User-Managed Access (UMA) 101

Alec Laws, Kantara Initiative UMA Work Group chair

@aleclaws | @UMAWG

IIWXXXV | 15 Nov 2022

Topics

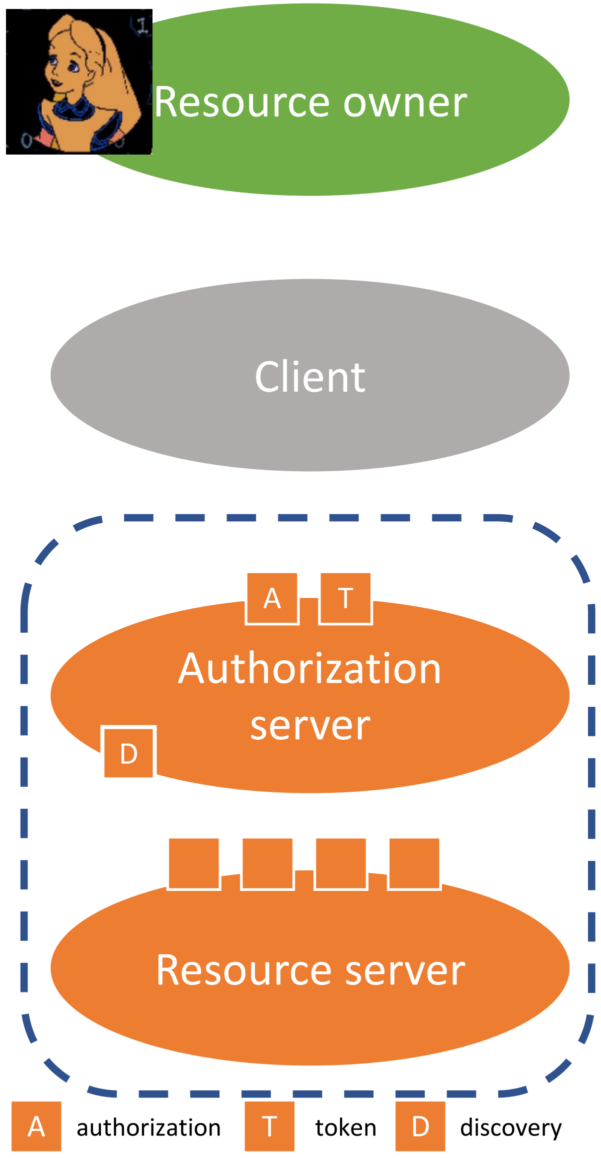
- Overview in OAuth terms
- UMA in action
- The technical big picture
- The UMA grant
- Federated authorization
- Authorization assessment
- Privacy and “BLT” (business-legal-technical) implications

Overview in OAuth terms

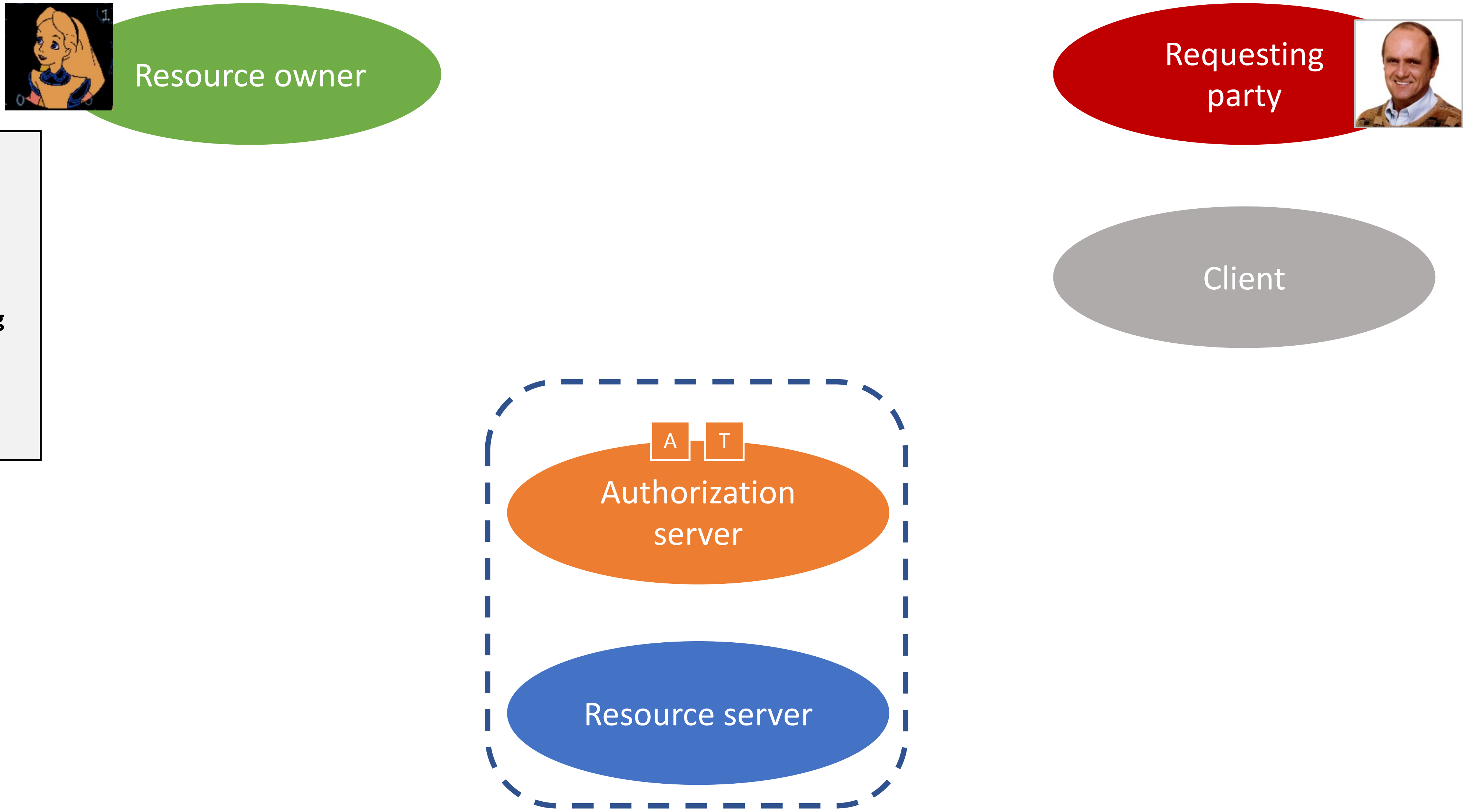
OAuth enables constrained delegation of access to apps

Benefits:

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



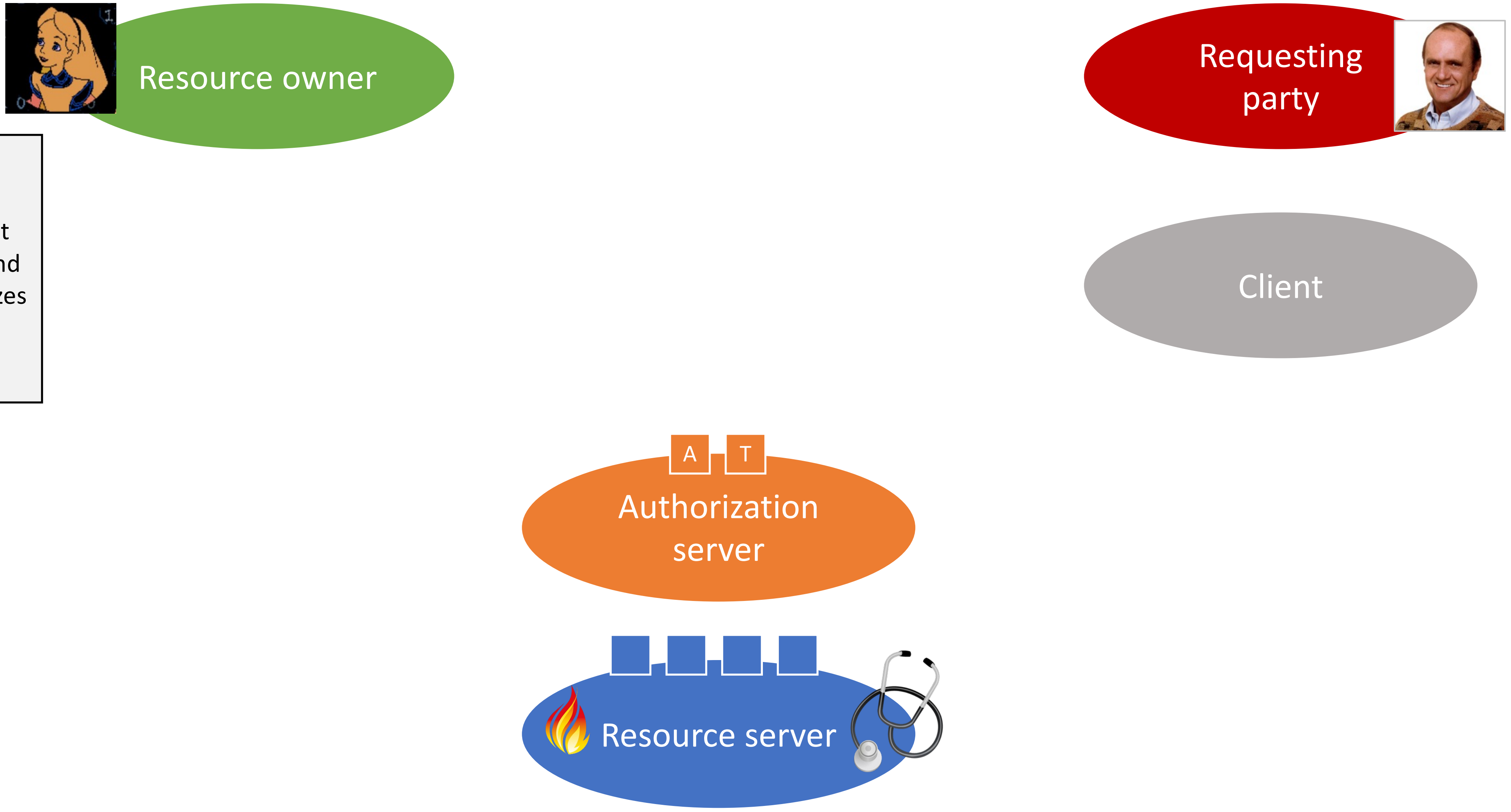
UMA adds cross-party sharing…



Benefits:

- **Secure** delegation
- Alice **can be absent** when Bob attempts access
- Helpful **error handling** for client applications

…in a wide ecosystem…



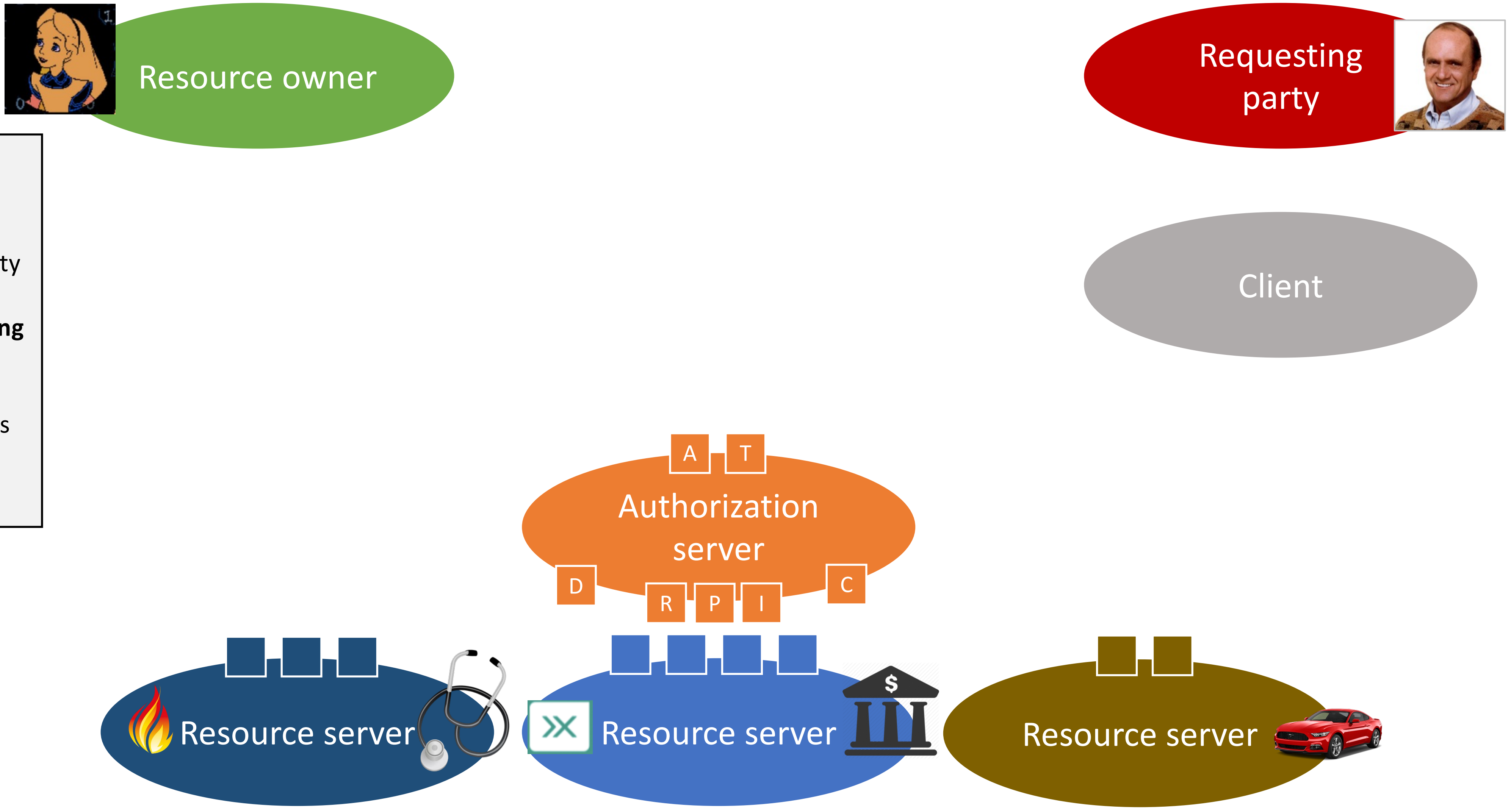
Benefits:

- Alice **controls trust** between a service that hosts her resources and a service that authorizes access to them

…of resource hosts

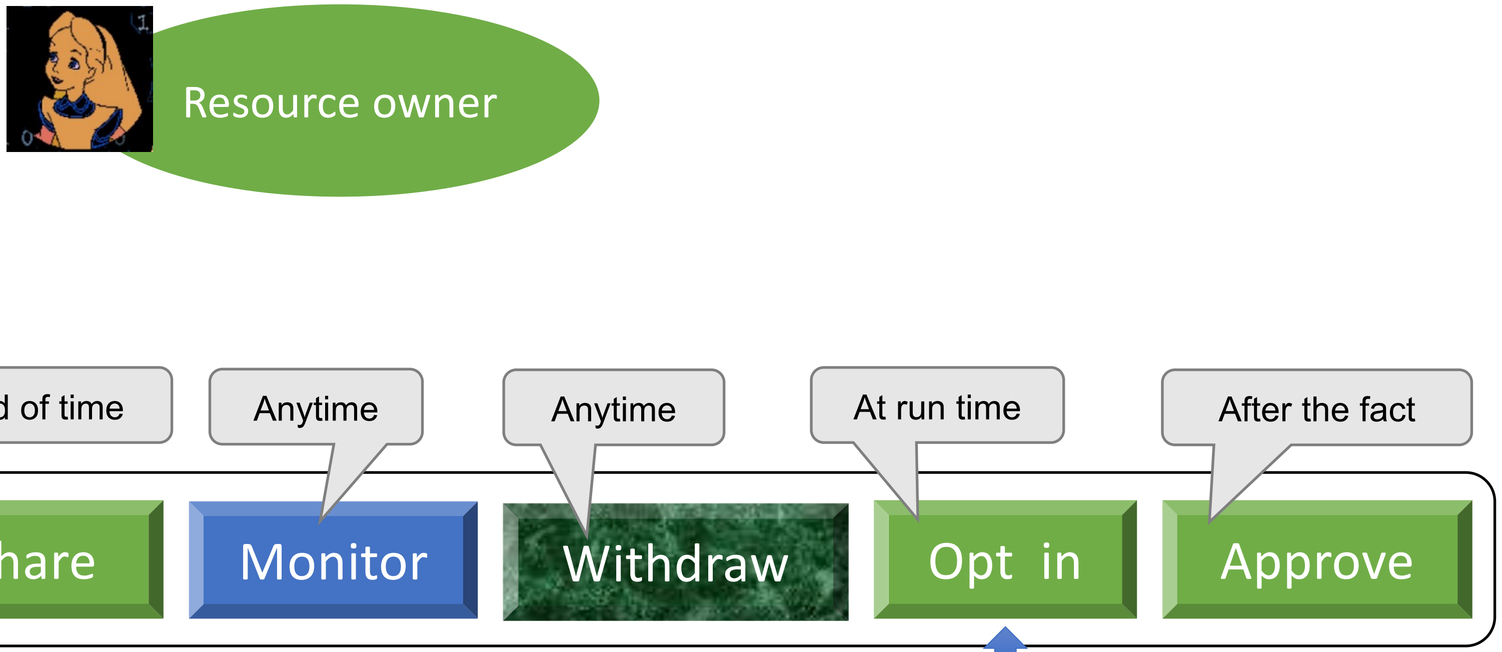
Benefits:

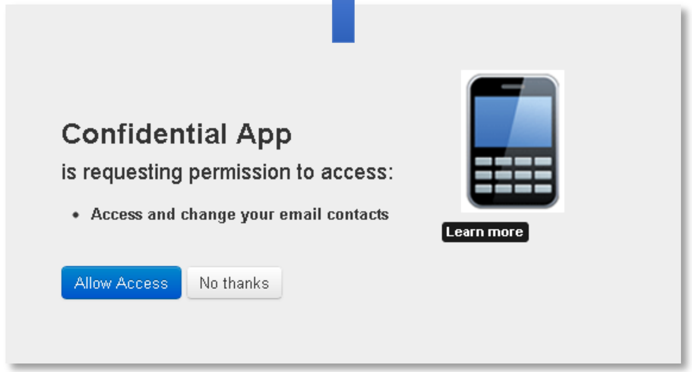
- Resource hosts can **outsource authorization** management – and liability – to a specialist service
- Alice can **manage sharing** at a centralizable service
- Bob can **revoke *his* access** to *Alice's* resources



A	authorization	T	token	D	discovery	R	resource registration	P	permission	I	token introspection	C	claims interaction

UMA user experience opportunities



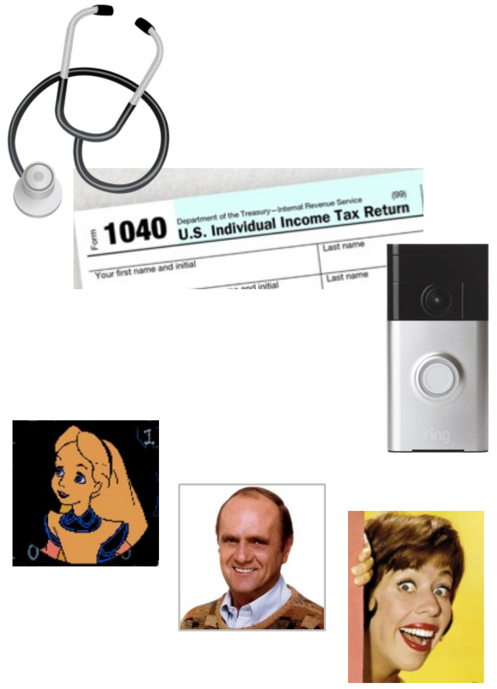

Confidential App
is requesting permission to access:

- Access and change your email contacts

Benefits for service providers: a summary

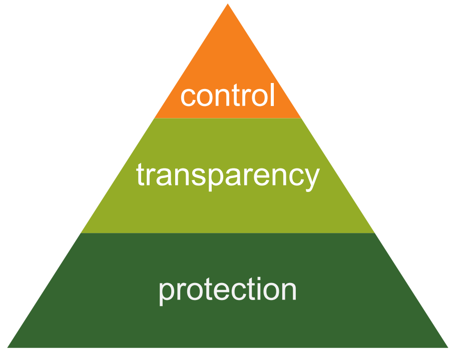
True secure delegation; no password sharing

Scale permissioning through self-service



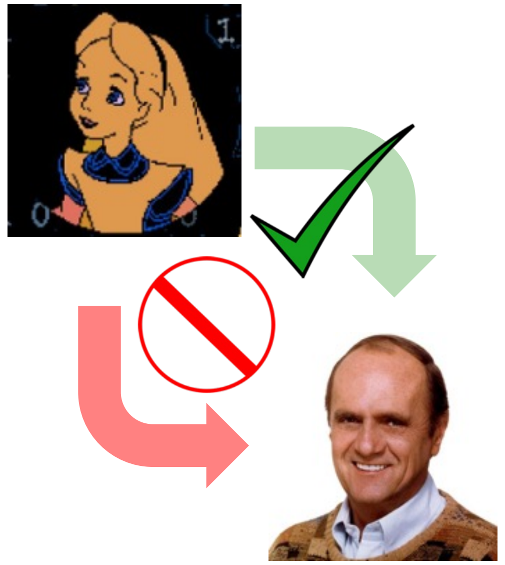
API-first protection strategy

Foster compliance through standards

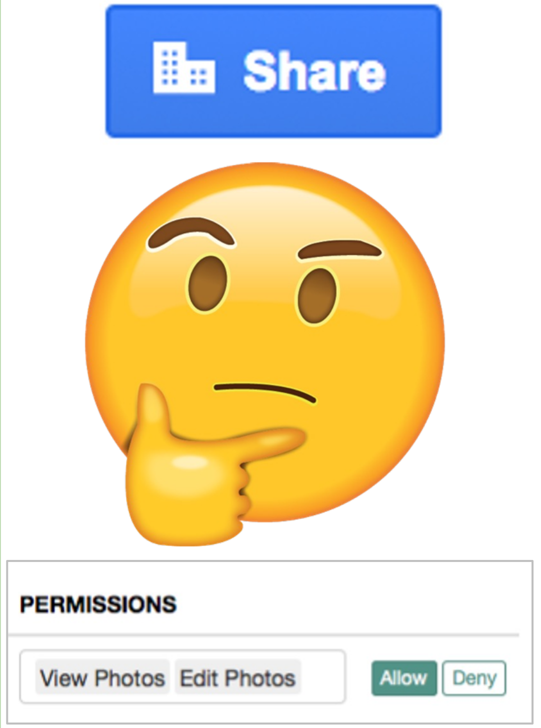


Benefits for patients and consumers: a summary

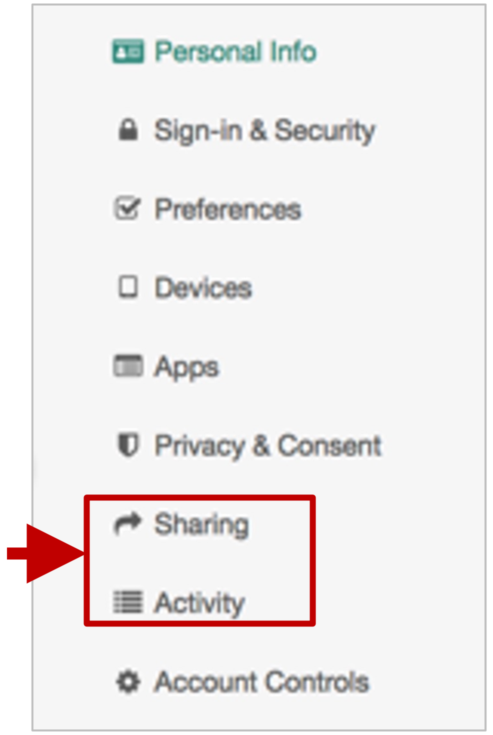
Choice in sharing with other parties



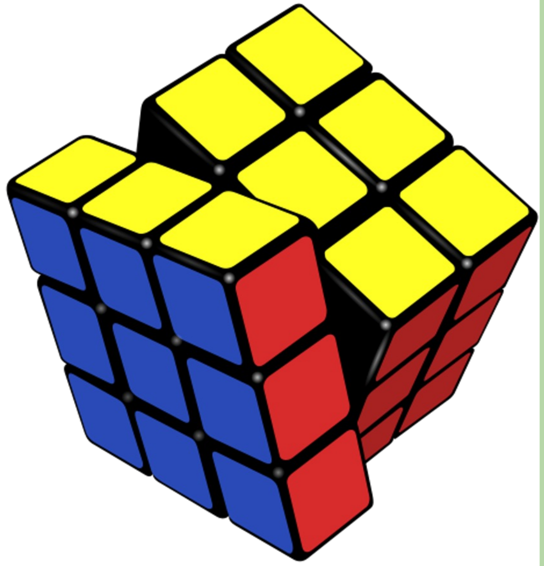
Convenient sharing/approval with no outside influence



Centralizable monitoring and management



Control of who/what/how at a fine grain



Typical use cases

- Alice to Bob (person to person):
  - Patient-directed health data/device sharing
  - Discovering/aggregating pension accounts and sharing access to financial advisors
  - Connected car data and car sharing
- Enterprise to Alice (initial RO is an organization):
  - Enterprise API access management
  - Access delegation between employees
- Alice to Alice (person to self/app):
  - Proactive policy-based control of app connections

- Profiled or referenced by:
  - OpenID Foundation HEART Working Group
  - UK Department for Work and Pensions

Known implementations

(more detail at tinyurl.com/umawg)

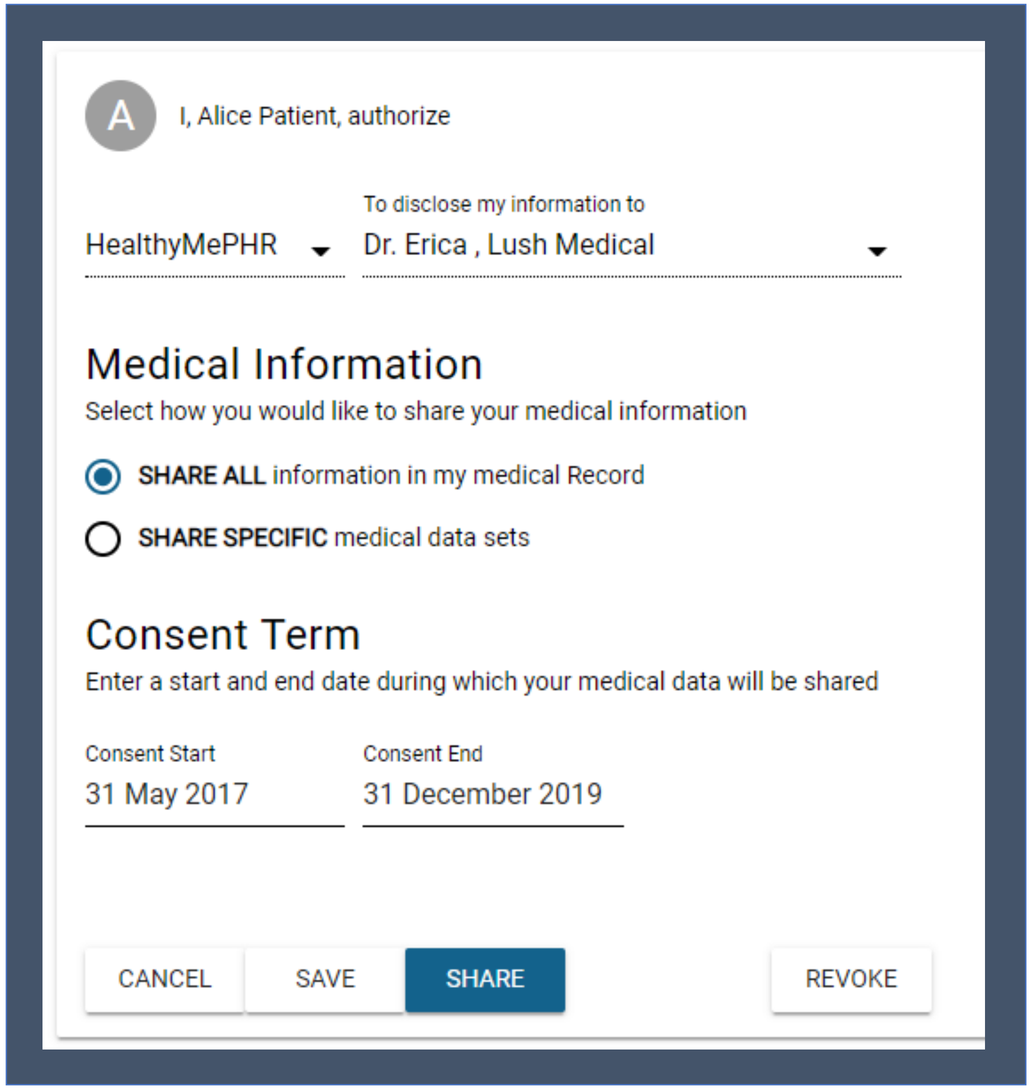
- ForgeRock – financial, healthcare, IoT, G2C…
- IDENTOS – healthcare, G2C
- Patient Centric Solutions – healthcare
- HIE of One / Trustee (open source) – healthcare
- Gravitee – API protection, financial
- Gluu (open source) – API protection, enterprise, G2C…
- Pauldron (open source) – healthcare
-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

UMA in a nutshell

- Developed at Kantara Initiative
  - V2.0 complete in Jan 2018
- Leverages existing open standards:
  - OAuth2
  - OpenID Connect and SAML
- Profiled by multiple industry sectors
  - Financial, healthcare
- UMA business model effort ("BLT") supports **legal licensing** for personal digital assets
  - Example: Mother (legal guardian) manages sharing for child (data subject); child becomes old enough and starts to manage sharing herself

UMA in action

PatientShare



A

I, Alice Patient, authorize

HealthyMePHR

To disclose my information to

Dr. Erica , Lush Medical

Medical Information

Select how you would like to share your medical information

- [✓ **SHARE ALL** information in my medical Record]
- **SHARE SPECIFIC** medical data sets

Consent Term

Enter a start and end date during which your medical data will be shared

Consent Start: 31 May 2017

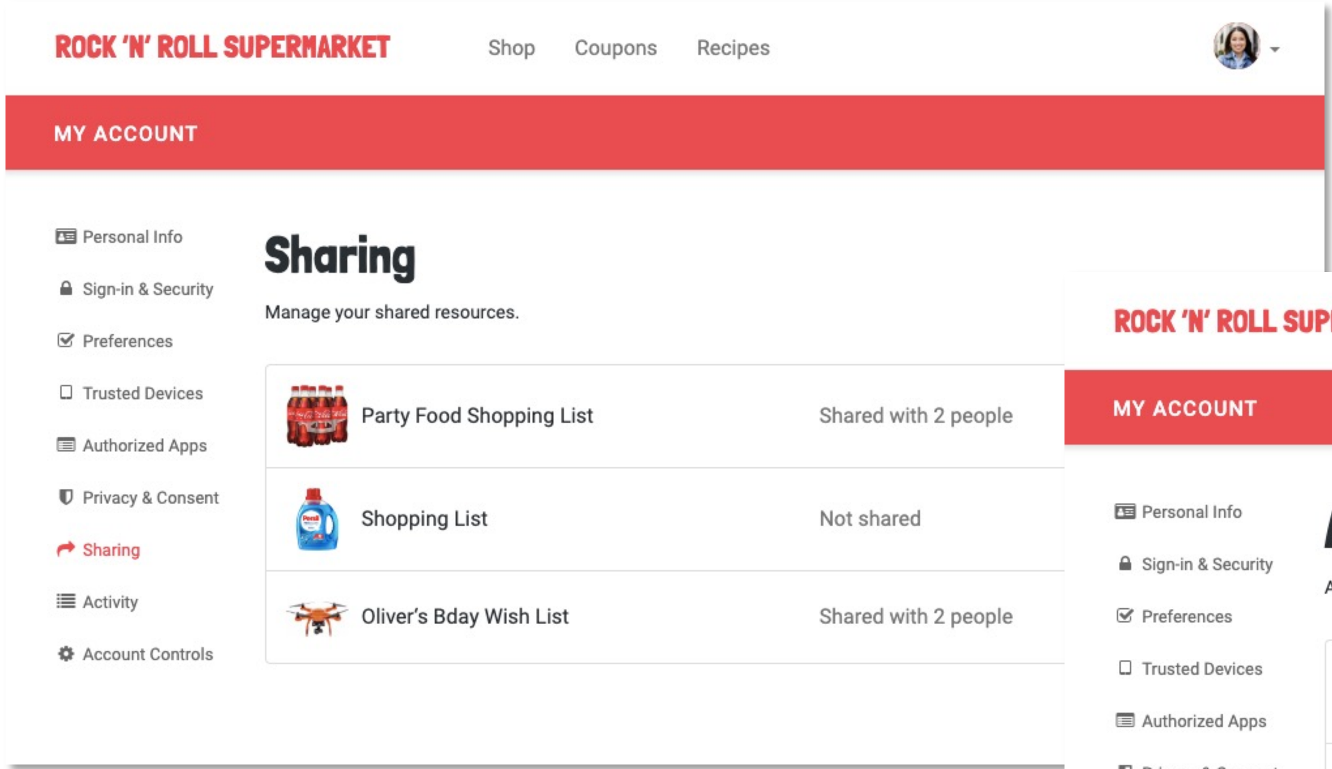
Consent End: 31 December 2019

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

SHARE

- Patient sharing is easy!

ForgeRock Identity Platform



ROCK 'N' ROLL SUPERMARKET — Shop | Coupons | Recipes

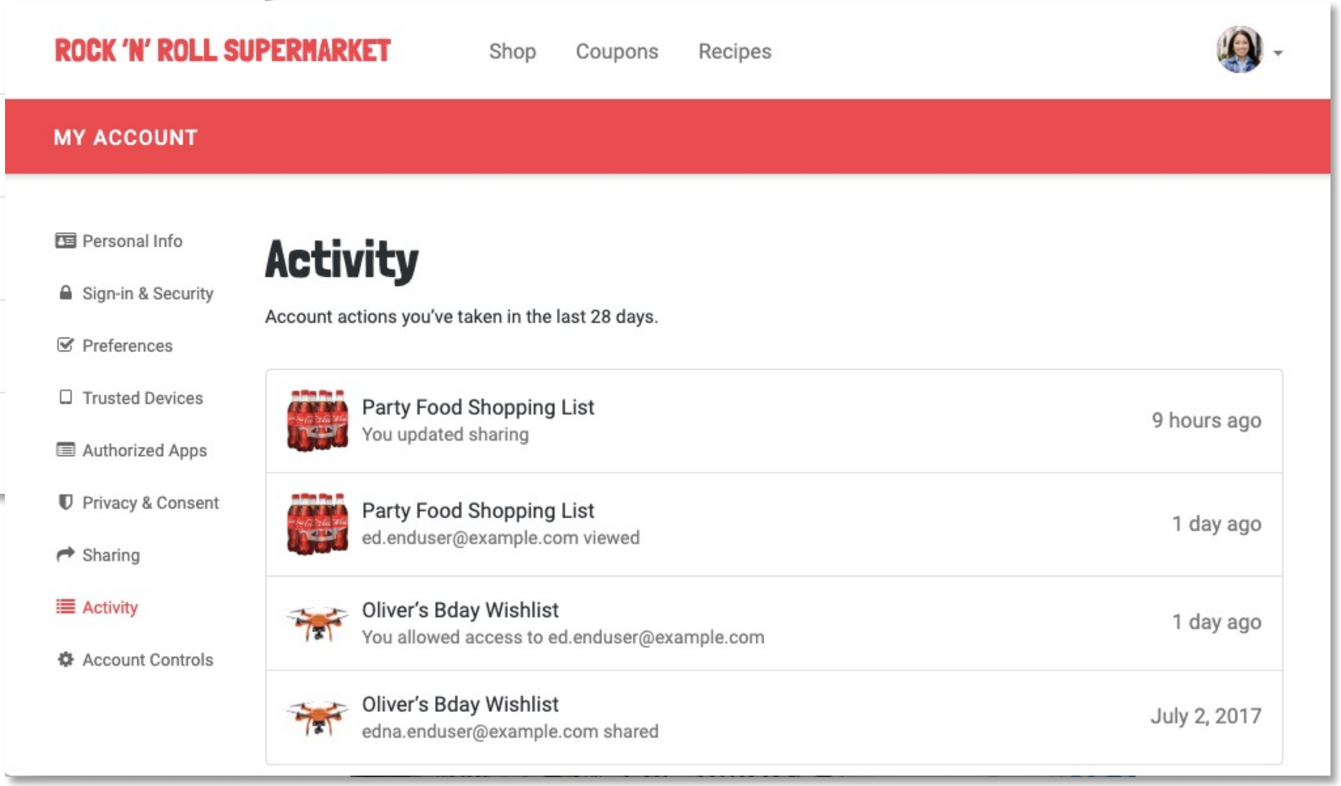
MY ACCOUNT

- 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

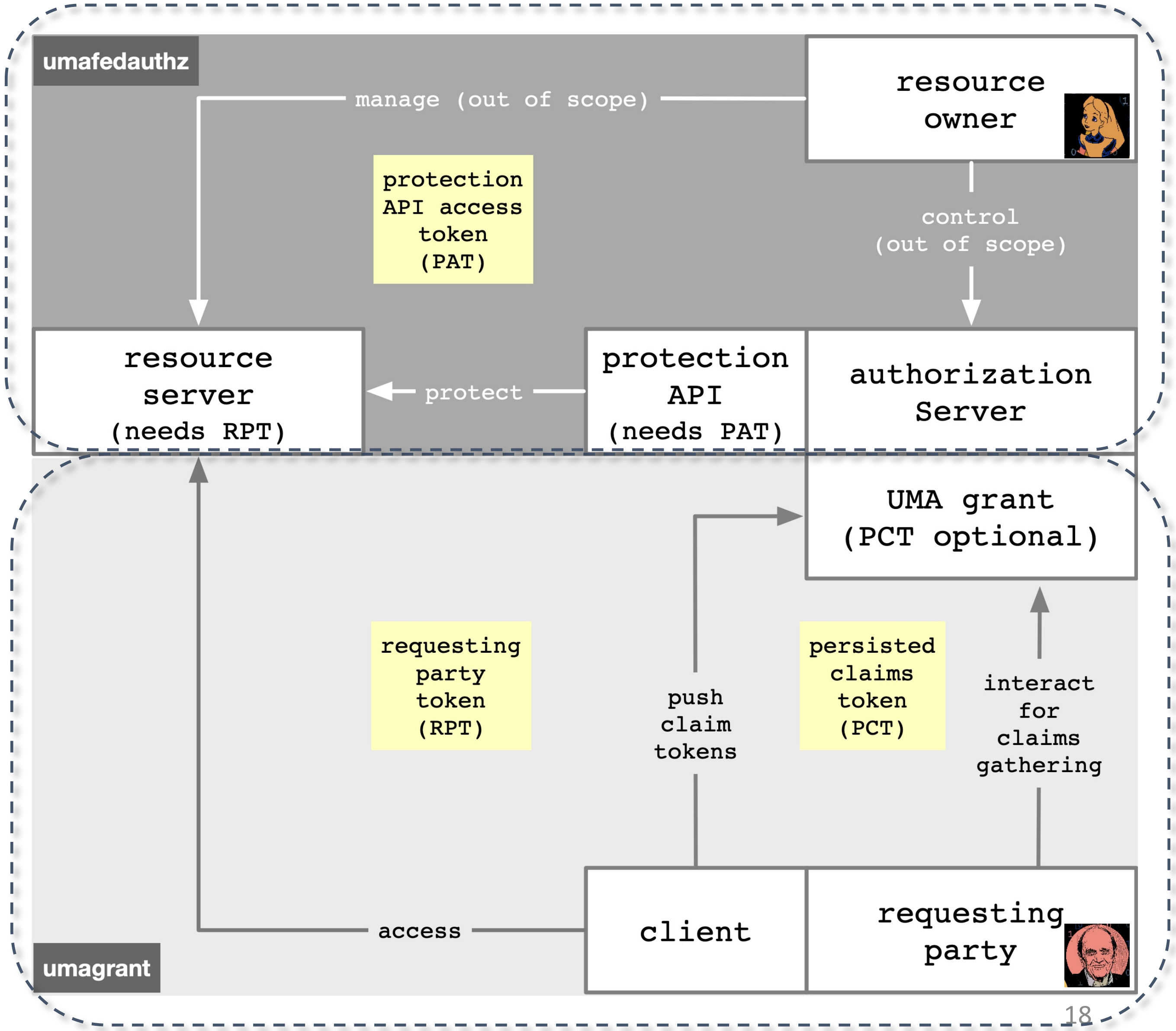
Account actions you've taken in the last 28 days.

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

The marvelous spiral of delegated sharing, squared

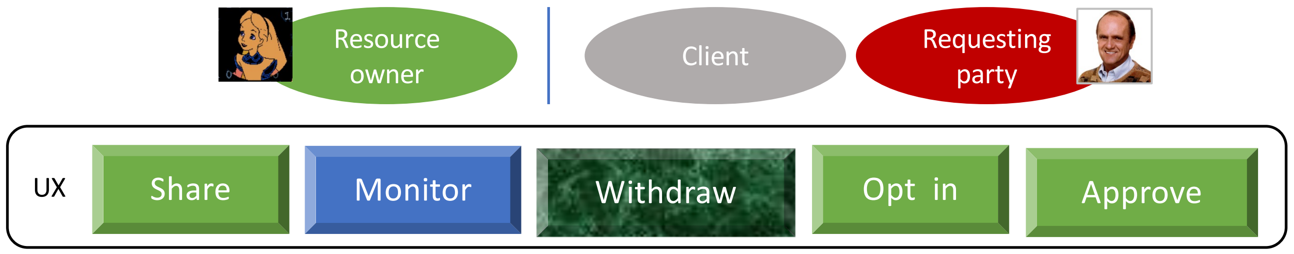
1. The **UMA grant of OAuth** enables Alice-to-Bob delegation
2. UMA **standardized an API for federated authorization** at the AS to make it centralizable
3. There are **nicknames** for enhanced and new tokens to keep them straight



The UMA extension grant adds…

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

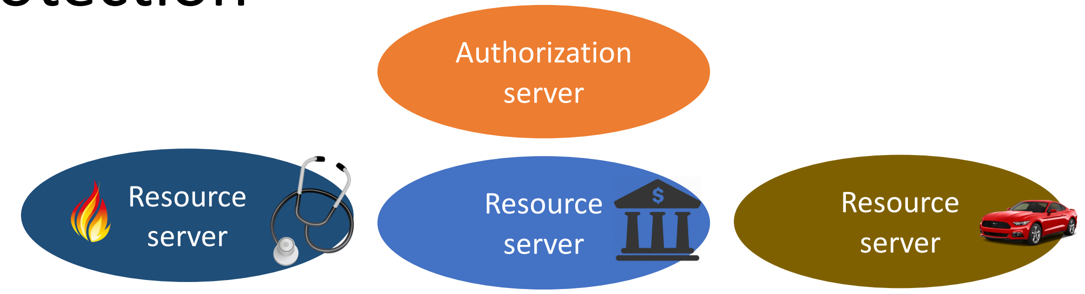
- **Party-to-party:** Resource owner authorizes protected-resource access to clients used by requesting parties
- **Asynchronous:** Resource owner interactions are asynchronous with respect to the authorization grant
- **Policies:** Resource owner can configure an AS with rules (policy conditions) for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



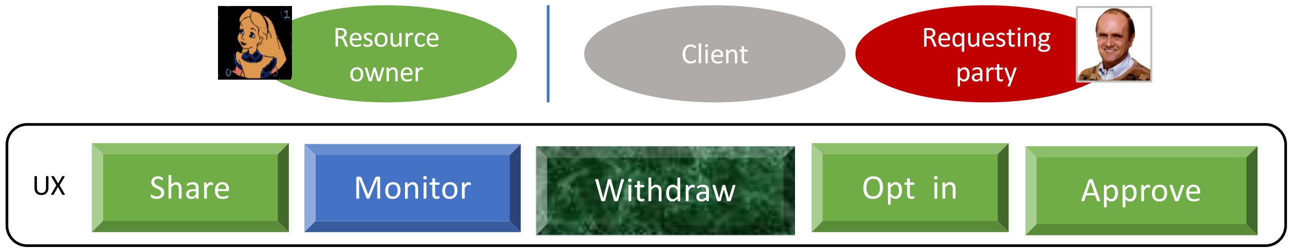
UMA federated authorization adds…

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



The UMA grant



Grant Prerequisites

- The Authorization Server knows about Alice’s resources
- The Authorization Server knows Alice’s policies for Bob to access
- The Client has an OAuth Client at the Authorization Server (or a way to create one dynamically)

The UMA extension grant flow and its options

The AS is acting as an **agent** for an absent RO

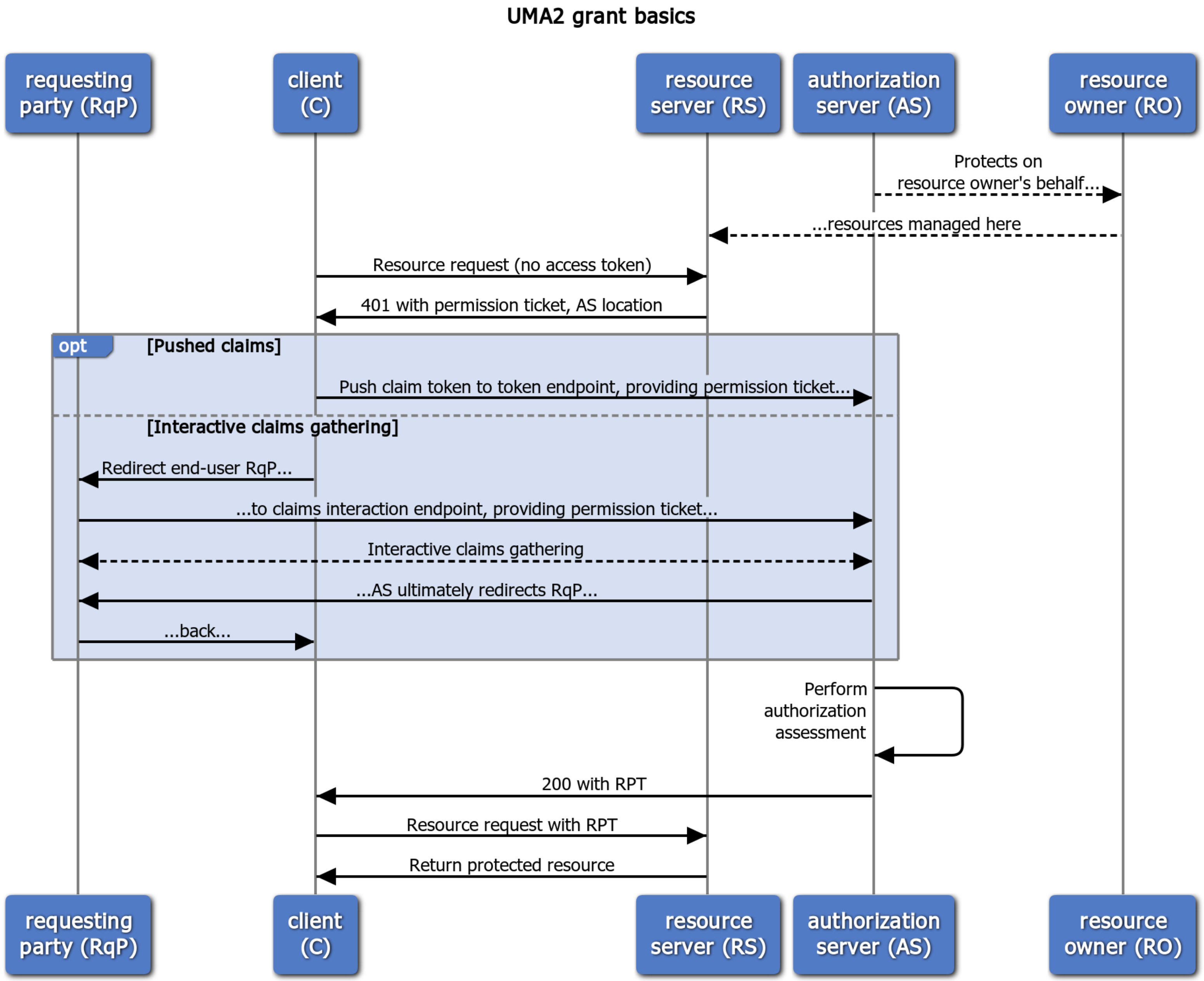
The client's first resource request is **tokenless**

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
requesting party (RqP)
client (C)
resource server (RS)
authorization server (AS)
resource owner (RO)
Protects on resource owner's behalf...
...resources managed here
Resource request (no access token)
401 with permission ticket, AS location
opt
[Pushed claims]
Push claim token to token endpoint, providing permission ticket...
[Interactive claims gathering]
Redirect end-user RqP...
...to claims interaction endpoint, providing permission ticket...
Interactive claims gathering
...AS ultimately redirects RqP...
...back...
Perform authorization assessment
200 with RPT
Resource request with RPT
Return protected resource

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

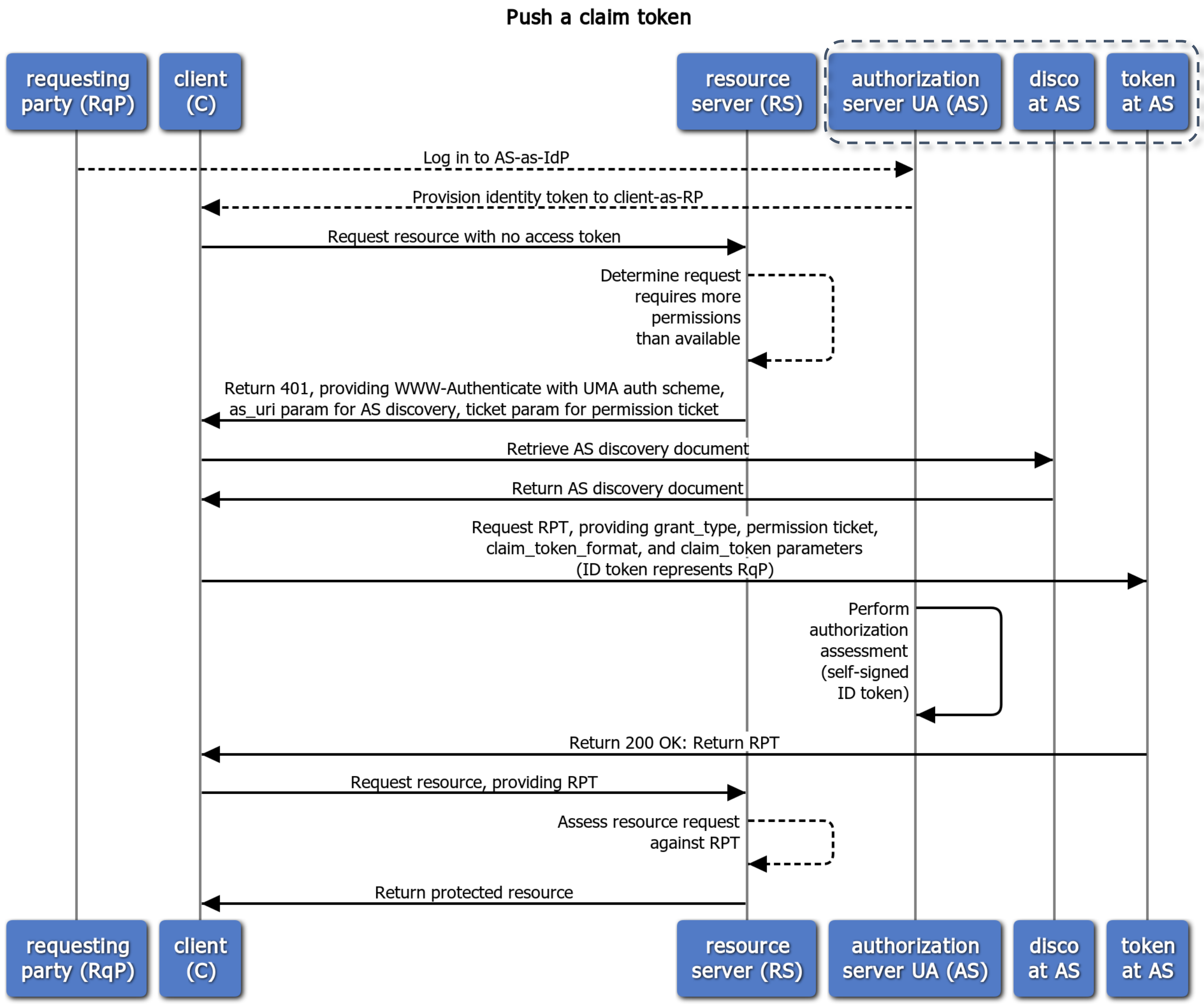
The AS is the requesting party's IdP and the client is the RP

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"


Push a claim token
requesting party (RqP)
client (C)
resource server (RS)
authorization server UA (AS)
disco at AS
token at AS
Log in to AS-as-IdP
Provision identity token to client-as-RP
Request resource with no access token
Determine request requires more permissions than available
Return 401, providing WWW-Authenticate with UMA auth scheme, as_uri param for AS discovery, ticket param for permission ticket
Retrieve AS discovery document
Return AS discovery document
Request RPT, providing grant_type, permission ticket, claim_token_format, and claim_token parameters (ID token represents RqP)
Perform authorization assessment (self-signed ID token)
Return 200 OK: Return RPT
Request resource, providing RPT
Assess resource request against RPT
Return protected resource
requesting party (RqP)
client (C)
resource server (RS)
authorization server UA (AS)
disco at AS
token at AS

Interactive claims gathering scenario: for wide ecosystems

(eliding detail already seen)

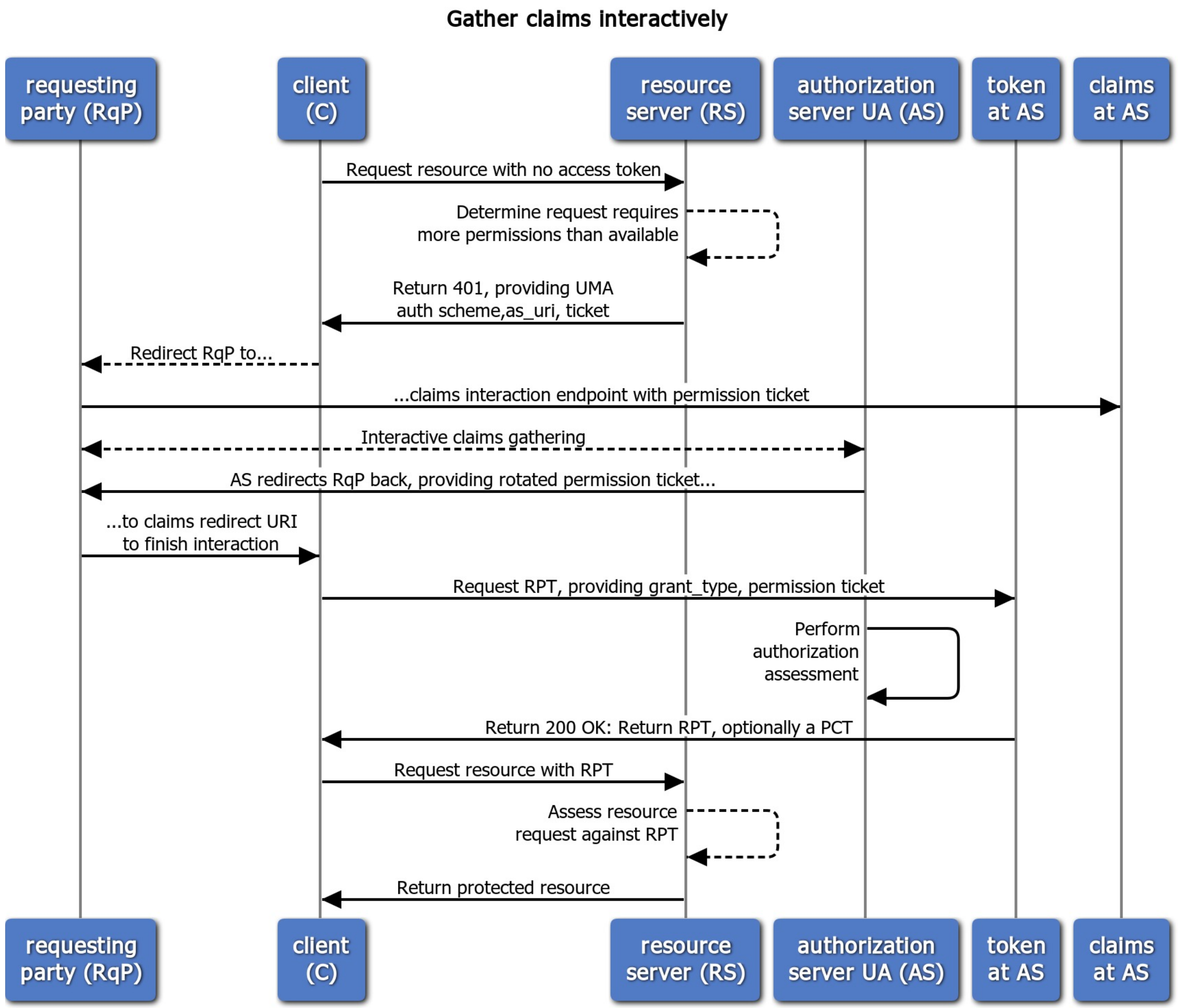
A claims interaction endpoint **must have been declared** in the discovery document to allow this flow

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

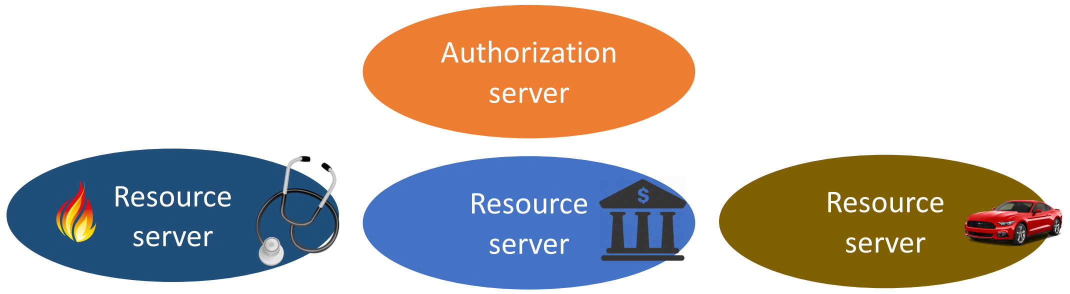
Resembles the **authorization code grant**, but can apply to non-unique identities and is repeatable and "buildable"


Gather claims interactively
requesting party (RqP)
client (C)
resource server (RS)
authorization server UA (AS)
token at AS
claims at AS
Request resource with no access token
Determine request requires more permissions than available
Return 401, providing UMA auth scheme,as_uri, ticket
Redirect RqP to...
...claims interaction endpoint with permission ticket
Interactive claims gathering
AS redirects RqP back, providing rotated permission ticket...
...to claims redirect URI to finish interaction
Request RPT, providing grant_type, permission ticket
Perform authorization assessment
Return 200 OK: Return RPT, optionally a PCT
Request resource with RPT
Assess resource request against RPT
Return protected resource

Grant Review

- The client makes a **tokenless** request for a resource on behalf of Bob
  - And receives a permission ticket and AS location
- The client makes a /token request with the ticket
  - and receives next steps -- **push claims** and/or **interactive claims gathering**
- The client and Bob fulfill the policy
- The client makes a final /token request and receives an RPT (Oauth access token)
- The client makes a request for the resource with the RPT
  - And receives the response!

Federated authorization



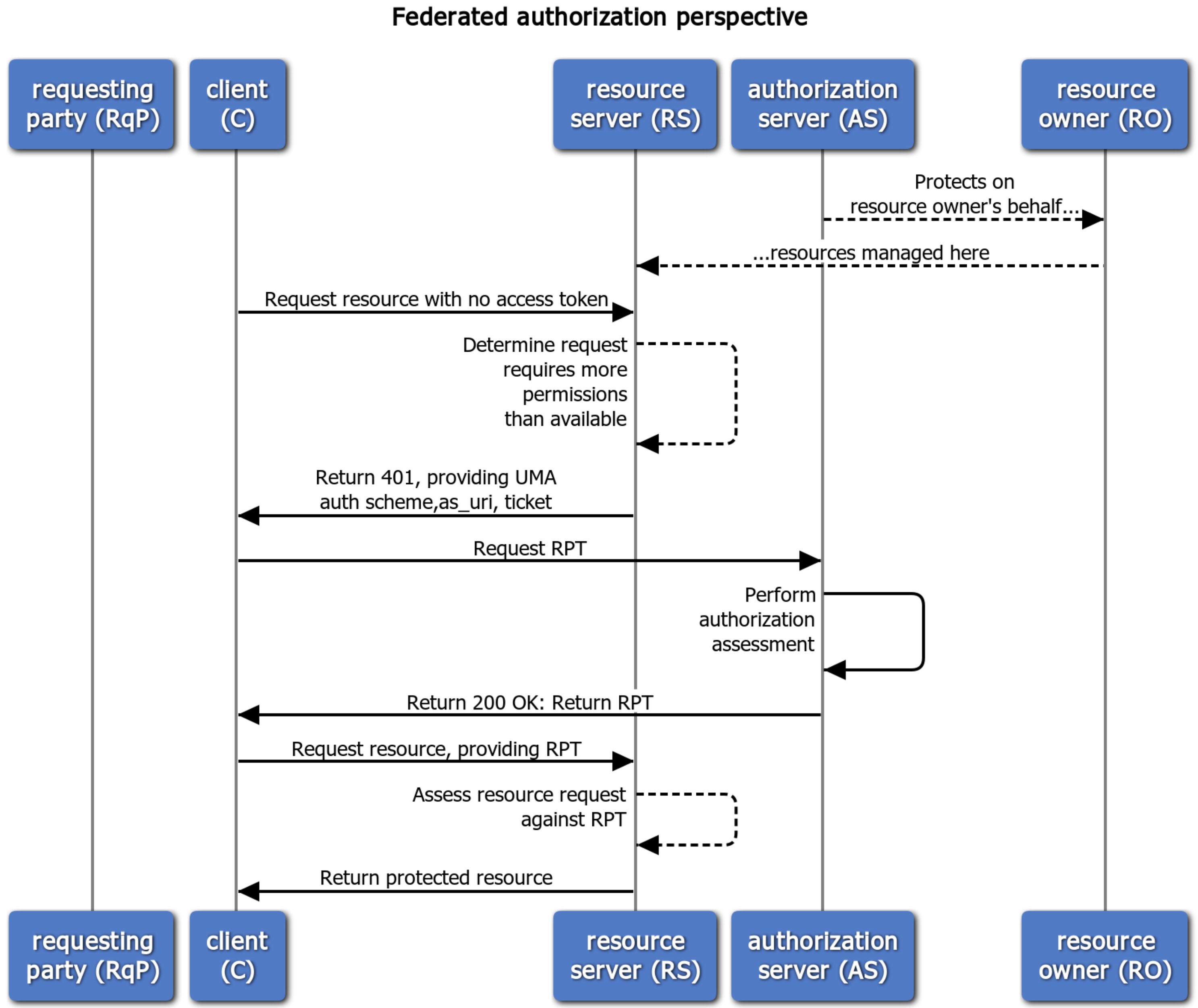
A new perspective on the UMA grant

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**?

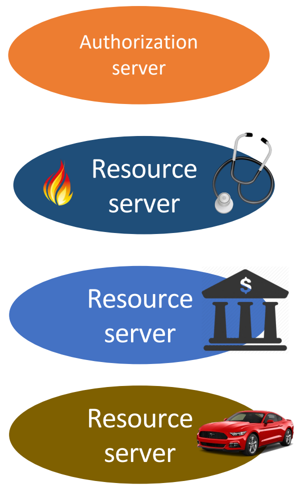
Is there anything special about **token introspection**?

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


Federated authorization perspective
requesting party (RqP)
client (C)
resource server (RS)
authorization server (AS)
resource owner (RO)
Protects on resource owner's behalf...
...resources managed here
Request resource with no access token
Determine request requires more permissions than available
Return 401, providing UMA auth scheme,as_uri, ticket
Request RPT
Perform authorization assessment
Return 200 OK: Return RPT
Request resource, providing RPT
Assess resource request against RPT
Return protected resource

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**

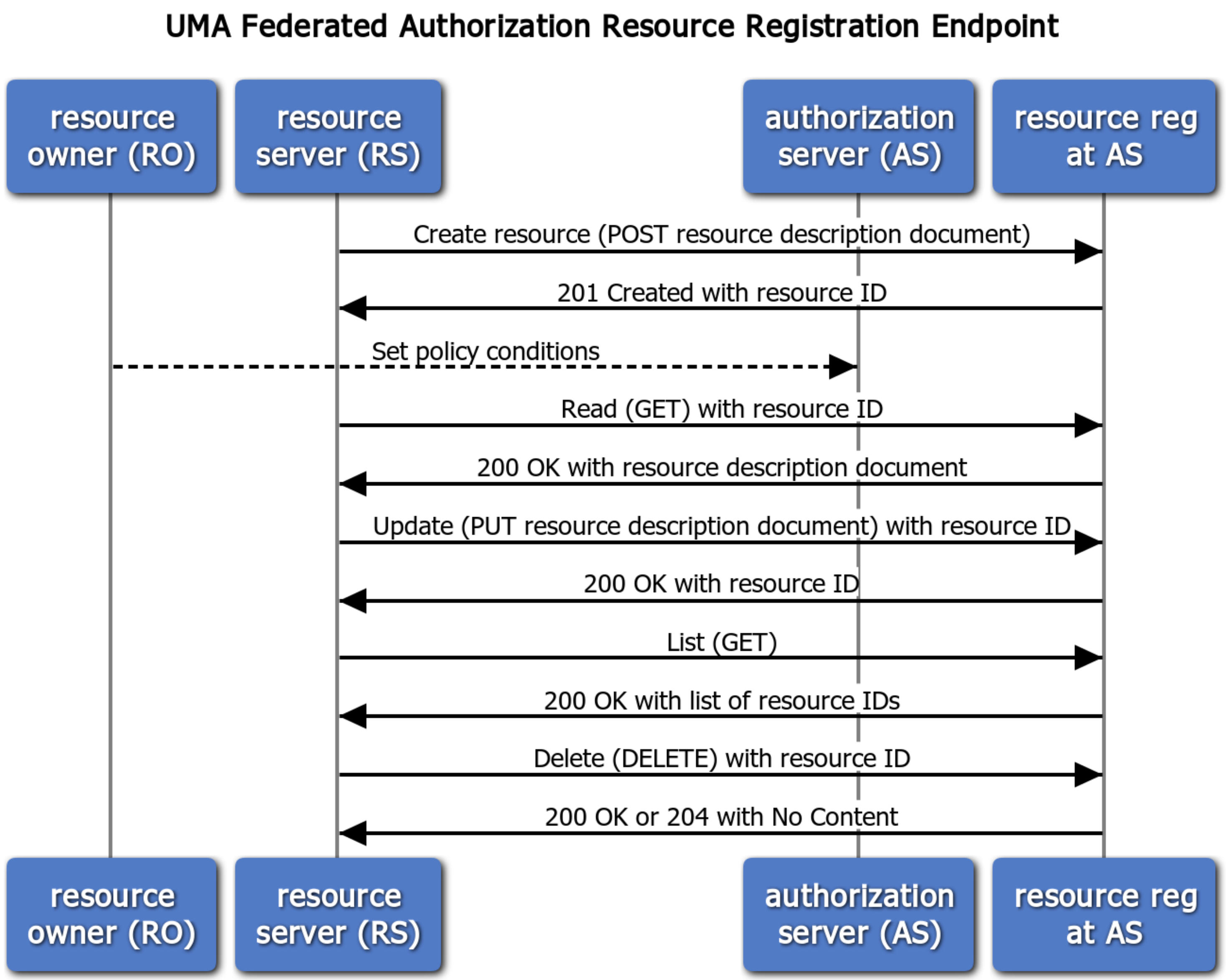


The resource registration endpoint

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 owner (RO)
resource server (RS)
authorization server (AS)
resource reg at AS
Create resource (POST resource description document)
201 Created with resource ID
Set policy conditions
Read (GET) with resource ID
200 OK with resource description document
Update (PUT resource description document) with resource ID
200 OK with resource ID
List (GET)
200 OK with list of resource IDs
Delete (DELETE) with resource ID
200 OK or 204 with No Content

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"
}
```

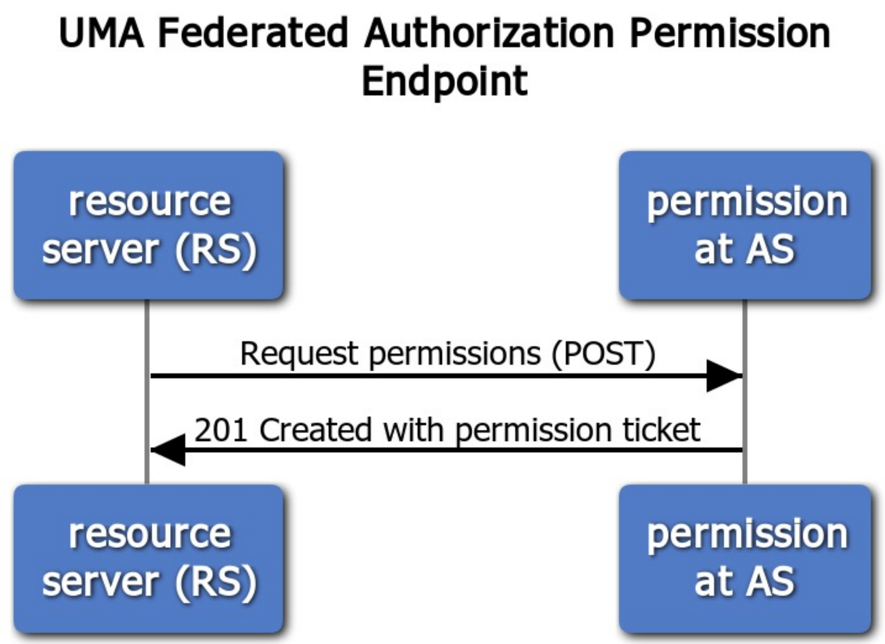
Response:

```
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**


UMA Federated Authorization Permission Endpoint
resource server (RS)
permission at AS
Request permissions (POST)
201 Created with permission ticket
resource server (RS)
permission at AS

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:

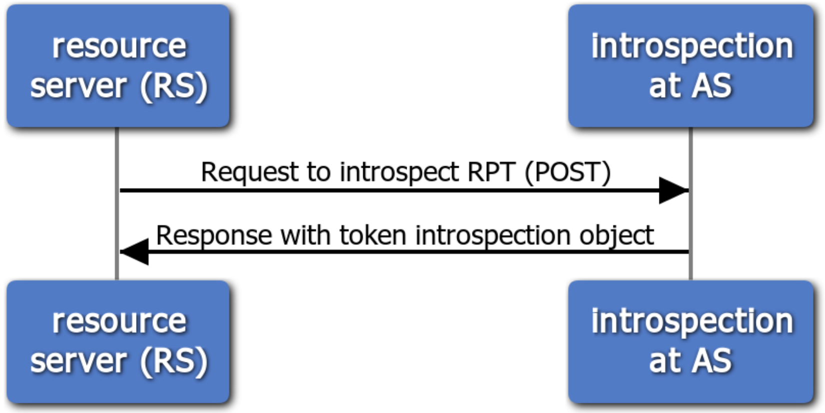
```
HTTP/1.1 201 Created
Content-Type: application/json
...
{
   "Ticket":"016f84e8-f9b9-11e0-bd6f-
0021cc6004de"
}
```

The token introspection endpoint

UMA **enhances** the token introspection response object

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

UMA Federated Authorization Token Introspection Endpoint



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
```

FedZ Review

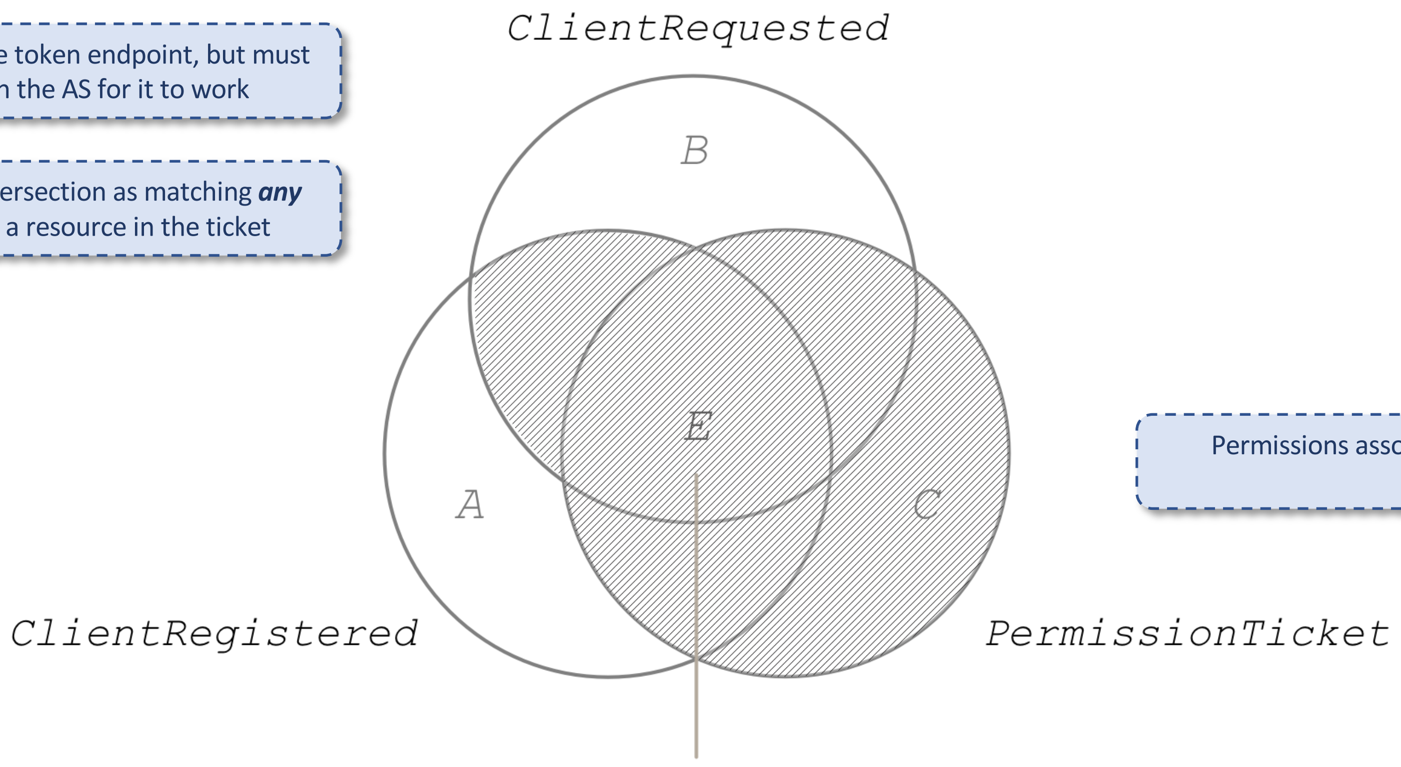
- UMA provides a reusable **description of resources and scopes**
- The resource server is able to dynamically register resources and scopes that it has – and knows how to enforce
- The RS and AS determine the appropriate access without the Clients involvement
    - Based on request hints, RO policy, presented RqP, etc
- The RS enforces access based on the AS direction (on behalf of Alice)

Authorization assessment

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

$$RequestedScopes = C \cup (A \cap B)$$

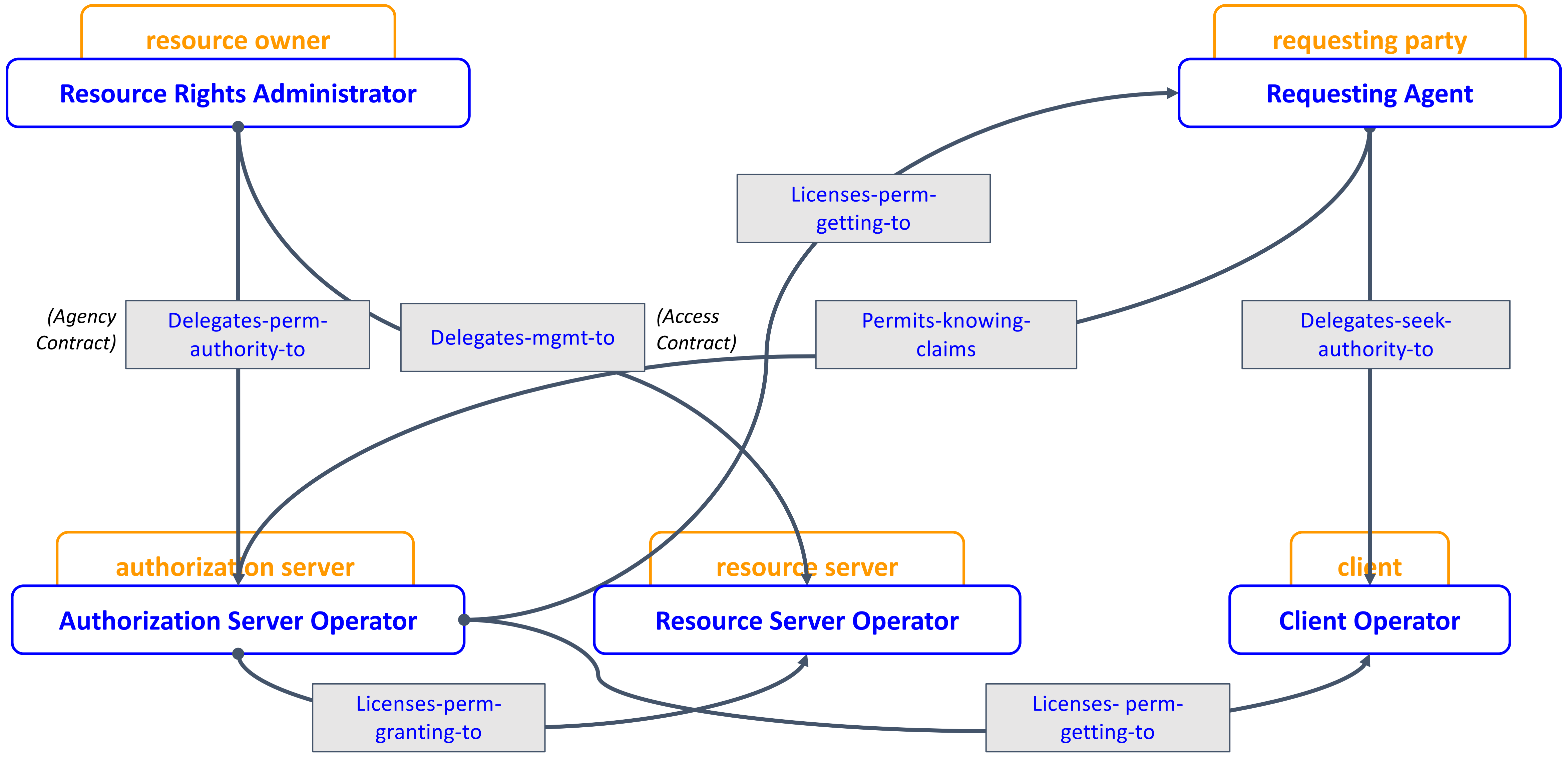
If authorization assessment results in only a subset of client-desired scopes, the AS can **choose to error**

Privacy and “BLT” implications

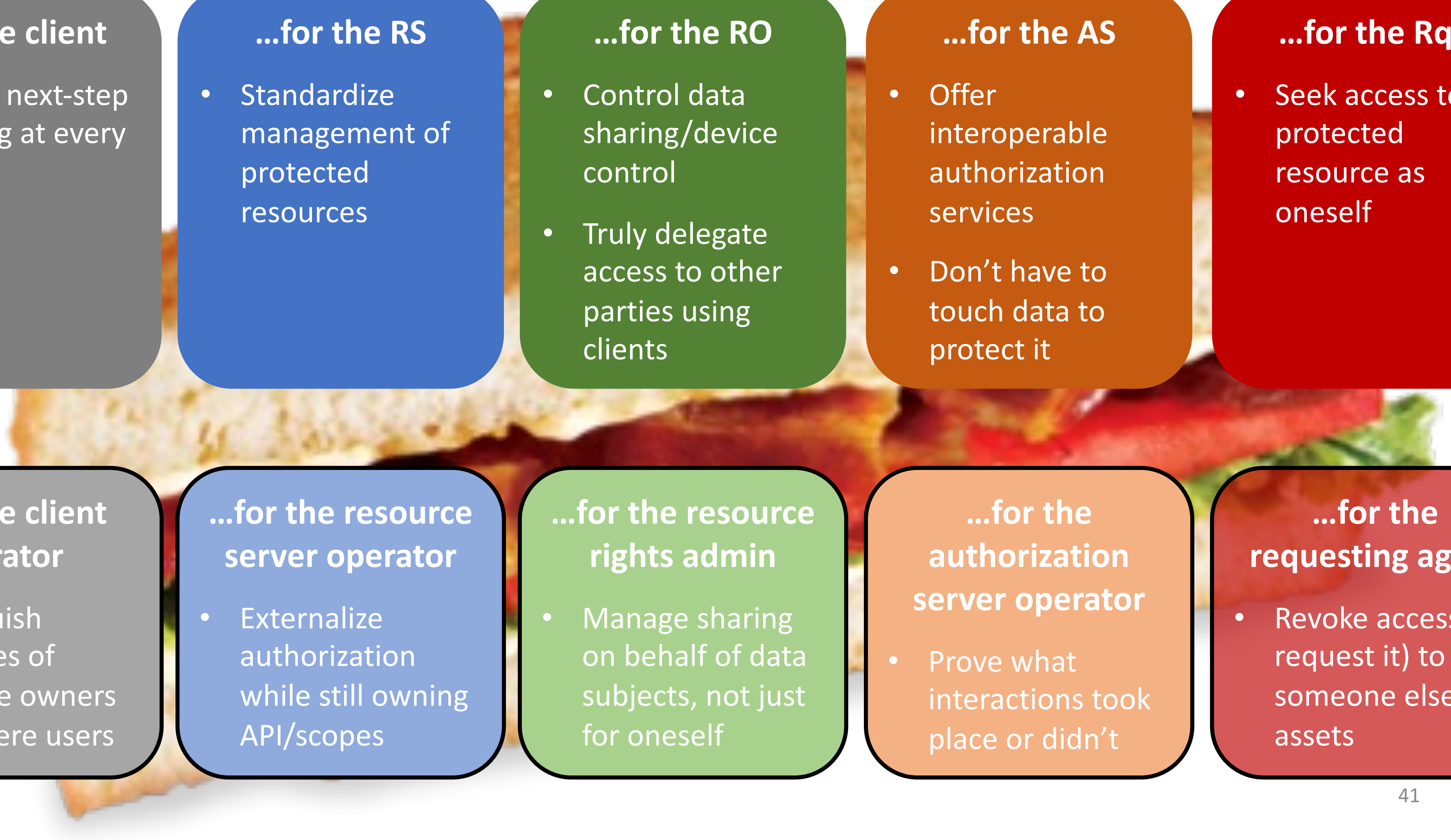
Relevance for privacy

- 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

(Most) legal relationships in the business model



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

What is the UMA WG up to?

- Julie Adam's use-case report – describes how UMA can be applied to complex patient centric data sharing, from Child to Adult
- UMA alignment to other specifications
  - How UMA and UDAP can be used together
  - How UMA can support the FAPI security profile
  - How UMA could be more backwards compatible with Oauth 2

Join us!
Thank you!
Questions?

Alec Laws, Kantara Initiative UMA Work Group chair

@aleclaws | @UMAWG

https://kantara.atlassian.net/wiki/spaces/uma/

IIWXXXV | 15 Nov 2022