

# Client Authentication in the VO

Mark Taylor (Bristol)

*with input from:* Sara Bertocco, Pat Dowler, Brian Major, Jesus Salgado,  
James Tocknell, GWS/DSP WG

Distributed Services & Protocols WG

IVOA Interop

College Park, MD, USA

4 June 2025

`$Id: auth.tex,v 1.11 2025/06/04 12:30:54 mbt Exp $`

# The Problem

## Question:

- How does a VO client know whether/how to authenticate with a VO service?
  - given only the base URL of that service (e.g. TAP access URL)
  - given only the URL of a protected resource? (e.g. the location of a protected FITS file)

## Answer:

- Web clients (e.g. archive front-end web pages):
  - ▷ usually there's some knowledge about authentication details present in the client itself
- Desktop clients (e.g. TOPCAT, Aladin, Python scripts):
  - ▷ slowly converging on answer over last several years (SSO\_next wiki page, various GWS presentations)
  - ▷ Now there is a Working Draft: <https://github.com/ivoa-std/IAP>

# Standardisation Status

## New Working Draft document (last week)

- Tentatively named **Interoperable Authentication Protocol**
- github: <https://github.com/ivoa-std/IAP>
- latest draft [PDF](#)
- Issues/PRs welcome

## Based on standard HTTP technology, but with some extensions

- ▶ Not trying to invent new auth machinery in parallel with industry standards, just plug gaps
- ▶ VO services are not required to use anything here, but it gives options for client interoperability



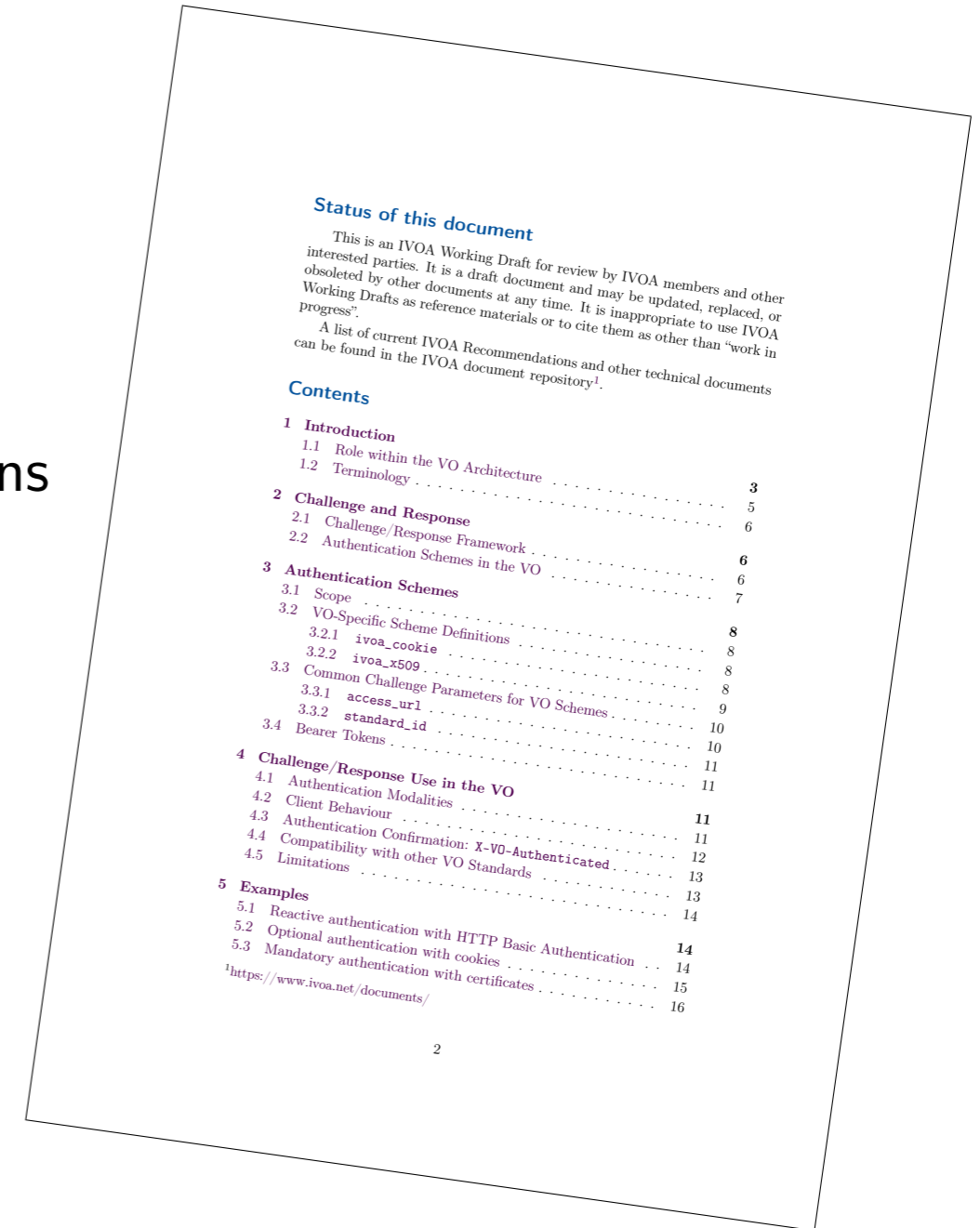
# Standardisation Status

New Working Draft document (last week)

- Tentatively named **Interoperable Authentication Protocol**
- github: <https://github.com/ivoa-std/IAP>
- latest draft [PDF](#)
- Issues/PRs welcome

Based on standard HTTP technology, but with some extensions

- ▶ Not trying to invent new auth machinery in parallel with industry standards, just plug gaps
- ▶ VO services are not required to use anything here, but it gives options for client interoperability



# IAP Content

## IVOA Interoperable Authentication Protocol (*or Process?*)

- Service advertises authentication options using HTTP `WWW-Authenticate` **challenges** (RFC 9110)
  - ▷ Returned with HTTP response when client must or may authenticate
  - ▷ IAP recommends some standard Authentication Schemes and defines some VO-specific ones
    - These provide all information necessary to log in (no out-of-band info required)
- VOSI-compliant services (e.g. TAP) can advertise optional authentication using `capabilities` HEAD request:
  - ▷ 401/403 with challenges: mandatory authentication
  - ▷ 200 with challenges: optional authentication
  - ▷ 200 without challenges: no authentication
  - ▷ Response header `X-VO-Authenticated` tells client whether optional auth has succeeded
- Client behaviour recommendation:
  - ▷ VOSI-compliant service (e.g. TAP):
    - First probe `capabilities` endpoint to see if authentication is mandatory/optional/unavailable
    - Optionally log in using challenge to get auth permit;  
then use auth permit for subsequent requests to same domain (same service)
  - ▷ Single resource (e.g. DataLink):
    - Make HTTP request for resource as usual
    - If 401 response, log in using challenge to get auth permit;  
then use auth permit for subsequent requests to same domain

# Authentication Schemes

IAP defines challenges providing all necessary auth information to clients

- Challenge format: `WWW-Authenticate: <auth-scheme> <scheme-specific-params>`
- Necessary information is auth-scheme dependent, but typically:
  - ▷ What URL do I present username+password at?
  - ▷ How do I present username+password?
  - ▷ What response (“*permit*”) do I get back when I present them? (e.g. cookie, token, certificate)
  - ▷ What is the usage domain of that permit?
- Auth-schemes recommended in IAP:
  - ▷ `Basic` (defined by [RFC 7617](#))
    - Domain is challenge *Realm* (as per RFC7617)
  - ▷ `ivoa_cookie` (defined by IAP)
    - Challenge parameters communicate [login URL](#) and [login method](#) to acquire cookie
    - Domain is cookie domain (defined by [RFC 6265](#))
  - ▷ `ivoa_x509` (defined by IAP)
    - Challenge parameters communicate [login URL](#) and [login method](#) to acquire client certificate
    - Domain defined ad-hoc by IAP: currently challenge URL *Origin*
  - ▷ More may be added
    - we especially need something for Bearer Tokens (OAuth2)

## Example

### Auth Scheme `ivoa_cookie` in place at GACS (ESAC)

- Query `capabilities` endpoint using HTTP HEAD:

```
% curl --head https://gea.esac.esa.int/tap-server/tap/capabilities
HTTP/1.1 200 OK
Content-Type: text/xml;charset=UTF-8
WWW-Authenticate: Bearer
WWW-Authenticate: ivoa_cookie access_url="https://gea.esac.esa.int/tap-server/login", standard_id="ivo://ivoa.net/sso#tls-with-password"
```

- 200 response with WWW-Authenticate challenge means optional authentication
- Challenge parameters tell us how (`standard_id`) and where (`access_url`) to exchange username+password for a cookie

```
% curl -D - --data username=mtaylo02 --data password=xxxx https://gea.esac.esa.int/tap-server/login
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=474DEF8CBB046D47294; Path=/tap-server; Secure; HttpOnly
Content-Type: text/plain; charset=UTF-8
```

- Retrieve cookie from Set-Cookie header and use in subsequent requests to the cookie's scope

```
% curl -D - --header 'Cookie: JSESSIONID=474DEF8CBB046D47294' https://gea.esac.esa.int/tap-server/tap/async
HTTP/1.1 200 OK
X-VO-Authenticated: mtaylo02
Content-Type: text/xml;charset=UTF-8

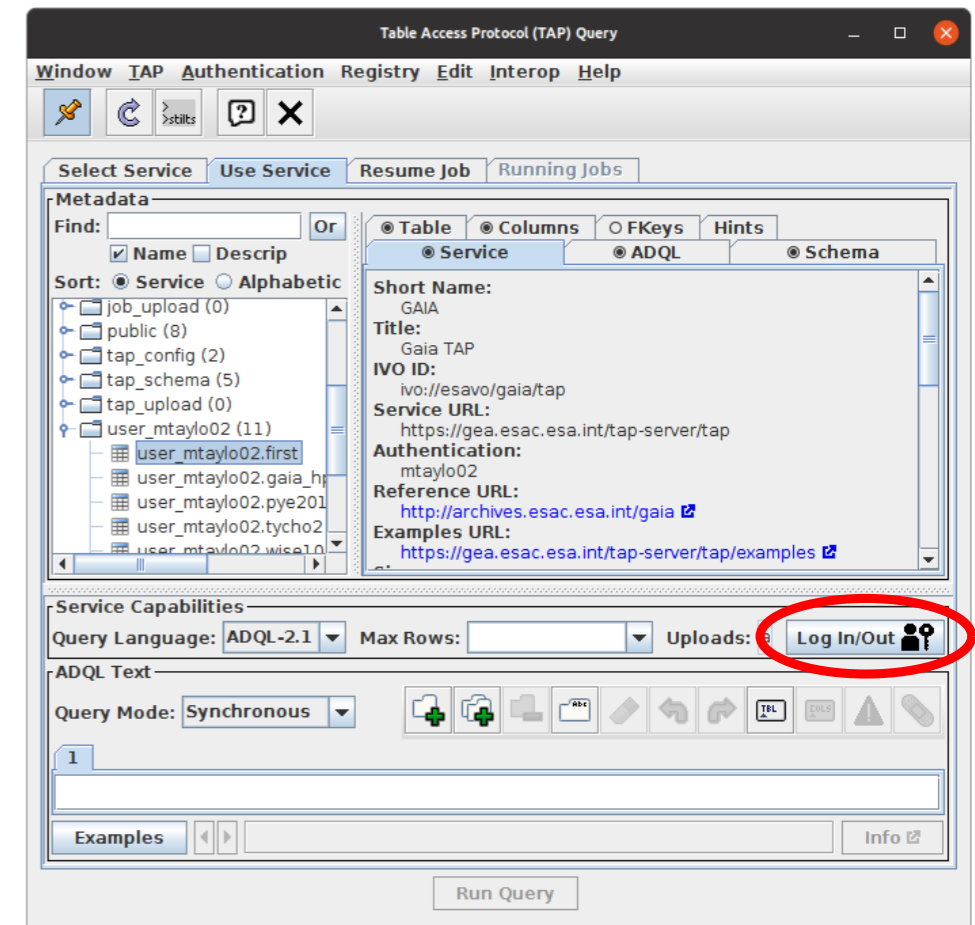
<?xml version="1.0" encoding="UTF-8"?>
<uws:jobs xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0">
  <uws:jobref id="17268355285350" > ... </uws:jobref>
  ...
</uws:jobs>
```

This example (and others) are in the IAP document.

# Implementations

IAP is a codification of a protocol already in production use

- Services:
  - ▶ ESAC TAP services (Gaia, Euclid, PSA, others?) and DataLink etc
    - Using `ivoa_cookie`
  - ▶ CADC TAP services (Youcat, CADC TAP) and DataLink etc
    - Using `ivoa_x509`
  - ▶ DaCHS TAP and DataLink etc
    - Using HTTP Basic auth
  - ▶ All of the above report authentication modality (mandatory/optional/no auth) on the VOSI `capabilities` endpoint
- Clients:
  - ▶ TOPCAT/STILTS
  - ▶ Java AUTH library can be used by other Java applications



## Open Question: Document Name

Document name (SSO → IAM → IAP → ...?)

- “Interoperable Authentication Protocol” → IAP? ← *Current draft says this*
- “Interoperable Authentication Process” → IAP? ← *... which was a typo for this*
- “Authentication in the Virtual Observatory” → Auth-VO? ← *Suggestion by James Tocknell*
- something else?

## Open Question: OAuth 2

OAuth 2.0 (Bearer tokens), often with OIDC, is very widely used among data providers

We really have to come up with some solution for it

- Requirements
  - ▷ Client must be able to get *Bearer Token* and domain information (e.g. a list of Origins)
    - (plus maybe refresh token and other things)
  - ▷ At least offer the option of headless operation
  - ▷ Align as much as possible with existing OAuth2/OIDC infrastructure
- Suggestions:
  - ▷ James Tocknell: <https://github.com/ivoa-std/SSO/pull/22>
  - ▷ Jesus Salgado: <https://github.com/ivoa-std/IAP/issues/6>
- Active discussion at <https://github.com/ivoa-std/IAP/issues/6>
- I'd suggest prototype something in service+client, then add to text if it works

## Open Questions: Others

### Miscellaneous items:

- Impacts on other standards
  - ▷ Update to VOSI to require HEAD on capabilities? Or use GET instead for IAP?
  - ▷ Changes to recommended use of VOResource `securityMethod` element?
  - ▷ Status of SSO document?
  - ▷ Use of keys from SSO StdRegExt
- More `standard_id` options (cookie, token, ...)
- Some items of terminology
- Scoping for certificates
- ... and more — all input welcome