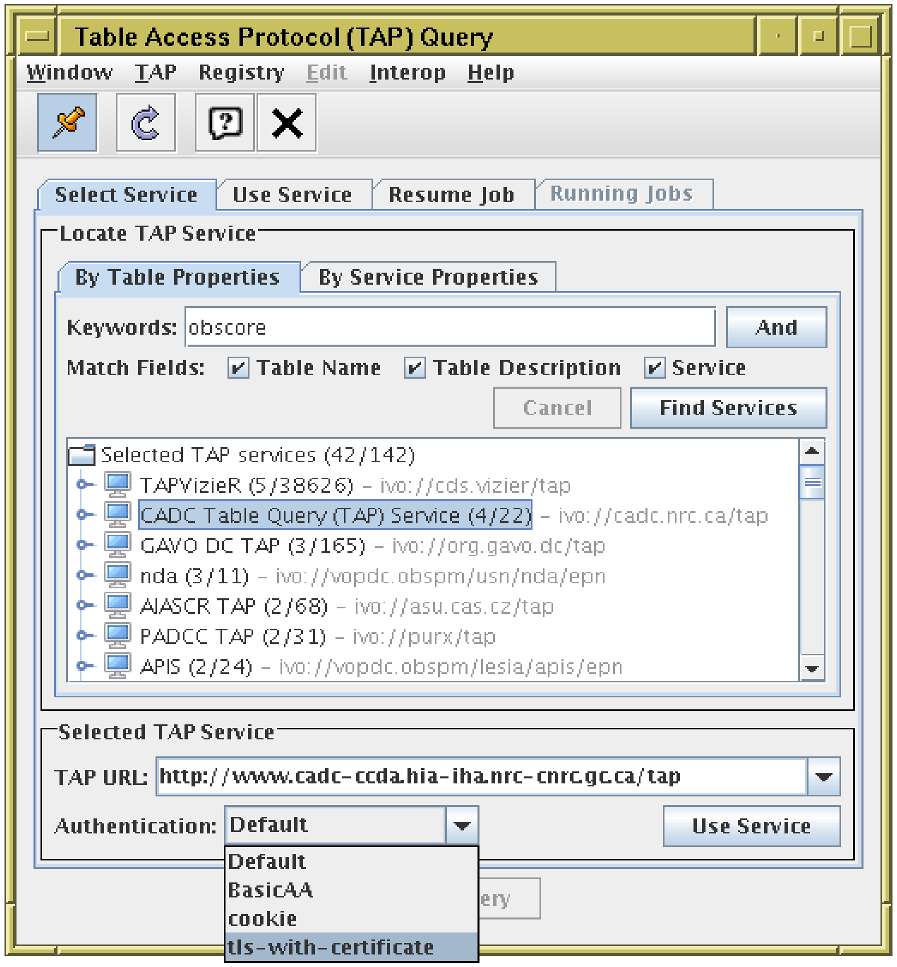
Authentication: Default

ftp://andromeda.star.bris.ac.uk/pub/star/topcat/pre/topcat-full_tap11.jar

TOPCAT TAP UI

TOPCAT TAP client supports authenticated use

- New **Authentication** selector below TAP URL selector
- Populated asynchronously when TAP URL is selected (or entered by hand)
- Select a non-default value if you like
- SecurityMethod-specific endpoint bundle is selected accordingly



Selected TAP services (42/142)
TAPVizieR (5/38626) – ivo://cds.vizier/tap
CADC Table Query (TAP) Service (4/22) – ivo://cadc.nrc.ca/tap
GAVO DC TAP (3/165) – ivo://org.gavo.dc/tap
nda (3/11) – ivo://vopdc.obspm/usn/nda/epn
AIASCR TAP (2/68) – ivo://asu.cas.cz/tap
PADCC TAP (2/31) – ivo://purx/tap
APIS (2/24) – ivo://vopdc.obspm/lesia/apis/epn

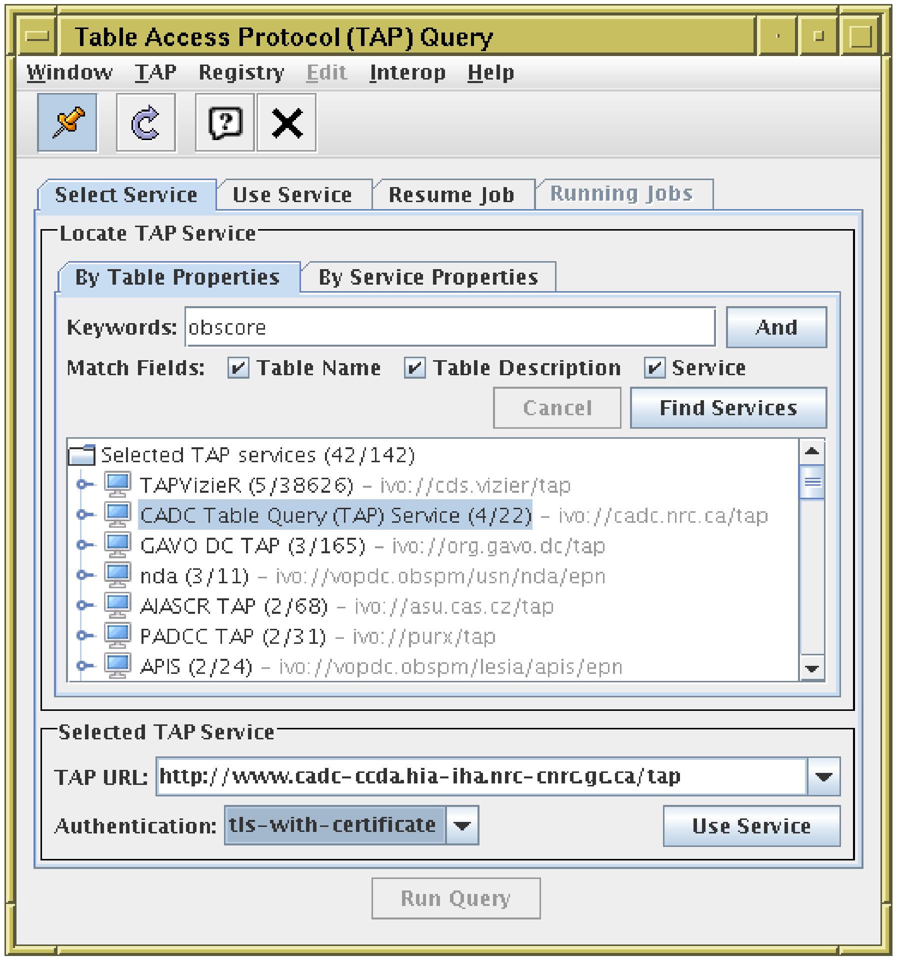
TAP URL: http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap
Authentication: Default

ftp://andromeda.star.bris.ac.uk/pub/star/topcat/pre/topcat-full_tap11.jar

TOPCAT TAP UI

TOPCAT TAP client supports authenticated use

- New **Authentication** selector below TAP URL selector
- Populated asynchronously when TAP URL is selected (or entered by hand)
- Select a non-default value if you like
- SecurityMethod-specific endpoint bundle is selected accordingly



Keywords: obscore

Match Fields: ✓ Table Name ✓ Table Description ✓ Service

Selected TAP services (42/142)
- TAPVizieR (5/38626) – ivo://cds.vizier/tap
- CADC Table Query (TAP) Service (4/22) – ivo://cadc.nrc.ca/tap
- GAVO DC TAP (3/165) – ivo://org.gavo.dc/tap
- nda (3/11) – ivo://vopdc.obspm/usn/nda/epn
- AIASCR TAP (2/68) – ivo://asu.cas.cz/tap
- PADCC TAP (2/31) – ivo://purx/tap
- APIS (2/24) – ivo://vopdc.obspm/lesia/apis/epn

TAP URL: http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap

Authentication: tls-with-certificate

ftp://andromeda.star.bris.ac.uk/pub/star/topcat/pre/topcat-full_tap11.jar

STILTS TAP UI

STILTS TAP clients support authenticated use

- New parameter `interface` for TAP client commands
- Affected commands: `tapquery`, `tapskymatch`, `taplint`
- Options:

  `interface=tap1.0`
  use standard TAP 1.0 endpoints and TAP 1.0 protocol *(default)*

  `interface=tap1.1`
  use standard TAP 1.0 endpoints and TAP 1.1 protocol

  `interface=auth:xxx`
  read `/capabilities` document and find endpoints
  with securityMethod/@standardID (approximately) matching "`xxx`"
  (unknown `xxx` e.g. "`auth:what`" lists available securityMethods)

  `interface=unauth`
  read `/capabilities` document and find endpoints
  with no declared `securityMethod`

- Examples:
  - ▷ `taplint interface=unauth tapurl=...`
  - ▷ `tapquery interface=auth:tls-with-certificate tapurl=...`

ftp://andromeda.star.bris.ac.uk/pub/star/stilts/pre/stilts_tap11.jar

Service Interaction Implementation

Current behaviour (prototype)

- Perform default or user-controlled JVM auth configuration on startup
- Identify service bundle for authentication options
- Talk to endpoints as normal — no change in most application code
  - ▷ JVM-level code triggers auth negotiation for some securityMethods:
    - BasicAA: HttpURLConnection calls Authenticator static methods to request username/password; from popup window or `star.basicauth.user`/`star.basicauth.password` sys properties
    - TLS-with-certificate: user can install cert for all HTTPS communications using `star.cert.pem` sys property *(thanks Brian!)*
    - More details in College Park DAL talk
  - ▷ Other security methods (e.g. OAuth) not currently supported

Possible future improvements

- More/better options to supply authentication tokens
  - ▷ Smarter user interaction based on selected authentication method (e.g. ask for certificate in case of tls-with-certificate)
  - ▷ Possibility for per-service rather than JVM-wide certificate config
  - ▷ `.netrc` file?
- Support other securityMethods (OAuth for LSST)
- Authenticated Cone Search, SIA, SSA, ...?

Summary

What does client software need from the VO authentication ecosystem?

- A way to find out what auth and unauth services are available (registry)
- A way to specify custom auth/unauth services to applications (base URL)
- A way to determine endpoint bundles per authentication method (capabilities structure)
- Rules for talking to authenticated services (SSO)
- Conventions or standards for users to supply authentication tokens (certificate serialization standards, .netrc files, ??)
- Library support for talking to authenticated services (language-specific; VO client libraries?)

Issues

- Identifying/specifying service-interface *bundles*
  - PR-TAP-1.1-20181024 capabilities: not very nice, but it can be done
    - ▷ Complicated rules in PR required to unflatten capability/interface list
    - ▷ TAP-1.0 fallbacks required to support custom services
    - ▷ Mirror support would require some rules relating to `mirrorURL/@title`
    - ▷ Current text relies on new draft UWSRegExt, but later proposal avoids that
  - Markus's `vs:DALIinterface` suggestion would be much cleaner
    - ▷ Base URL with endpoints at well-known sub-locations (like TAP 1.0)
    - ▷ ... but not everybody likes it
- Determine if available credentials are suitable for a given service
  - Would be nice to tell the user: *"you are authenticated for these services"*
  - ... but I don't see how
- SecurityMethod-specific authentication code
  - Some implementation work to be done
  - not clear how much, but likely to be tractable