User-Managed Access (UMA) 101

George Fletcher and Eve Maler
Kantara Initiative UMA Work Group
@UMAWG | tinyurl.com/umawg
IIWXXXII | 20 Apr 2021

kantara™ INITIATIVE

Topics

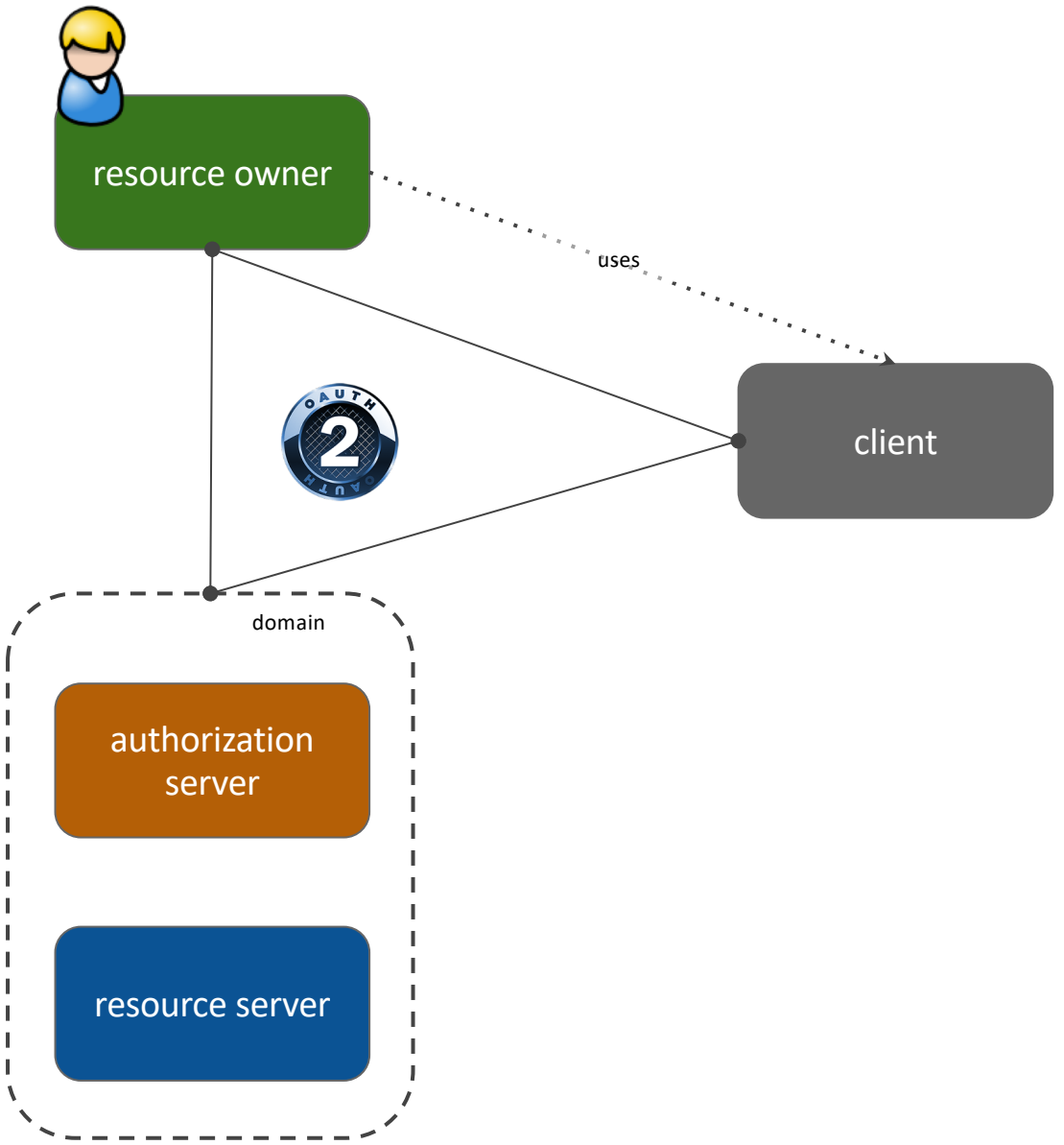
- Overview
- Use cases
- New work
- UMA and decentralized identity
- Business-legal-technical (BLT) implications
- Technical big picture
- Technical deep dive

OAuth and UMA

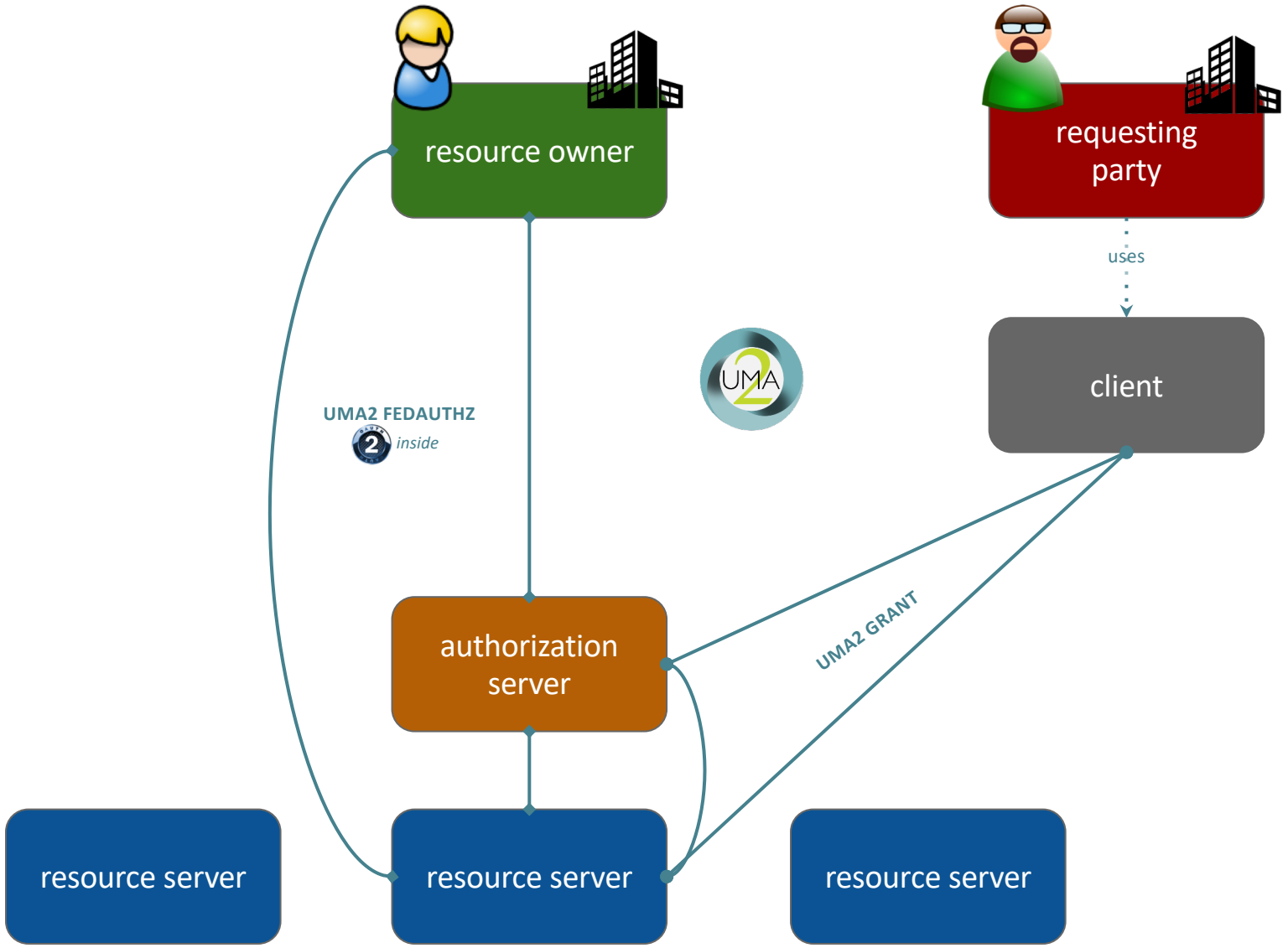
“ALICE-TO-SELF” SHARING

OAuth enables **constrained delegation** of access to **apps** on request

Alice can **agree** to app connections and also **revoke** them



OAuth and UMA



“ALICE-TO-BOB” SHARING

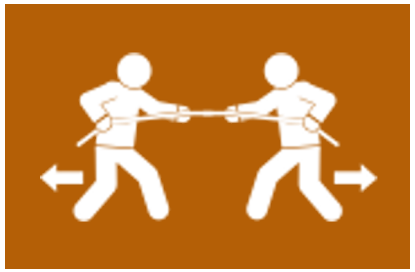
UMA adds **control** of **cross-party sharing**, letting Alice be **absent** when Bob uses a client to attempt access

Alice **controls trust** between resource hosts and authorization services – enabling a **wide ecosystem** of resource hosts, so Alice can manage sharing **across** them

UMA and consent

Consent (and consent to contract) legally require **Manifestation**, **Knowledge**, and **Voluntariness** – more often honored in the breach

Cookie consent
App permissions
Marketing preferences
Third-party permissions
ToS agreements



Digital consent has serious practical challenges achieving revocability, contract meeting of the minds, choice in relationship building, and consent seeker good faith

UMA enables permissioning that is **asynchronous**

Share with parties, with groups, by relationship
Respond to pending requests
Monitor all current shares across sources
Modify one or more shares
(Respond to request at run time à la consent)

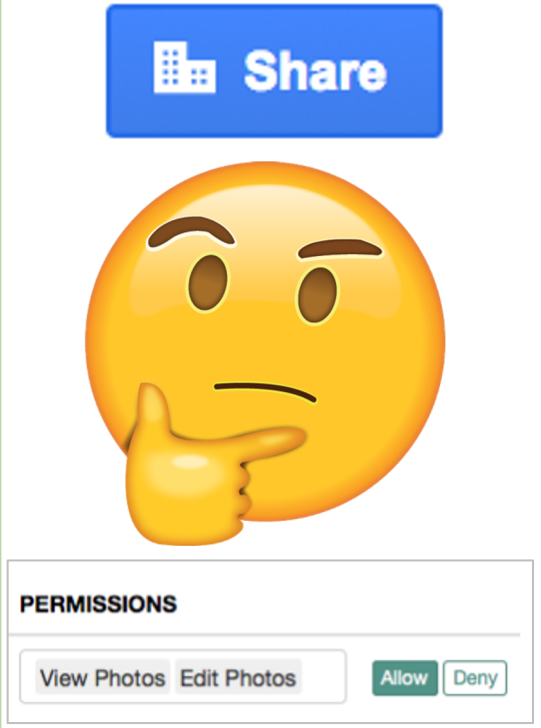
It is a technology that can enable **right-to-use licensing** within a Me2B framework of mutual agency and value exchange

L. LeVasseur and E. Maler, "Beyond Consent: A Right-to-Use License for Mutual Agency," in IEEE Communications Standards Magazine, vol. 3, no. 4, pp. 52-59, December 2019, doi: 10.1109/MCOMSTD.001.1900031.

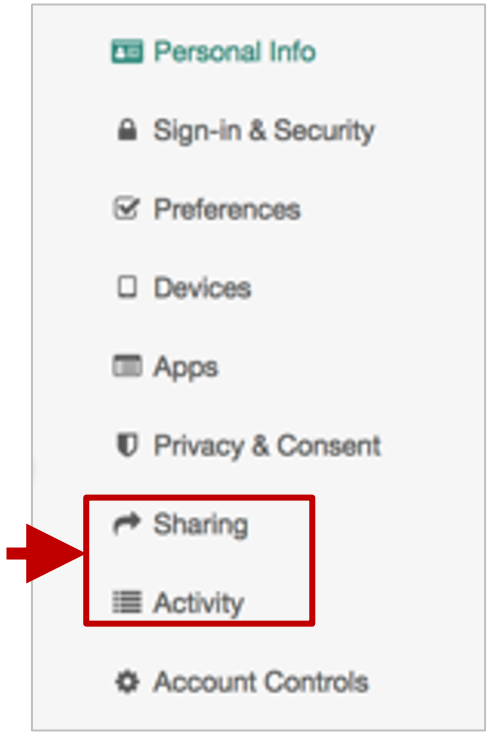
Benefits for individuals: a summary

Choice in sharing with other parties

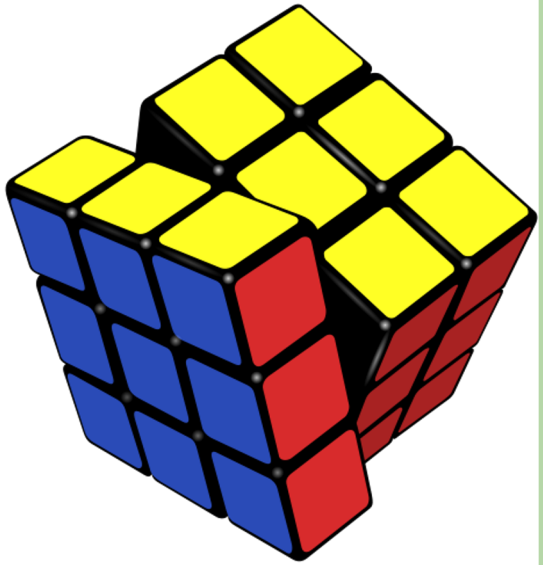
Convenient sharing/approval with no outside influence



Centralizable monitoring and management



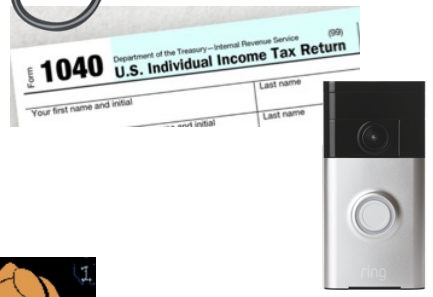
Control of who/what/how at a fine grain

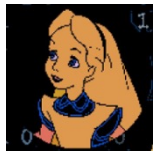


Benefits for service providers: a summary

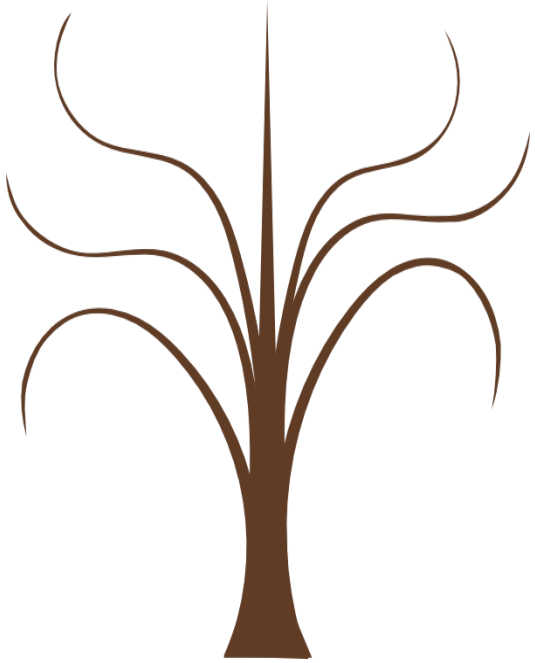
True secure delegation; no password sharing

Scale permissioning through self-service



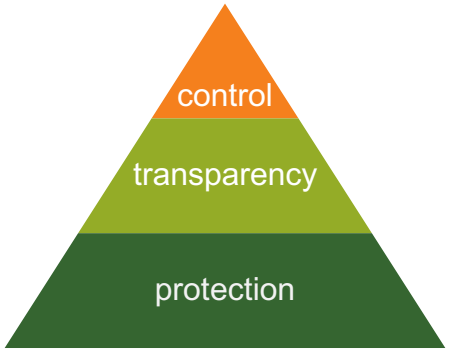


Resources accessed from distributed locations

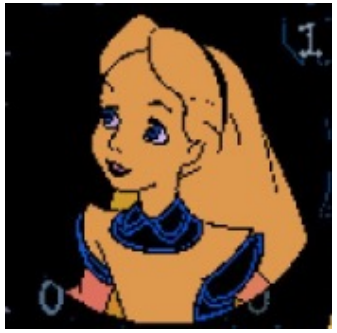


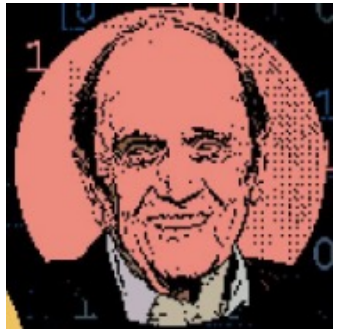
Foster compliance through standards

CMS
CENTERS FOR MEDICARE & MEDICAID SERVICES



Typical patterns

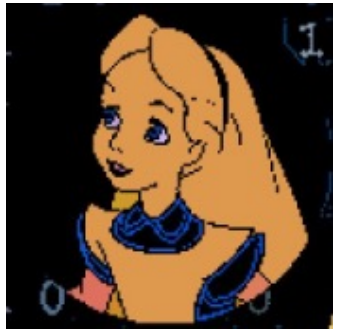


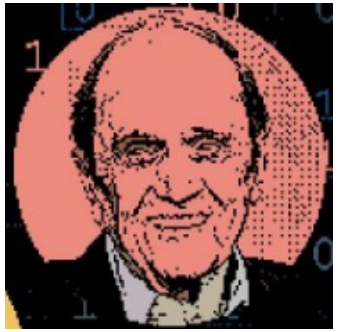


Alice-to-Bob (person-to-person) delegated sharing of **health** data/devices, **financial** data, **connected cars**...

Pensions Dashboards Programme

E.g., Alice shares selected accounts with selected financial advisors

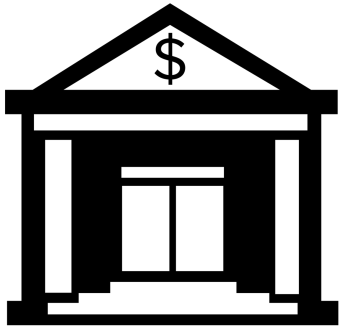


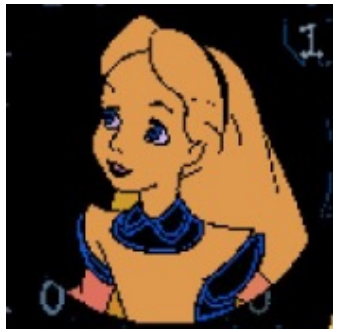


Enterprise-initiated delegated sharing – enterprise **API access management**, access delegation between **employees**

KEYCLOAK

E.g., RS acting as RO





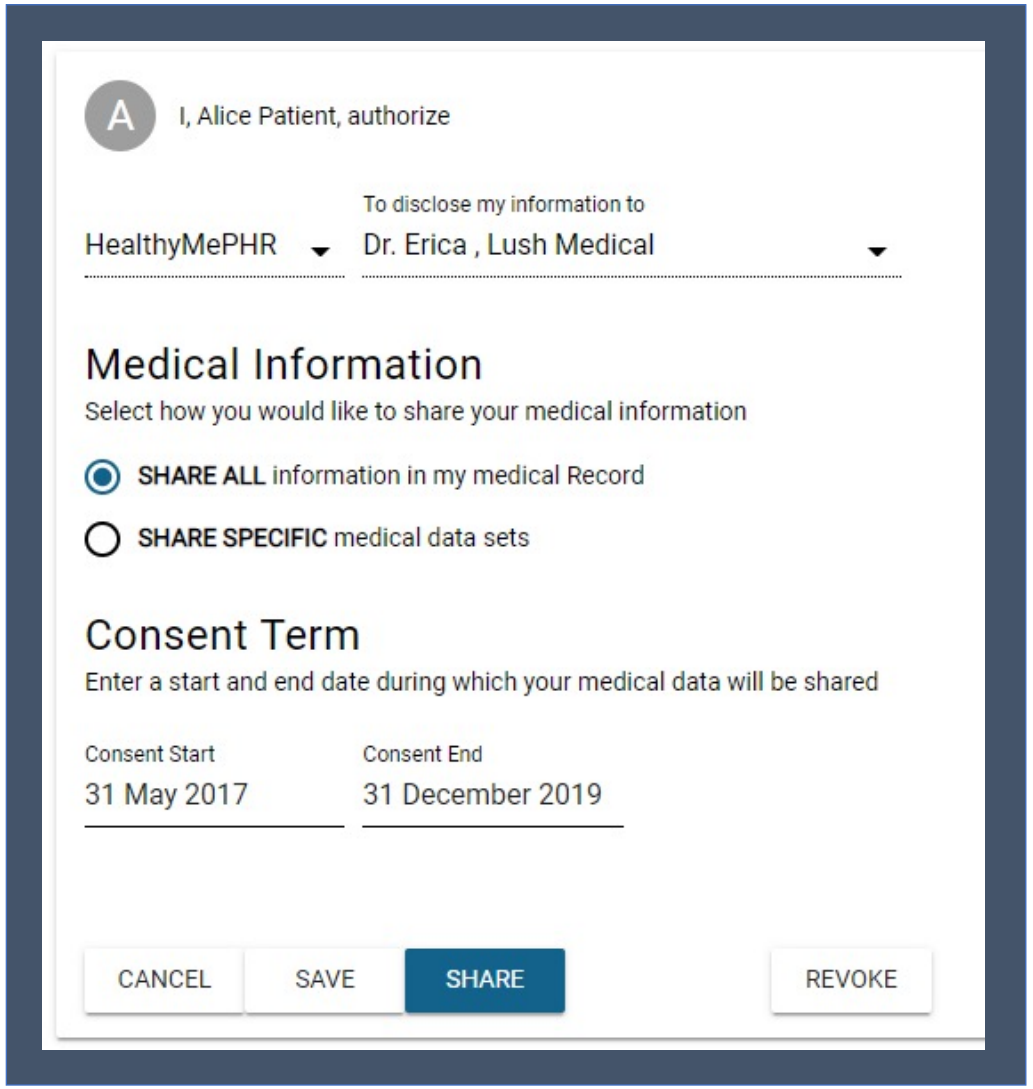
Alice-to-Alice (person-to-self) delegated sharing – **proactive** policy-based sharing of OAuth-style **app connections**

Pensions Dashboards Programme

...but first Alice enables the Pension Finder Service to find and display her accounts

Lush Group

HealthyMePHR – also ShareMedData

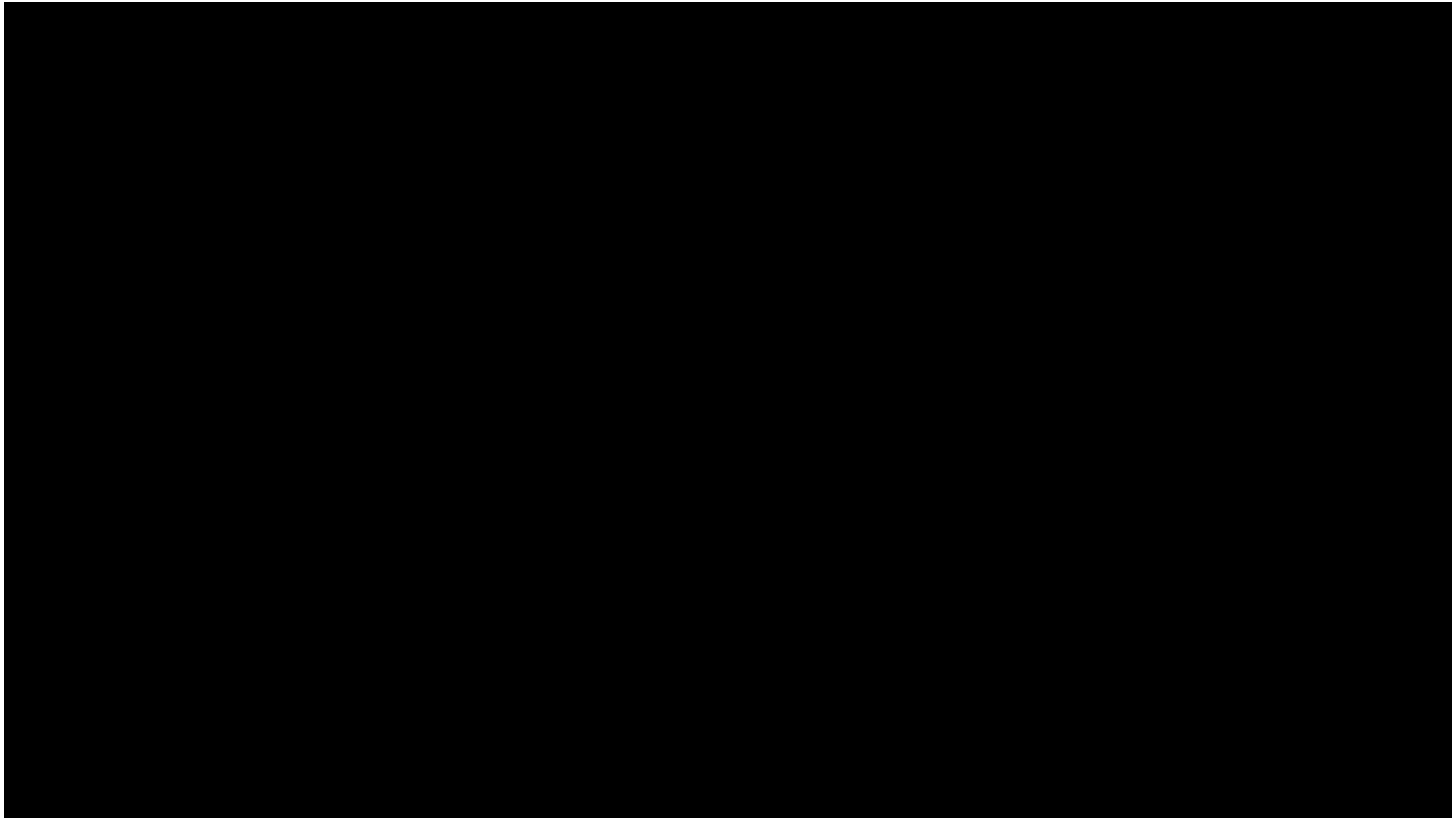


- Patient Alice creates a policy to share with Dr. Erica, she selects her sharing preferences, and presses SHARE

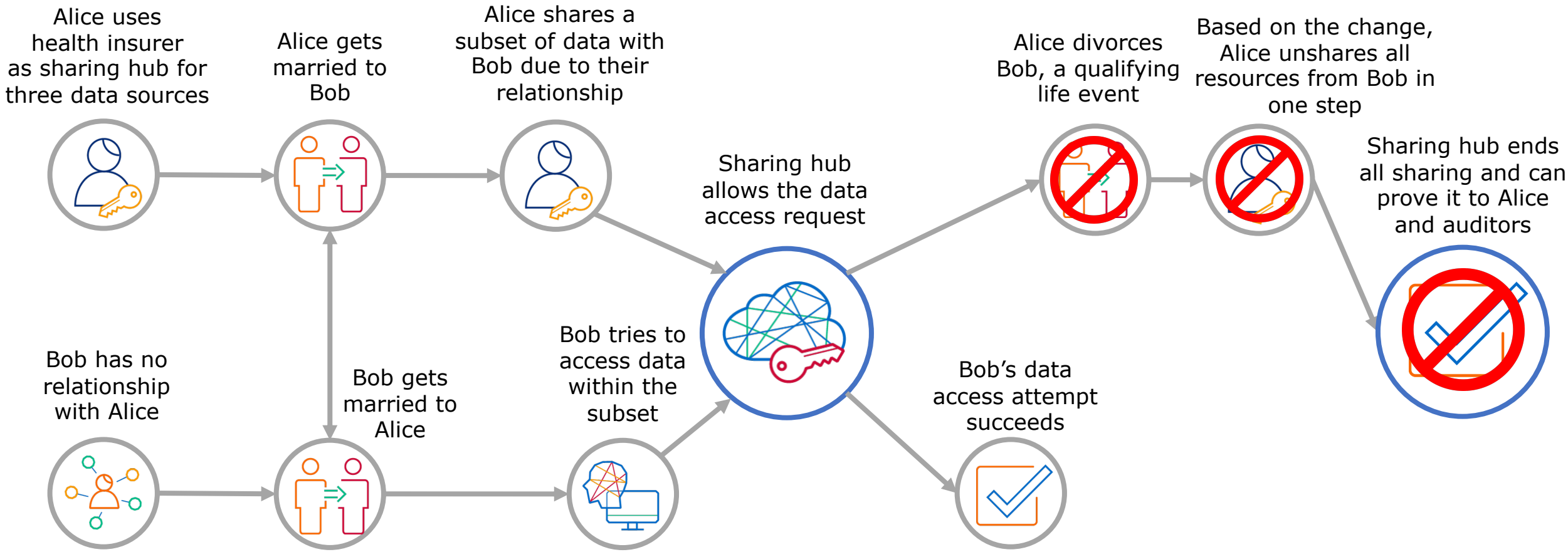
SHARE

- Patient sharing is easy!
  - *See HEART webinar recording*

ForgeRock Identity Platform – financial services example



Relationship-based health data sharing scenario



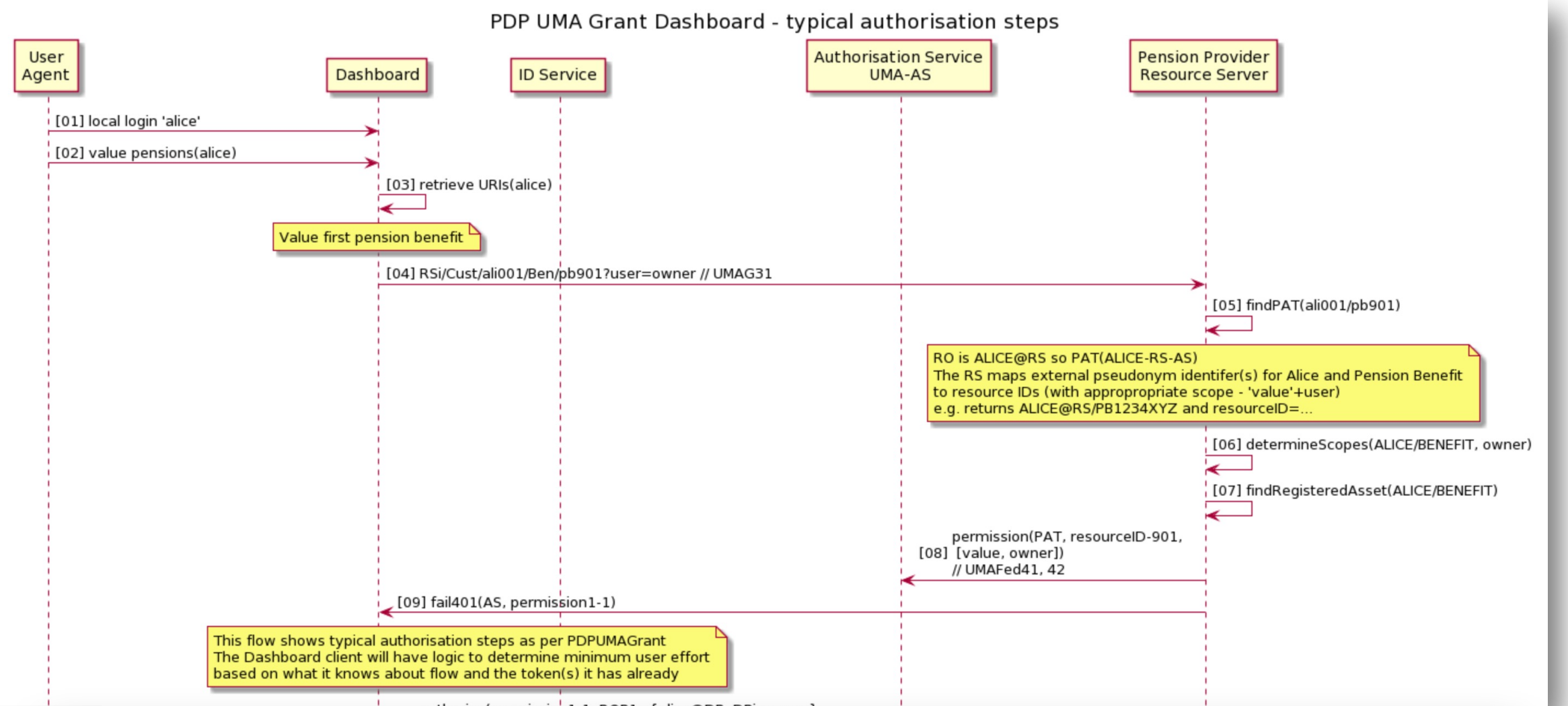
Key implementations

(more detail at tinyurl.com/umawg)

- ForgeRock – financial, healthcare, IoT, G2C…
- Gluu (open source) – API protection, enterprise, G2C…
- ShareMedData – healthcare
- HIE of One / Trustee (open source) – healthcare
- IDENTOS – healthcare, G2C
- Pauldron (open source) – healthcare
- RedHat Keycloak (open source) – API protection, enterprise, IoT…
- WSO2 (open source) – enterprise…

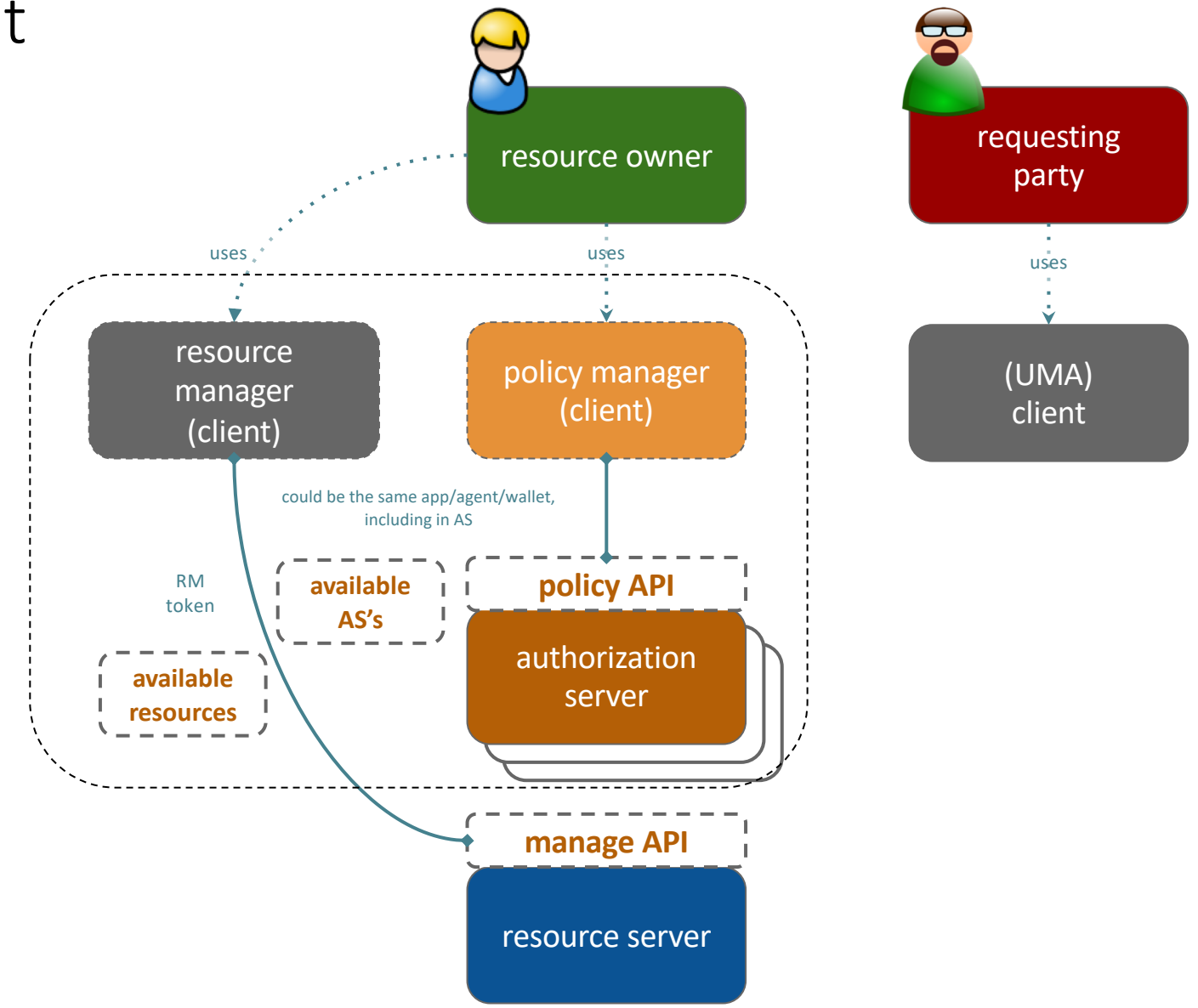
New profiling work: Pensions Dashboard profile (contributed)

(partial sample)



New profiling work: RO-side relationship management

Resource Manager extension: Extends Fed Authz, specifying an interface that allows an RS to work with any number of AS's to enable resource management by one RO



UMA and (decentralized) identity

UMA is identity-agnostic

AS, RS, and client may be **single-user** (dedicated) or **multi-user** (typically requiring identity and authentication)

AS and RS **establish trust** in (pseudonymous) resource owner context

Policy conditions need requesting party **claims** for authorization

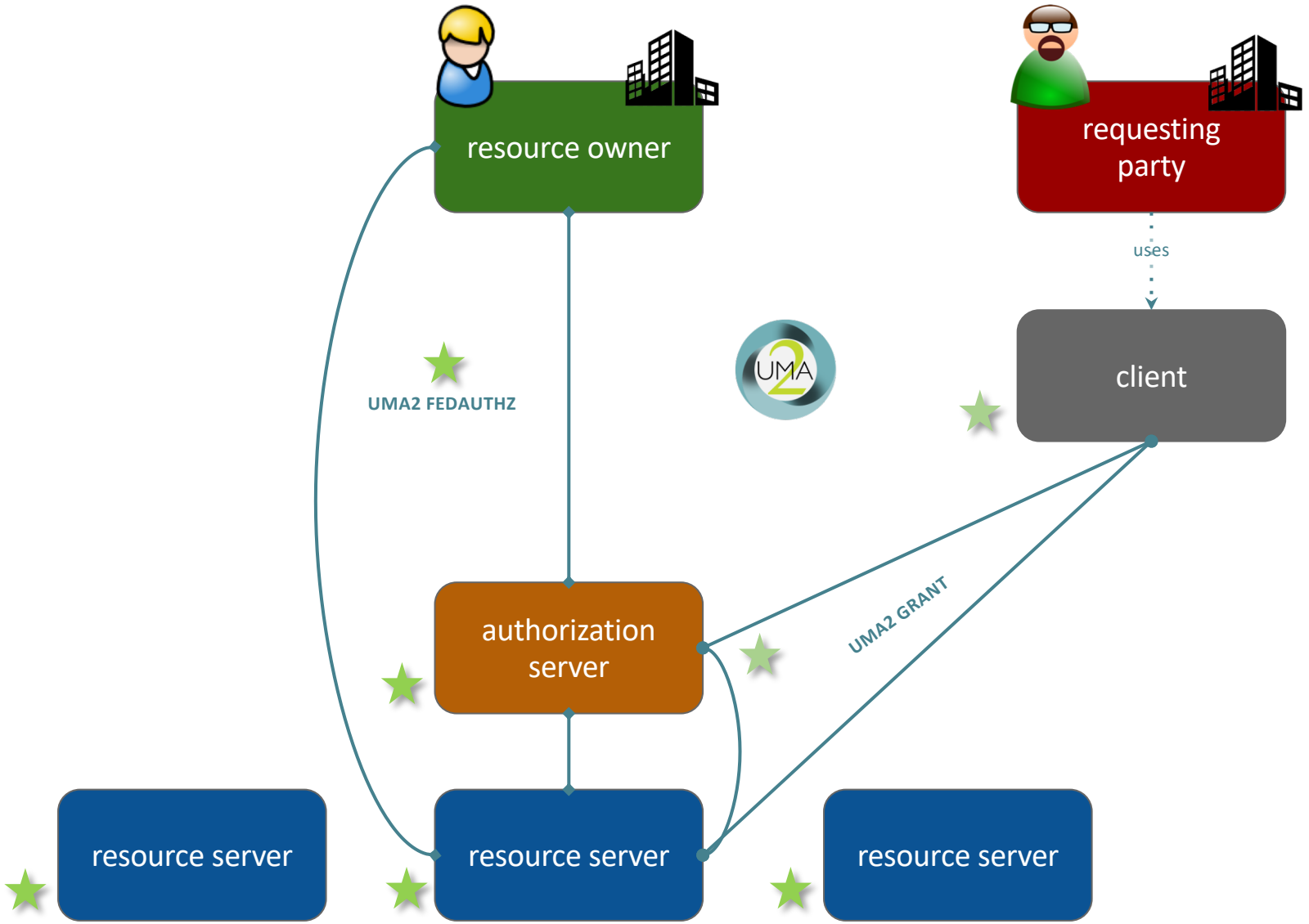
Claims can be **pushed** by smart client ahead of token request (narrower ecosystem)

Requesting party can be redirected to AS for **interactive claims gathering** at AS or further services (wider ecosystem)

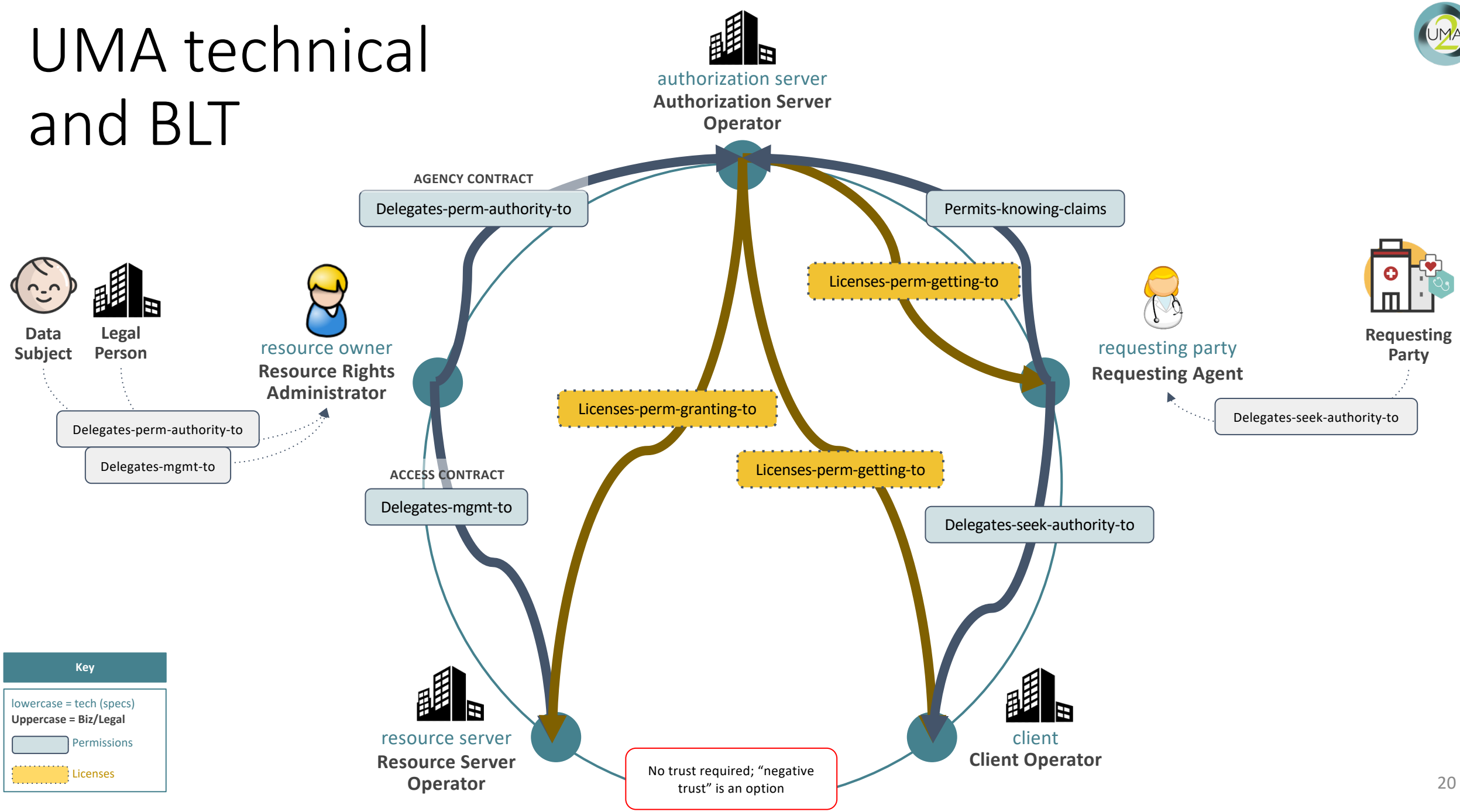
RS outsources all claims knowledge to AS

DID / VC approaches have been integrated at UMA's various identity touchpoints by various implementers (e.g., HIE of One with uPort)

(decentralized) identity may be relevant here



UMA technical and BLT

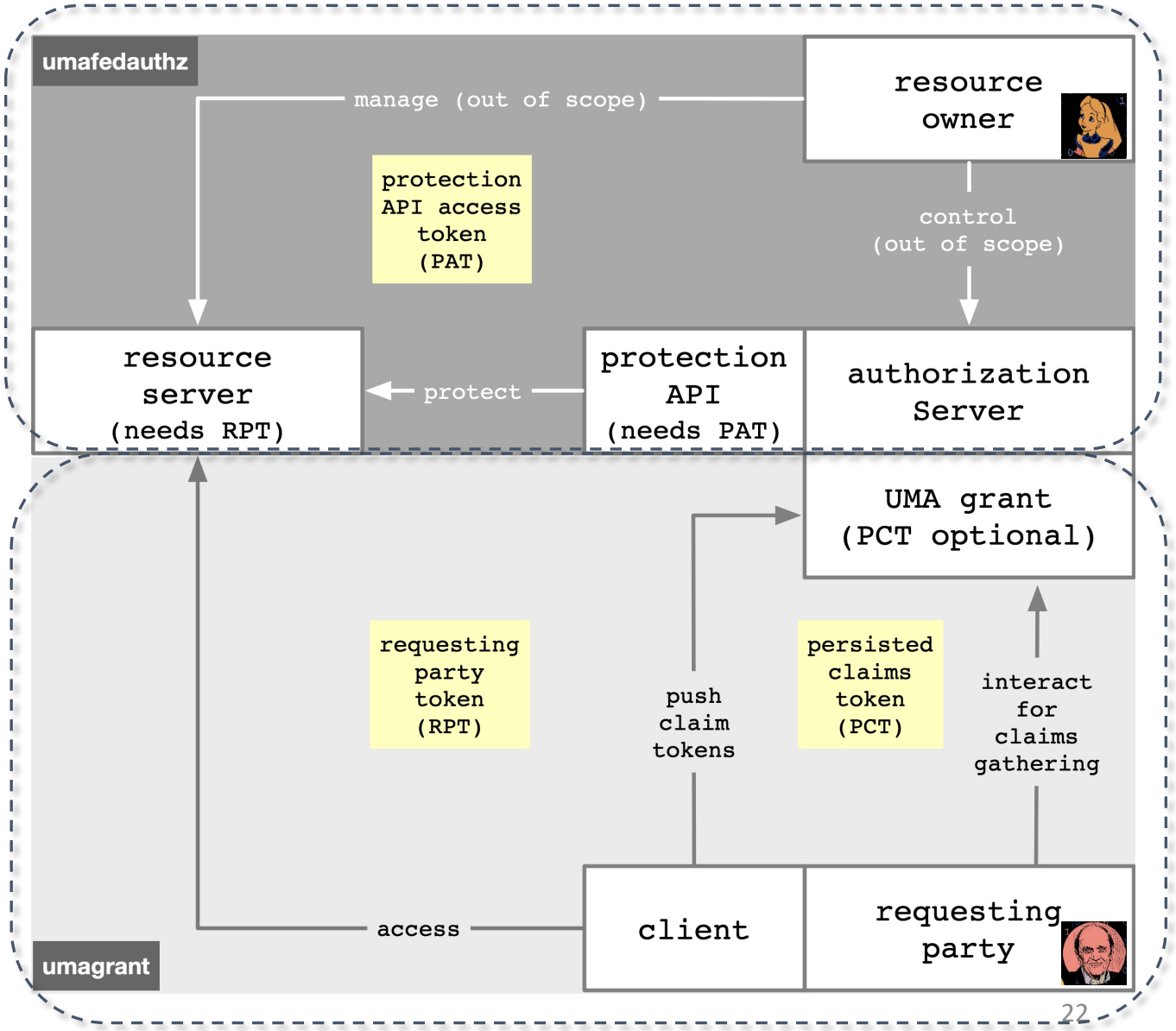


The technical big picture

A technical summary of the two UMA 2.0 specifications and their tokens

The marvelous spiral of delegated sharing, squared

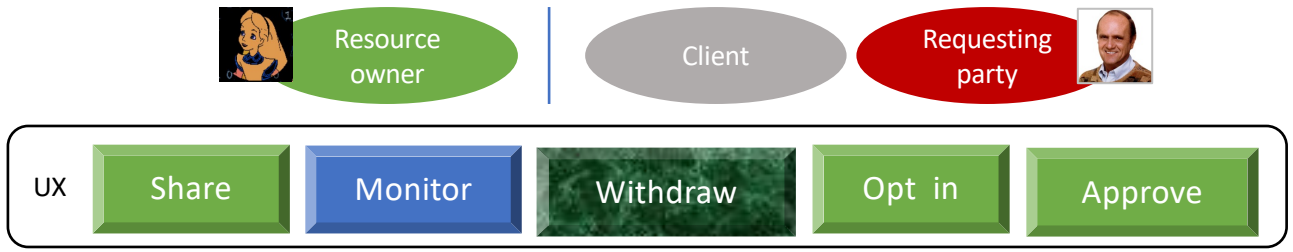
1. The **UMA grant of OAuth** enables Alice-to-Bob delegation
2. UMA **standardized an API for federated authorization** at the AS to make it centralizable
3. There are **nicknames** for enhanced and new tokens to keep them straight



The UMA extension grant adds…

docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html

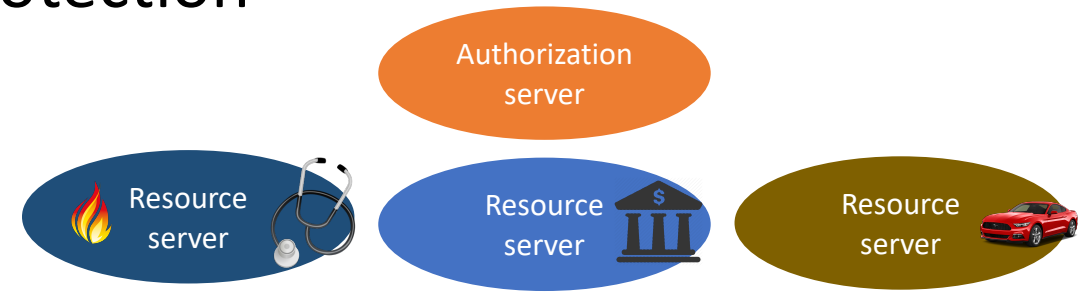
- **Party-to-party:** Resource owner authorizes protected-resource access to clients used by requesting parties
- **Asynchronous:** Resource owner interactions are asynchronous with respect to the authorization grant
- **Policies:** Resource owner can configure an AS with rules (policy conditions) for the grant of access, vs. just authorize/deny
  - Such configurations are outside UMA's scope



UMA federated authorization adds…

docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html

- **1-to-n:** Multiple RS's in different domains can use an AS in another domain
  - "Protection API" automates resource protection
  - Enables resource owner to monitor and control grant rules from one place
- **Scope-grained control:** Grants can increase/decrease by resource and scope
- **Resources and scopes:** RS registers resource details at the AS to manage their protection



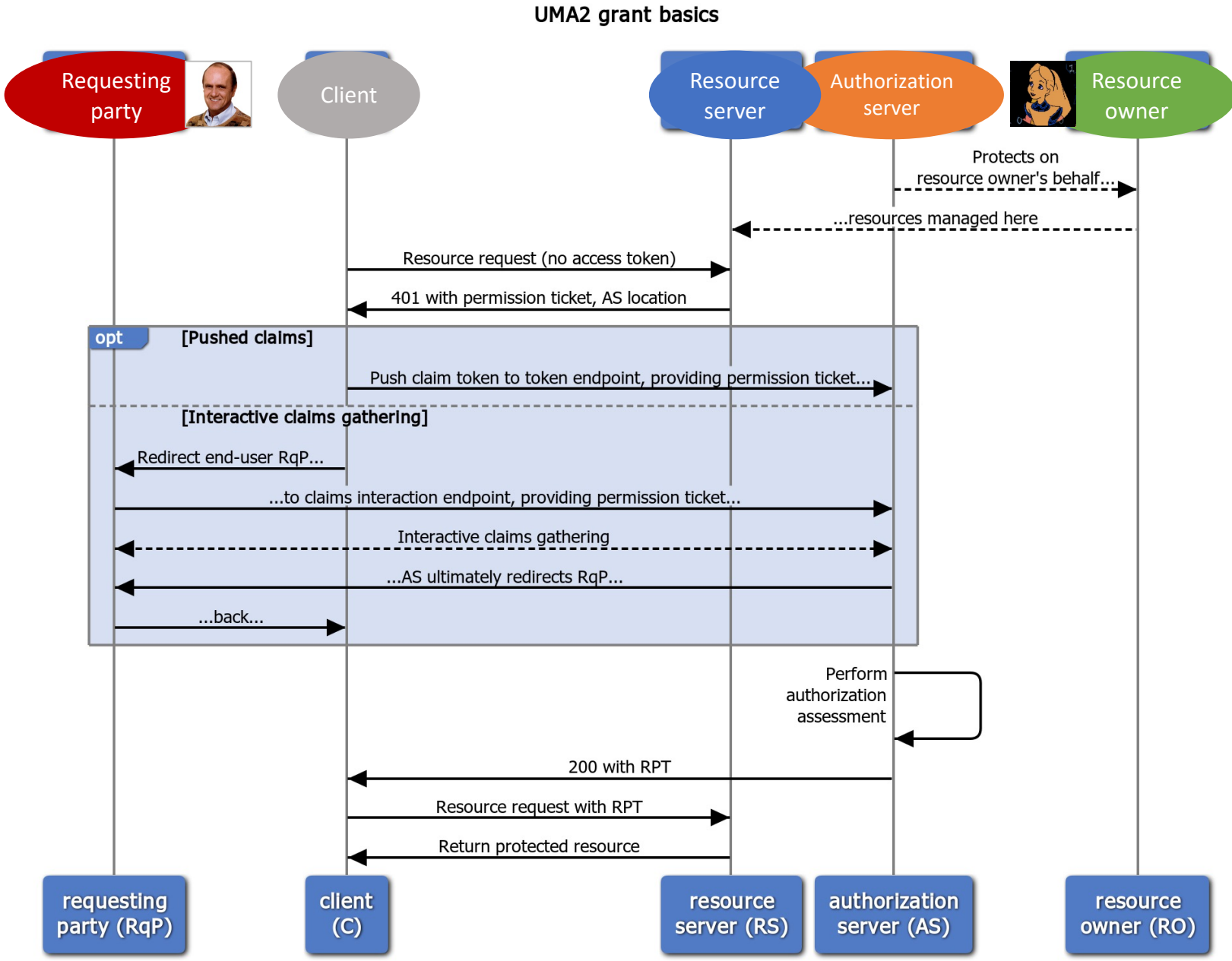
Technical Deep Dive

The UMA grant

A walkthrough of the UMA extension grant of OAuth2 and permission tickets

The UMA extension grant flow and its options

- The AS is acting as an **agent** for an absent RO
- The client's first resource request is **tokenless**
- The RS provides a **permission ticket** and allows **AS discovery**
- There are two **claims collection options** for meeting policy
- Authorization assessment and token issuance has **guardrails**
- RPTs can be **upgraded**, **revoked**, **introspected**, and **refreshed**



The permission ticket: how you *start* building a bridge of trust

- **Binds client, RS, and AS:** Every entity may be **loosely coupled**; the whole flow needs to be bound
  - It's like an overarching state parameter or "ticket-getting ticket"
  - Or maybe even a bit like an authorization code
- **Refreshed for security:** The client can **retry** RPT requests after non-fatal AS errors, using either claims collection option of the grant flow
  - The AS **refreshes** the permission ticket when responding with such errors

Pushed claims scenario: for wide-ish ecosystems

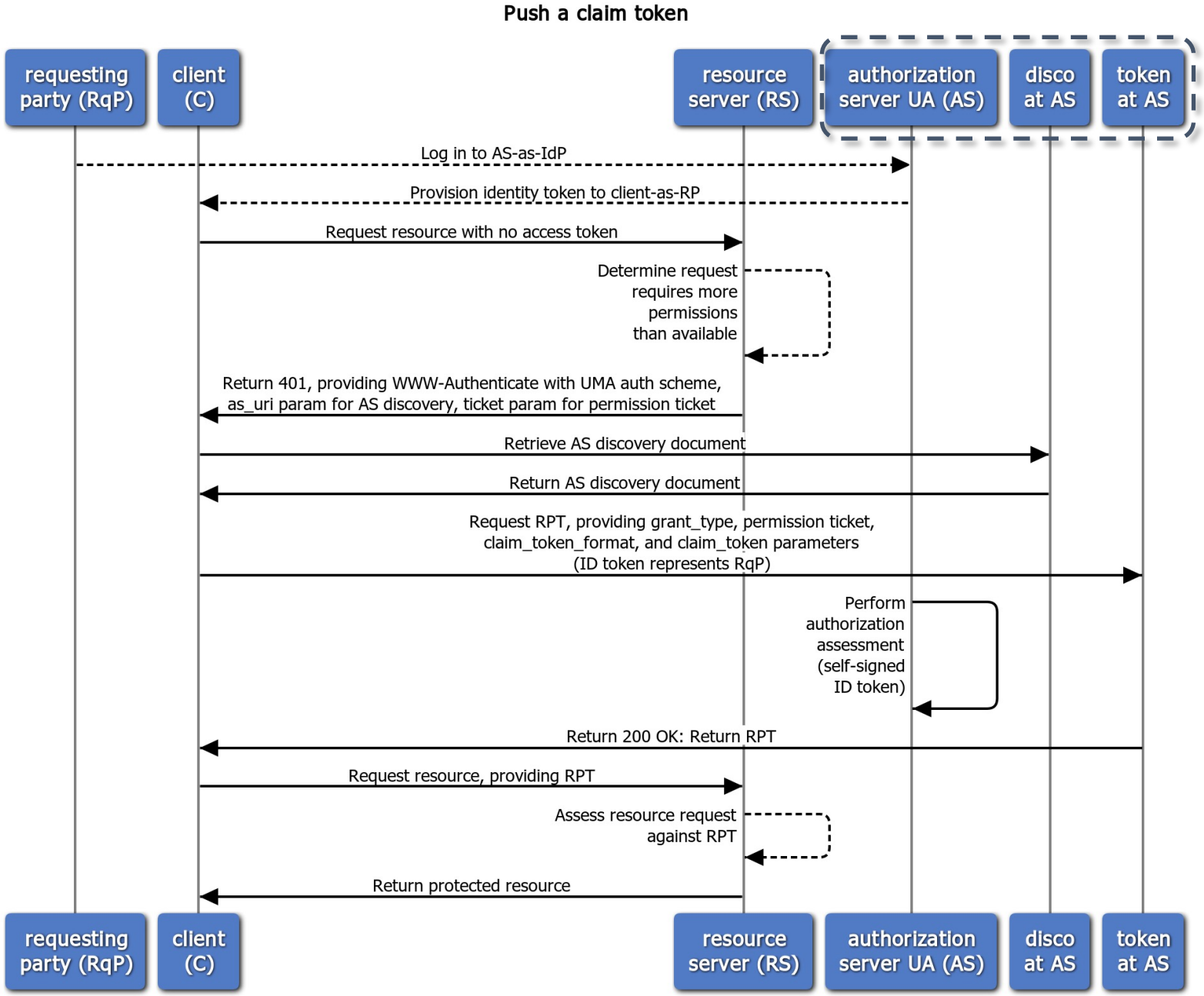
The AS is the requesting party's IdP and the client is the RP

More detail on the **RS's initial response** to the client

The client **pushes its existing ID token** to the token endpoint

The AS is **in the primary audience** for this token

Somewhat resembles SSO or the OAuth assertion grant, where a token of expected type and contents is "turned in"



Interactive claims gathering scenario: for wide ecosystems

(eliding detail already seen)

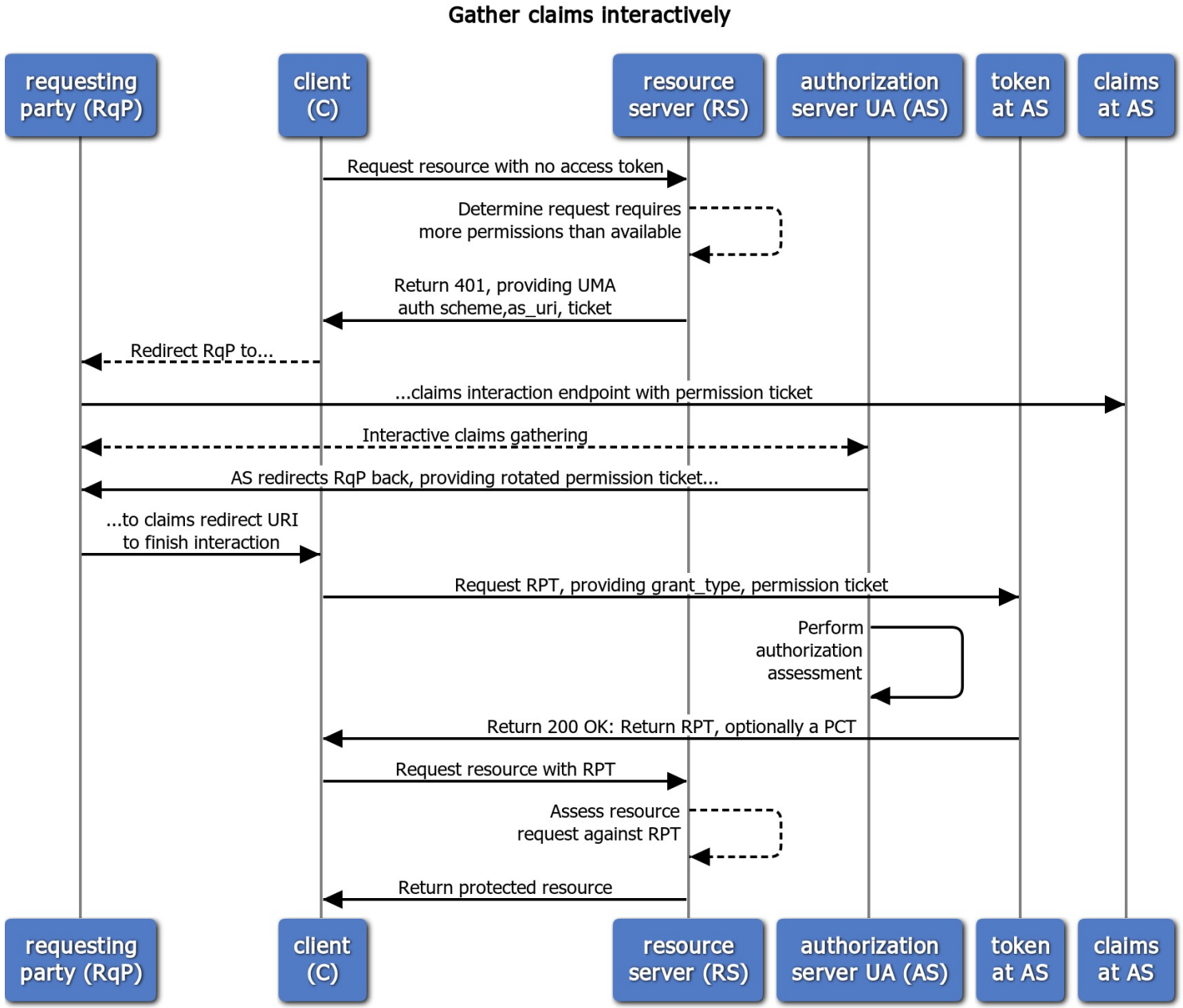
A claims interaction endpoint **must have been declared** in the discovery document to allow this flow

The AS mediates gathering of **claims from any source**

A key "metaclaim" to think about: **consent to persist claims**

A PCT potentially enables a **better RqP experience** next time; the AS can then re-assess using claims on hand

Resembles the **authorization code grant**, but can apply to non-unique identities and is repeatable and "buildable"



Federated authorization

A walkthrough of UMA federated authorization and its protection API

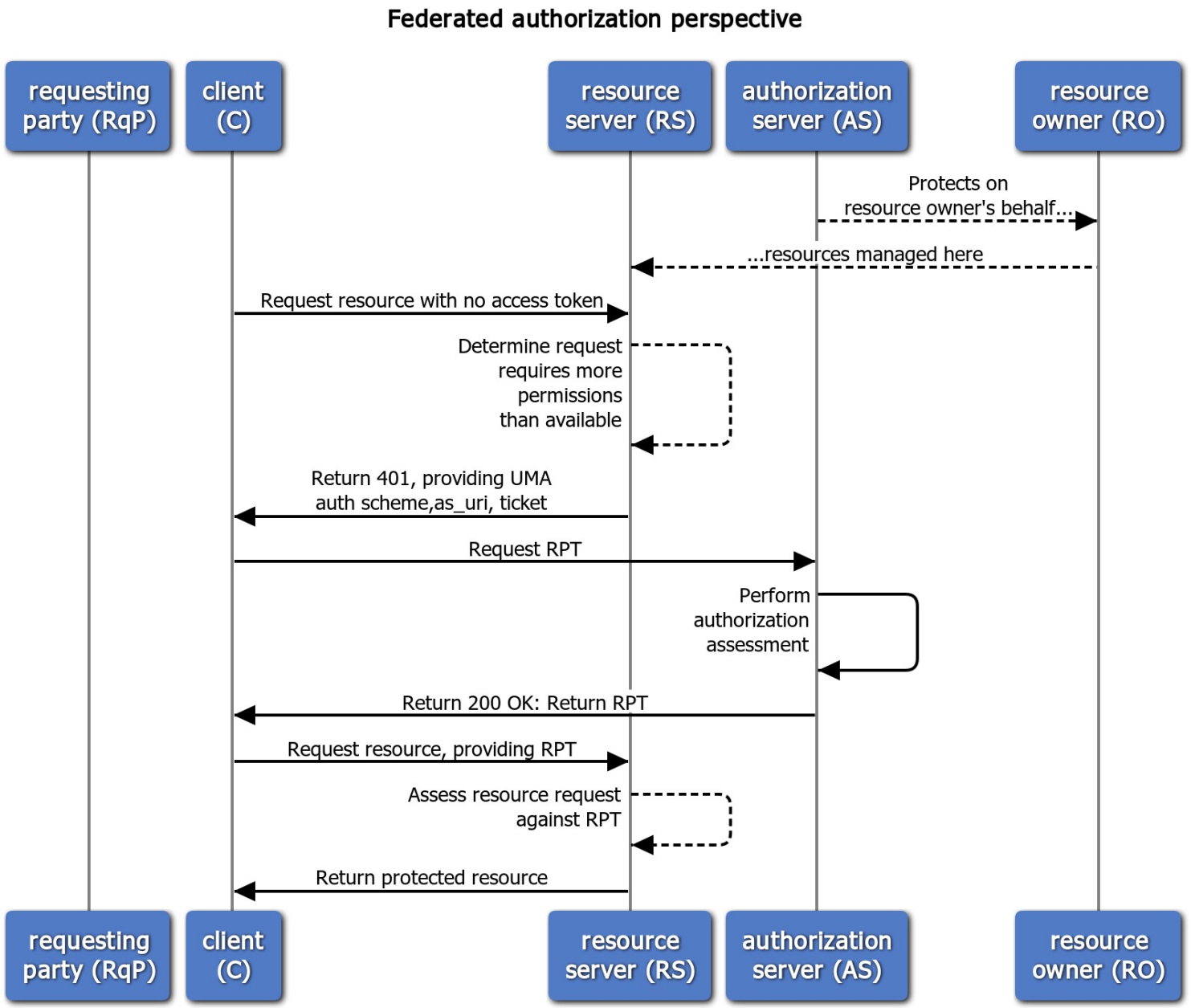
A new perspective on the UMA grant

How does the AS know when to **start protecting resources**?

How does the RS know what **ticket** the AS is associating with the RS's recommended **permissions**?

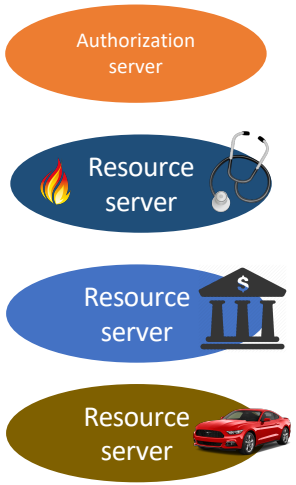
Is there anything special about **token introspection**?

Let's **standardize an interface** at the AS for these jobs



The protection API: how you *federate* authorization

- **RS registers resources:** This is required for an AS to be “on the job”
  - Scopes can differ per resource
  - Resource and scope metadata assist with policy setting interfaces
- **RS chooses permissions:** The RS **interprets** the client’s tokenless resource request and **requests** permissions from the AS
  - The AS then issues the initial permission ticket
- **RS can introspect the RPT:** UMA **enhances** the token introspection response object
- **RO controls AS-RS trust:** The protection API is **OAuth-protected**
  - The resource owner authorizes the scope **uma_protection**
  - The issued token is called the **PAT**

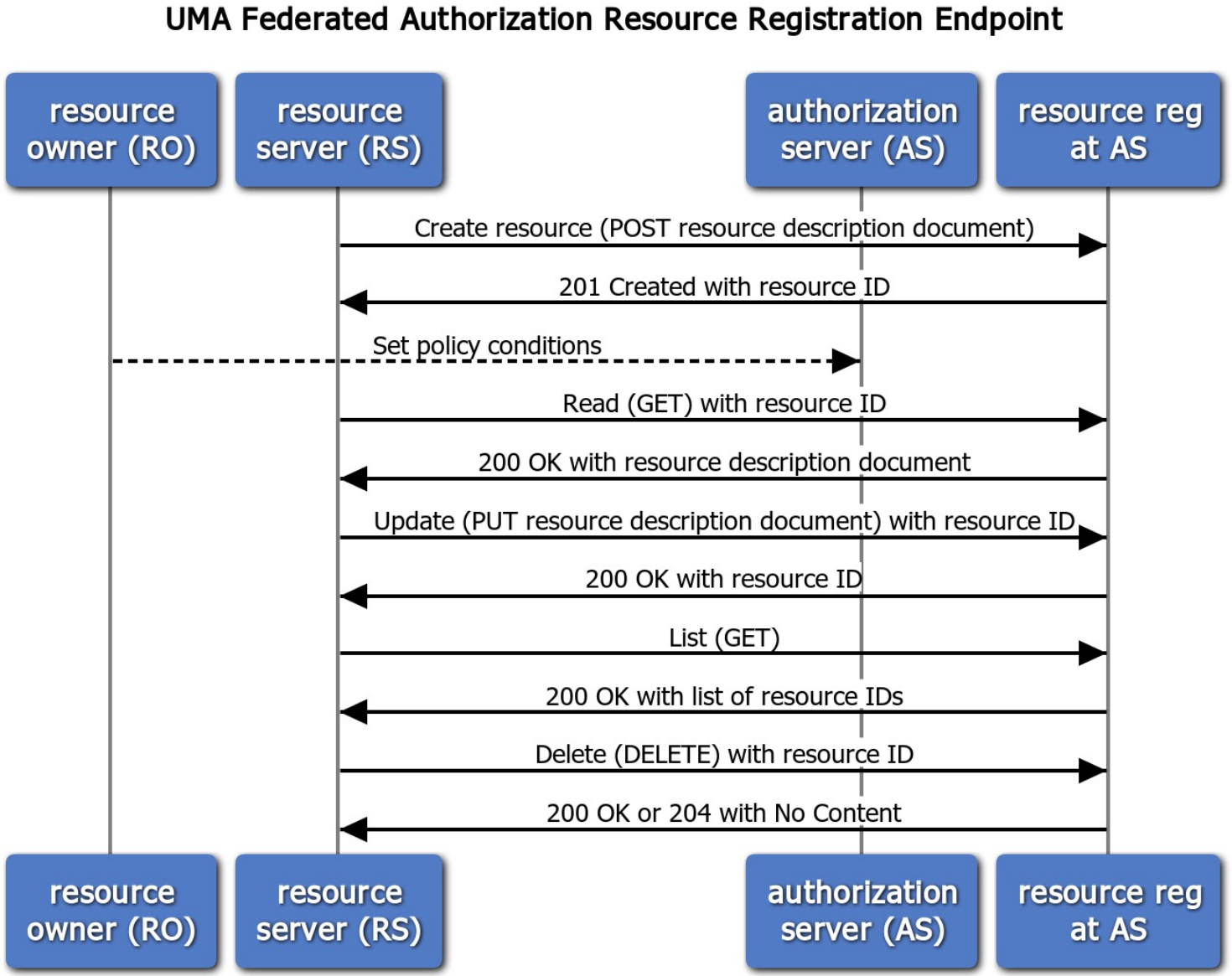


The resource registration endpoint

Registering a resource **puts it under protection**

Setting policies can be done **anytime after creation**

Deregistering a resource **removes it from protection**



Resource and scope registration

- The RS is authoritative for what its resource boundaries are
  - It registers them as JSON-based descriptions
  - There is a resource “type” parameter
- Scopes can be simple strings or URIs that point to description documents

Create request:

```
POST /rreg/ HTTP/1.1 Content-Type: application/json
Authorization: Bearer MHg3OUZEQkZBMjcx
...
{
  "resource_scopes":[
     "patient/*.read"
  ],
  "icon_uri":"http://www.example.com/icons/device23",
  "name":"Awesome Medical Device Model 23",
  "type":"https://www.hl7.org/fhir/observation.html"
}
```

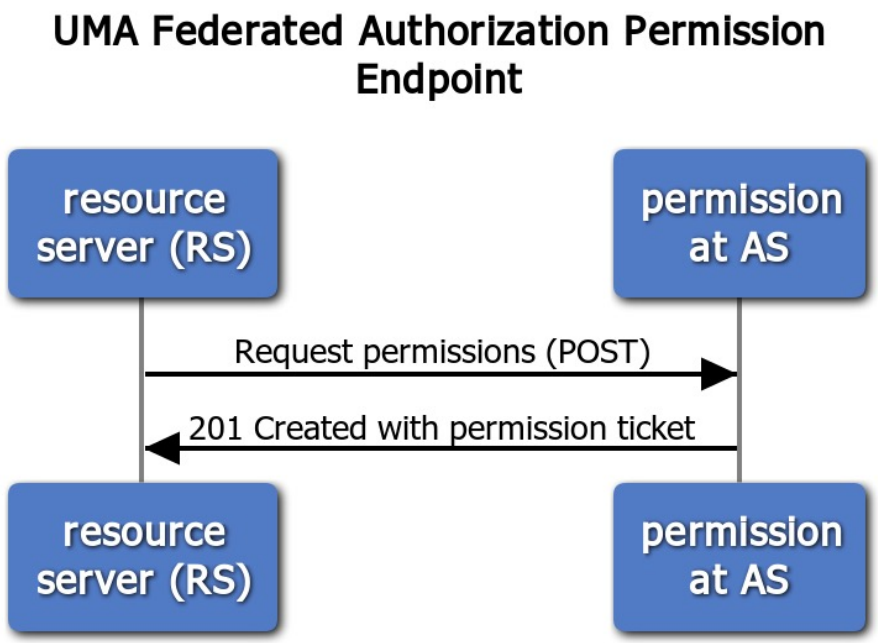
Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /rreg/rsrc1
...
{
  "_id":"rsrc1"
}
```

The permission endpoint

The RS **interprets** the client's tokenless (or insufficient-token) resource request

The RS must be able to tell from the client's request context **which RO and AS were meant**



Request:

```
POST /perm/ HTTP/1.1
Content-Type: application/json
Host: as.example.com
Authorization: Bearer MHg3OUZEQkZBMjcx
...
{
   "resource_id":"rsrc1",
   "resource_scopes":[
      "patient/*.read"
   ]
}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
...
{
   "Ticket":"016f84e8-f9b9-11e0-bd6f-
0021cc6004de"
}
```

The token introspection endpoint

UMA **enhances** the token introspection response object

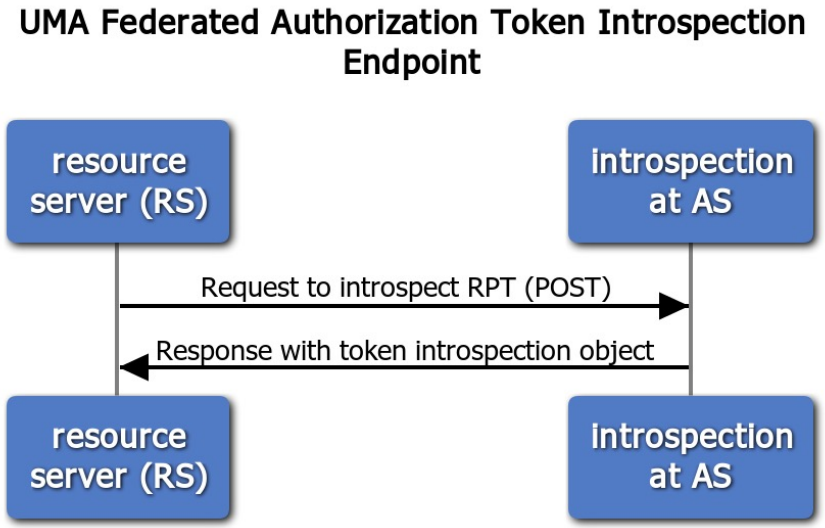
A **permissions claim** is added, with resource ID-bound scopes

Request:

```
POST /introspect HTTP/1.1
Host: as.example.com
Authorization: Bearer MHg3OUZEQkZBMjcx
…
token=mF_9.B5f-4.1JqM
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
…
{
   "active":true,
   "exp":1256953732,
   "iat":1256912345,
   "permissions":[
      {
         "resource_id":"rsrc1",
         "resource_scopes":[
            "patient/*.read"
         ],
         "exp":1256953732
      }
   ]
}
```



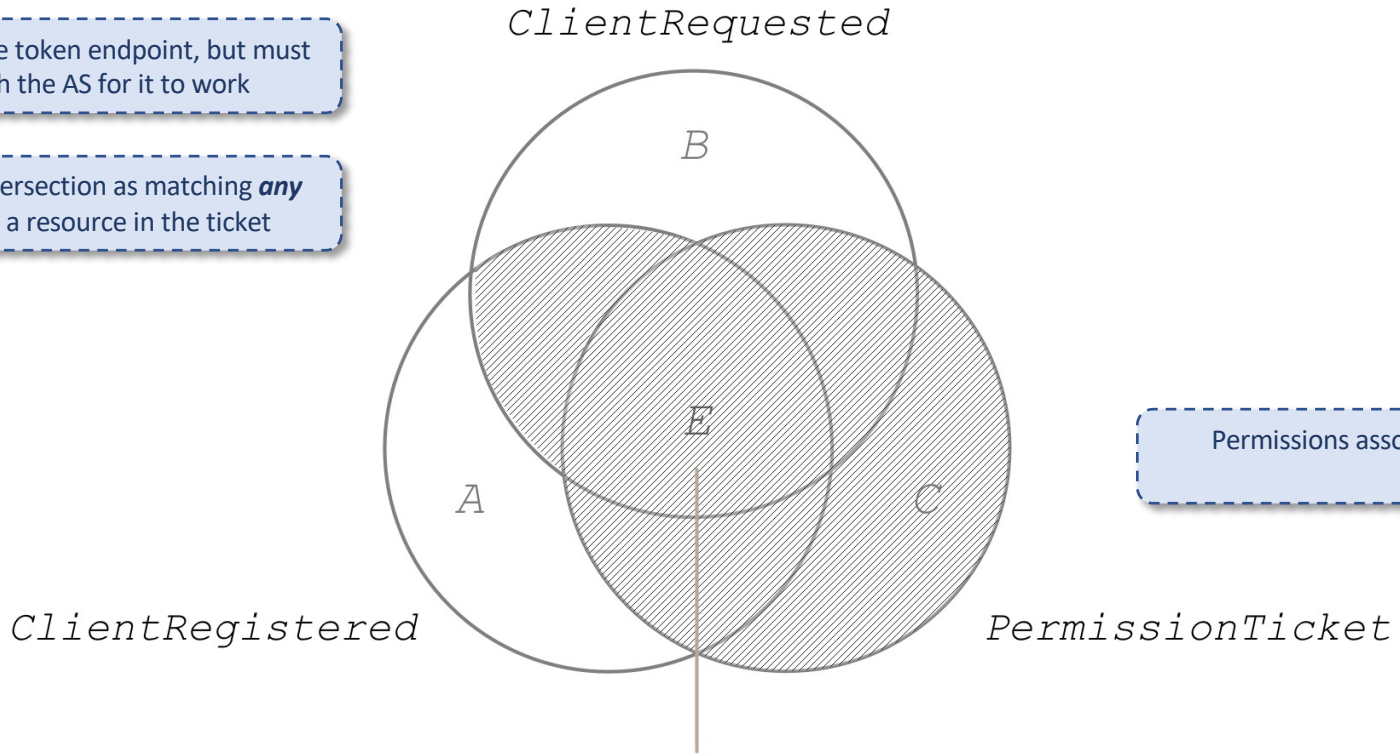
Authorization assessment

The UMA guardrails around issuing permissions

Authorization assessment: how the AS adheres to the RO's wishes in the larger context

The client can request scopes at the token endpoint, but must have **pre-registered** them with the AS for it to work

The AS treats the scopes in this intersection as matching ***any* available scope** associated with a resource in the ticket



Permissions associated with the ticket can **add** to total requested scopes

If authorization assessment results in only a subset of client-desired scopes, the AS can **choose to error**

Use case: Calendar sharing

The UMA protocol in action

Detailed use case

- Alice needs to coordinate a meeting with an important client Bob
- Alice wants to allow Bob to view her calendar so he can pick a time that works for both of them
- Bob can schedule over normal calendar events but not ones designated as high priority

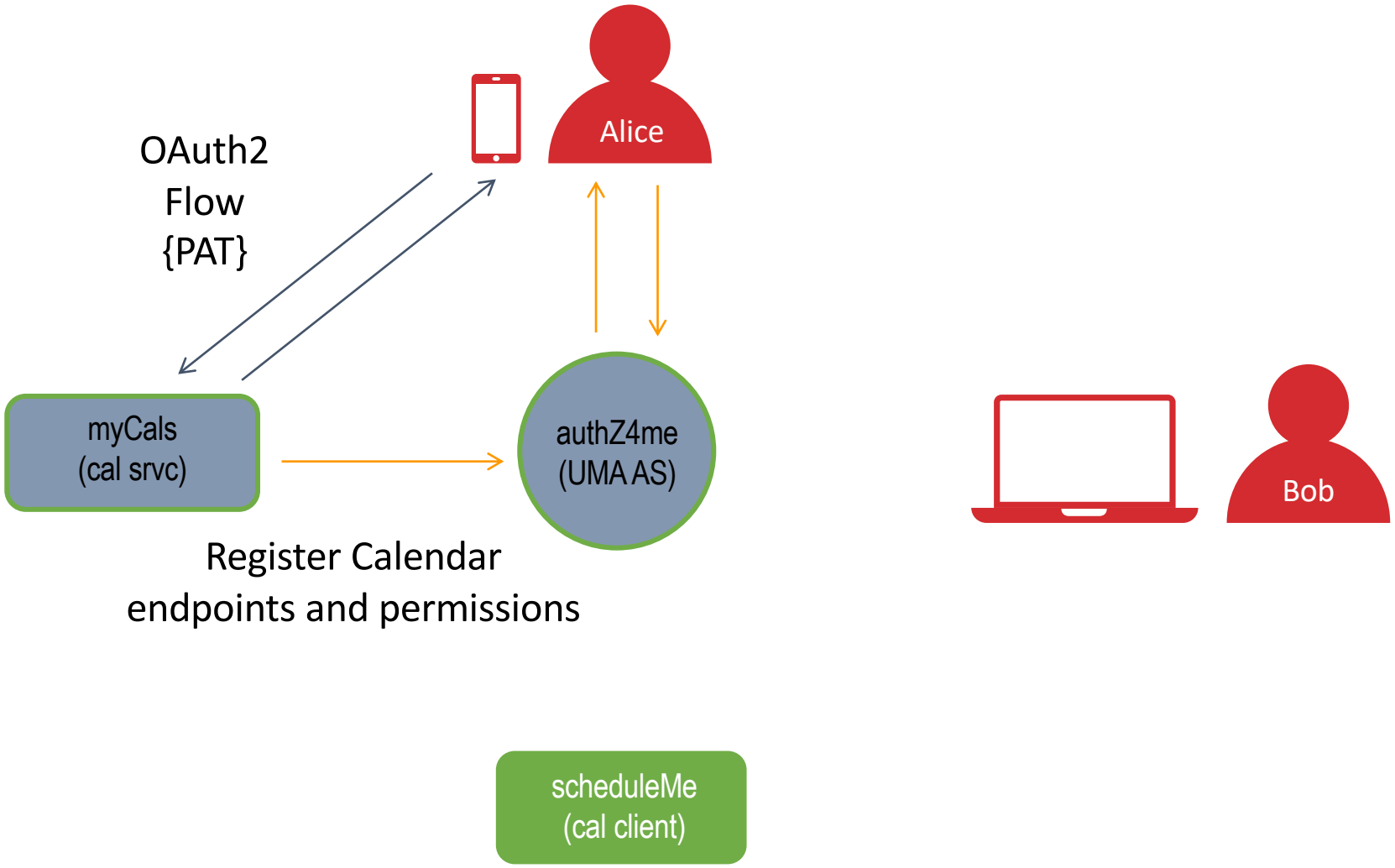
Use Case Actors

myCals
(cal srvc)

authZ4me
(UMA AS)

scheduleMe
(cal client)

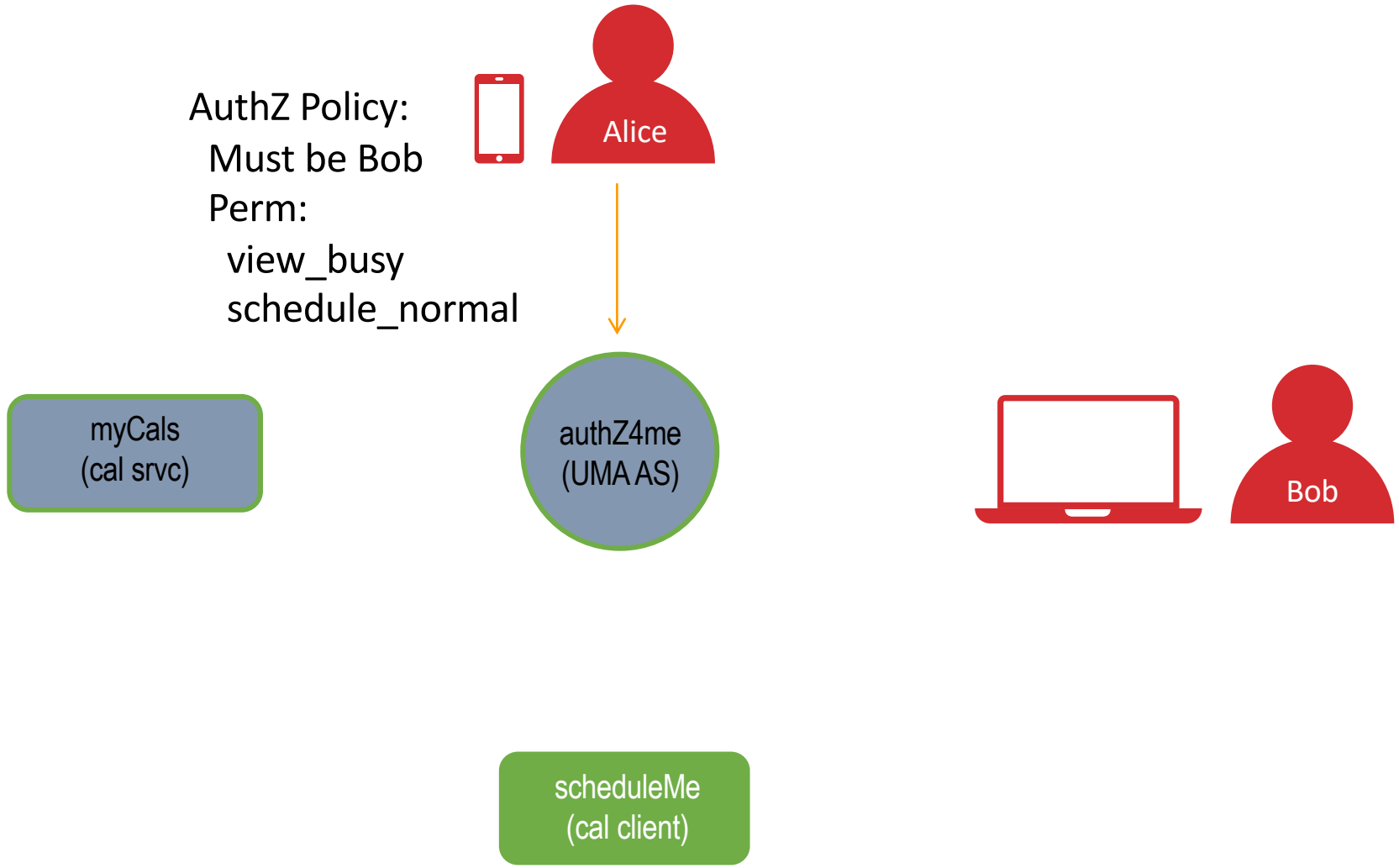
Alice registers protection for her calendar



Alice UMA protects her calendar

- Standard OAuth2 flow between myCals and authZ4me to obtain a “PAT”
- myCals registers Alice’s calendar
  - https://mycals.example.com/cal/alice/work
    - View, view_busy, delete, update, download, publish
    - Schedule_all, schedule_normal

Alice defines authorization policy



Alice sends Bob an email

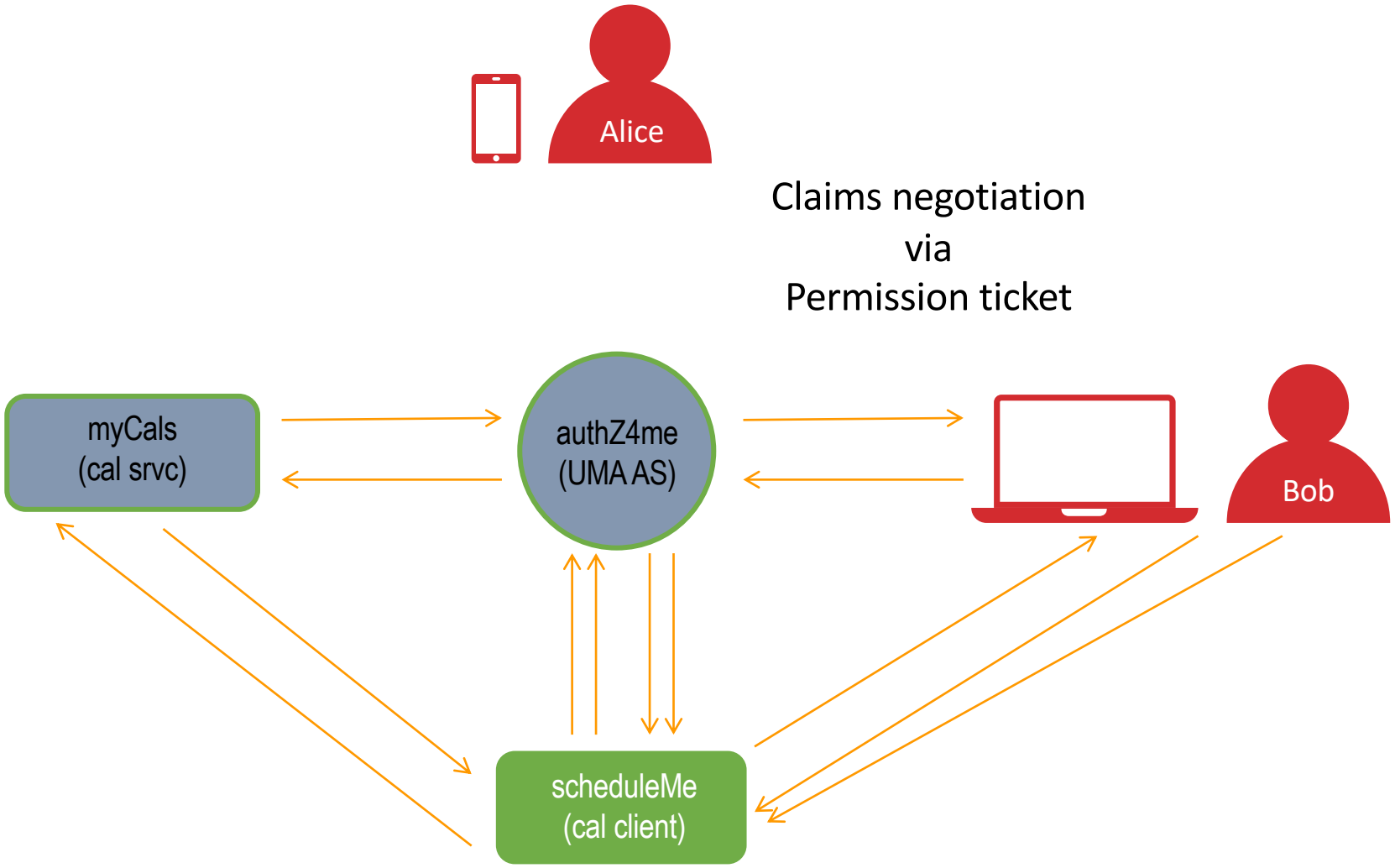
Hi Bob,

Please view my calendar and schedule the meeting we spoke about today.

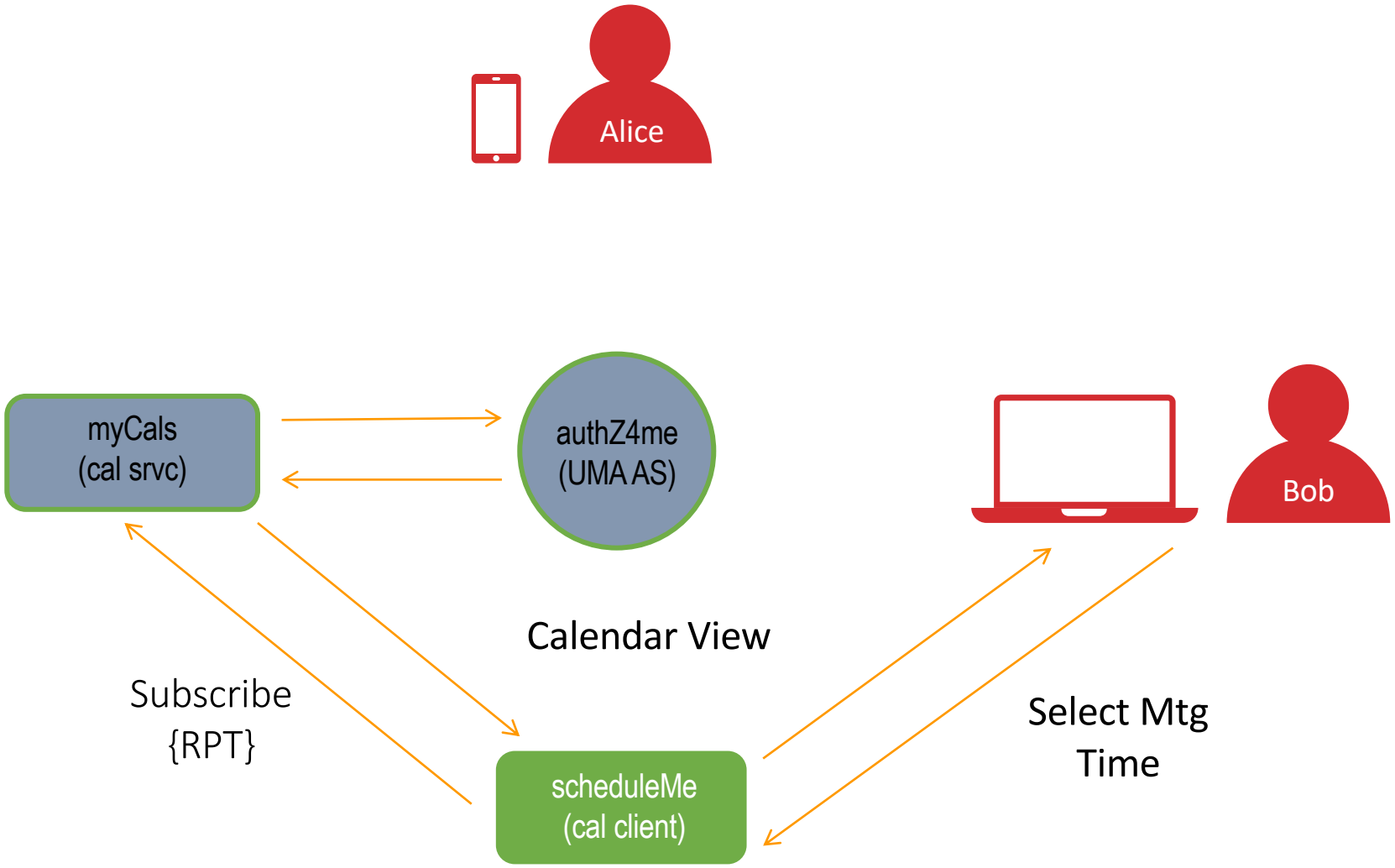
https://mycals.example.com/cal/alice/work

Thanks,
Alice

Bob meets claims to access Alice's calendar



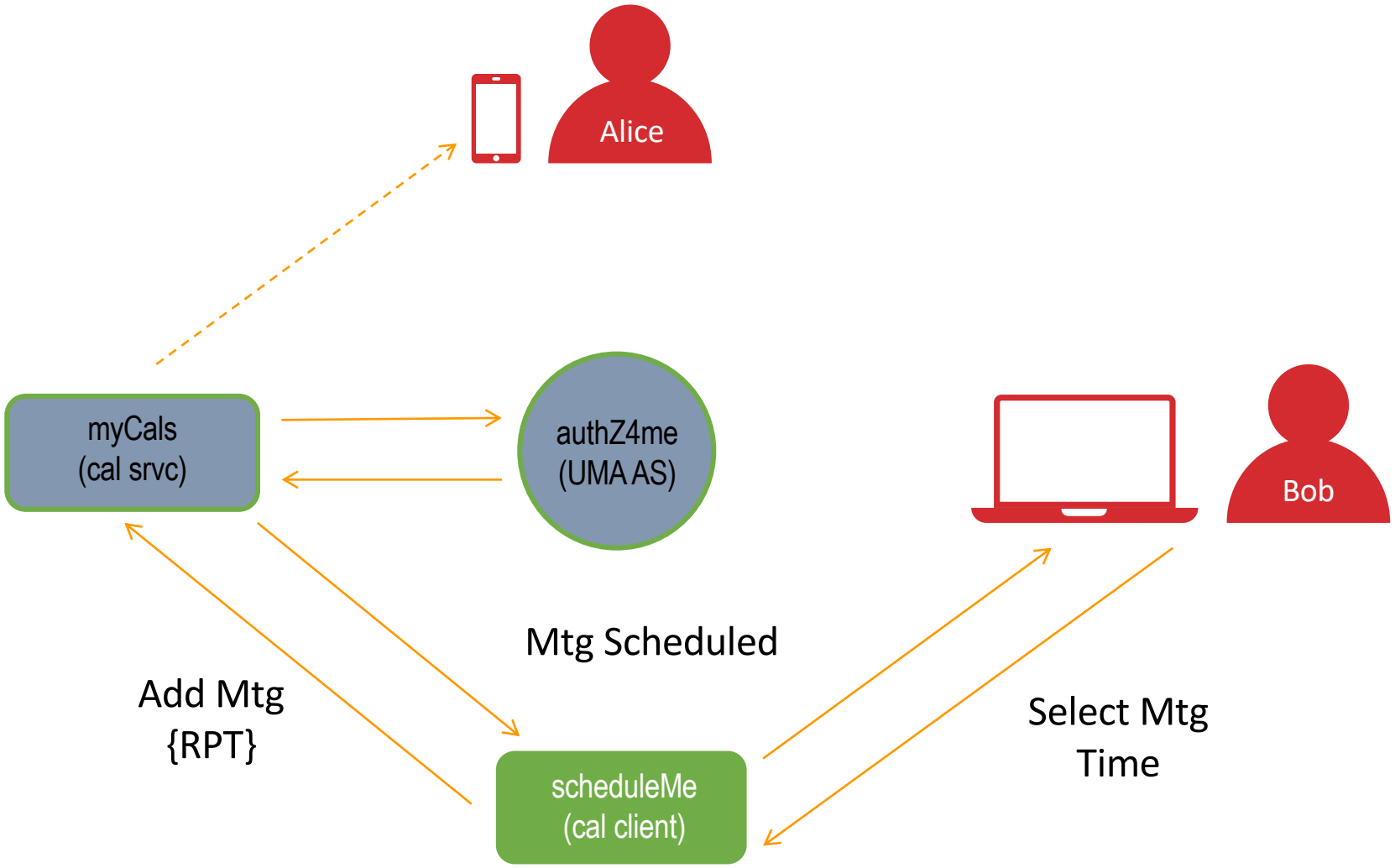
Bob subscribes to Alice's calendar



Bob schedules a meeting with Alice

- Scheduleme POST's to
  - https://mycals/cal/alice/work/meeting
    - Date, time, location
    - Passes RPT in the HTTP Authorzation header

Meeting added to Alice's calendar



Privacy and “BLT” implications

The bigger business-legal-technical picture

Relevance for privacy beyond “empowered flows”

- Features relevant to privacy regulations (GDPR, CCPA, OB, PSD2, CDR, HHS ONC info blocking rules...):
  - Asynchronous resource owner control of grants
  - Enabling resource owner to monitor and manage grants from a “dashboard”
  - Auditability of grants (consent) and PAT-authorized AS-RS interactions
- Work is well along on an UMA business model
  - Modeling real-life data-sharing relationships and legal devices
  - Technical artifacts are mapped to devices
  - Goal: tear down artifacts and build up new ones in response to state changes

UMA implications…

…for the client

- Simpler next-step handling at every point

…for the RS

- Standardize management of protected resources

…for the RO

- Control data sharing/device control
- Truly delegate access to other parties using clients

…for the AS

- Offer interoperable authorization services
- Don’t have to touch data to protect it

…for the RqP

- Seek access to a protected resource as oneself

…for the client operator

- Distinguish identities of resource owners from mere users

…for the resource server operator

- Externalize authorization while still owning API/scopes

…for the resource rights admin

- Manage sharing on behalf of data subjects, not just for oneself

…for the authorization server operator

- Prove what interactions took place or didn’t

…for the requesting agent

- Revoke access (or request it) to someone else’s assets

Join us!
Thank you!
Questions?

George Fletcher and Eve Maler

Kantara Initiative UMA Work Group

@UMAWG | tinyurl.com/umawg

IIWXXXII | 20 Apr 2021

kantara™ INITIATIVE