



User-Managed Access (UMA) 101

George Fletcher, Kantara Initiative UMA Work Group

@UMAWG | tinyurl.com/umawg

IIWXXX | 20 Oct 2020



Topics

- Overview
- UMA in action
- The technical big picture
- The UMA grant
- Federated authorization
- Authorization assessment
- Privacy and business-legal-technical implications

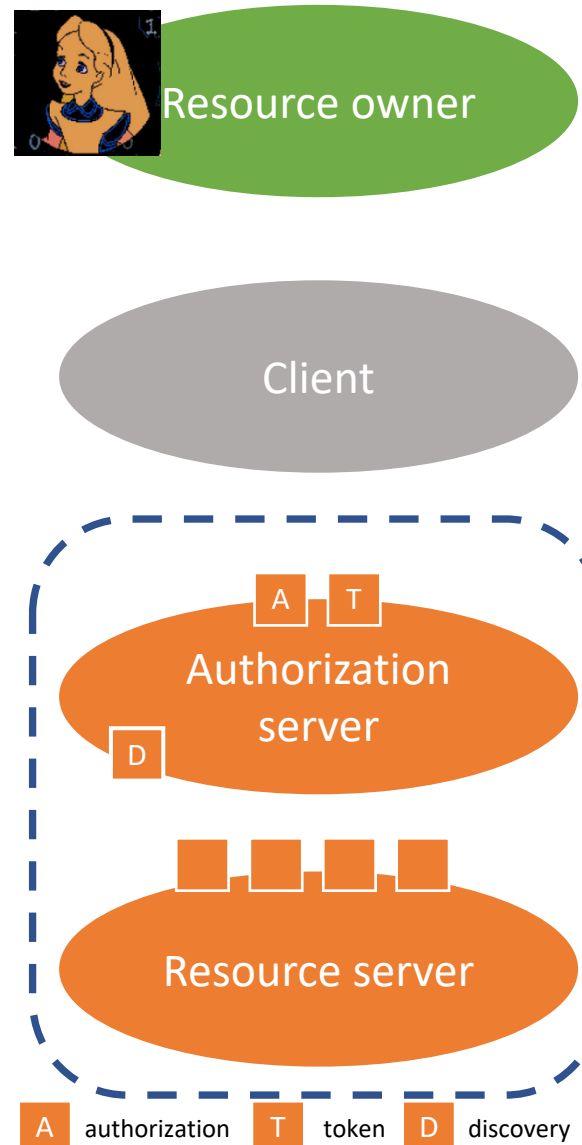
Overview

What UMA adds to OAuth

OAuth enables constrained delegation of access to apps

Benefits:

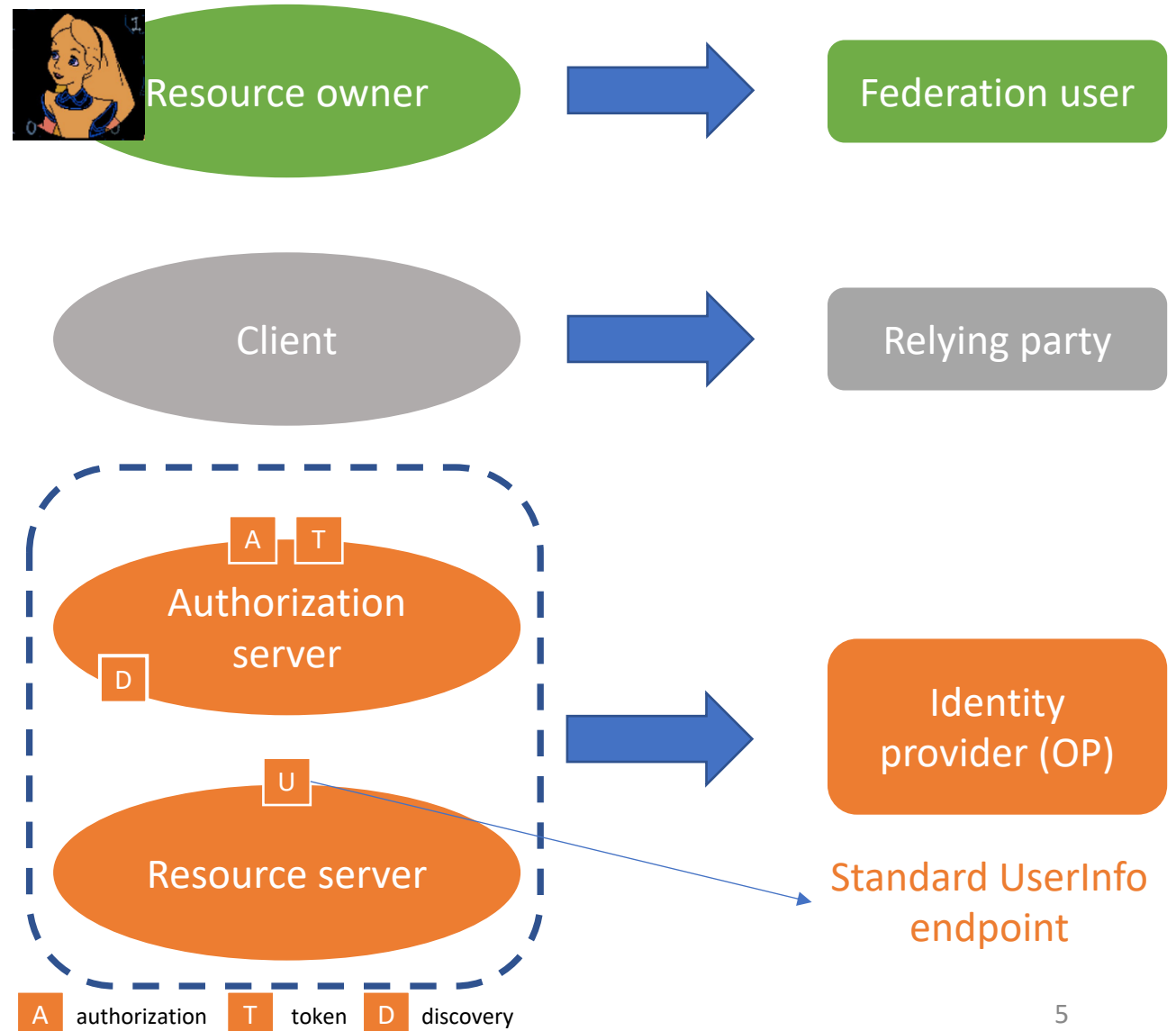
- Flexible, clever API security **framework**
- Alice can **agree** to app connections and also **revoke** them



OpenID Connect does modern-day federation

Benefits:

- **Layers** identity/ authentication tech with delegation/ authorization tech
- **Translates** federated identity for mobile and the API economy

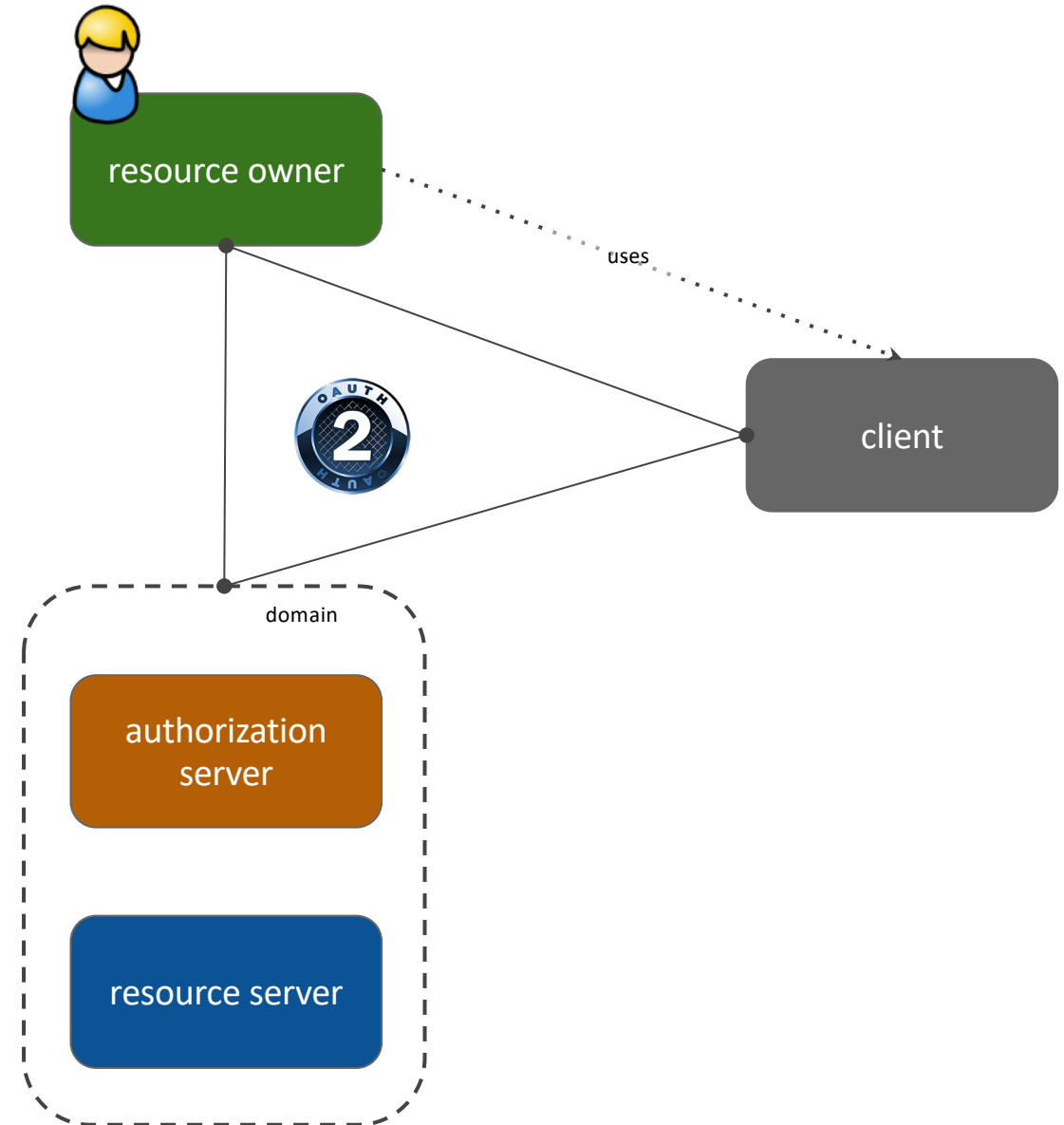


OAuth and UMA

"ALICE-TO-SELF" SHARING

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

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

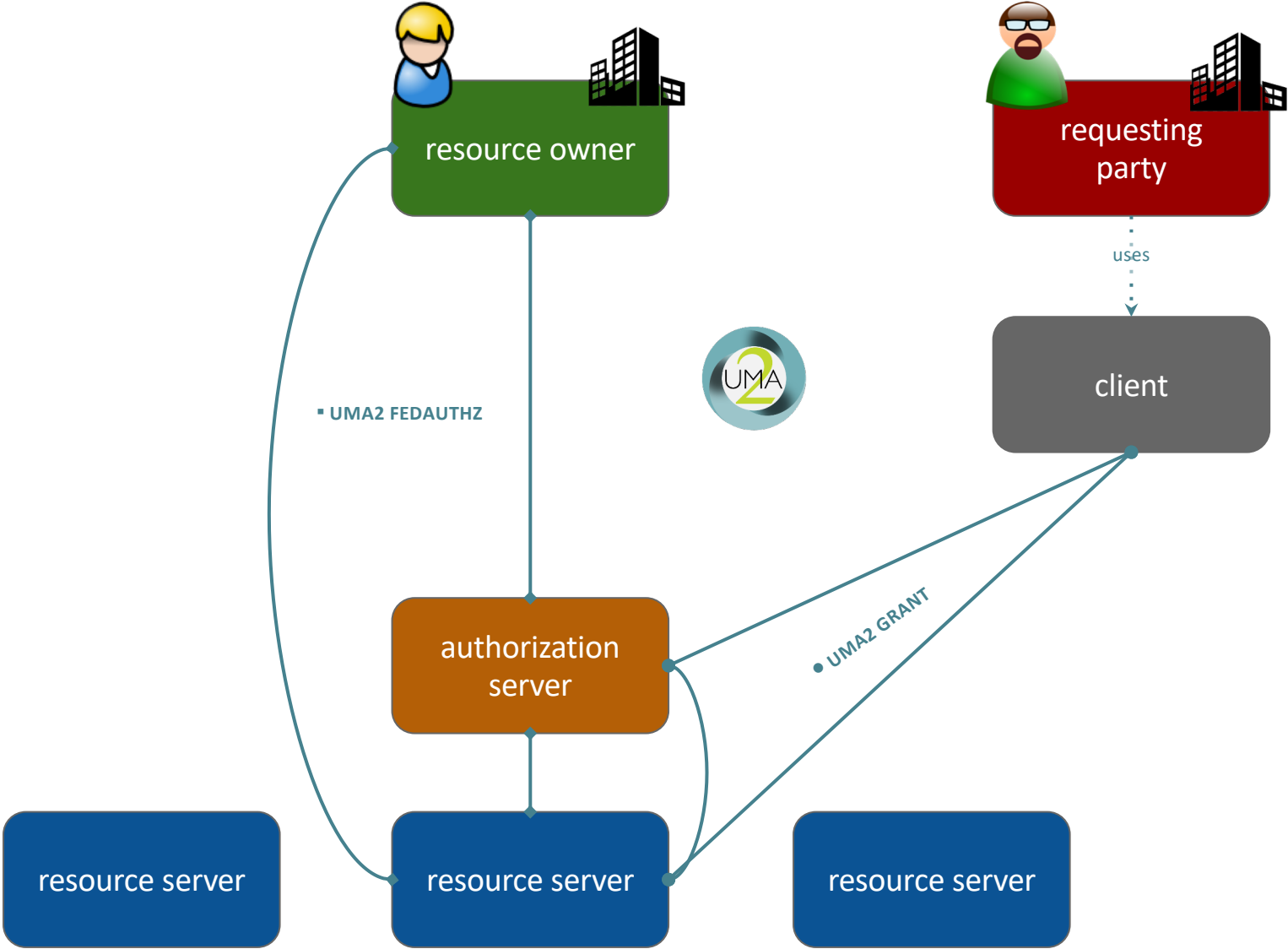


OAuth and UMA

“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



can be in different domains

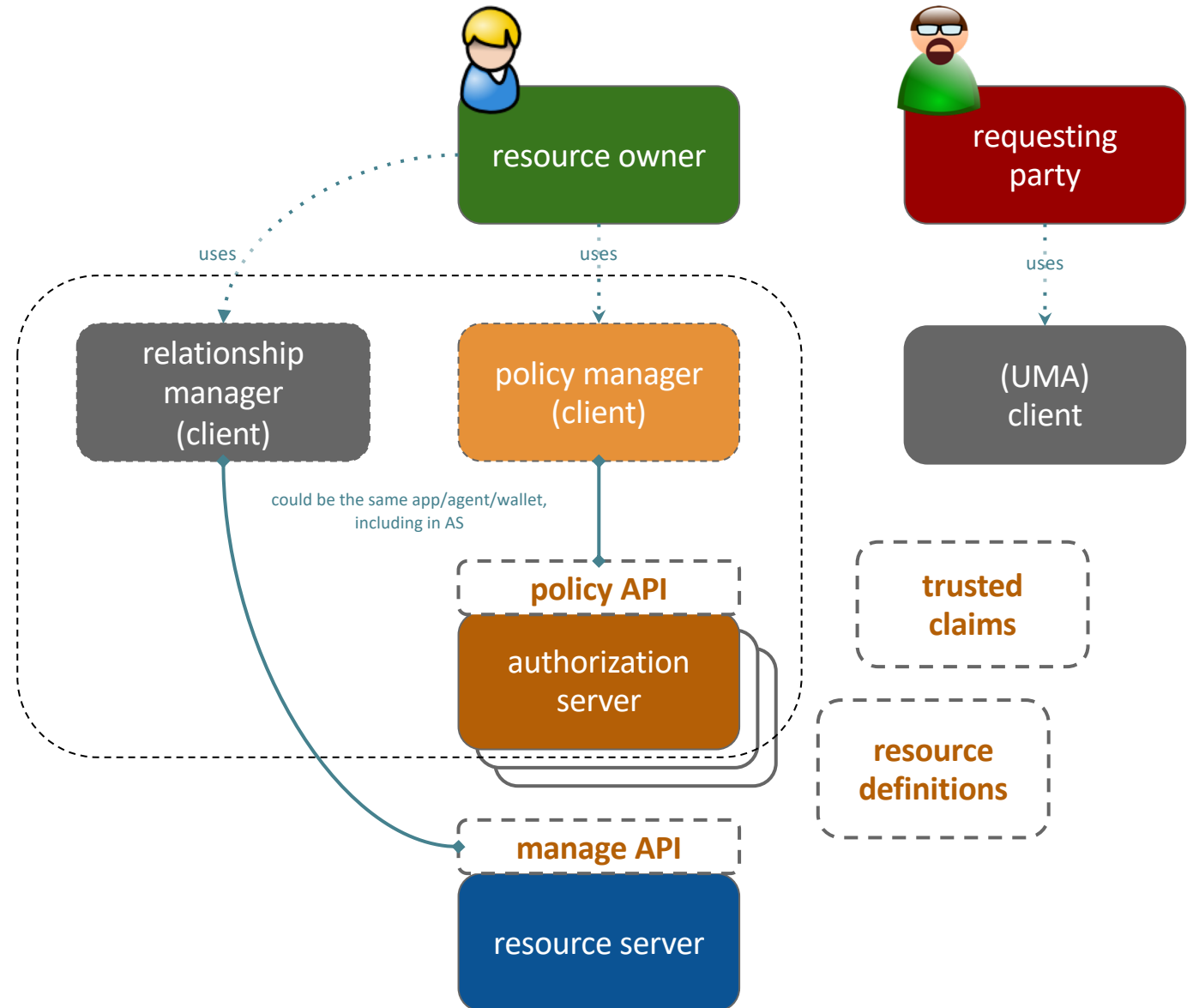
UMA and New Work

Policy Manager extension: AS can delegate policy handling; RO can choose how to manage policy; RO can aggregate management across AS's at one trusted place

Manage API extension (TBD): RO can manage details of resource registration in an interoperable way

Resource definitions (extension? TBD): RS can register API resource and scope templates for UMA clients to follow, to increase interop as well as extent of AS abilities to manage client communities of trust

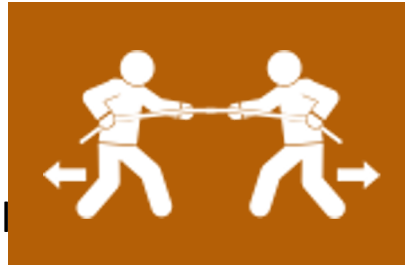
Trusted claims (TBD): AS delegates claims collection about RqP to other AS's in an interoperable way, with predictable set math



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



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

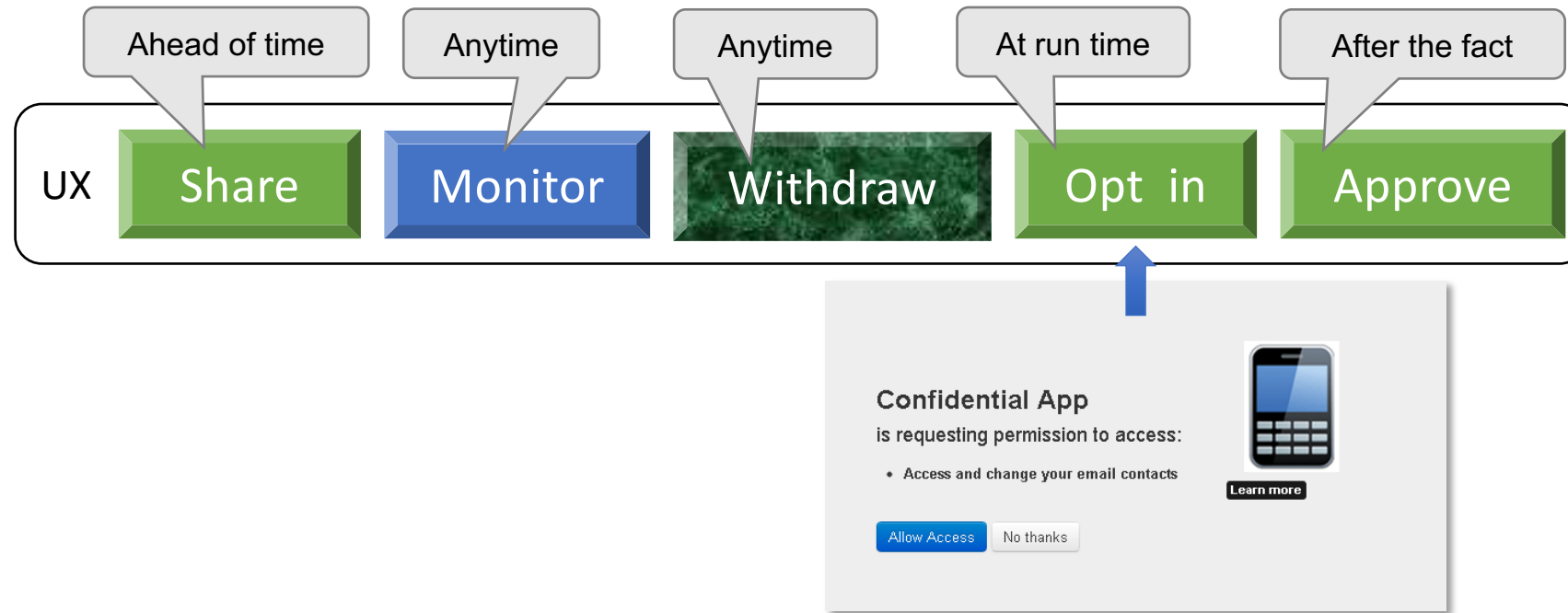
UMA enables permissioning that is **asynchronous**

- 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

UMA user experience opportunities

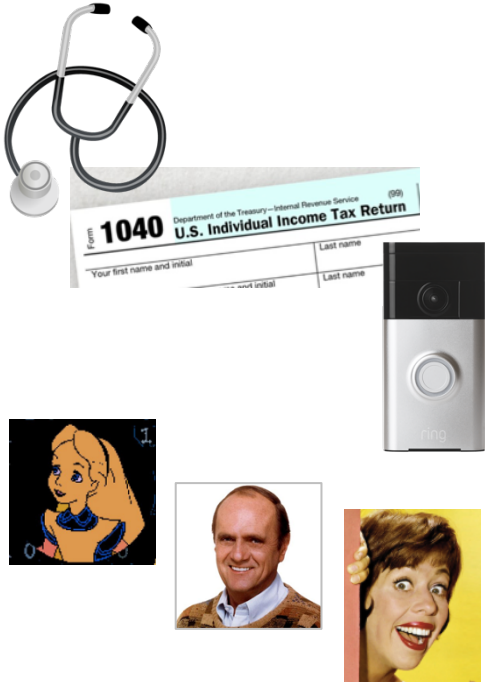


Benefits for service providers: a summary

True secure delegation; no password sharing



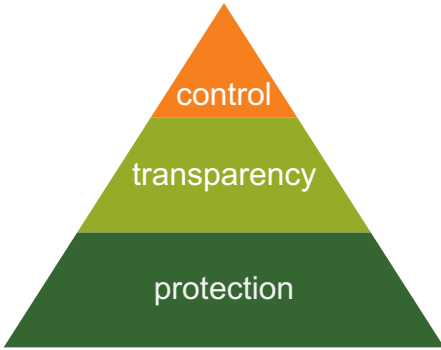
Scale permissioning through self-service



API-first protection strategy

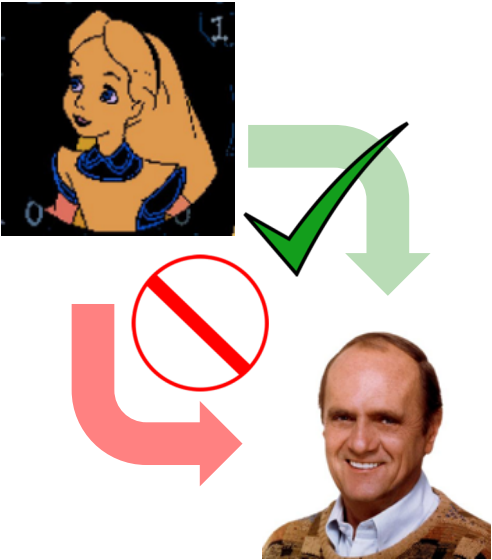


Foster compliance through standards

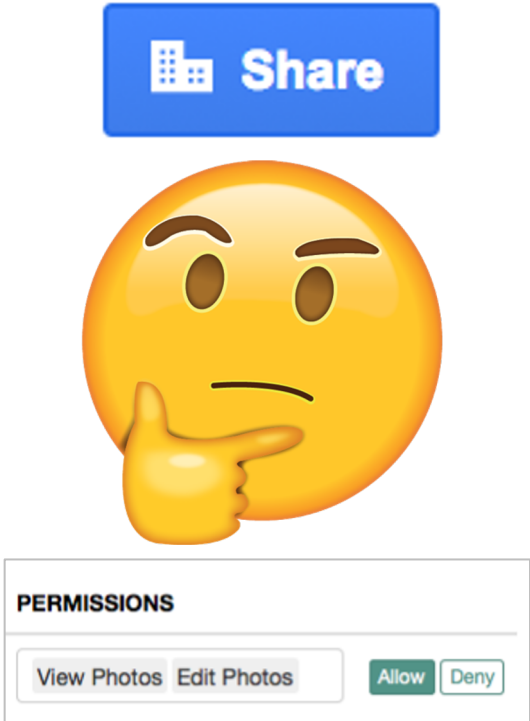


Benefits for individuals: a summary

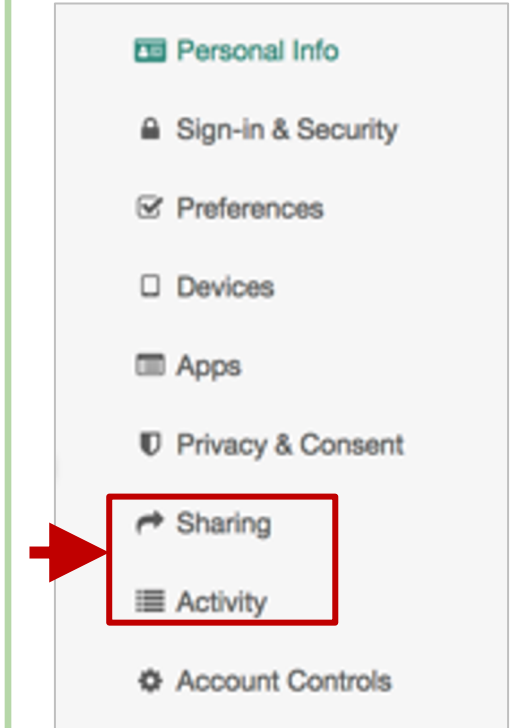
Choice in sharing with other parties



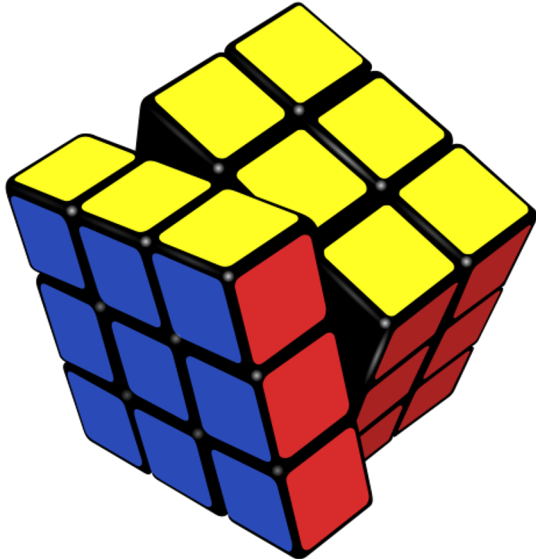
Convenient sharing/approval with no outside influence



Centralizable monitoring and management



Control of who/what/how at a fine grain

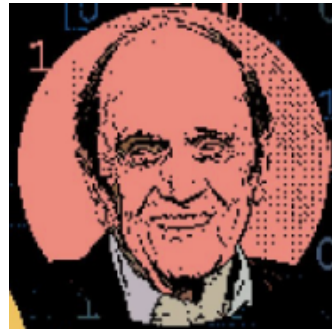
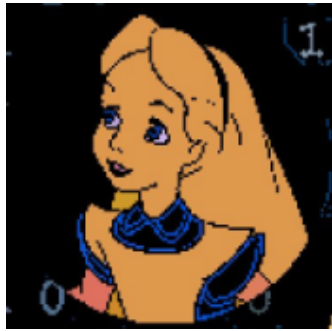


Typical use cases

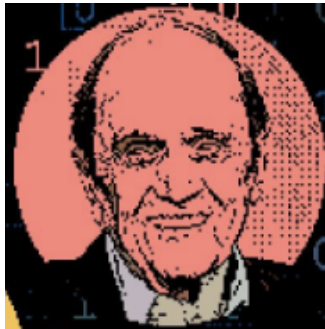


Profiles / references:

- Health Relationship Trust
- UK Pensions Dashboard
- OpenMedReady Alliance



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



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



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

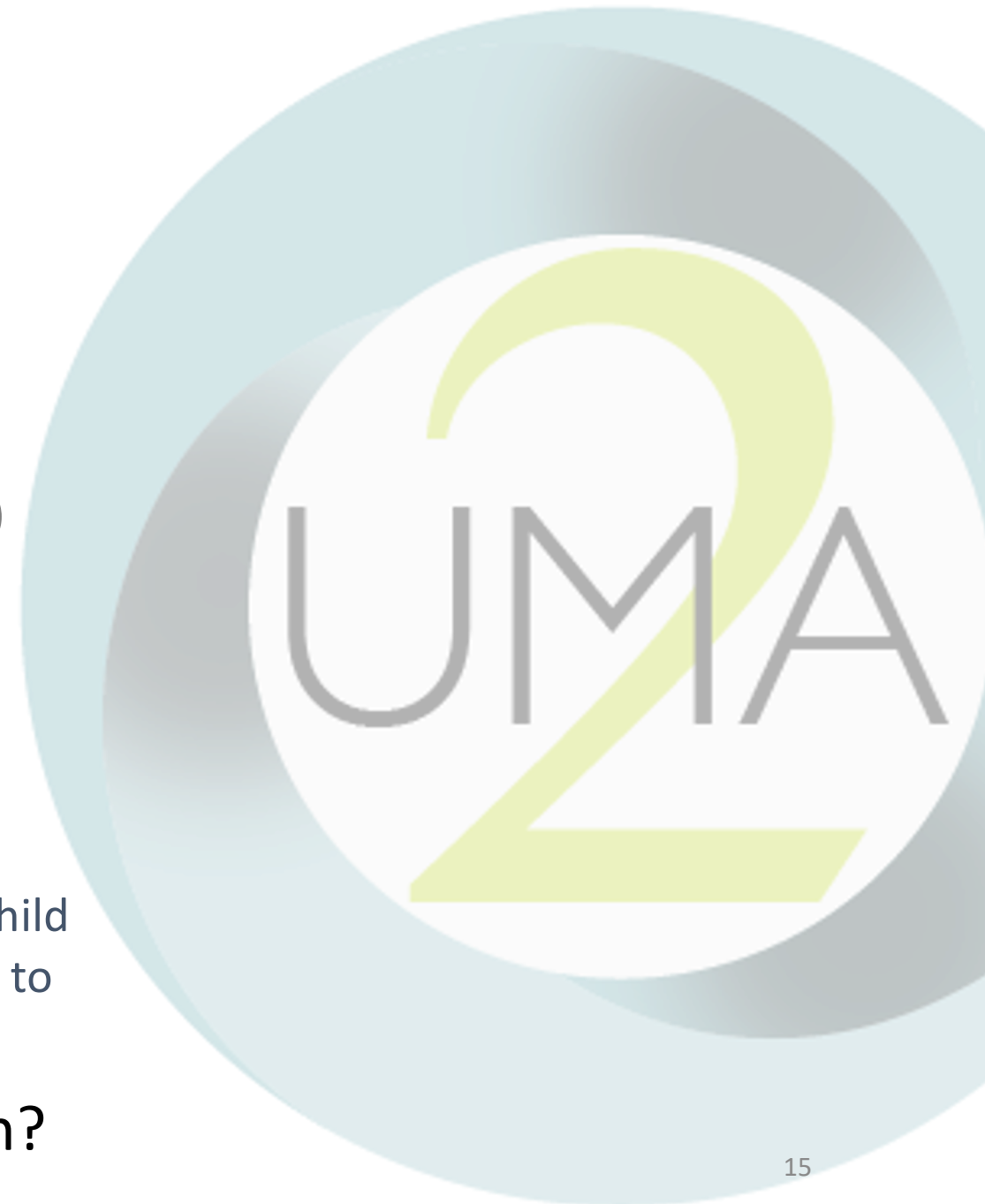
Known implementations

(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...

UMA in a nutshell

- Developed at **Kantara Initiative**
 - V1 done in 2015, V2 done in 2018
- Leverages existing **open standards**
 - OAuth2
 - OpenID Connect and SAML (optional but popular)
- Profiled by multiple **industry sectors**
 - Financial, healthcare
- UMA business model effort supports **legal licensing** for personal digital assets
 - Example: Mother (guardian) manages sharing for child (data subject); child “ages in” to consent and starts to manage sharing herself
- Some **1:1 interop testing** done; more soon?



UMA in action

A couple of sample implementations

Lush Group

HealthyMePHR – also ShareMedData

A screenshot of a web interface for patient authorization. At the top, it says "I, Alice Patient, authorize". Below this, there are two dropdown menus: "HealthyMePHR" and "Dr. Erica, Lush Medical". The section is titled "Medical Information" with the instruction "Select how you would like to share your medical information". There are two radio button options: "SHARE ALL information in my medical Record" (which is selected) and "SHARE SPECIFIC medical data sets". Below this is the "Consent Term" section, which says "Enter a start and end date during which your medical data will be shared". It has two input fields: "Consent Start" with the value "31 May 2017" and "Consent End" with the value "31 December 2019". At the bottom, there are four buttons: "CANCEL", "SAVE", "SHARE" (highlighted in blue), and "REVOKE".


- 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 from 23 Apr 2019](#)

ForgeRock Identity Platform

Profile and Privacy Management Dashboard – also Access Management module




ROCK 'N' ROLL SUPERMARKET Shop Coupons Recipes 


MY ACCOUNT

- Personal Info
- Sign-in & Security
- Preferences
- Trusted Devices
- Authorized Apps
- Privacy & Consent
- Sharing**
- Activity
- Account Controls

Sharing

Manage your shared resources.

	Party Food Shopping List	Shared with 2 people
	Shopping List	Not shared
	Oliver's Bday Wish List	Shared with 2 people





ROCK 'N' ROLL SUPERMARKET Shop Coupons Recipes 

MY ACCOUNT

- Personal Info
- Sign-in & Security
- Preferences
- Trusted Devices
- Authorized Apps
- Privacy & Consent
- Sharing
- Activity**
- Account Controls

Activity

Account actions you've taken in the last 28 days.

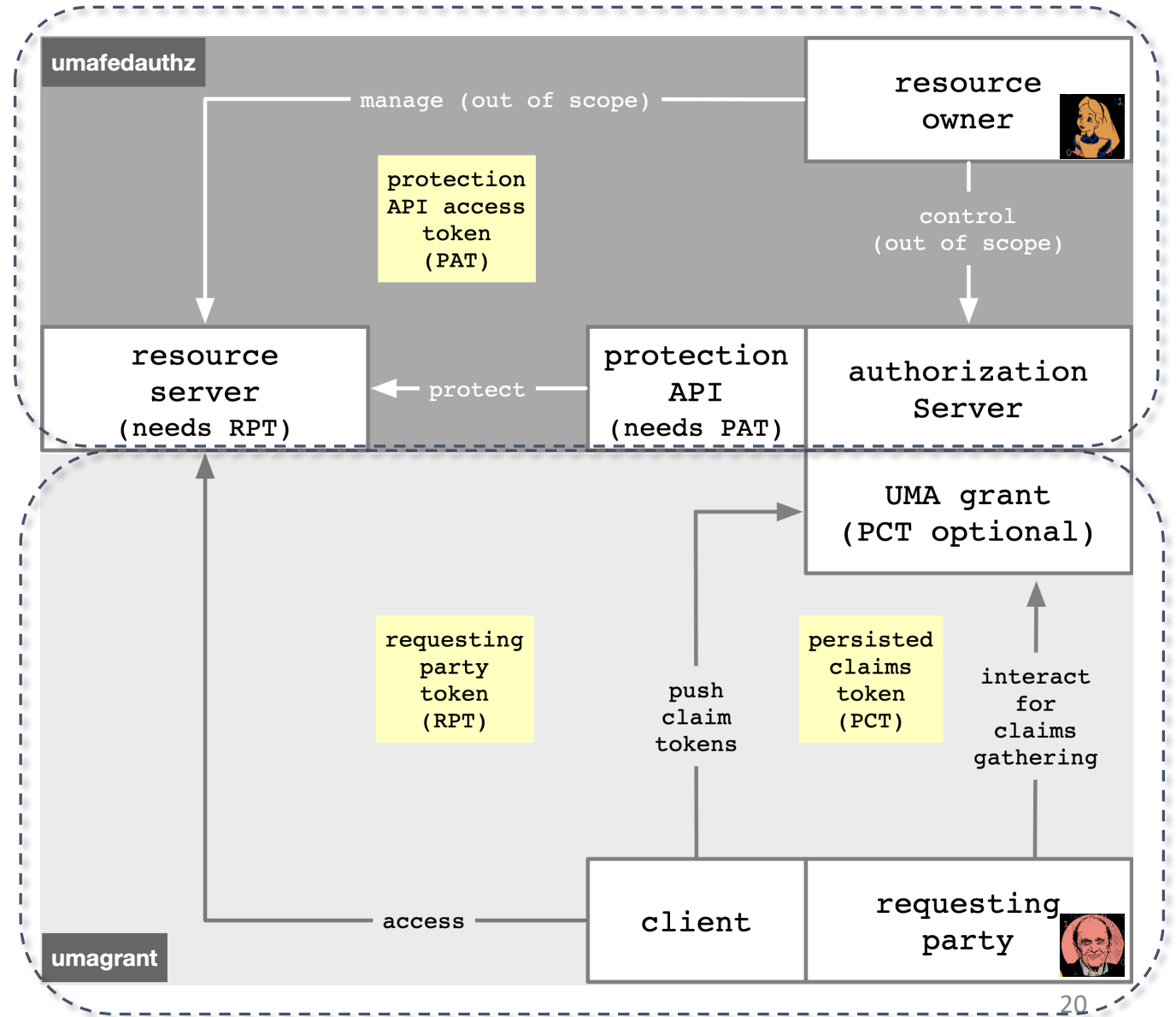
	Party Food Shopping List You updated sharing	9 hours ago
	Party Food Shopping List ed.enduser@example.com viewed	1 day ago
	Oliver's Bday Wishlist You allowed access to ed.enduser@example.com	1 day ago
	Oliver's Bday Wishlist edna.enduser@example.com shared	July 2, 2017

The technical big picture

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

The marvelous spiral of delegated sharing, squared

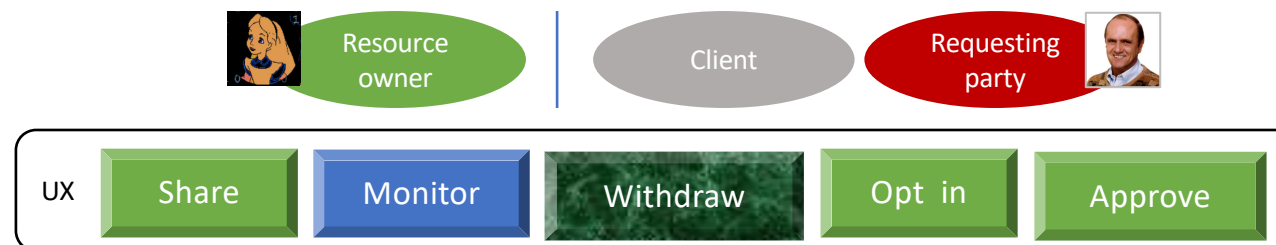
1. The **UMA grant of OAuth** enables Alice-to-Bob delegation
2. **UMA standardized an API for federated authorization** at the AS to make it centralizable
3. There are **nicknames** for enhanced and new tokens to keep them straight



The UMA extension grant adds...

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

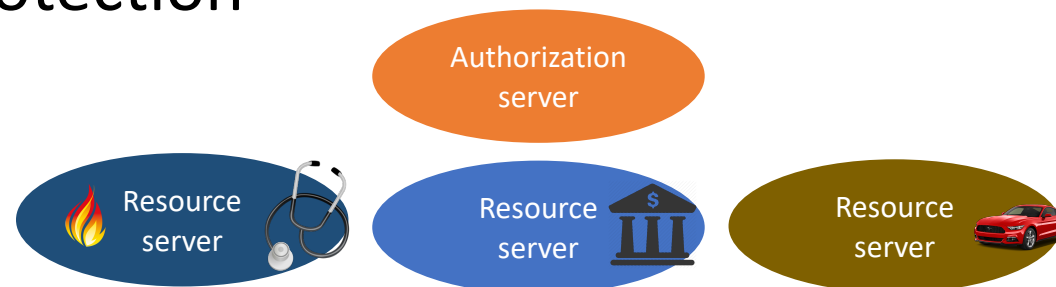
- **Party-to-party:** Resource owner authorizes protected-resource access to clients used by requesting parties
- **Asynchronous:** Resource owner interactions are asynchronous with respect to the authorization grant
- **Policies:** Resource owner can configure an AS with rules (policy conditions) for 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



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**

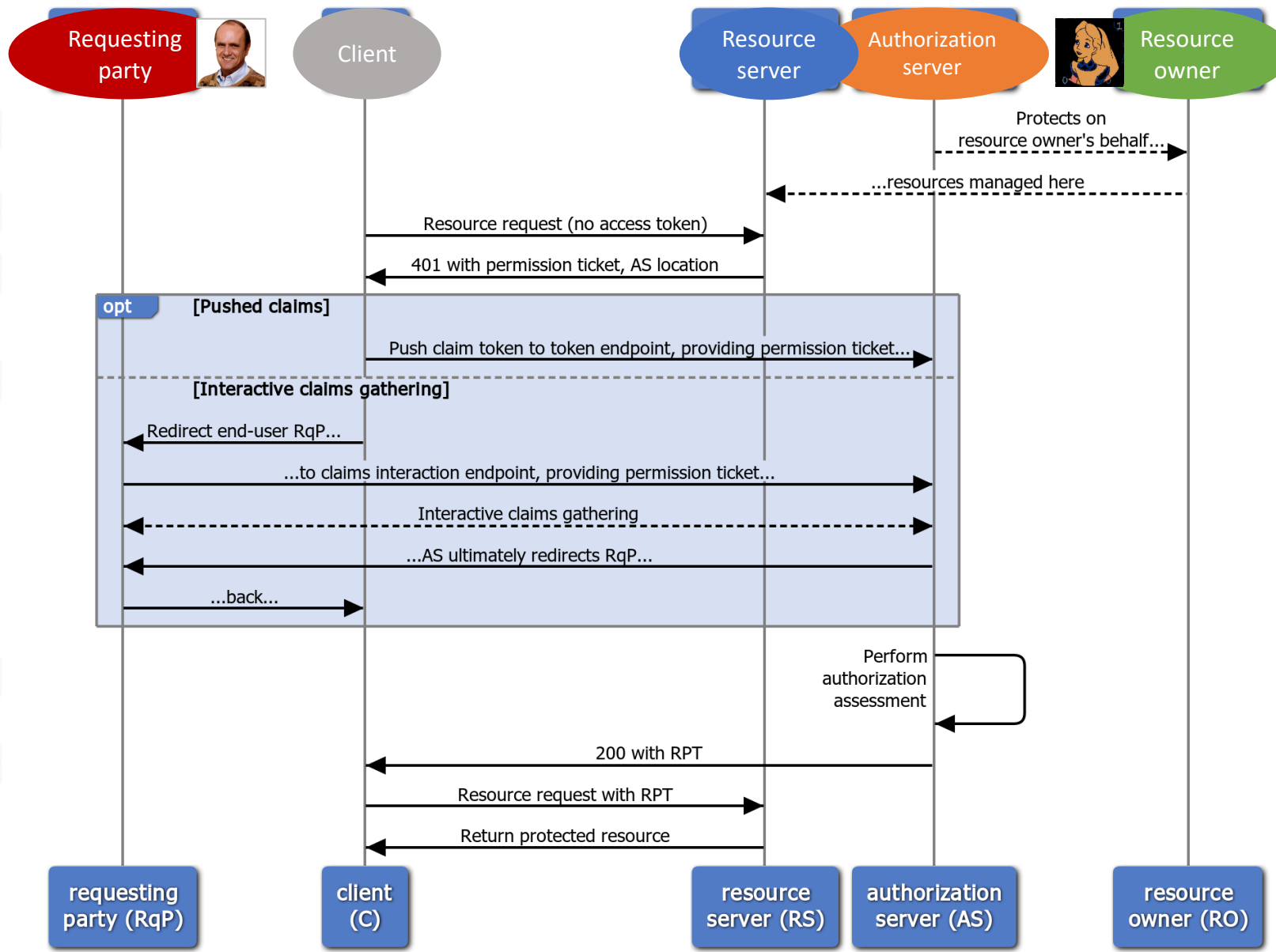
The RS provides a **permission ticket** and allows **AS discovery**

There are two **claims collection options** for meeting policy

Authorization assessment and token issuance has **guardrails**

RPTs can be **upgraded, revoked, introspected, and refreshed**

UMA2 grant basics



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

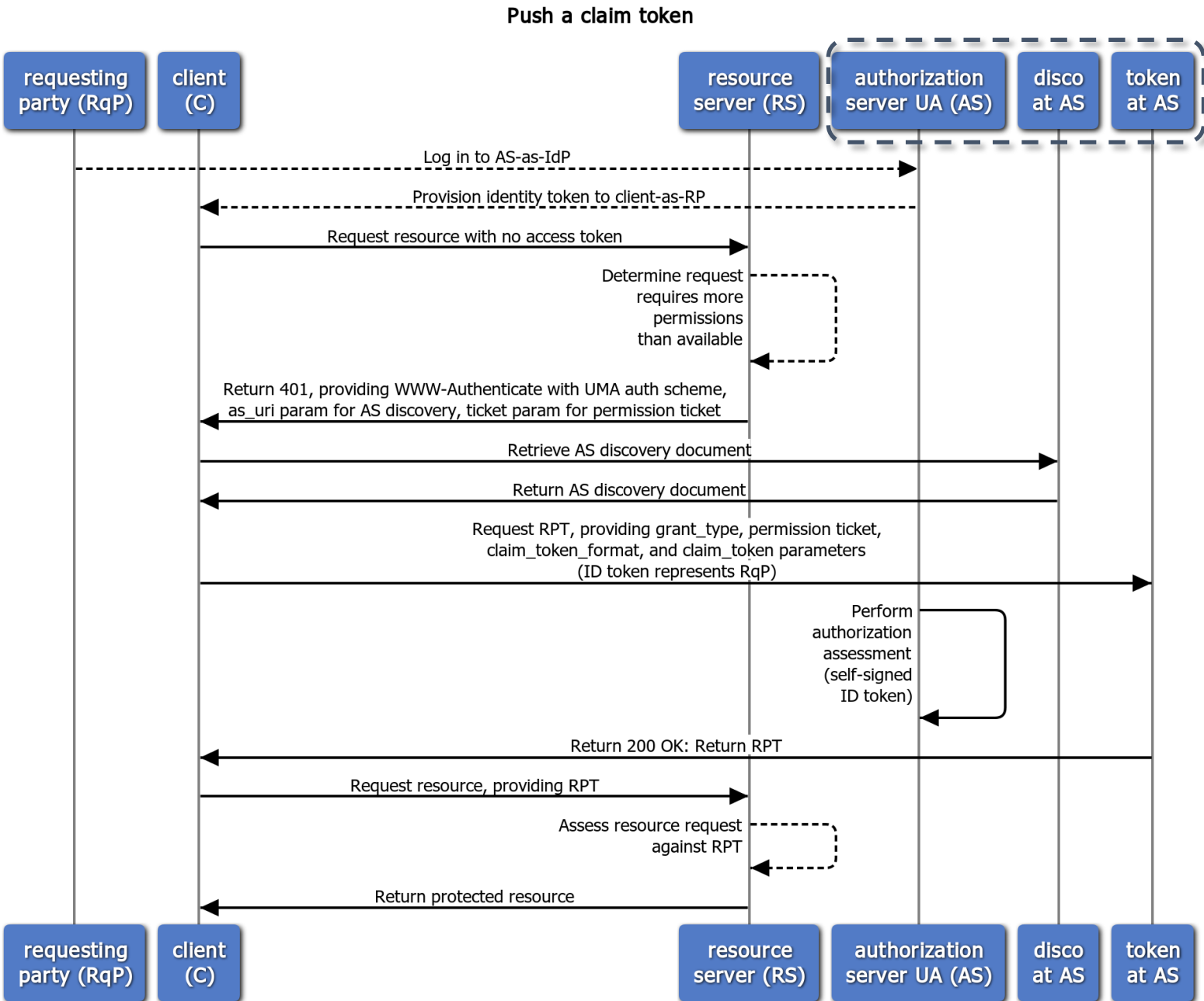
The AS is the requesting party's IdP and the client is the RP

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

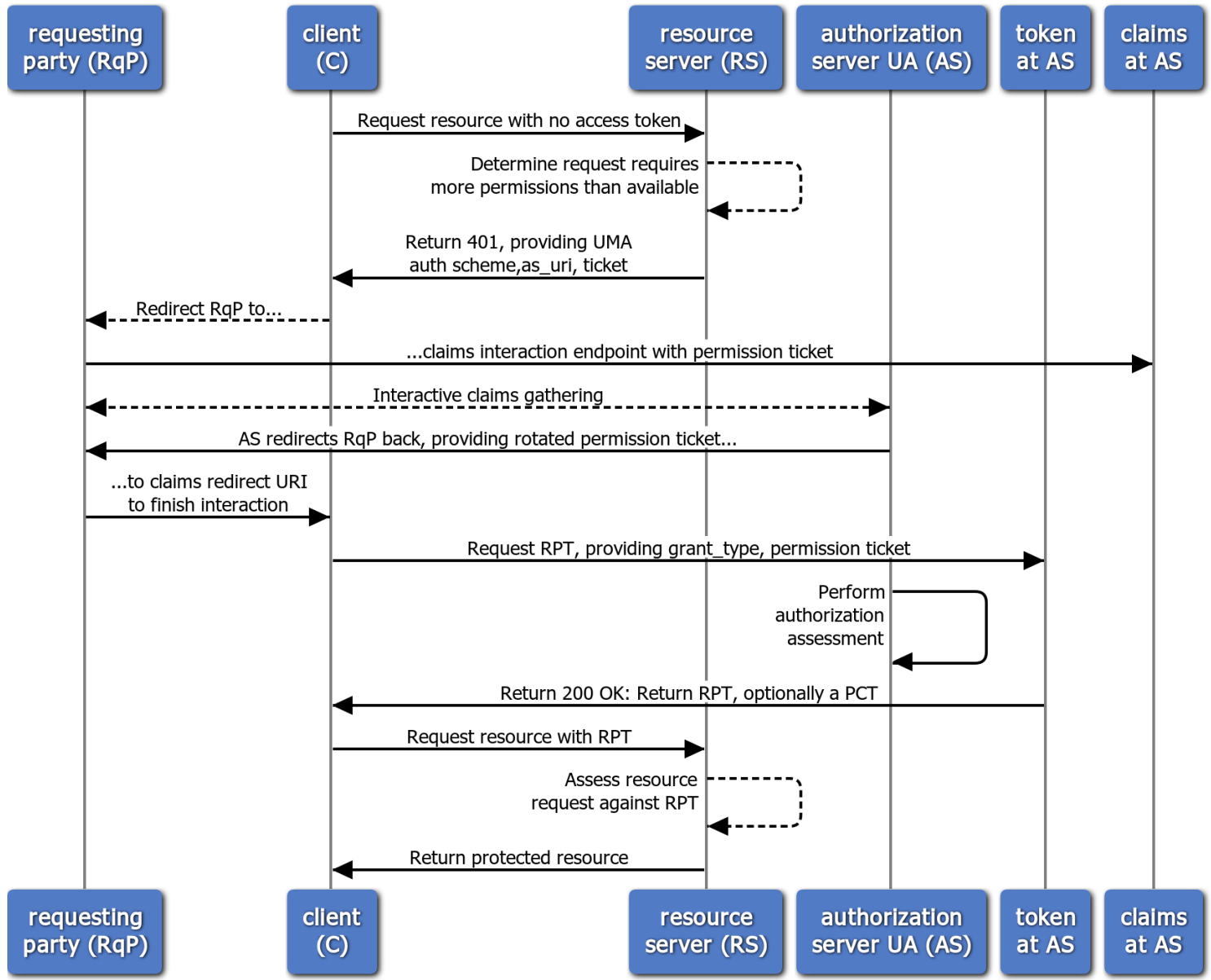
The AS mediates gathering of **claims from any source**

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”

Gather claims interactively



Federated authorization

A walkthrough of UMA federated authorization and its protection API

A new perspective on the UMA grant

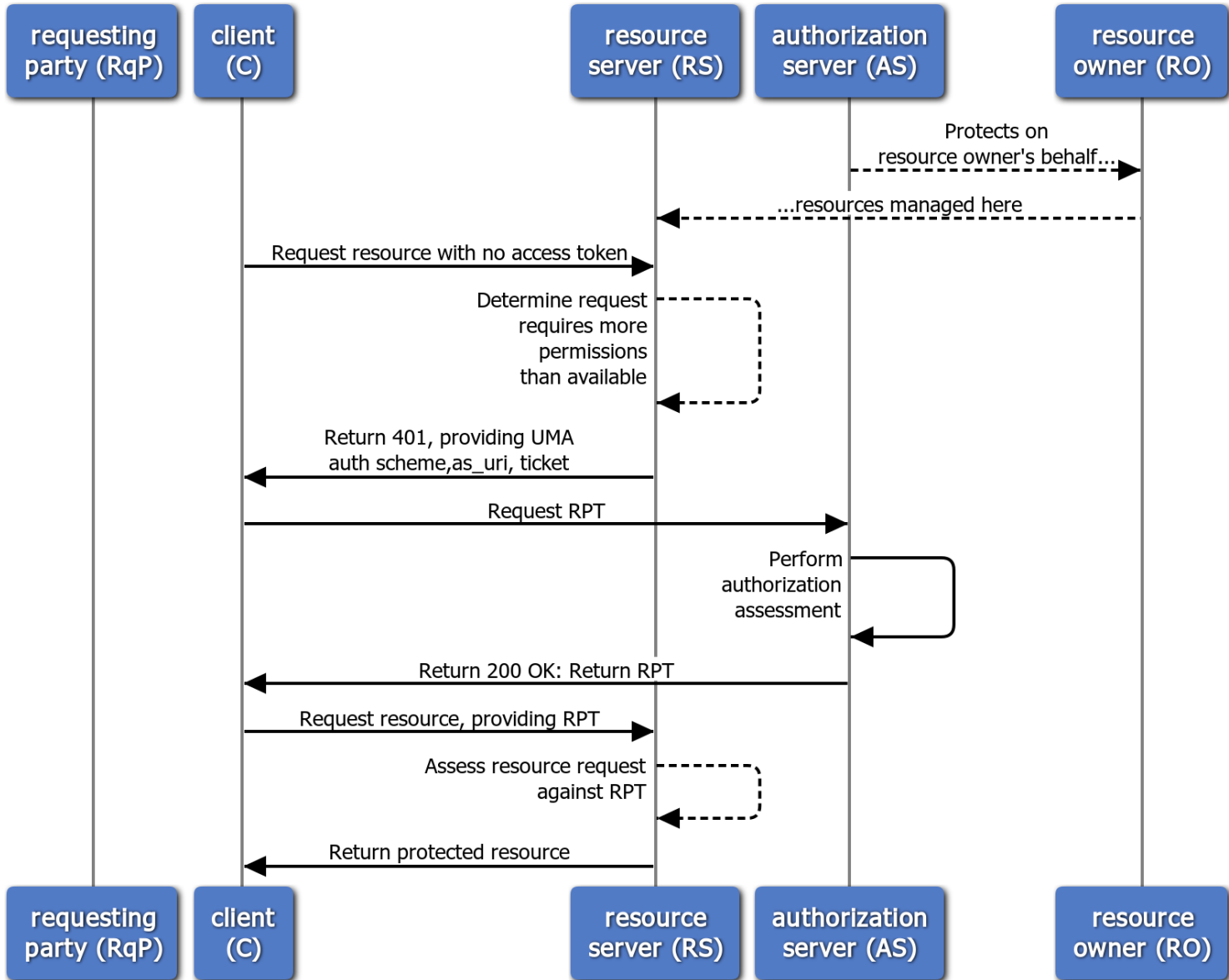
How does the AS know when to **start protecting resources**?

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

Federated authorization perspective



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**



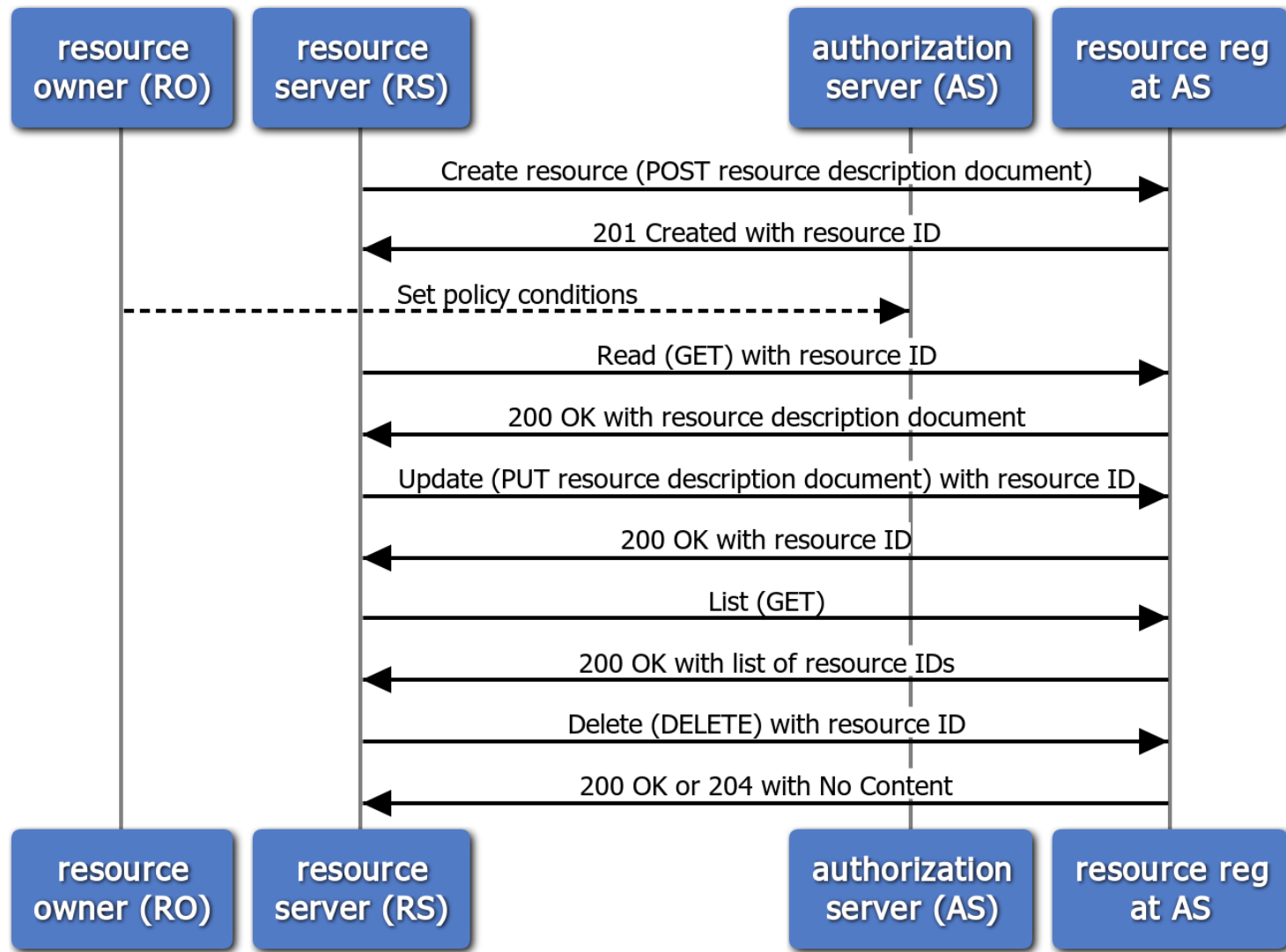
The resource registration endpoint

Registering a resource **puts it under protection**

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

Deregistering a resource **removes it from protection**

UMA Federated Authorization Resource Registration Endpoint



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/rsrcl
...
{
  "_id": "rsrcl"
}
```

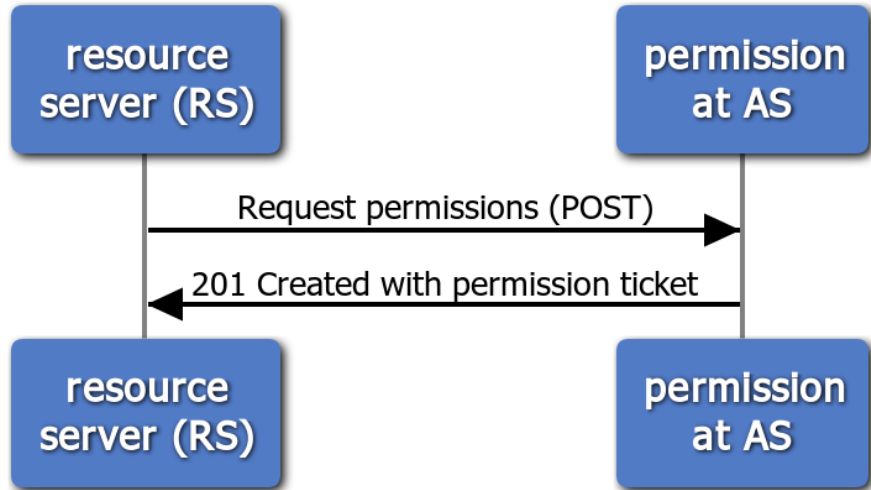
The permission endpoint

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"  
  ]  
}
```

UMA Federated Authorization Permission Endpoint



```
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

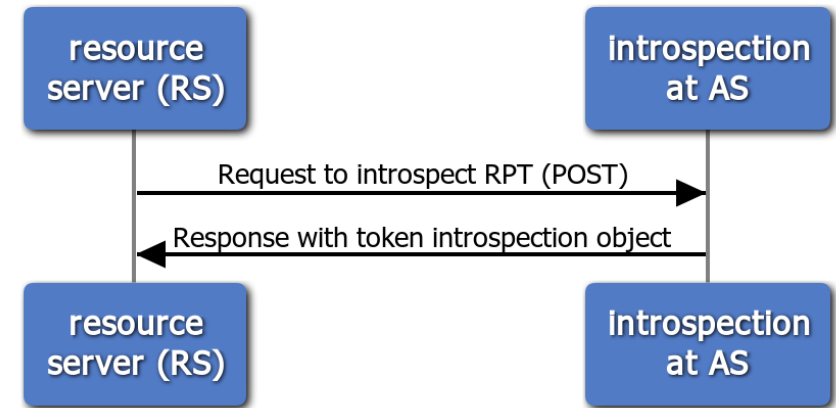
Response:

```
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
    }
  ]
}
```

Request:

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

UMA Federated Authorization Token Introspection Endpoint



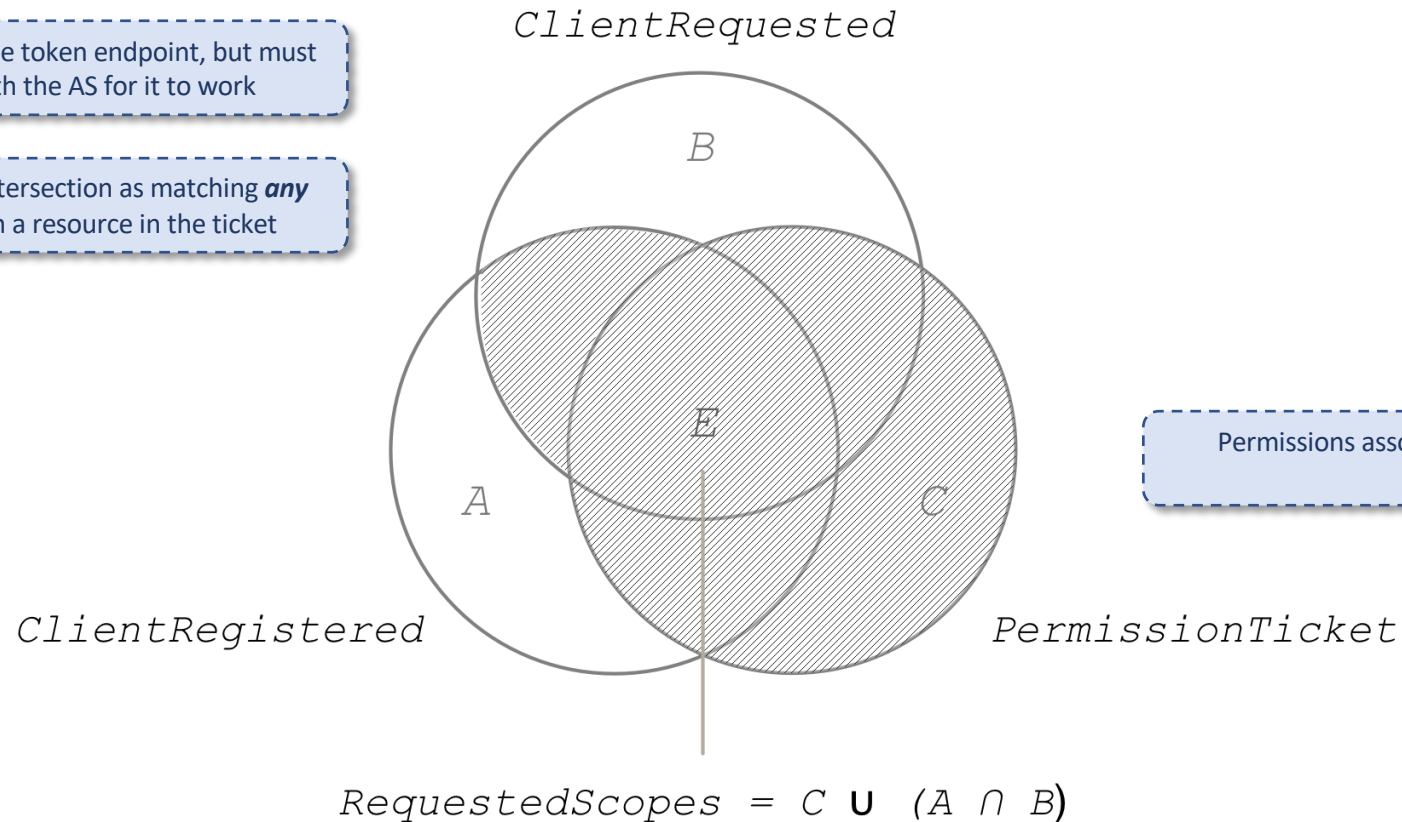
Authorization assessment

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**

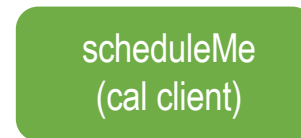
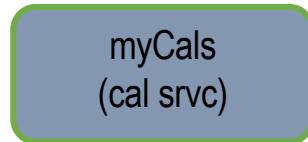
Use case: Calendar sharing

The UMA protocol in action

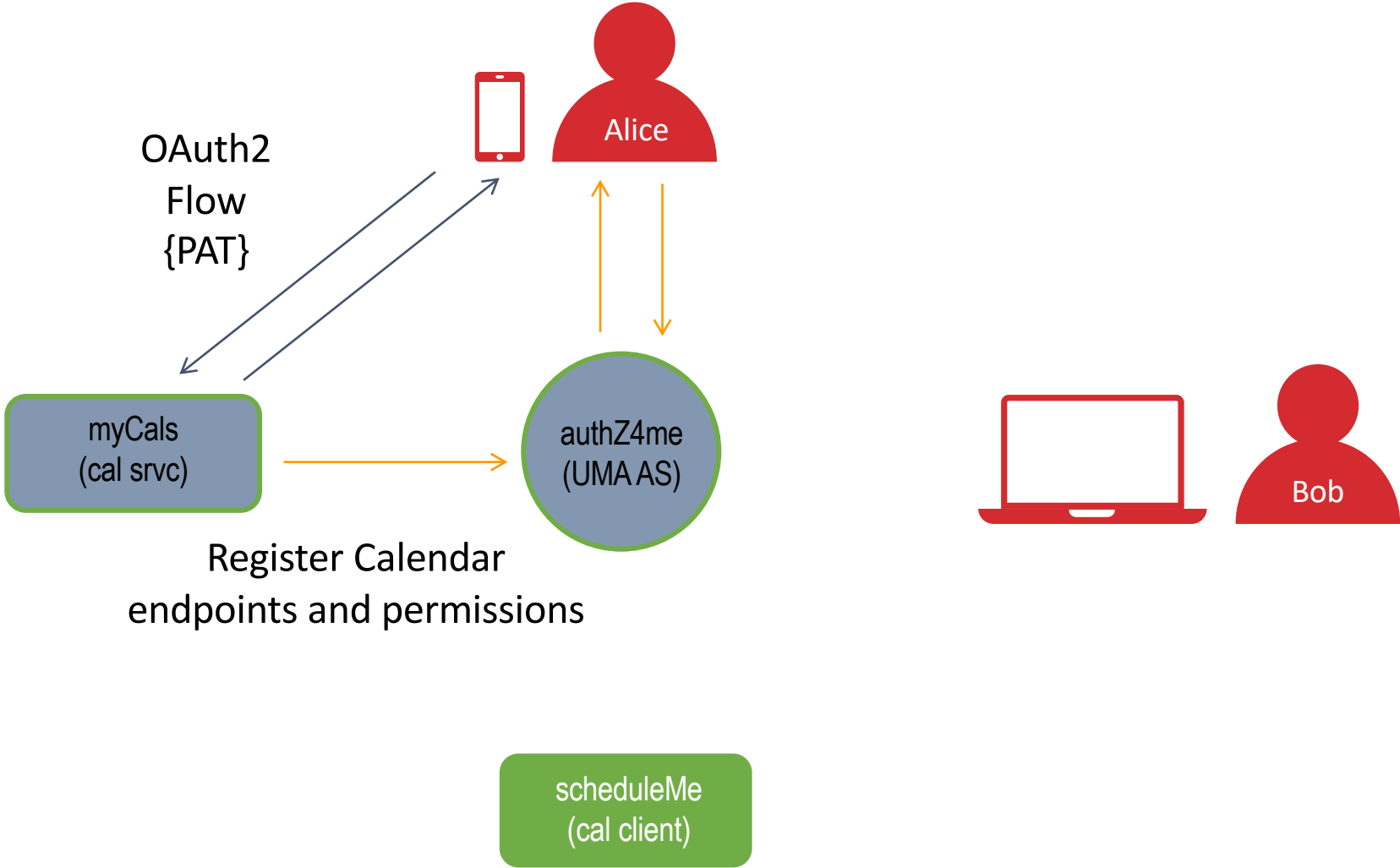
Detailed use case

- 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



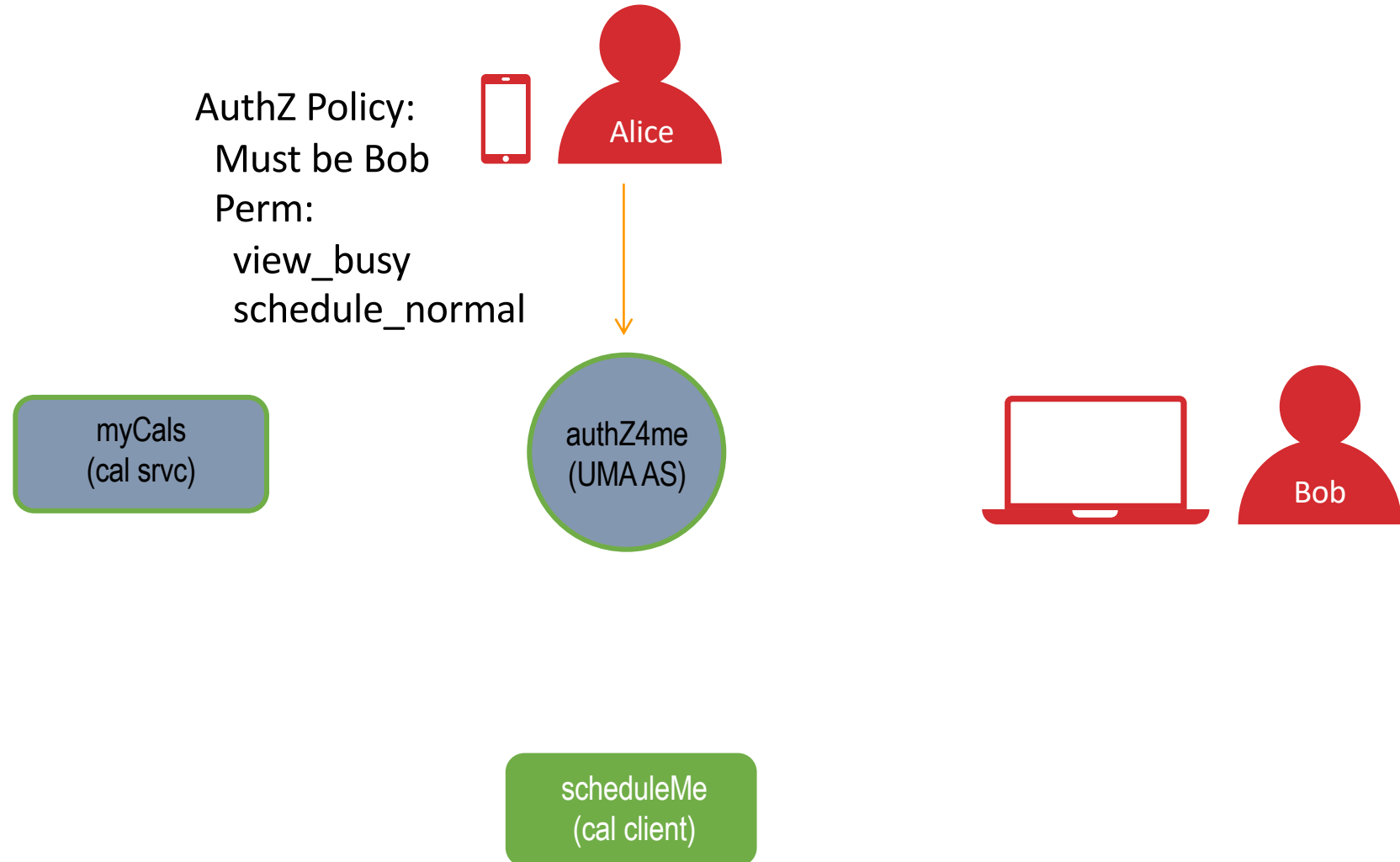
Alice registers protection for her calendar



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,

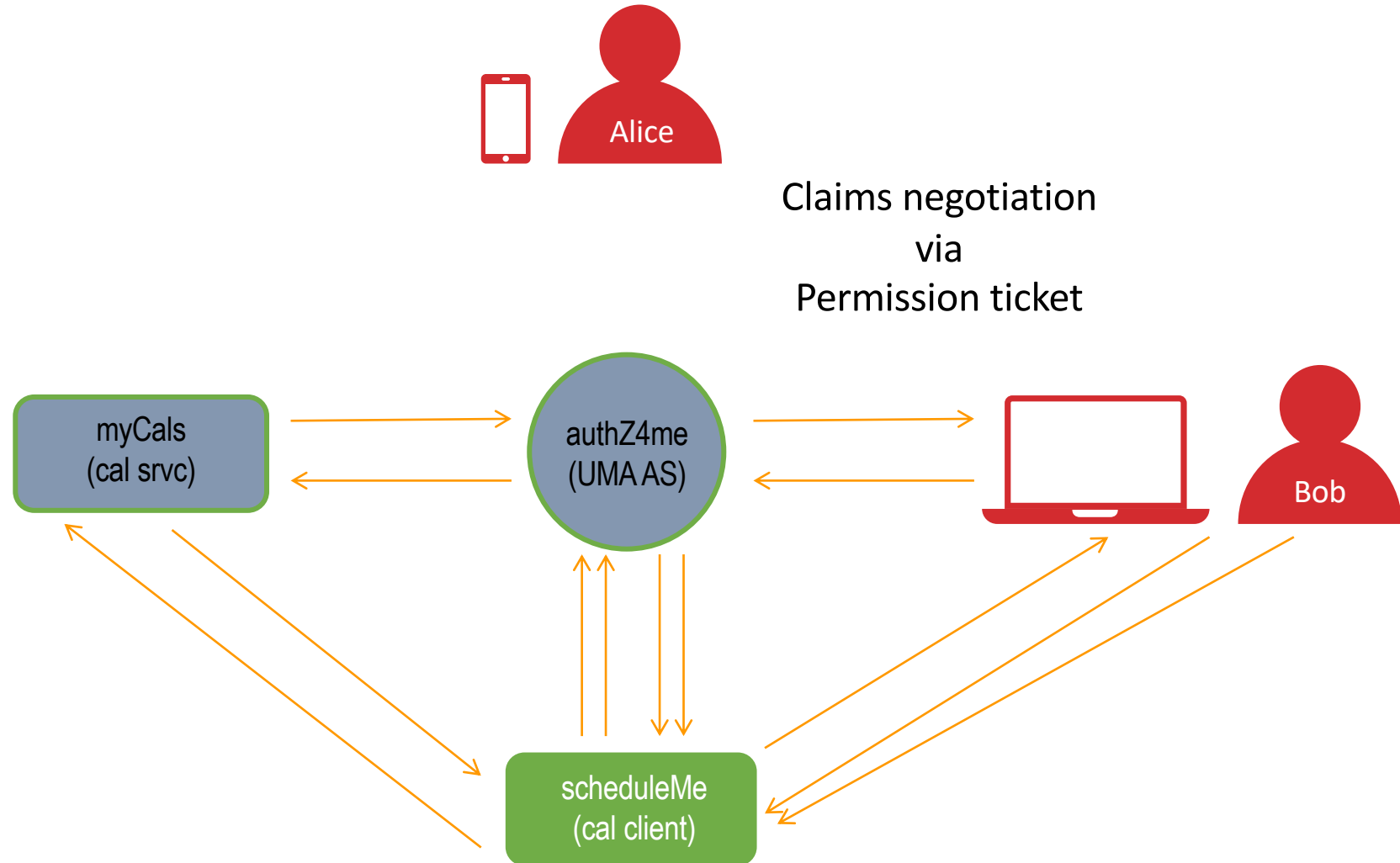
Please view my calendar and schedule the meeting we spoke about today.

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

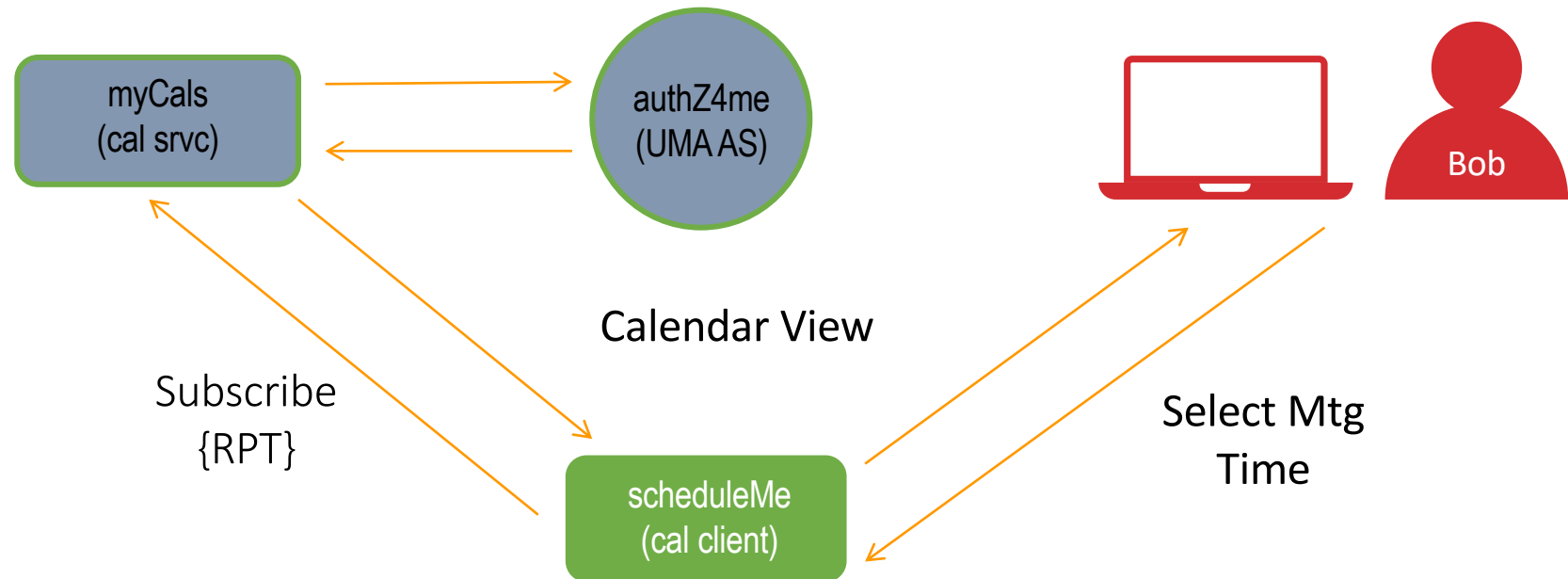
Thanks,
Alice



Bob meets claims to access Alice's calendar



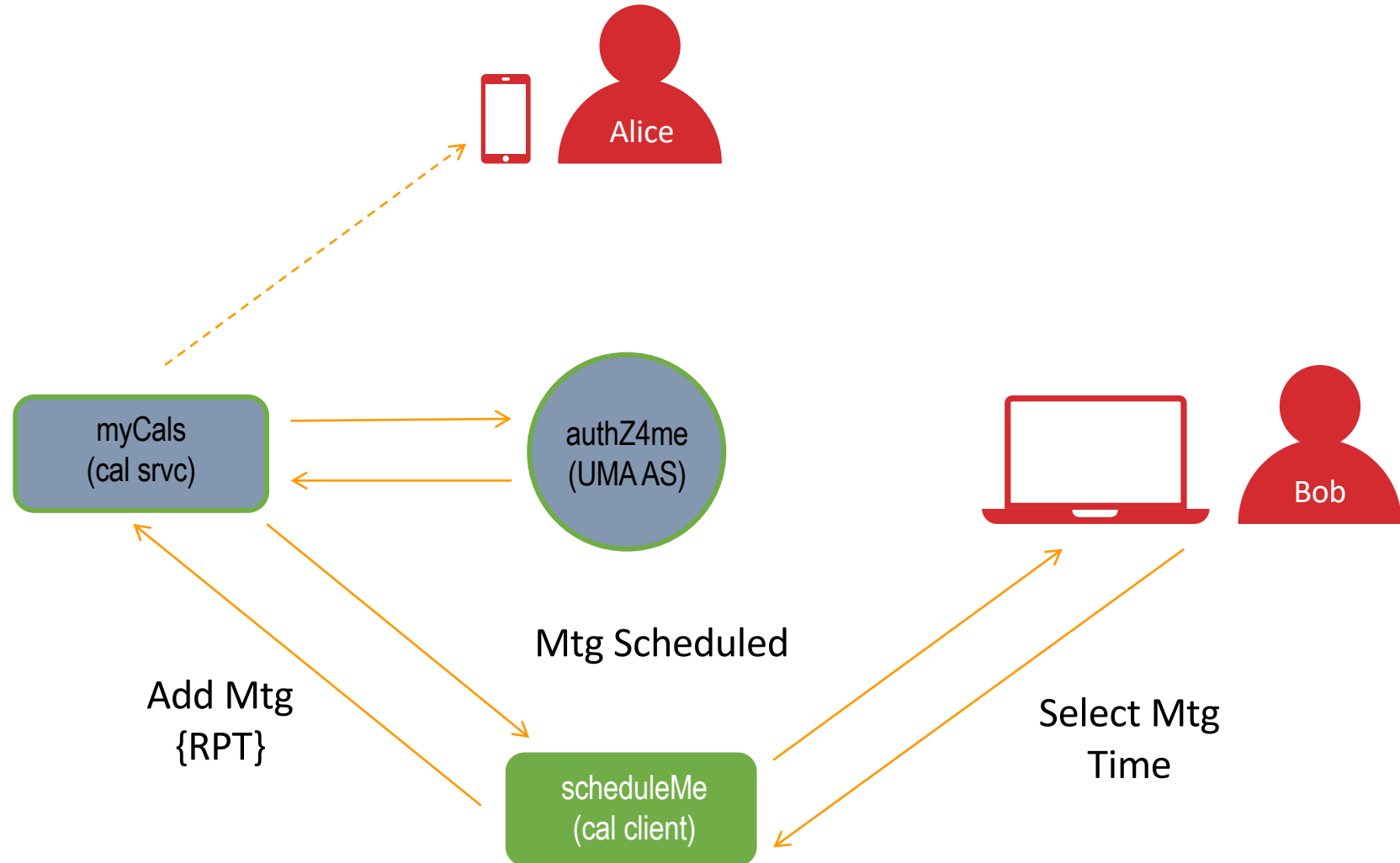
Bob subscribes to Alice's calendar



Bob schedules a meeting with Alice

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

Meeting added to Alice's calendar



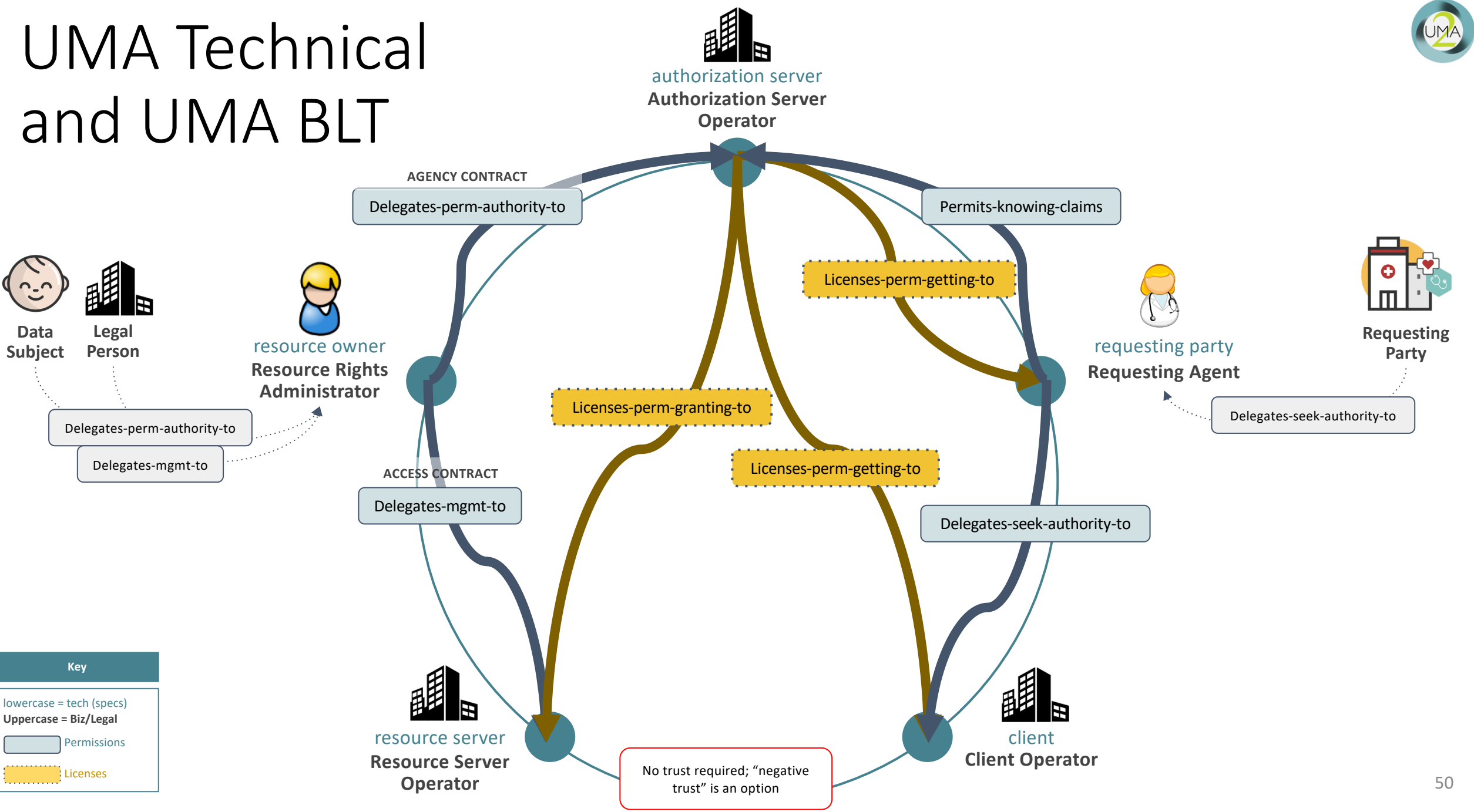
Privacy and “BLT” implications

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 Technical and UMA BLT



Data Subject
Legal Person

resource owner
Resource Rights Administrator

requesting party
Requesting Agent

Requesting Party

AGENCY CONTRACT
Delegates-perm-authority-to

Permits-knowing-claims

Licenses-perm-getting-to

Licenses-perm-granting-to

Licenses-perm-getting-to

Delegates-seek-authority-to

ACCESS CONTRACT
Delegates-mgmt-to

Delegates-seek-authority-to

resource server
Resource Server Operator

client
Client Operator

No trust required; "negative trust" is an option

Key

lowercase = tech (specs)
Uppercase = Biz/Legal

Permissions
Licenses

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, Kantara Initiative UMA WG member

@gffletch | @UMAWG | tinyurl.com/umawg

IIWXXVIII | 20 Oct 2020

