Cross-party delegation with User-Managed Access 2.0 (UMA2)

draft-maler-oauth-umagrant
draft-maler-oauth-umafedauthz

Eve Maler, Maciej Machulak

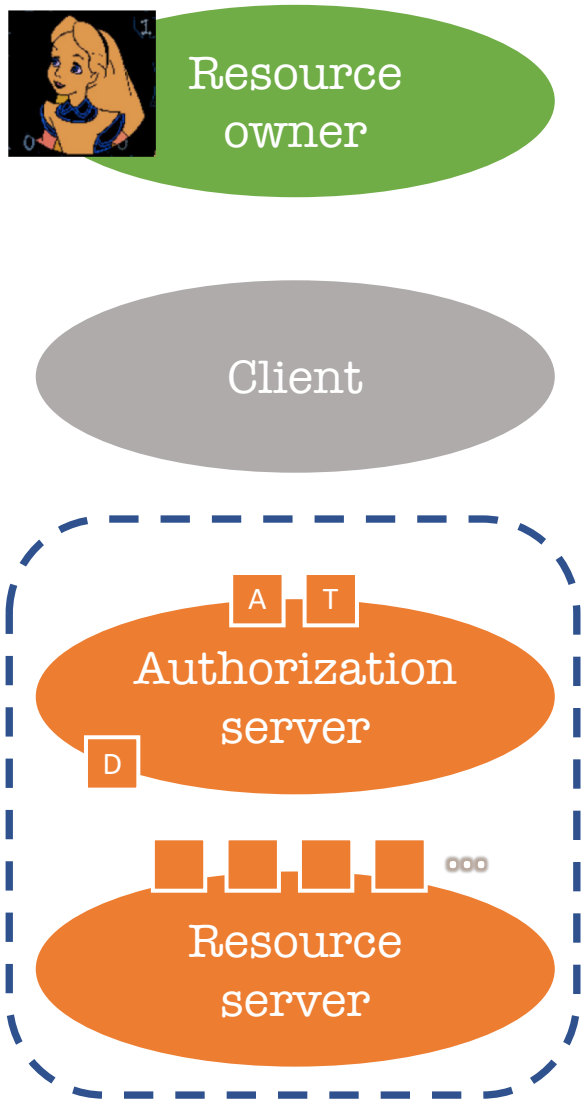
IETF 104 – OAuth Working Group

Topics

- Overview(s)
- Implementation and use in the wild
- Basic flows in the UMA grant
- Basic flows in UMA federated authorization
- If time: UMA Grant details
- If time: Privacy implications and the UMA business model
- Conclusion and next steps

Overview(s)

While OAuth enables constrained delegation of access to apps…



A authorization T token D discovery

…UMA enables cross-party sharing, in a wide ecosystem

Resource owner

Requesting party

Ahead of time | Anytime | Anytime | At run time | After the fact

UX | Share | Monitor | Withdraw | Opt in | Approve

Client

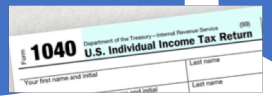
A T

Authorization server

D R P I C

Resource server

Resource server

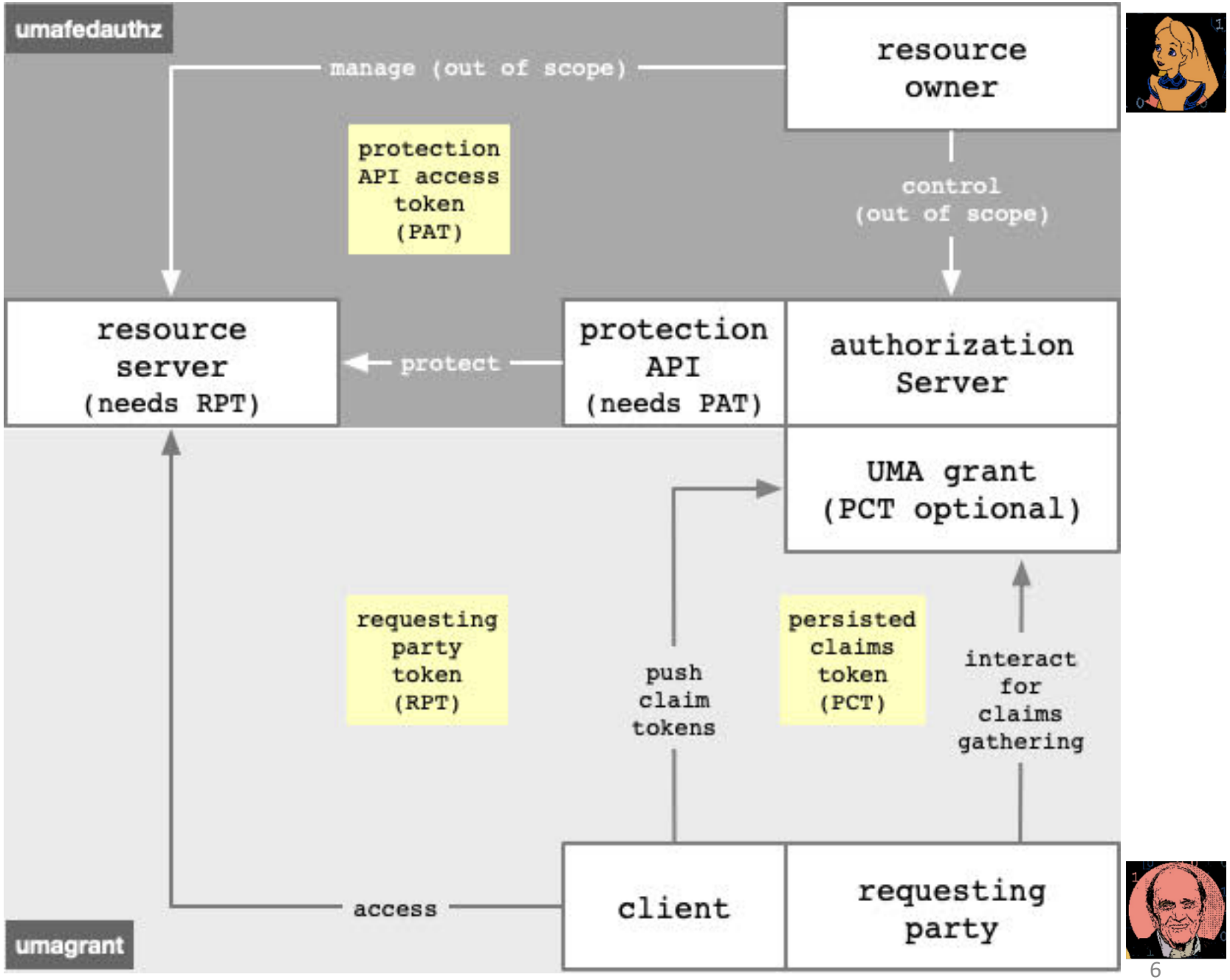


Resource server

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

UMA 2.0 overview

- OAuth grant for Alice-to-Bob delegation
- Standard API makes the AS centralizable
- Nicknames for new/enhanced tokens



IETF history

- Contributed **draft-oauth-dyn-reg**
- Contributed UMA 1.0 and 1.0.1 as **draft-hardjono-oauth-umacore** and **draft-hardjono-oauth-resource-reg**
- Influenced ACE actor and usecase work with **draft-maler-ace-oauth-uma**
- Contributed an UMA “business model” as **draft-maler-oauth-umatrust**
- Contributed a client claims framework as **draft-catalano-oauth-umaclaim**
- Refactored the specs for UMA2: **core**/**resource-reg** became **grant**/**fedauthz**

UMA grant overview

draft-maler-oauth-umagrant

- **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

"For example, bank customer (resource owner) Alice with a bank account service (resource server) can use a sharing management service (authorization server) hosted by the bank to manage access to her various protected resources by spouse Bob, accounting professional Charline, and and financial information aggregation company Decide Account, all using different client applications. Each of her bank accounts is a protected resource, and two different scopes of access she can control on them are viewing account data and accessing payment functions."

UMA federated authorization overview

draft-maler-oauth-umafedauthz

- **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
- ***Resource and scope concepts:** Slightly different in UMA

"…bank customer (resource owner) Alice has a bank account service (resource server), a cloud file system (different resource server hosted elsewhere), and a dedicated sharing management service (authorization server) hosted by the bank. She can manage access to her various protected resources by spouse Bob, accounting professional Charline, financial information aggregation company DecideAccount, and neighbor Erik (requesting parties), all using different client applications. Her bank accounts and her various files and folders are protected resources, and she can use the same sharing management service to monitor and control different scopes of access to them by these different parties, such as viewing, editing, or printing files and viewing account data or accessing payment functions.."

Quick reference to tokens

Requesting party token (RPT)

Access token for the **UMA grant** → used by an **UMA client**

When introspected, response is enhanced with a **`permissions`** parameter

Protection API access token (PAT)

Access token used with protection API → used by an **UMA RS** acting as an OAuth client

Has scope **`uma_protection`**

Accompanies requests sent to an **UMA AS** acting as an OAuth RS

Persisted claims token (PCT)

Correlation handle issued by an **UMA AS** to a client

Helps the AS “remember” claims about **a requesting party** during a future resource access attempt

Implementation and use in the wild

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
  - OpenMedReady Alliance

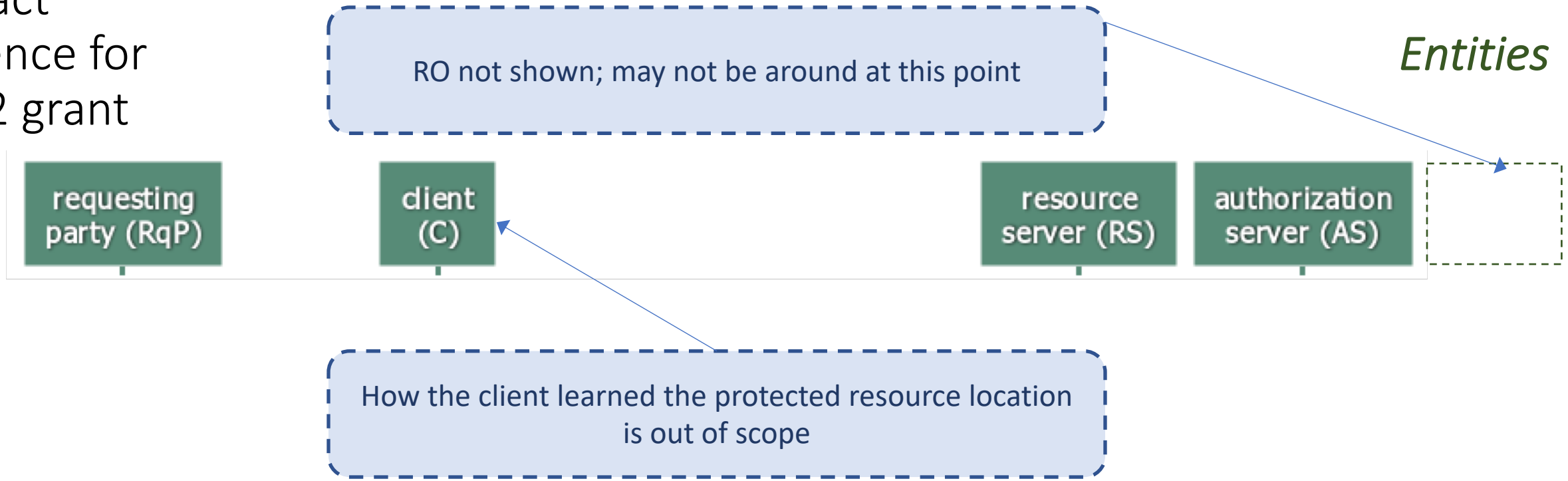
Known implementations

(see also: kantarainitiative.org/confluence/display/uma/UMA+Implementations)

- 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…

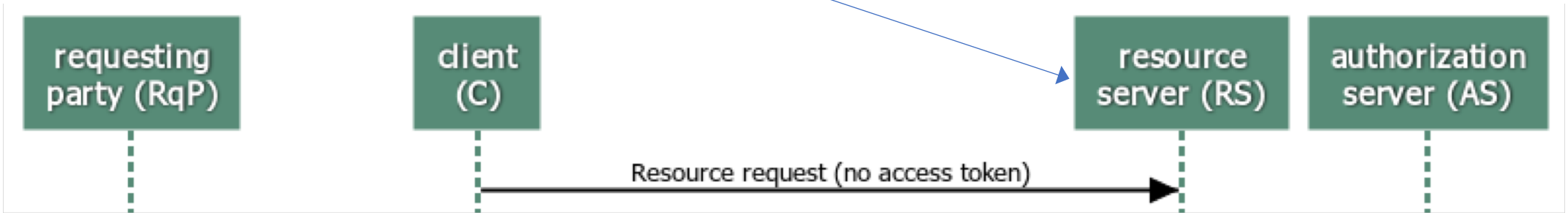
Basic flows in the UMA grant

Abstract sequence for UMA2 grant



Abstract sequence for UMA2 grant

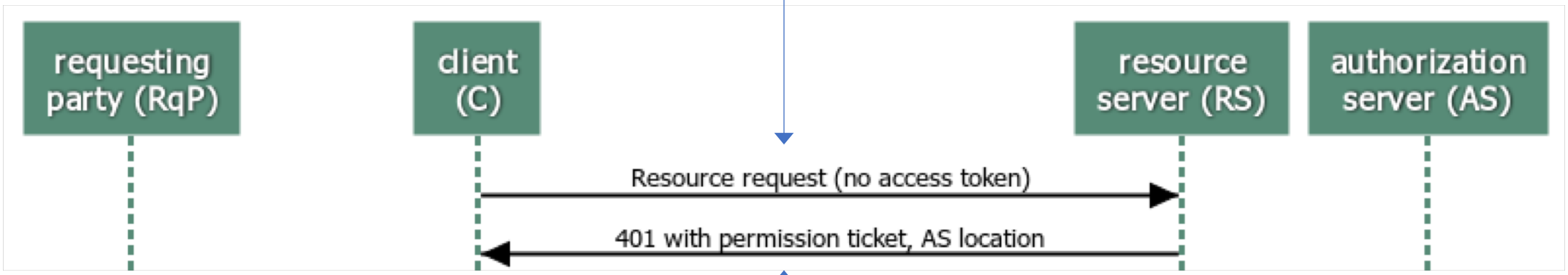
Initial resource request



Abstract sequence for UMA2 grant

Discovery/ binding

RS gets a **permission ticket** from AS and sends in header (`ticket=`); client must present at AS later



RS indicates AS to use in header (`as_uri=`); OAuth discovery doc at `.well-known/uma2-configuration` has extension metadata

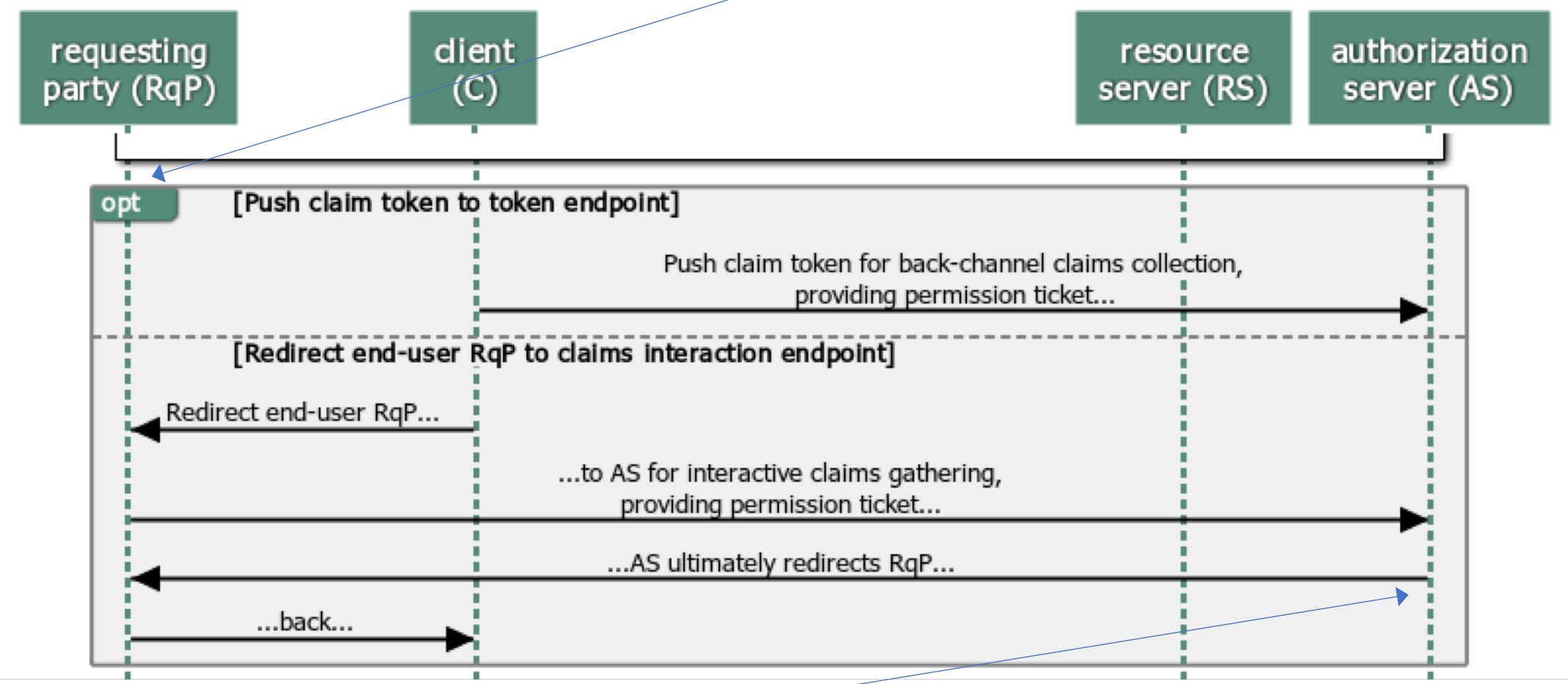
Permission ticket: like an authorization code

- **Binds client, RS, and AS:** The client is untrusted and the RS and AS may also be loosely coupled; whole flow needs to be bound
- **RS chooses permissions:** The RS interprets the client's resource request and chooses what permissions to request from the AS
- **Refreshed for security:** The client can return to the AS after non-fatal errors; the AS refreshes the ticket value when responding with such errors

Abstract sequence for UMA2 grant

Grant flow (`urn:ietf:params:oauth:grant-type:uma-ticket`) has options: front-channel (interaction)/back-channel (push) initiation and retrying

Main grant flow/claims collection

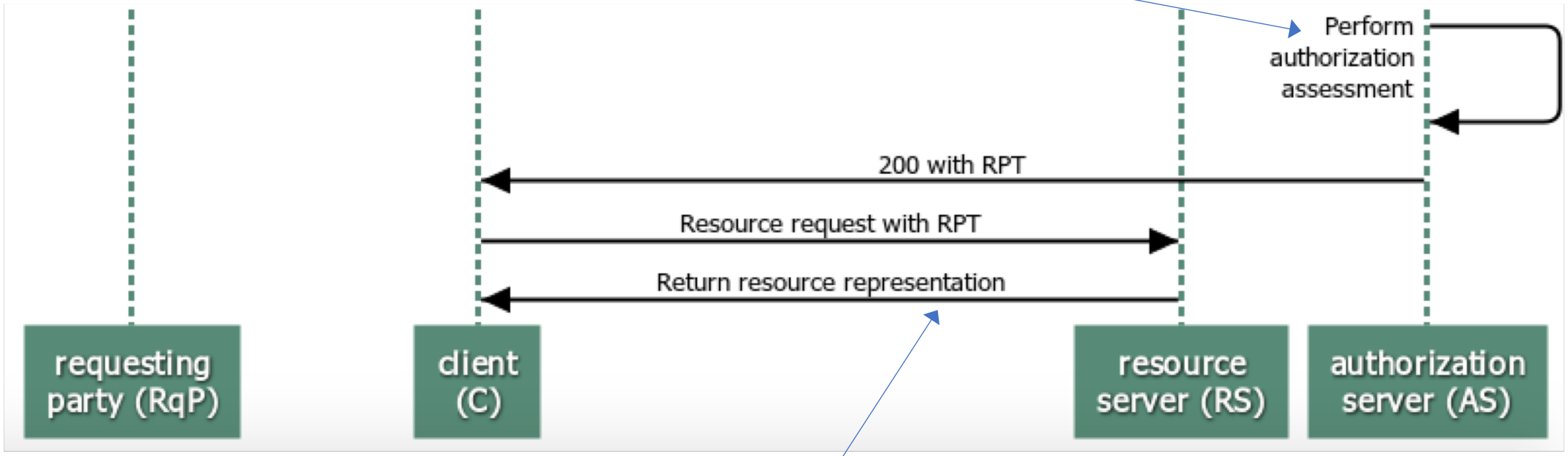


Token endpoint has additional WAM-like errors `need_info`, `request_submitted`, `request_denied`

Abstract sequence for UMA2 grant

Success/ token issuance

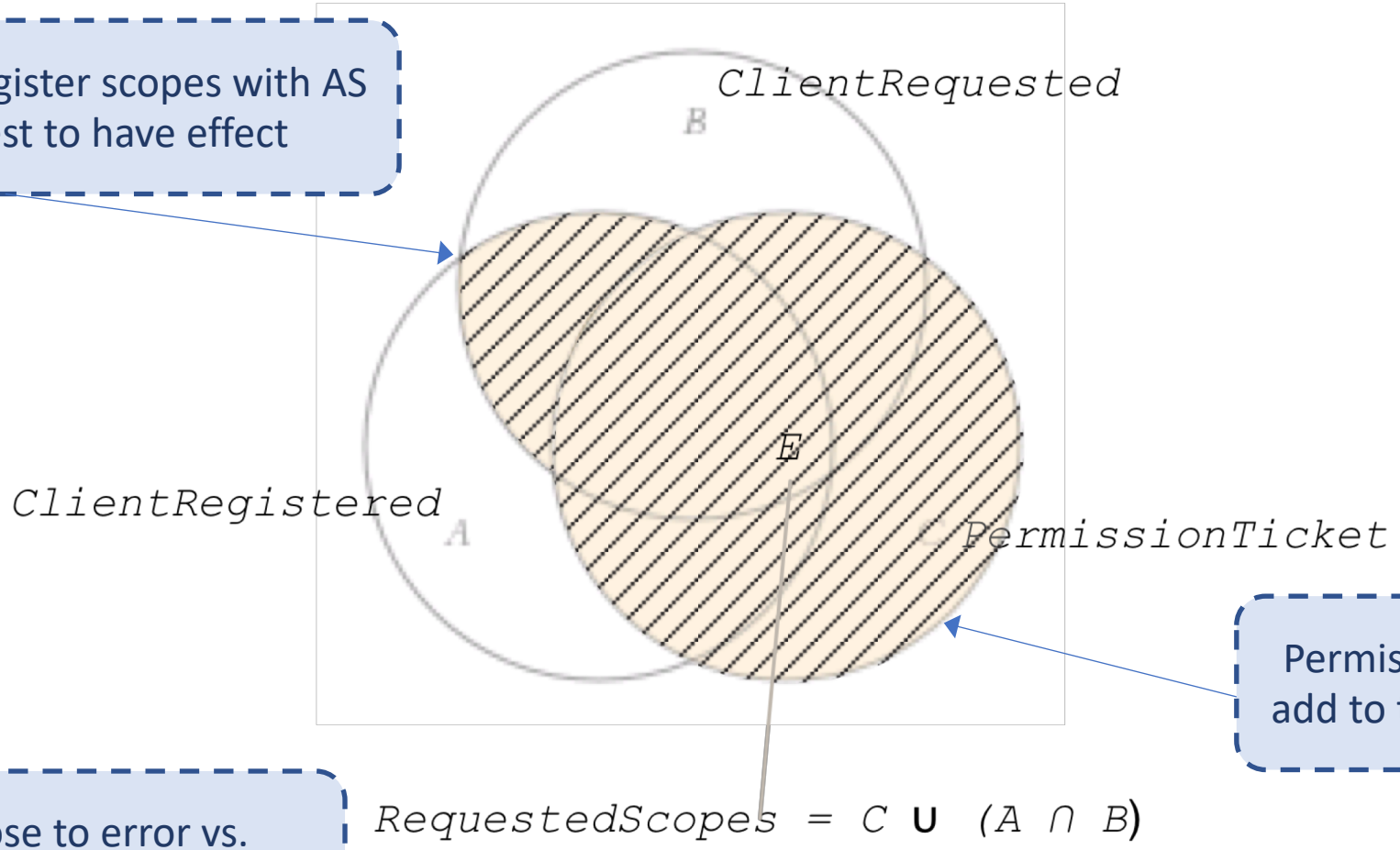
Assessment is AS-internal, but there are “set math” requirements



RPT can be upgraded, revoked, and introspected; AS can issue a refresh token (can’t re-assess later) and a PCT (must re-assess later)

Authorization assessment

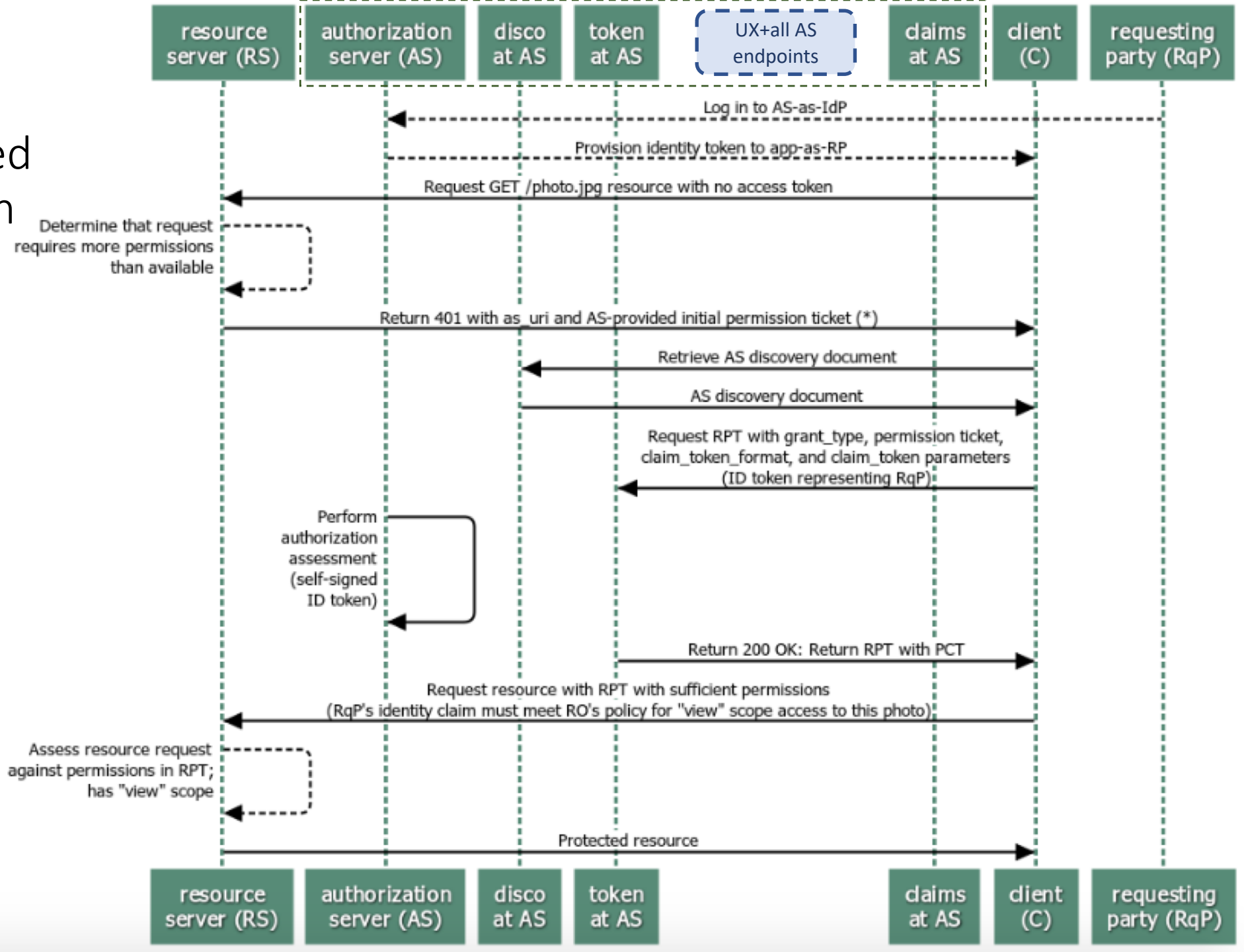
Client had to pre-register scopes with AS for scope request to have effect



Permissions in the ticket can add to total requested scopes

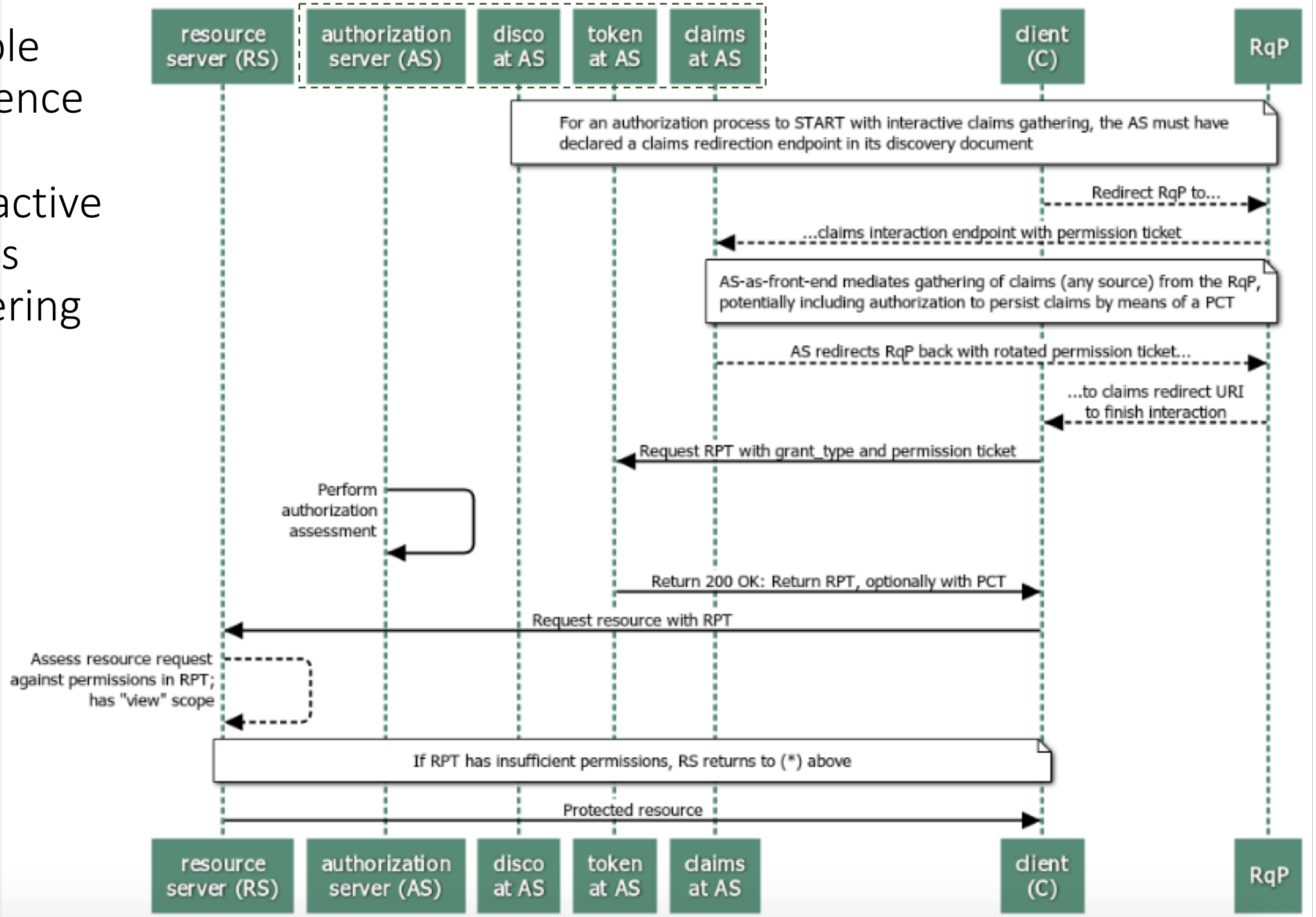
The AS can choose to error vs. granting only some requested scopes

Sample sequence with pushed claim token



"Assertion" pattern

Sample sequence with interactive claims gathering



"Authz endpoint" pattern

Observations on UMA clients

- API developers don’t have to introduce scope/entitlement logic as directly; enables externalization of entitlement logic without imposing a policy language
  - “Clients can be a lot dumber about scopes”
  - “Permissions and policies that govern access to resources get a lot more manageable”
- Error flows are easier to handle
  - “Clients can be smarter about bad flows”
  - “Clients can recover better when they don’t get an access token, without introducing risk – a ‘unauthorized’ error returns permission ticket and AS location”

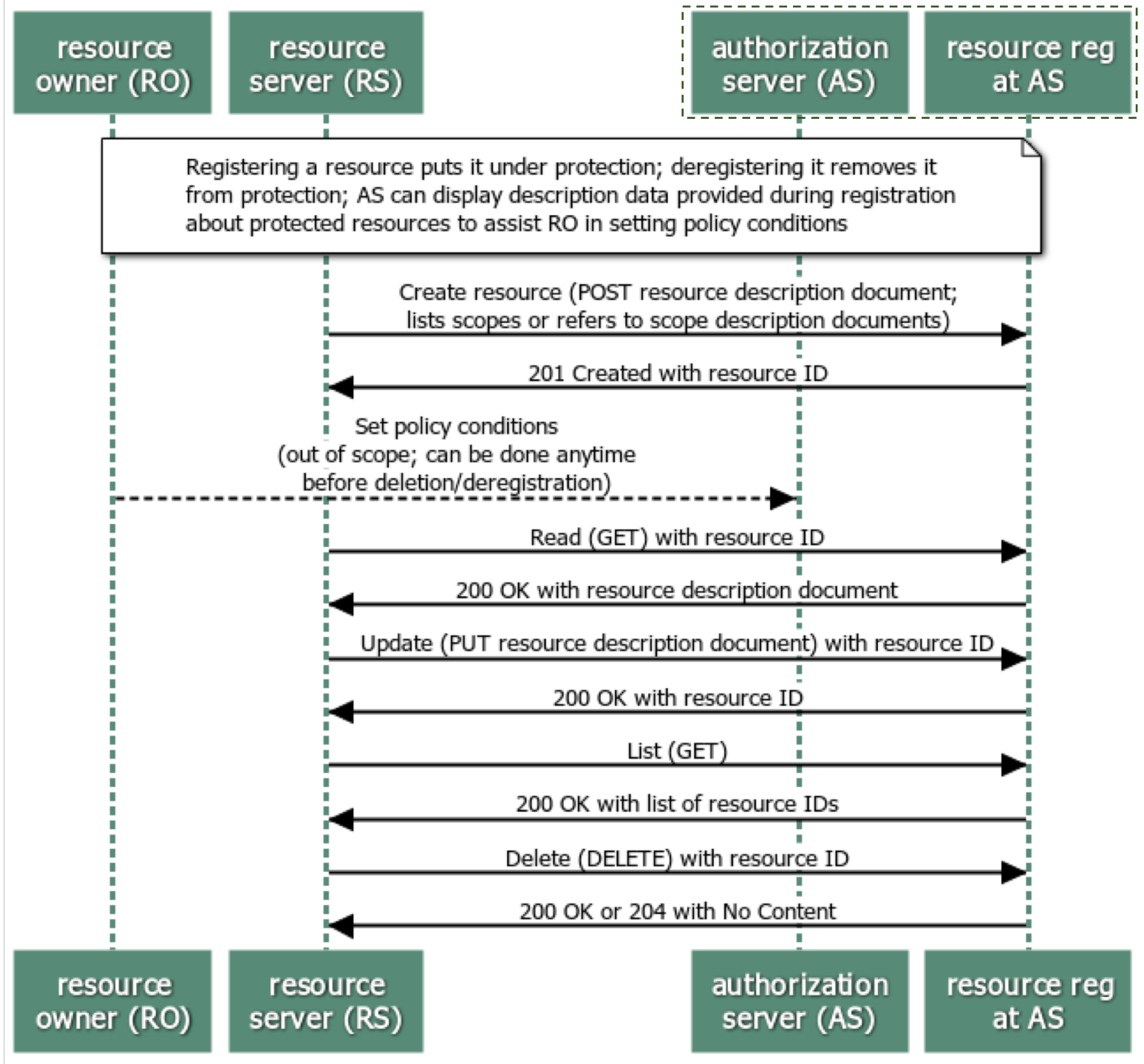
Basic flows in UMA federated authorization

The AS can present a standard protection API to enable “federating” RS’s

- RS calls a **resource registration endpoint** to protect a resource
- RS calls a **permission endpoint** to request a permission ticket
- RS can call the **token introspection endpoint**

- Protection API is protected with OAuth
  - A PAT == an access token with `uma-protection` scope
- This gives an RO a formal means of authorizing resource protection
  - Client credentials for an “enterprise RO”

Protection API



Resource registration endpoint/API

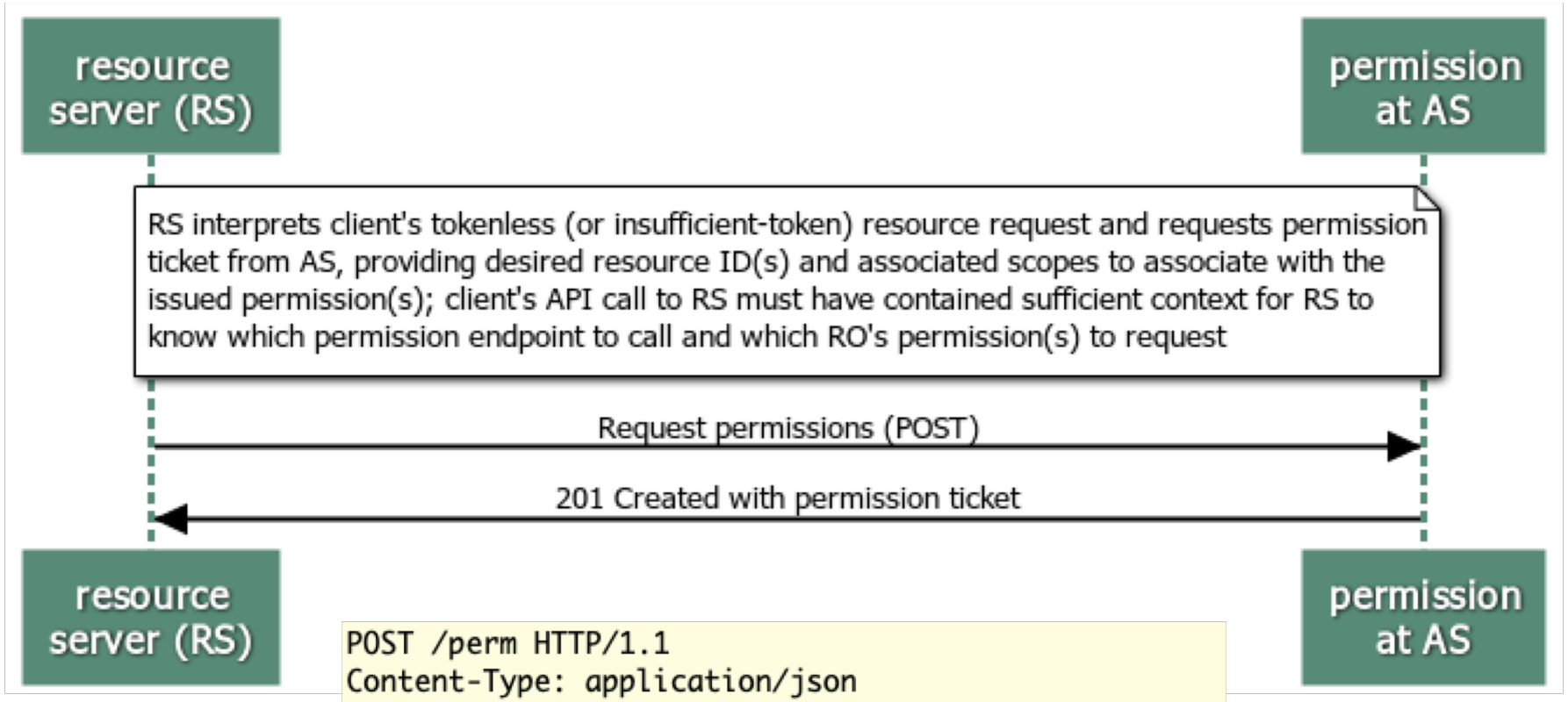
Resources and scopes

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

```
{
   "resource_scopes":[
      "view",
      "http://photoz.example.com/dev/scopes/print"
   ],
   "description":"Collection of digital photographs",
   "icon_uri":"http://www.example.com/icons/flower.png",
   "name":"Photo Album",
   "type":"http://www.example.com/rsrcs/photoalbum"
}
```

Protection API

Permission endpoint



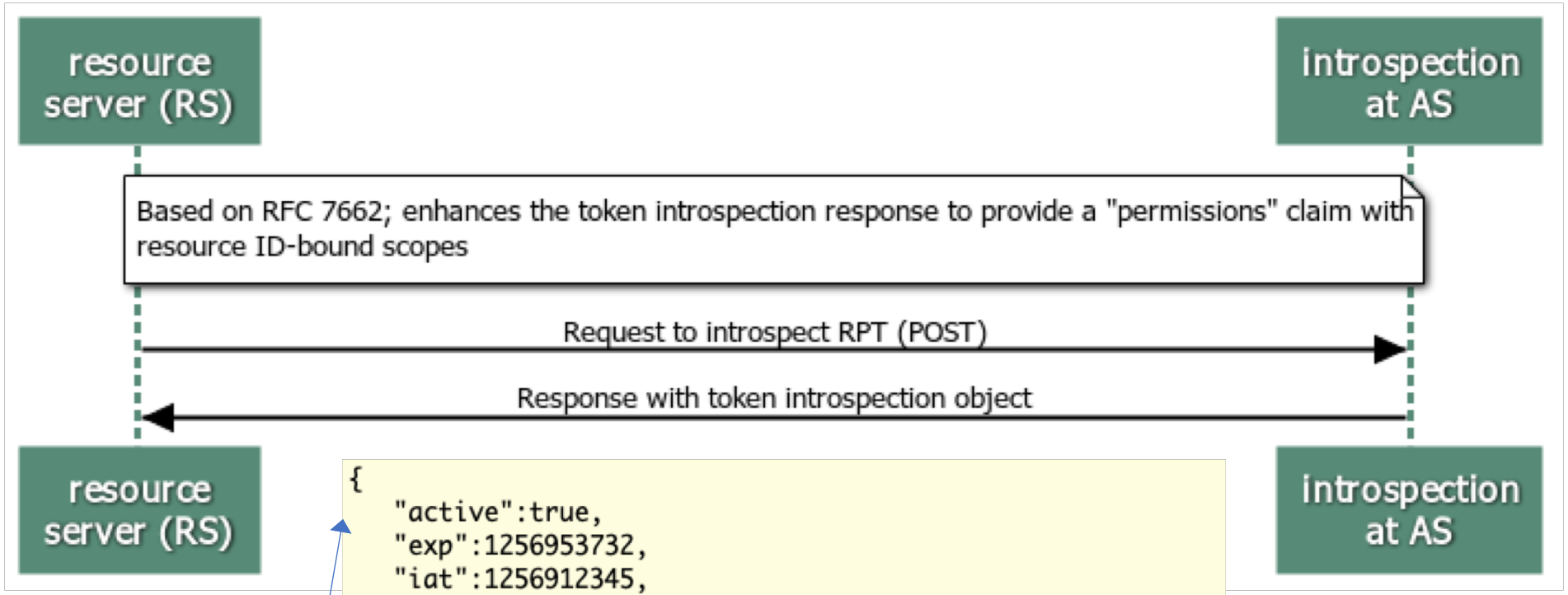
…or could put multiple permissions in one request

```
POST /perm HTTP/1.1
Content-Type: application/json
Host: as.example.com
Authorization: Bearer 204c69636b6c69
...

{
   "resource_id":"112210f47de98100",
   "resource_scopes":[
      "view",
      "http://photoz.example.com/dev/actions/print"
   ]
}
```

Protection API

Token introspection endpoint



Sample token introspection object

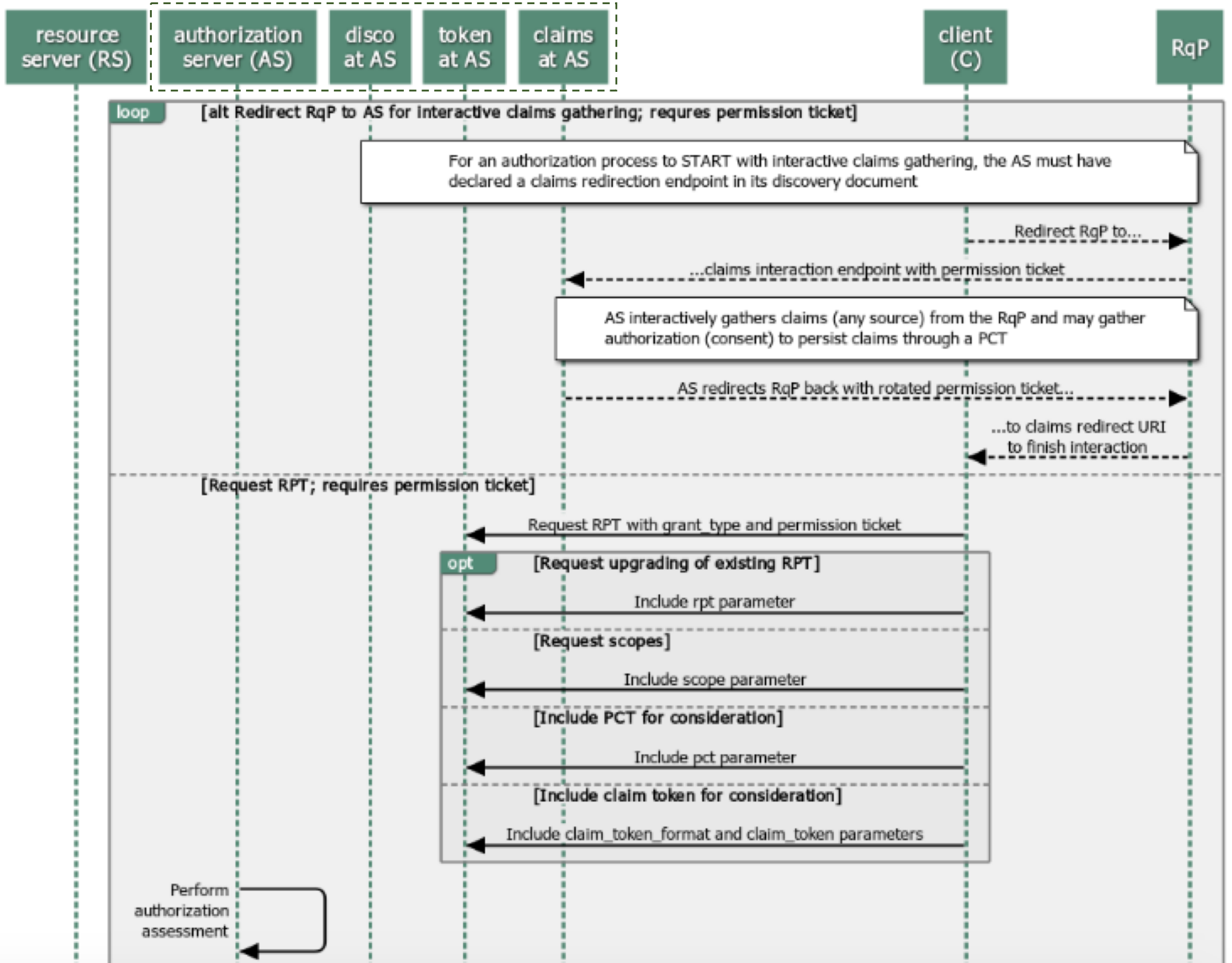
```
{
   "active":true,
   "exp":1256953732,
   "iat":1256912345,
   "permissions":[
      {
         "resource_id":"112210f47de98100",
         "resource_scopes":[
            "view",
            "http://photoz.example.com/dev/actions/print"
         ],
         "exp":1256953732
      }
   ]
}
```

Observations on tokens, tickets, and discovery

- Permission ticket binds the RPT to its audience firmly
  - Doesn’t require the client to signal where it intends to use the access token (à la **draft-ietf-oauth-resource-indicators**)
  - Requires the client to attempt access to a resource that turns out to be protected (à la the “Dynamic Scopes” discussion of Jun 2018)
- RPT assumes bearer, and PAT makes bearer MTI, but PoP is also mentioned
  - ACE token is PoP
- Protection API assumes the RS is online to request a ticket
  - ACE has proposed an RS-offline binding mechanism
  - Returns an AS Request Creation Hints message as a CBOR map: mandatory **`AS`** element with absolute URI; optional parameters **`audience`**, **`kid`**, **`nonce`**, **`scope`**

If time: UMA Grant details

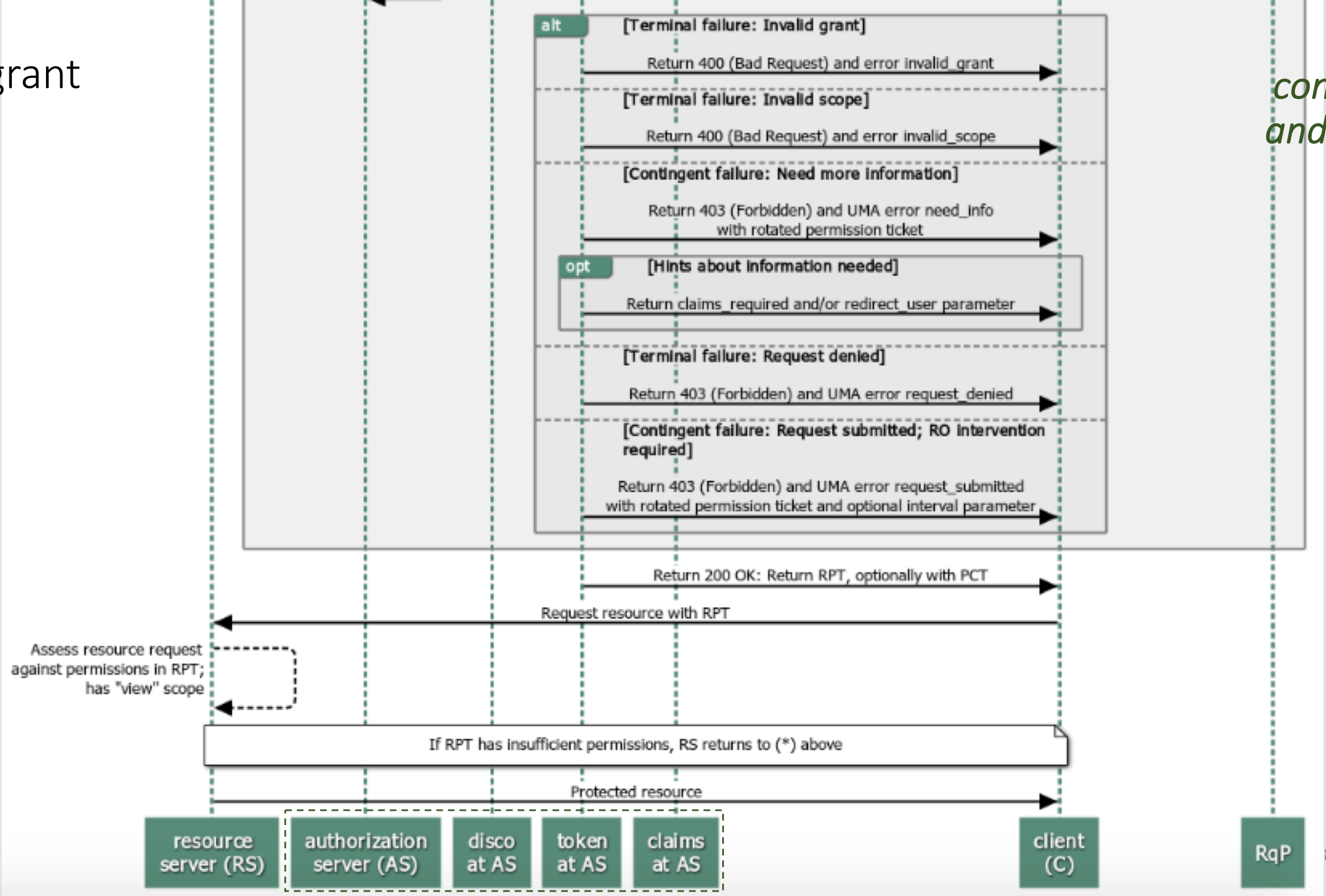
UMA grant details



Redirection and token endpoint request options

UMA grant details

Error conditions and retries

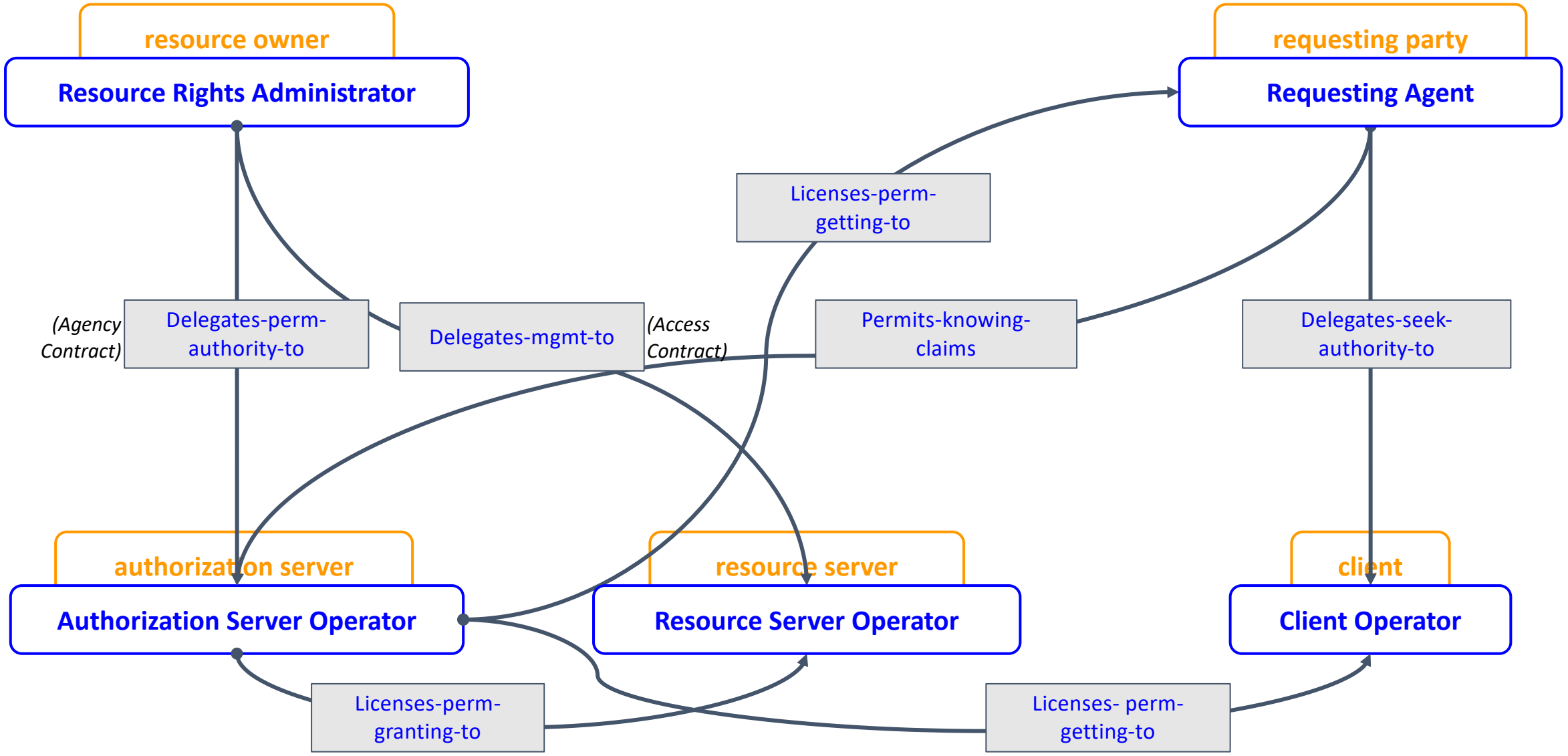


If time: Privacy implications and the UMA business model

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



Conclusion and next steps

UMA adds…

…to the client

- Simpler next-step handling at every point
- Can become “really dumb”

…to the RS

- Standardize management of protected resources
- Externalize authorization while still owning API/scopes

…to the RO

- Control data sharing and device control with others
- Truly delegate access to other client-using parties

…to the AS

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

…to the RqP

- Seek access to a protected resource as oneself
- Revoke access meaningfully

Relationship of the UMA and OAuth WGs

- We propose for the OAuth WG to adopt the UMA2 specs as work items
  - Operational interop and business model work currently continue at Kantara
  - The UMA WG can continue technical profiling and/or work out transition efforts as required
  - We can figure out WG-WG liaisons/communications; there are several mutual participants
- Discussion on these points?