TAP 1.1 Authentication in TOPCAT/STILTS

Mark Taylor (Bristol)

with input from Markus Demleitner (ARI)

IVOA Interop
College Park MD, USA

9 November 2018

• TAP 1.1 authentication refresher
• Prototype implementation in TOPCAT + STILTS
  • Client behaviour
  • Implementation details
    ▶ Overview
    ▶ By security method
• Complaints about TAP 1.1 auth specification
Authentication Options in TAP 1.1 PR

PR-1.1-TAP-20180830

- Section 2: Declared resource requirements

<table>
<thead>
<tr>
<th>resource type</th>
<th>resource name</th>
<th>required</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAP-sync</td>
<td>/sync</td>
<td>must</td>
</tr>
<tr>
<td>TAP-async</td>
<td>/async</td>
<td>must</td>
</tr>
<tr>
<td>TAP-sync</td>
<td>service specific</td>
<td>may (alternate authentication method)</td>
</tr>
<tr>
<td>TAP-async</td>
<td>service specific</td>
<td>may (alternate authentication method)</td>
</tr>
<tr>
<td>VOSI-availability</td>
<td>service specific</td>
<td>must (should be anonymous)</td>
</tr>
<tr>
<td>VOSI-availability</td>
<td>service specific</td>
<td>may (alternate authentication method)</td>
</tr>
<tr>
<td>VOSI-capabilities</td>
<td>/capabilities</td>
<td>must (must be anonymous)</td>
</tr>
<tr>
<td>VOSI-tables</td>
<td>/tables</td>
<td>should</td>
</tr>
<tr>
<td>VOSI-tables</td>
<td>service specific</td>
<td>may (alternate authentication method)</td>
</tr>
<tr>
<td>DALI-examples</td>
<td>/examples</td>
<td>should</td>
</tr>
<tr>
<td>DALI-examples</td>
<td>service specific</td>
<td>may (alternate authentication method)</td>
</tr>
</tbody>
</table>

- Capabilities document may declare multiple interface elements with different securityMethod/@standardID attributes
PR-1.1-TAP-2018030

- Section 2.4: Example capabilities document:

```xml
<vosi:capabilities ...>

<!-- TAP sync/async -->
<capability standardID="ivo://ivoa.net/std/TAP">
  <interface xsi:type="urx:Sync" role="std" version="1.1">
    <accessURL use="base">http://example.net/myTAP/sync</accessURL>
    <!-- no declared securityMethod -->
  </interface>
<capability standardID="ivo://ivoa.net/std/TAP">
  <interface xsi:type="urx:Sync" role="std" version="1.1">
    <accessURL use="base">https://example.net/myTAP/auth-sync</accessURL>
    <securityMethod standardID="ivo://ivoa.net/sso#BasicAA"/>
  </interface>
<capability>

<!-- VOSI tables -->
<capability standardID="ivo://ivoa.net/std/VOSI#tables-1.1">
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1">
    <accessURL use="base">http://example.net/myTAP/tables</accessURL>
    <!-- no declared securityMethod -->
  </interface>
<capability standardID="ivo://ivoa.net/std/VOSI#tables-1.1">
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.1">
    <accessURL use="base">https://example.net/myTAP/auth-tables</accessURL>
    <securityMethod standardID="ivo://ivoa.net/sso#BasicAA"/>
  </interface>
<capability>

...</vosi:capabilities>
```
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
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
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
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
STILTS 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”
  
  - `interface=unauth`
    - read `/capabilities` document and find endpoints
      with no declared `securityMethod`

- Examples:
  
  - `taplint interface=unauth tapurl=...`
  
  - `tapquery interface=auth:tls-with-security tapurl=...`
TOPCAT/STILTS TAP clients need a *bundle* of service endpoints

- To interact with the service, they need (several of) the endpoints `sync`, `async`, `tables`, `capabilities`, `examples`

What are these authenticated TAP client implementations doing?

- Read `/capabilities` document from service
- Sort declared `capability/interface` entries into per-securityMethod endpoint bundles
- Offer the available bundles to the user (choice of `securityMethod/@standardID`)
- Arrange that subsequent service interactions use the correct bundle-specific endpoints
- Nothing else!

▷ Actual authentication is done outside of application code (see later slides)
▷ They do not attempt to work out which bundle(s) can be used in the current context — the user has to look at the names and work it out.
Security Method Specifics

Application code just attempts to use given endpoints

- It doesn't know if they are authenticated or not, or which auth method is used
- Actual authentication is done at the JRE level (HttpURLConnection does the hard work)

Several authentication methods are defined by SSO 2.0:

- No authentication required:
  - Easy.
- HTTP Basic Authentication, TLS-with-Client-Certificate:
  - Tested and working using JRE-level mechanisms (see next slides)
- TLS-with-Password:
  - I think it *should* work just like HTTP Basic Auth
- Cookies:
  - Only appropriate for browser clients??
- SAML, OAuth, OpenID:
  - I have no idea what these are.

This seems to do the job so far … do I need to work any harder?
HTTP Basic Authentication

securityMethod/@standardID="ivo://ivoa.net/sso#BasicAA":

- Client initially attempts unauthenticated access
- This results in an HTTP 401 response
- When it sees a 401, the java.net.HttpURLConnection calls java.net.Authenticator static methods to get username/password
  ▶️ I think it keeps track of these by hostname
- Clients can install default Authenticator instances into the JVM
  ▶️ STILTS uses an instance that picks up username/password from system properties (star.basicauth.user, star.basicauth.password)
  ▶️ TOPCAT tries those system properties, but if they are not supplied, it uses a GUI prompt instead
- I think the same should work without code modification for securityMethod/@standardID="ivo://ivoa.net/sso#tls-with-password"

```
stilts -Dstar.basicauth.user=mbt
    -Dstar.basicauth.password=xxxxx
tapquery tapurl=http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap/
    interface='auth:BasicAA'
    adql='SELECT TOP 3 * FROM caom2.siav1'
```
TLS With Certificate

securityMethod/@standardID="ivo://ivoa.net/sso#tls-with-certificate":

- HTTPS client must present a (suitable) client certificate when opening connection
- It can be configured to do so by installing a (suitable) `javax.net.ssl.SSLSocketFactory`:
  - Either: per connection: `connection.setSSLSocketFactory()`
  - Or: system-wide: `HttpsURLConnection.setDefaultSSLSocketFactory()`

- STILTS/TOPCAT lets you install a system-wide one
  - Set system property `star.cert.pem` to name of file containing (e.g.) the PEM-format proxy certificate downloaded from CANFAR authenticated web page
  - Can only have one certificate installed per JVM (TOPCAT session)
  - ... which is OK for now, since only one service (CADC) is using this auth method

- Most of the code supplied by CADC (*thanks Brian!*)
- Future improvements under investigation:
  - TOPCAT GUI prompt for certificate file rather than requiring system properties
  - Multiple certificates for different services

- Hard to say whether this will work well for other future services (similar cert format?)

```
stilts -Dstar.cert.pem=/home/mbt/certs/cadcproxy.pem
tapquery tapurl=http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/tap/
  interface='auth:tls-with-certificate'
adql='select top 3 collection, dataRelease from caom2.siav1'
sync=true
```
Other Security Methods

Cookies, SAML, OAuth, OpenID:

- No attempt to deal with these so far
- Possibly some JRE-level configuration will work for these too?
- Does anyone plan to use these for desktop applications?
Availability

Working versions available:

- **URLs:**
  
  ftp://andromeda.star.bris.ac.uk/pub/star/topcat/pre/topcat-full_tap11.jar
  ftp://andromeda.star.bris.ac.uk/pub/star/stilts/pre/stilts_tap11.jar

- These are prototype versions, subject to change
  
  ▶ Maybe some improvements for supplying credentials
  ▶ Maybe changes following discussions here or IVOA feedback
Future Work

Better user interaction

• Better options for user to supply authentication tokens at runtime
• Smarter user interaction based on selected authentication method
• Possibility for per-service TLS certificate configuration
  (but is it worth it if only CADC is using tls-with-certificate?)
• Guess what authentication option the user will want based on available credentials?
  (but I can’t see how to do this)

Other protocols

• Authenticated Cone, SIA, SSA? (do services exist? how are they registered?)

More testing

• Try out non-CADC authenticated TAP services (are there any?)

Share code

• Collaborate on authentication library to share between VO Java applications?
• Already working with CADC

Release

• Incorporate functionality in public release — when stable
PR-TAP-1.1-20181024 Section 2.4 “VOSI-capabilities”:

- TAP 1.1 PR describes how to find TAP endpoints for a TAP service
- The implementation on previous slides follows this description
- ... so it can be done ... 
- ... but I don’t really like it.
- Quite a bit of discussion on this topic already:
  - My presentation Shanghai 2017 (some items cleared up since then: bundle assembly rules, unique capabilities file)
  - RFC page
  - DAL mailing list “TAP 1.1 authentication” thread in August and September

- Summary:
  - Some details have been cleaned up, but disagreements remain
  - Pat Dowler (TAP author) supports current approach
  - Mark Taylor, Markus Demleitner, Paul Harrison have concerns

- Details on following slides
Why I don’t much like TAP 1.1 authenticated service specification

- TAP service interaction uses several endpoints:
  
  sync, async, tables, capabilities, examples

- Clients like TOPCAT/STILTS need to work with a bundle of these, not just pick one

- TAP 1.0: service defined by base URL
  
  Bundle specification: base URL + well-known subpaths
  
  http://dc.g-vo.org/tap/sync
  http://dc.g-vo.org/tap/async
  ...

- TAP 1.1: service defined by capabilities document
  
  Capabilities doc provides an unstructured list of (endpoint, securityMethodID) pairs
  
  Bundle specification: capabilities doc + securityMethodID + bundle-assembly rules
  
  There is no base URL

- TAP 1.1 is hard work for bundle-oriented clients:
  
  Fairly difficult to list available bundles to offer to users
  (but not impossible — I implemented it)
  
  Can’t specify a bundle by just giving a URL e.g. on the command line or in an email
  (except by fallback to TAP 1.0 rules)
  
  Very difficult to deal with services that may be mirrored as well as authenticated
  (I gave up)
PR-TAP-1.1-20181024 Section 2.4 “VOSI-capabilities”:

- Explains use of capabilities file to specify endpoints
- Includes text explaining how to interpret it as bundles
  - Text supplied by me — as simple as I could make it for e.g. TOPCAT requirements, but still a bit involved
- Relies on (draft) UWSRegExt Note, so provisional and non-normative
  - PR-TAP-1.1-20181024 therefore does not tell clients how to use authenticated services (just guesses how it might work in future)
  - If UWSRegExt doesn’t work out as per its current draft, this section will be unhelpful/misleading
TAP 1.1 Capabilities: Suggestions

Bundles: return to **Base-URL-based** system

- Markus suggests new DALI subtype of `vr:interface` (see registry mailing list 17 Oct):

  ```xml
  <capability standardID="ivo://ivoa.net/std/TAP">
    <interface xsi:type="vs:DALIInterface">
      <accessURL>http://dc.g-vo.org/tap</accessURL>
      <endpoint>sync</endpoint>
      <endpoint>async</endpoint>
      <endpoint>capabilities</endpoint>
      <endpoint>tables</endpoint>
      <endpoint>examples</endpoint>
    </interface>
  </capability>
  ```

- I think this would reduce the complexity of Sec 2.4 and remove the need for UWSRegExt
- `<mirrorURL>` elements would become usable
- Looks reasonable to me? But I’m not a registry expert

**Capabilities discussion: extract to TAPRegExt**

- Capabilities section is not really core to TAP 1.1, and is anyway subject to change and non-normative
- TAPRegExt 1.1 is currently in WD

⇒ Punt capabilities details to TAPRegExt?

... blame Markus for this suggestion too