User Managed Access

User Managed Access

- Kantara project

  ".. address the harmonization and interoperability challenges that exist between enterprise identity systems, Web 2.0 applications and services, and Web-based initiatives."

- UMA WG

  ".. to develop specs that let an individual control the authorization of data sharing and service access made between online services on the individual's behalf"

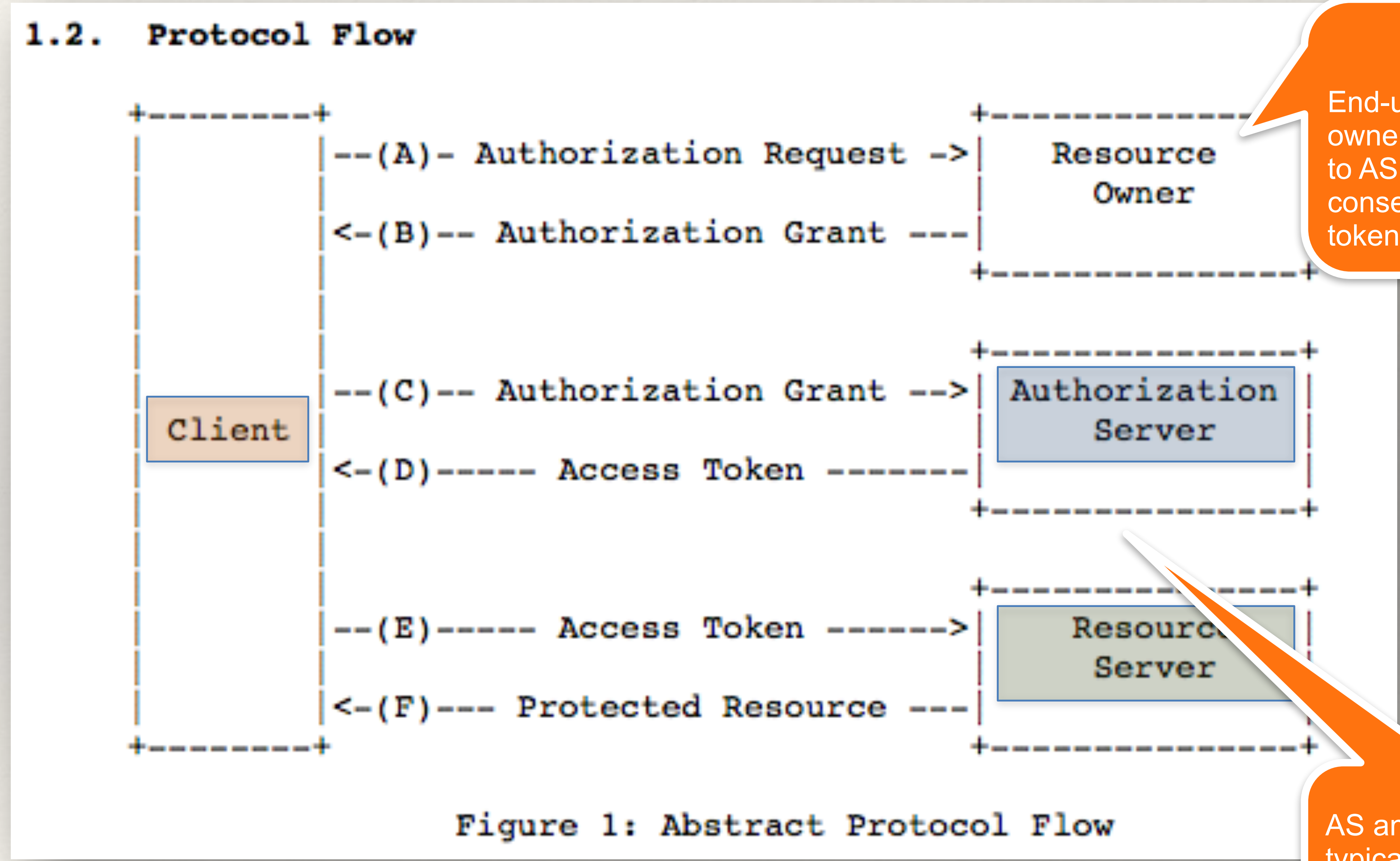
Privacy by design

- Proactive not reactive
- Benefits all actors in an online service ecosystem
- Proper testable behavior
- User centric

Use cases

- **Person-to-self**
  - Attribute release
- **Person-to-organization**
  - Health care
  - Work application
- **Person-to-person**
  - Student information
- IoT

OAuth2 is a 3-entity protocol for securing API-calls in a user context



1.2.   Protocol Flow

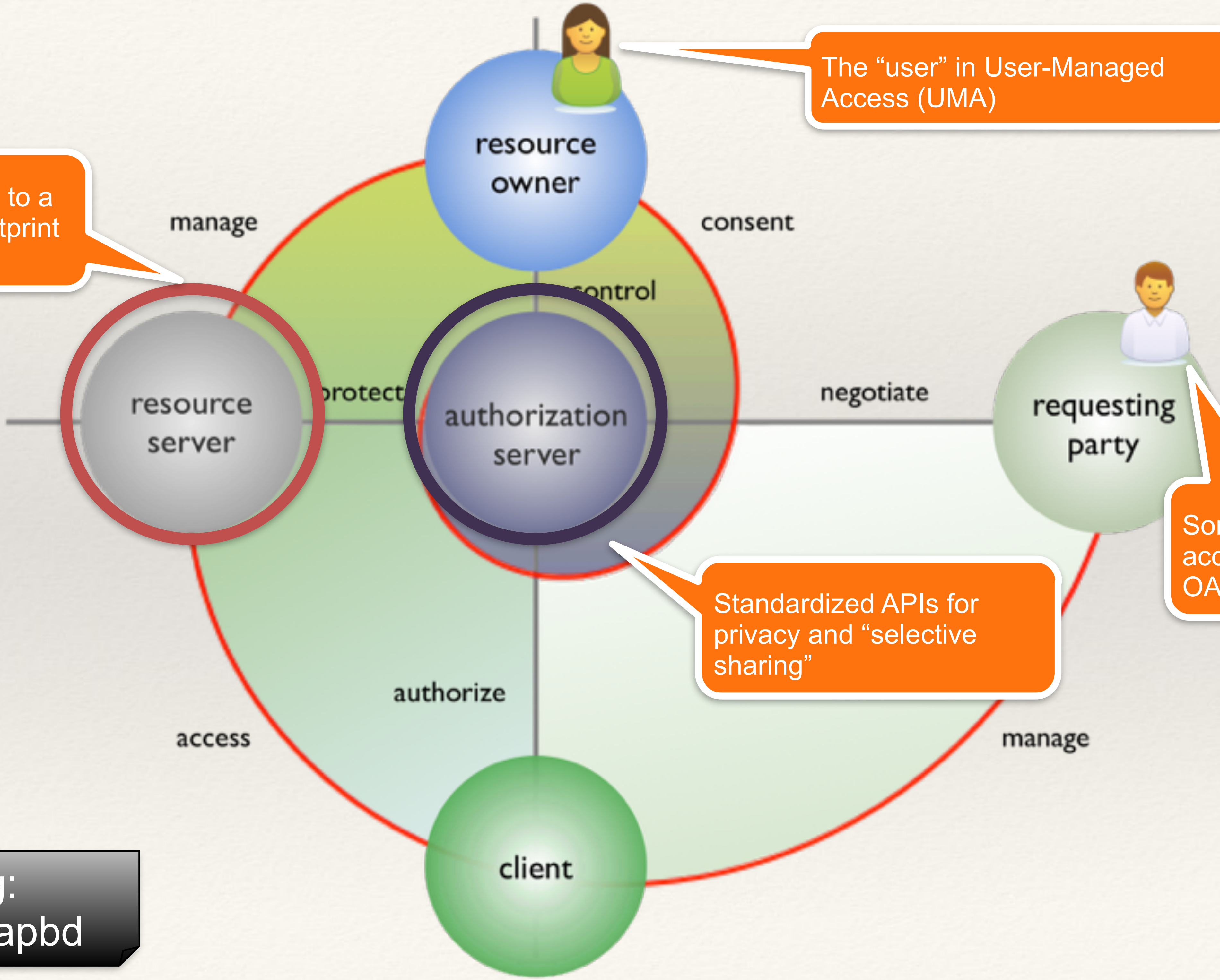
```
     +--------+                               +-------------
     |        |--(A)- Authorization Request ->|   Resource
     |        |                               |     Owner
     |        |<-(B)-- Authorization Grant ---|
     |        |                               +---------------+
     |        |
     |        |                               +---------------+
     |        |--(C)-- Authorization Grant -->| Authorization |
     | Client |                               |     Server    |
     |        |<-(D)----- Access Token -------|               |
     |        |                               +---------------+
     |        |
     |        |                               +---------------+
     |        |--(E)----- Access Token ------>|    Resourc
     |        |                               |     Server
     |        |<-(F)--- Protected Resource ---|
     +--------+                               +---------------+
```

Figure 1: Abstract Protocol Flow

End-user resource owner gets redirected to AS to log in and consent to access token issuance

AS and RS are typically in the same domain and communicate in a proprietary way

UMAs original goal: apply privacy-by-design to OAuth data sharing



The “user” in User-Managed Access (UMA)

Outsources protection to a centralized “digital footprint control console”

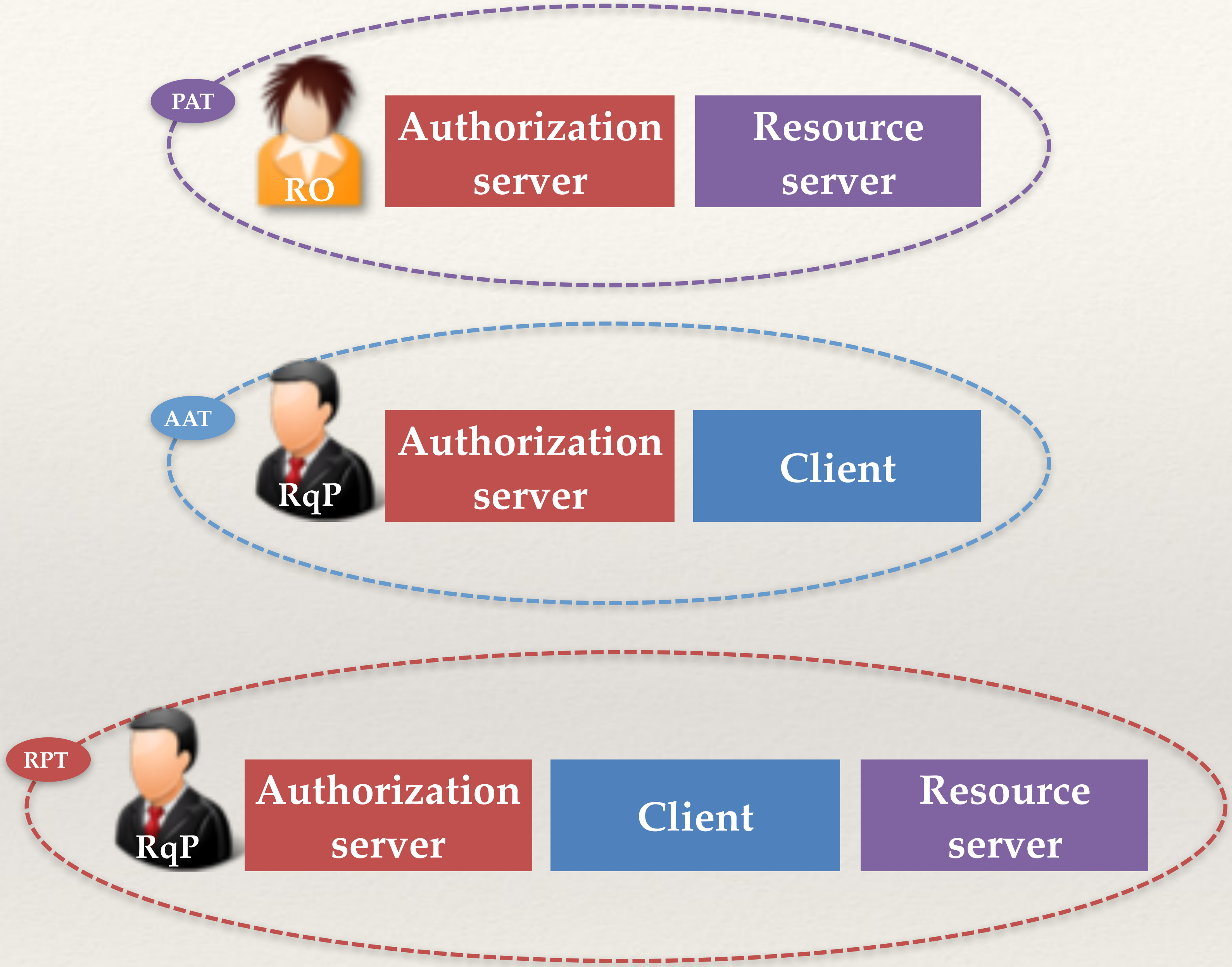
Some guy not accounted for in OAuth…

Standardized APIs for privacy and “selective sharing”

Further reading: tinyurl.com/umapbd

The 3 Phases of UMA -and associated tokens

- Protect a resource
  - Protection API Token (PAT)
- Get Authorization
  - Authorization API Token (AAT)
- Access a resource
  - Requesting Party Token (RPT)



PAT

RO

Authorization server

Resource server

AAT

RqP

Authorization server

Client

RPT

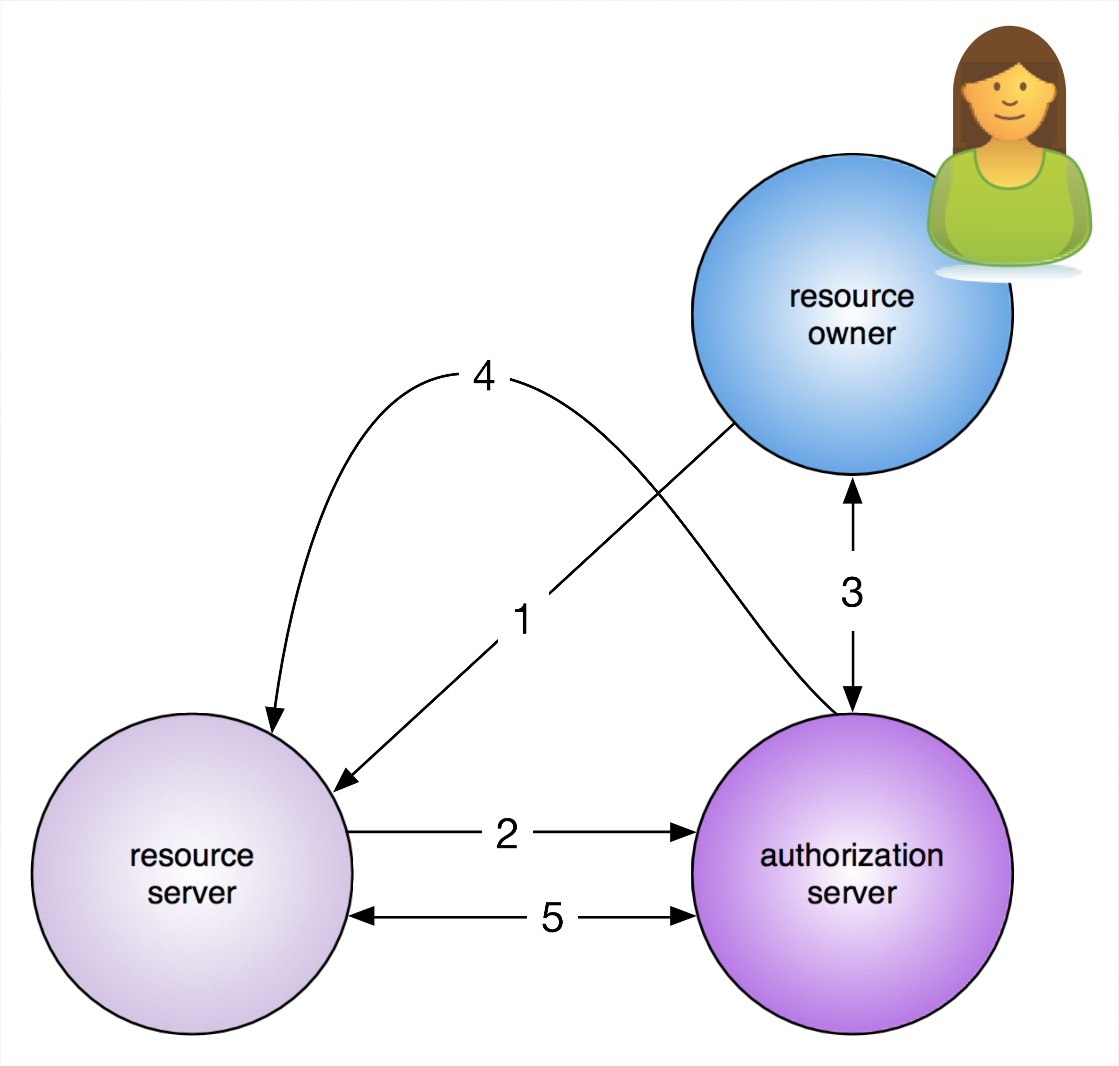
RqP

Authorization server

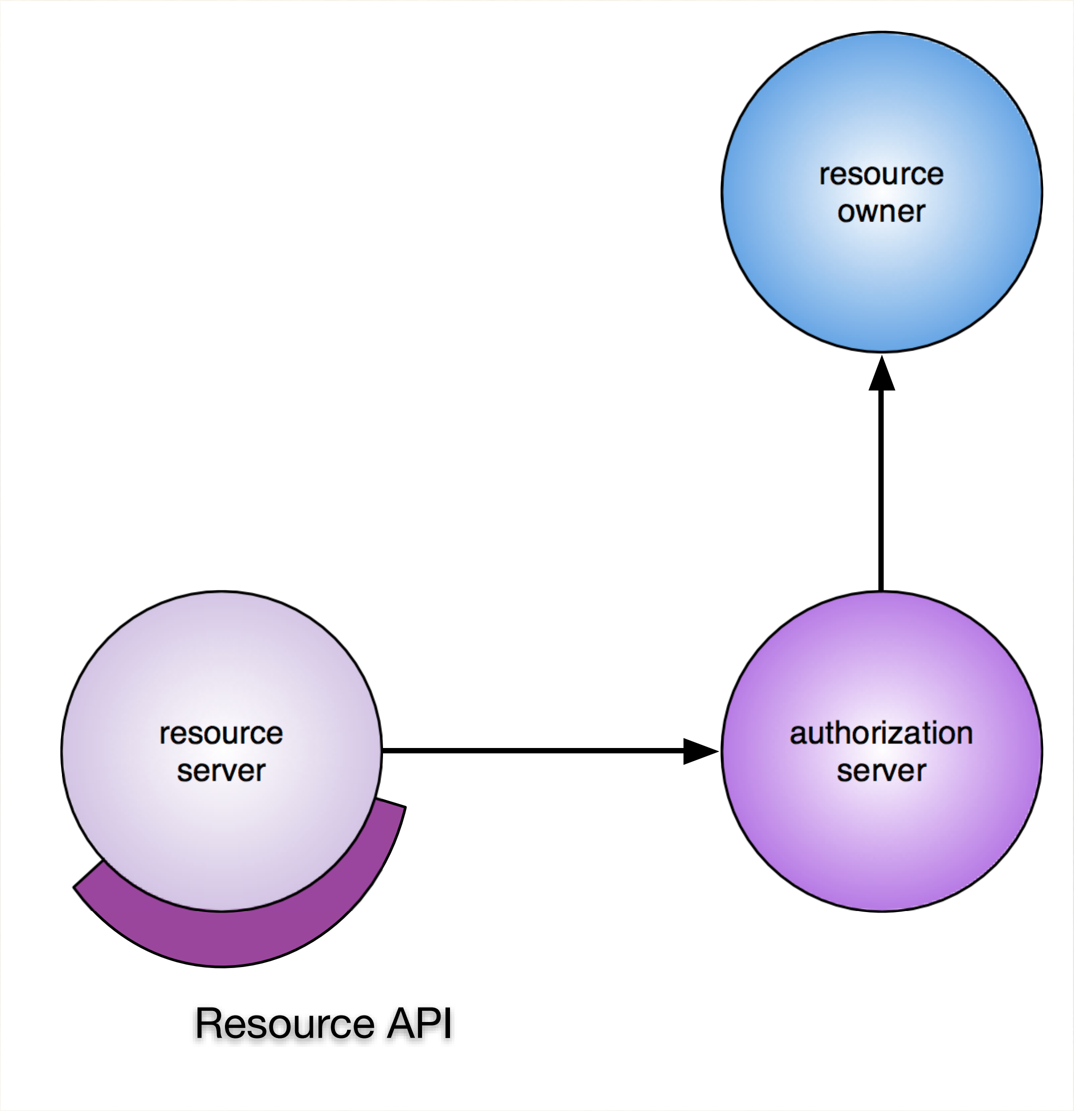
Client

Resource server

1. Introducing the Authorization Server



2. Translating resource API



Resource set description

- A JSON document with the following parameters
  - name

    REQUIRED. A human-readable string describing a set of one or more resources.
  - uri

    OPTIONAL. A URI that provides the network location for the resource set being registered
  - type

    OPTIONAL. A string uniquely identifying the semantics of the resource set
  - scopes

    REQUIRED. An array of strings indicating the available scopes for this resource set.
  - icon_uri

    OPTIONAL. A URI for a graphic icon representing the resource set

Scopes description

- a JSON document with the following properties:
    - name

      REQUIRED. A human-readable string describing some scope (extent) of access.
    - icon_uri

      OPTIONAL. A URI for a graphic icon representing the scope.

Standardized format for describing resource sets

REST interface

GET

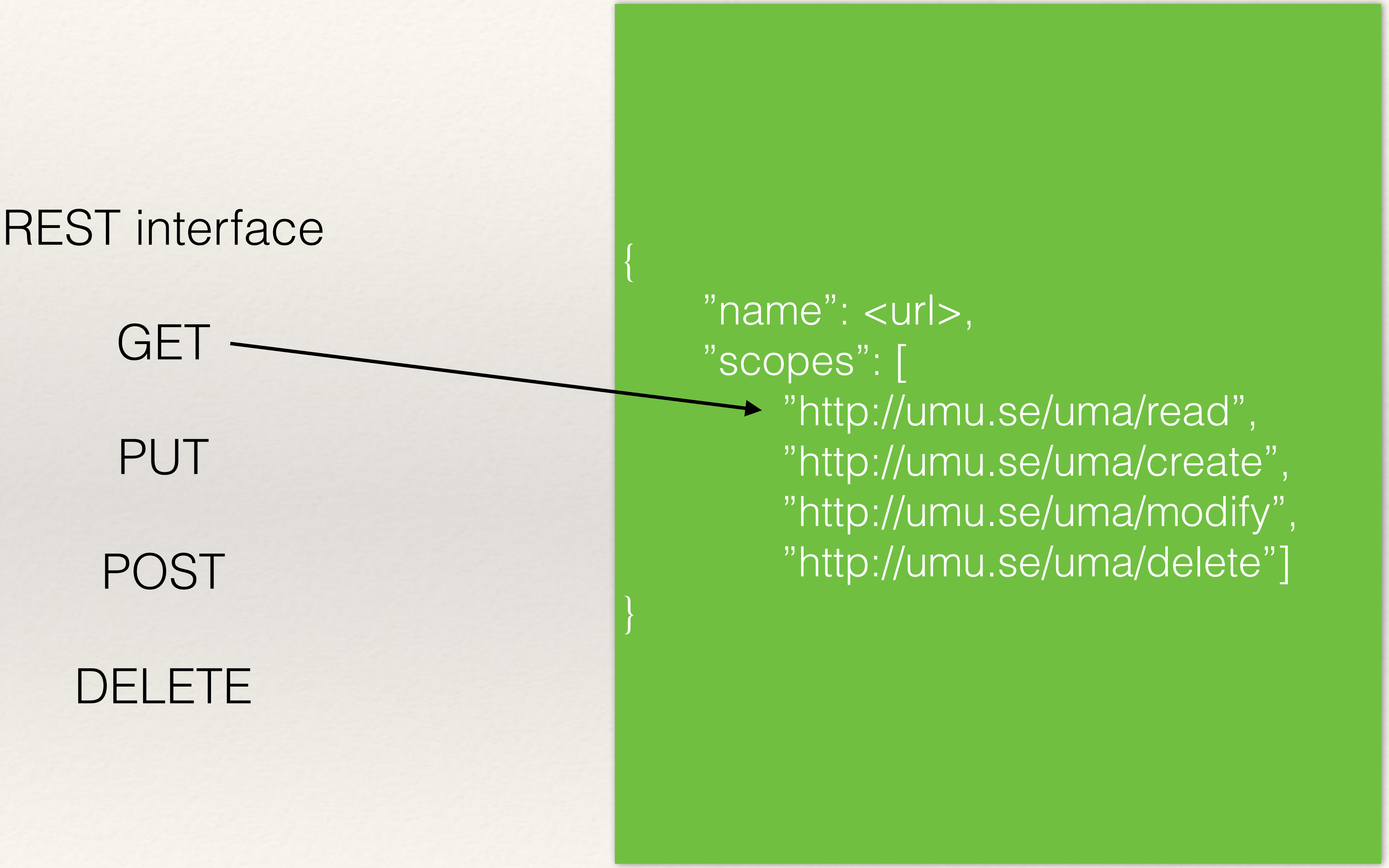
PUT

POST

DELETE

```
{
        ”name”: <url>,
        ”scopes”: [
                ”http://umu.se/uma/read”,
                ”http://umu.se/uma/create”,
                ”http://umu.se/uma/modify”,
                ”http://umu.se/uma/delete”]
}
```

Standardized format for describing resource sets



REST interface

GET

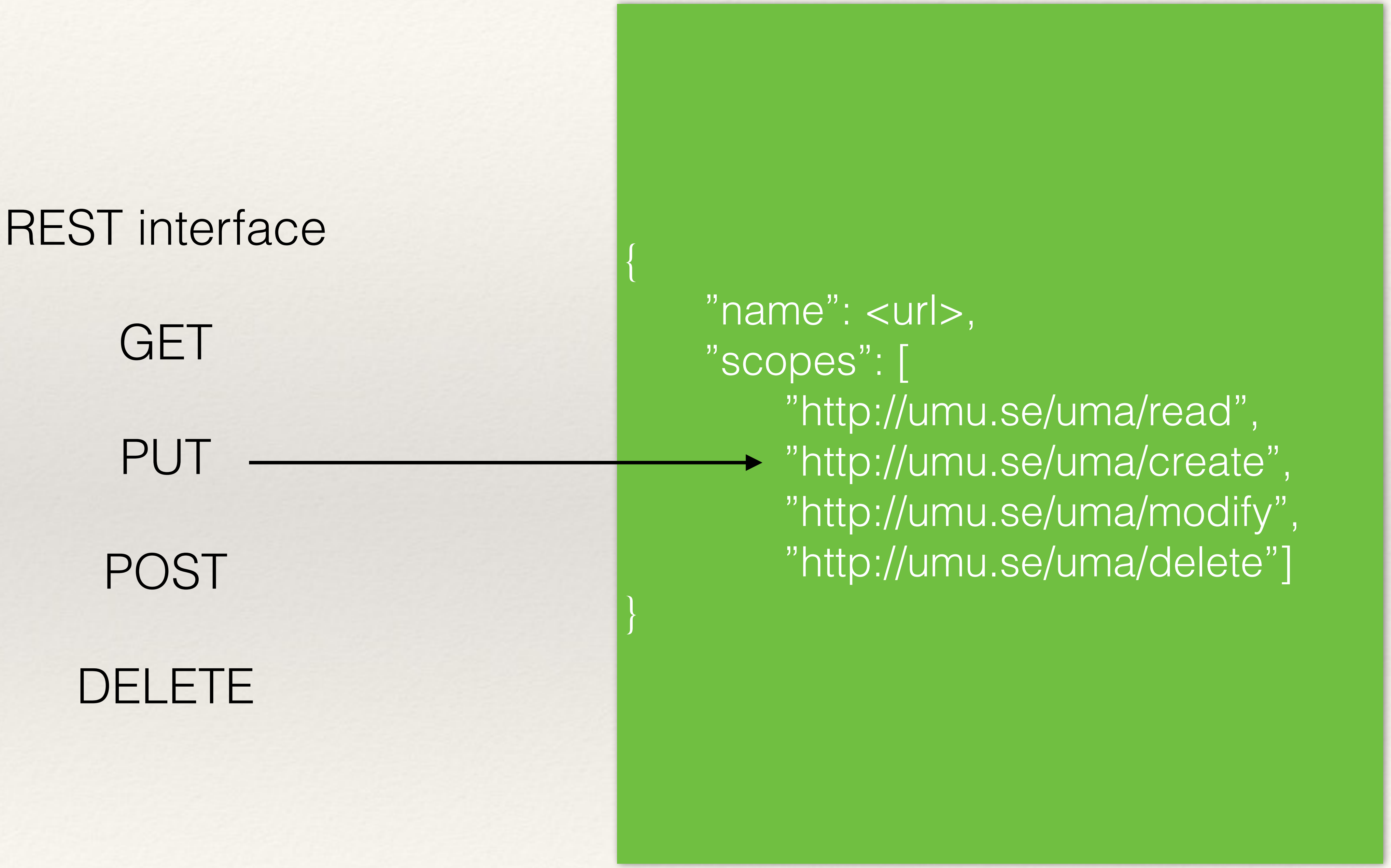
PUT

POST

DELETE

```
{
        ”name”: <url>,
        ”scopes”: [
              ”http://umu.se/uma/read”,
              ”http://umu.se/uma/create”,
              ”http://umu.se/uma/modify”,
              ”http://umu.se/uma/delete”]
}
```

Standardized format for describing resource sets

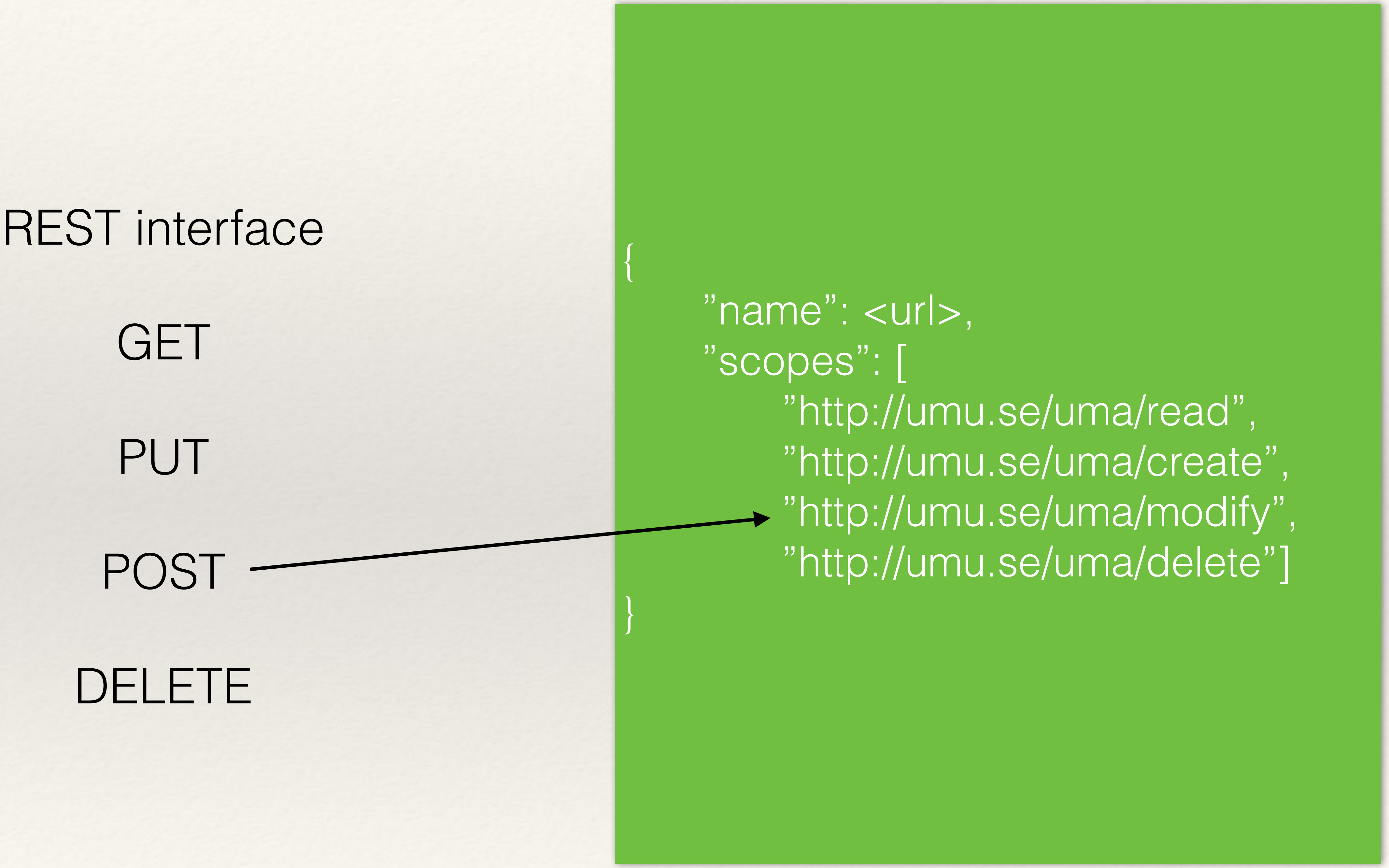


REST interface

- GET
- PUT
- POST
- DELETE

```
{
      ”name”: <url>,
      ”scopes”: [
            ”http://umu.se/uma/read”,
            ”http://umu.se/uma/create”,
            ”http://umu.se/uma/modify”,
            ”http://umu.se/uma/delete”]
}
```

Standardized format for describing resource sets



REST interface

GET

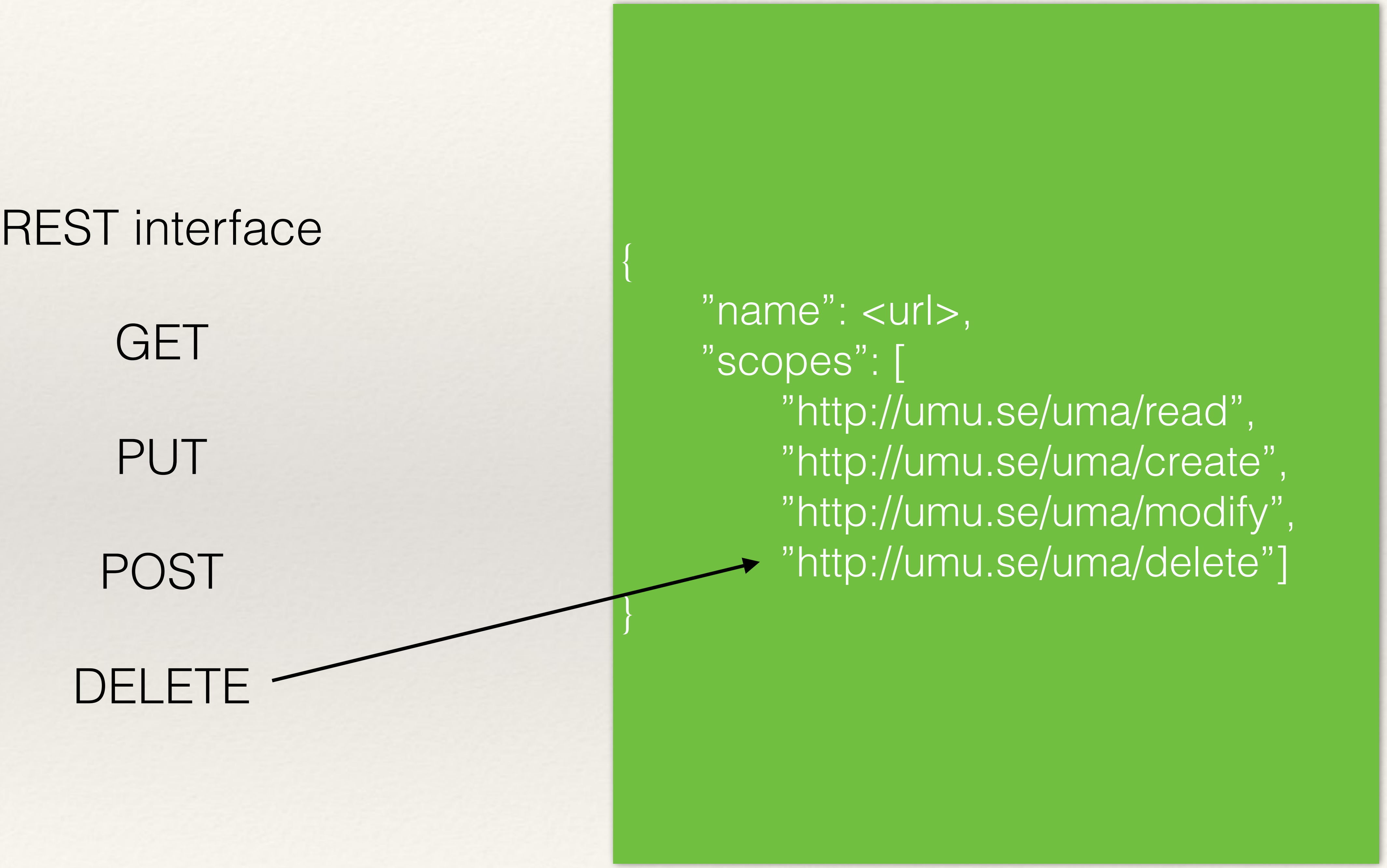
PUT

POST

DELETE

```
{
      ”name”: <url>,
      ”scopes”: [
            ”http://umu.se/uma/read”,
            ”http://umu.se/uma/create”,
            ”http://umu.se/uma/modify”,
            ”http://umu.se/uma/delete”]
}
```

Standardized format for describing resource sets



REST interface

GET

PUT

POST

DELETE

```
{
      ”name”: <url>,
      ”scopes”: [
            ”http://umu.se/uma/read”,
            ”http://umu.se/uma/create”,
            ”http://umu.se/uma/modify”,
            ”http://umu.se/uma/delete”]
}
```

Registration response

- JSON object with the following parameters:
  - _id
    REQUIRED The authorization server-defined identifier for the web resource corresponding to the resource set.
  - user_access_policy_uri
    OPTIONAL. A URI that allows the resource server to redirect an end-user resource owner to a specific user interface within the authorization server where the resource owner can immediately set or modify access policies subsequent to the resource set registration action just completed

Resource set registration API

- Create resource set description: POST {rsreguri}/resource_set
- Read resource set description: GET {rsreguri}/resource_set/{_id}
- Update resource set description: PUT {rsreguri}/resource_set/{_id}
- Delete resource set description: DELETE {rsreguri}/resource_set/{_id}
- List resource set descriptions: GET {rsreguri}/resource_set

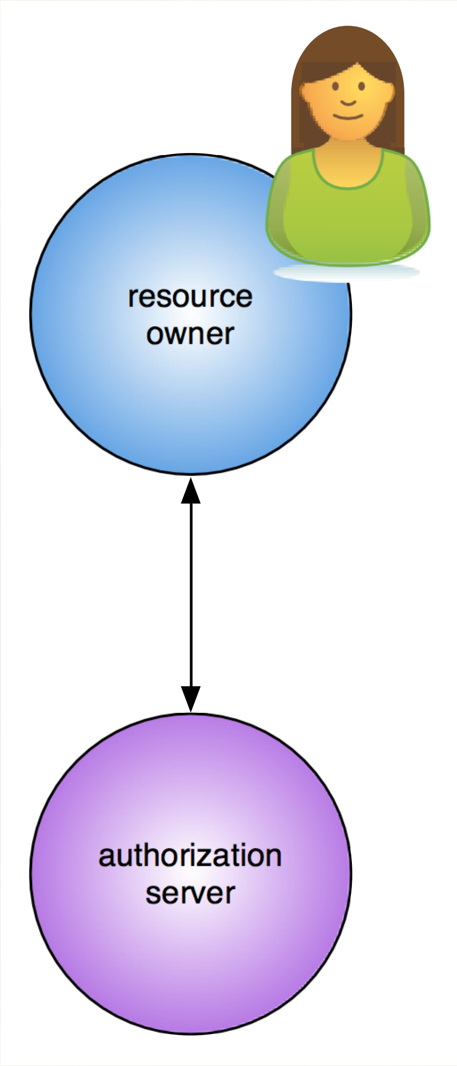
Create resource set example

```
POST /rs/resource_set HTTP/1.1
Content-Type: application/json
Authorization: Bearer MHg3OUZEQkZBMjcx
...

{
  "name" : "Tweedl Social Service",
  "icon_uri" : "http://www.example.com/icons/sharesocial.png",
  "scopes" : [
    "read-public",
    "post-updates",
    "read-private",
    "http://www.example.com/scopes/all"
  ],
  "type" : "http://www.example.com/rsets/socialstream/140-compatible"
}
```

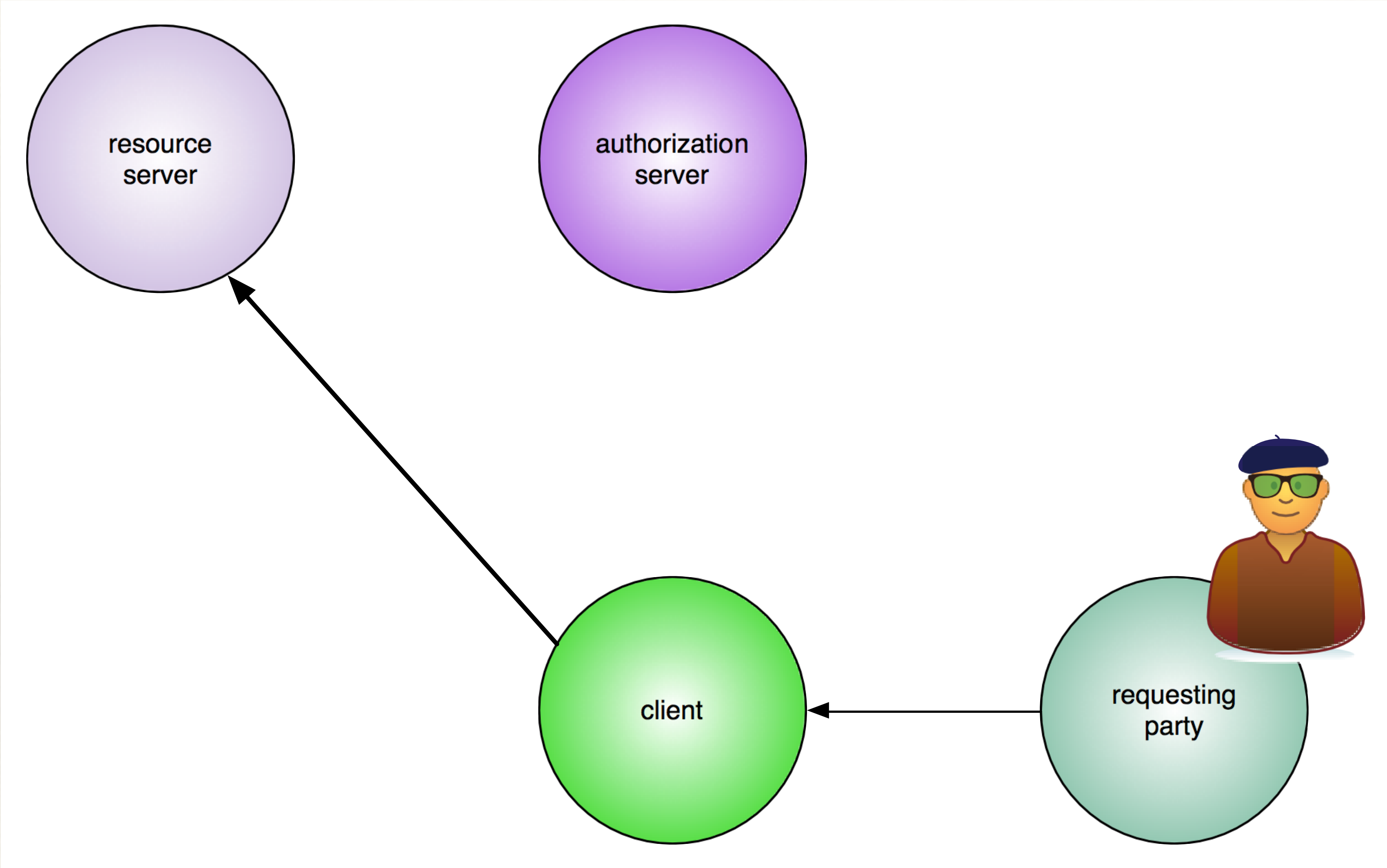
Registration response

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /rs/resource_set/KX3A-39WE
...

{
  "_id" : "KX3A-39WE",
  "user_access_policy_uri" : "http://as.example.com/rs/222/resource/KX3A-39WE/policy"
}
```

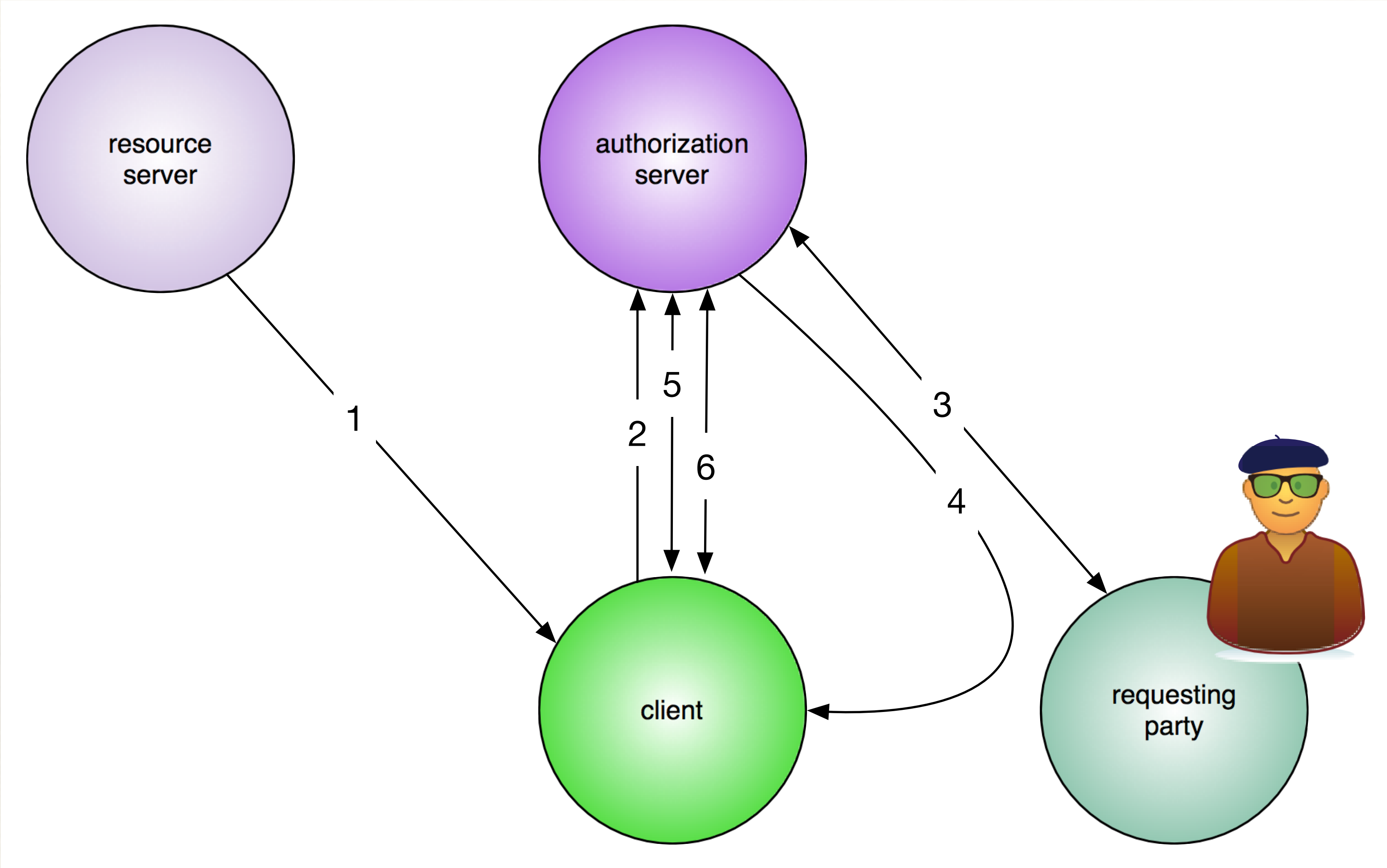
3. Setting permissions



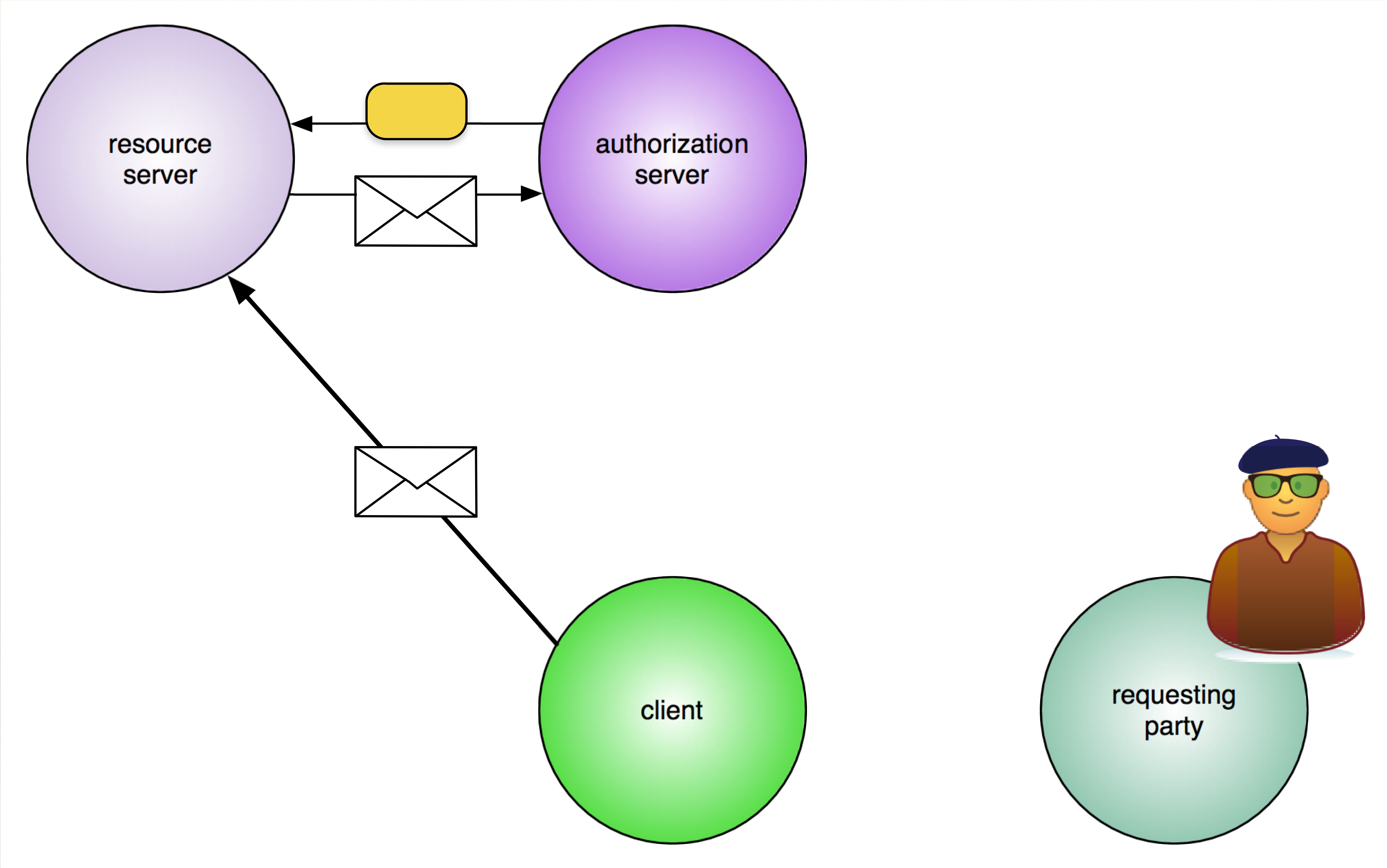
4a. Accessing resource



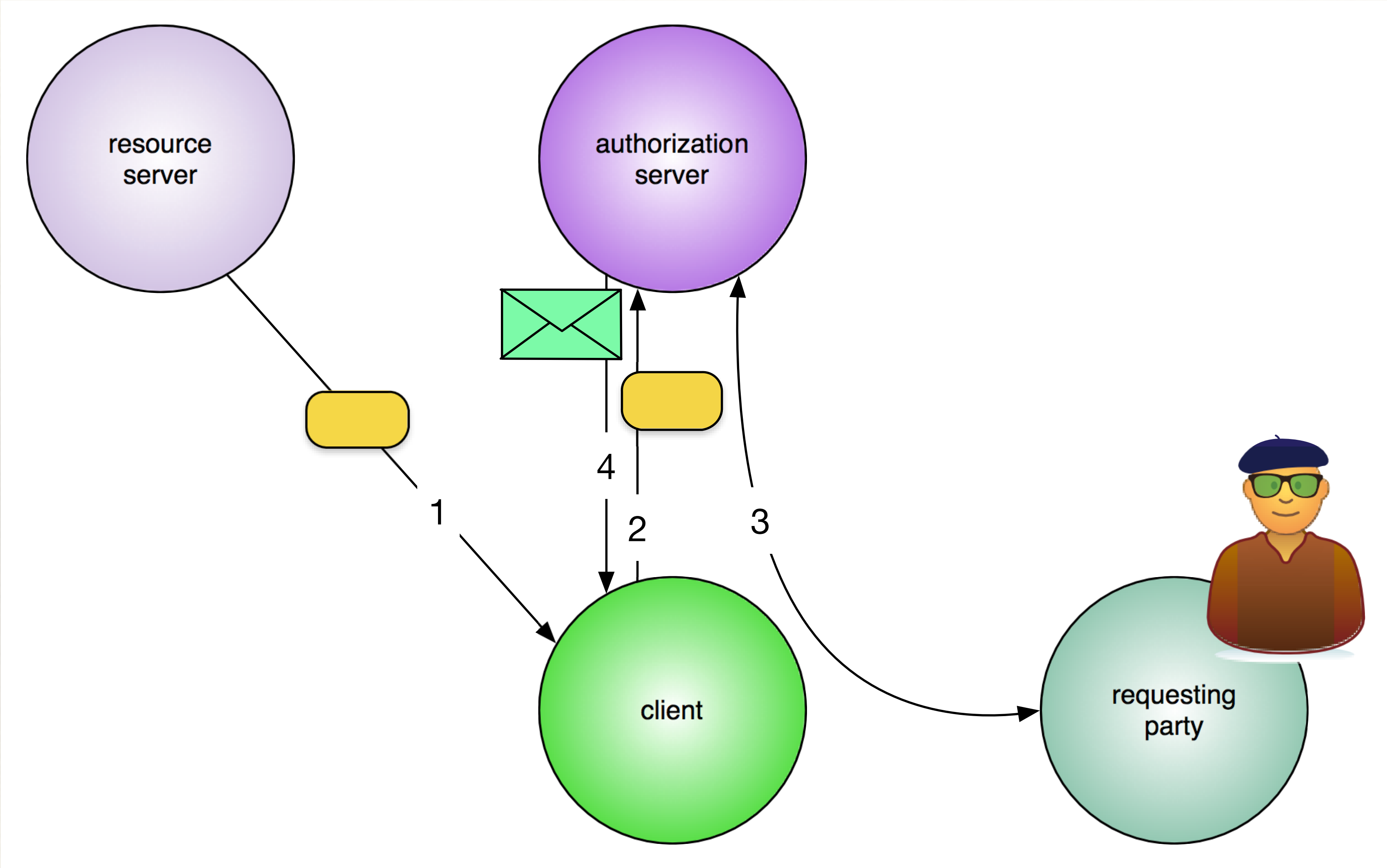
4b. Getting AAT&RPT



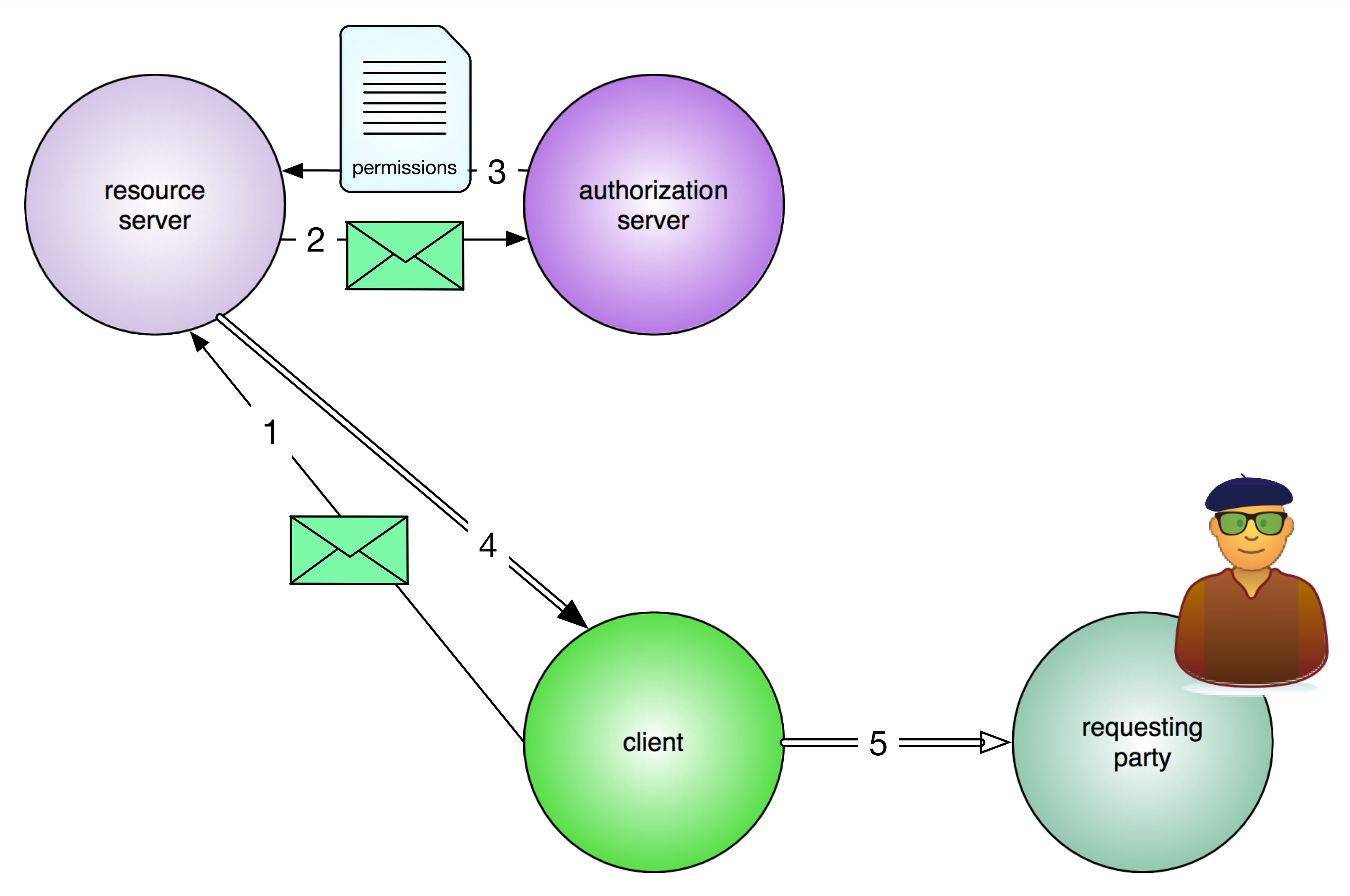
4c. Presenting empty RPT

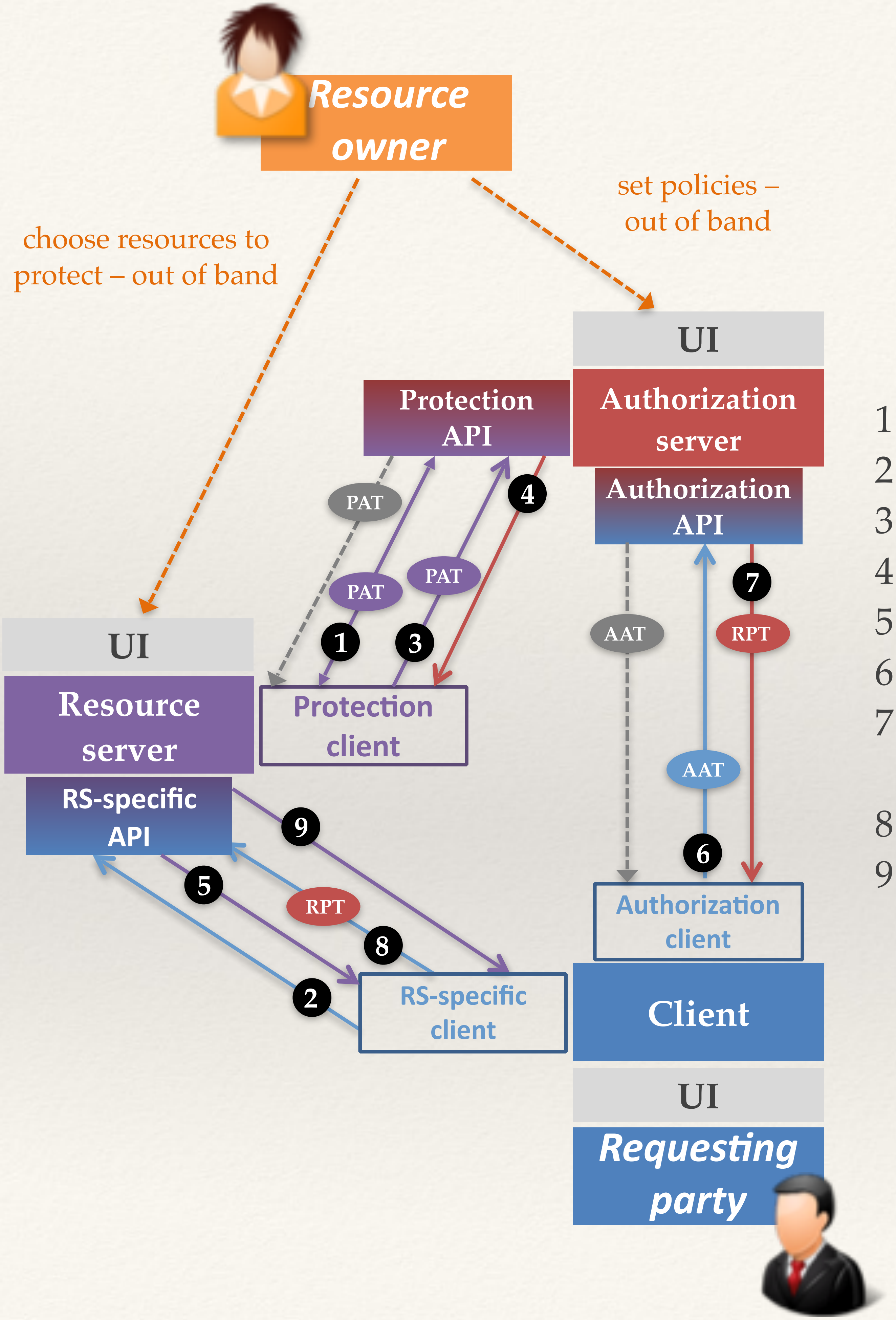


4d. Assigning permission



4e. With permission

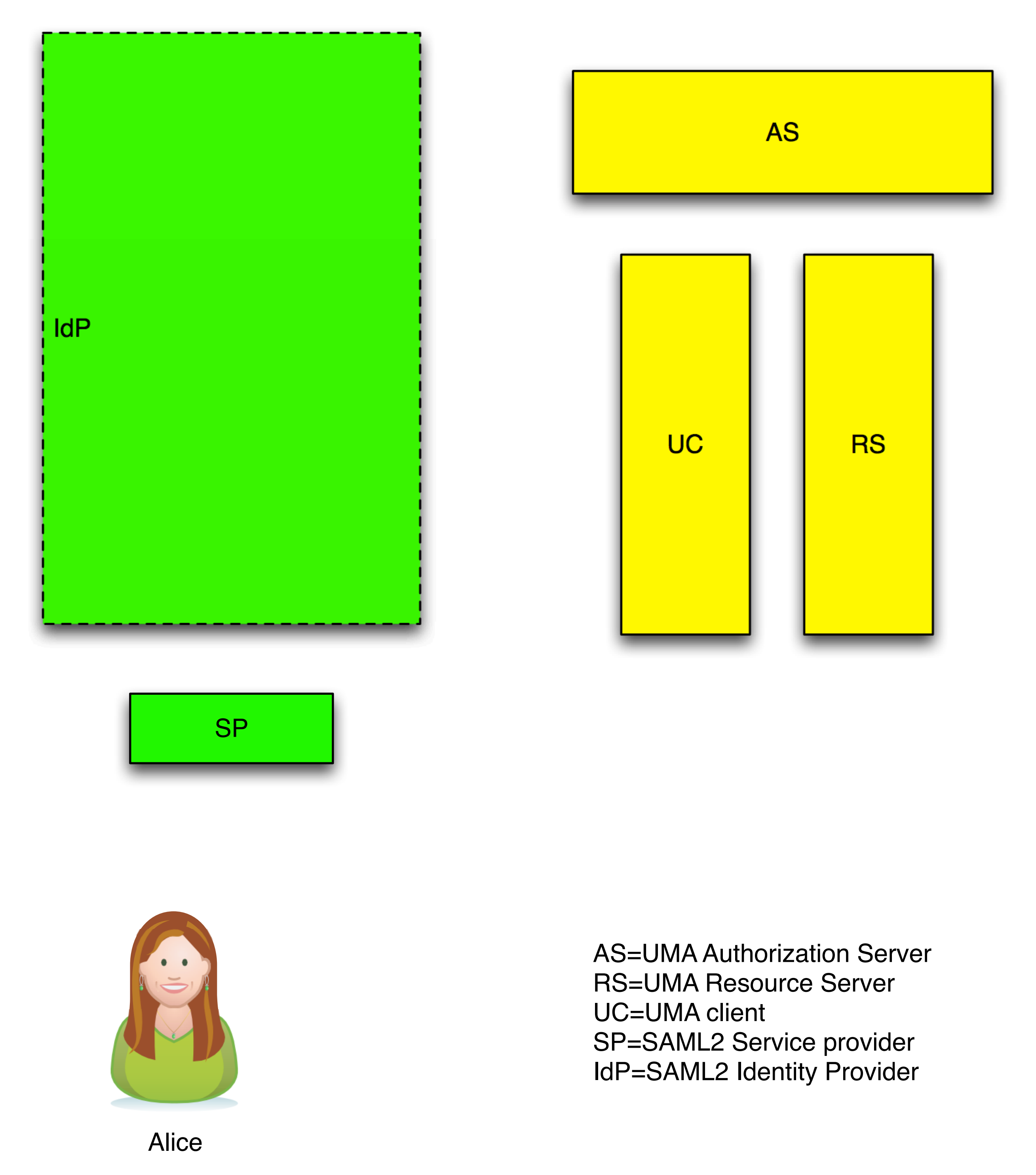




1. RS registers resource sets and scopes (ongoing)
2. C requests resource
3. RS registers permission
4. AS returns permission ticket
5. RS error with ticket
6. C requests authz data and RPT with ticket
7. AS gives RPT and authz data (after optional claim flows)
8. C requests resource with RPT
9. RS returns resource representation

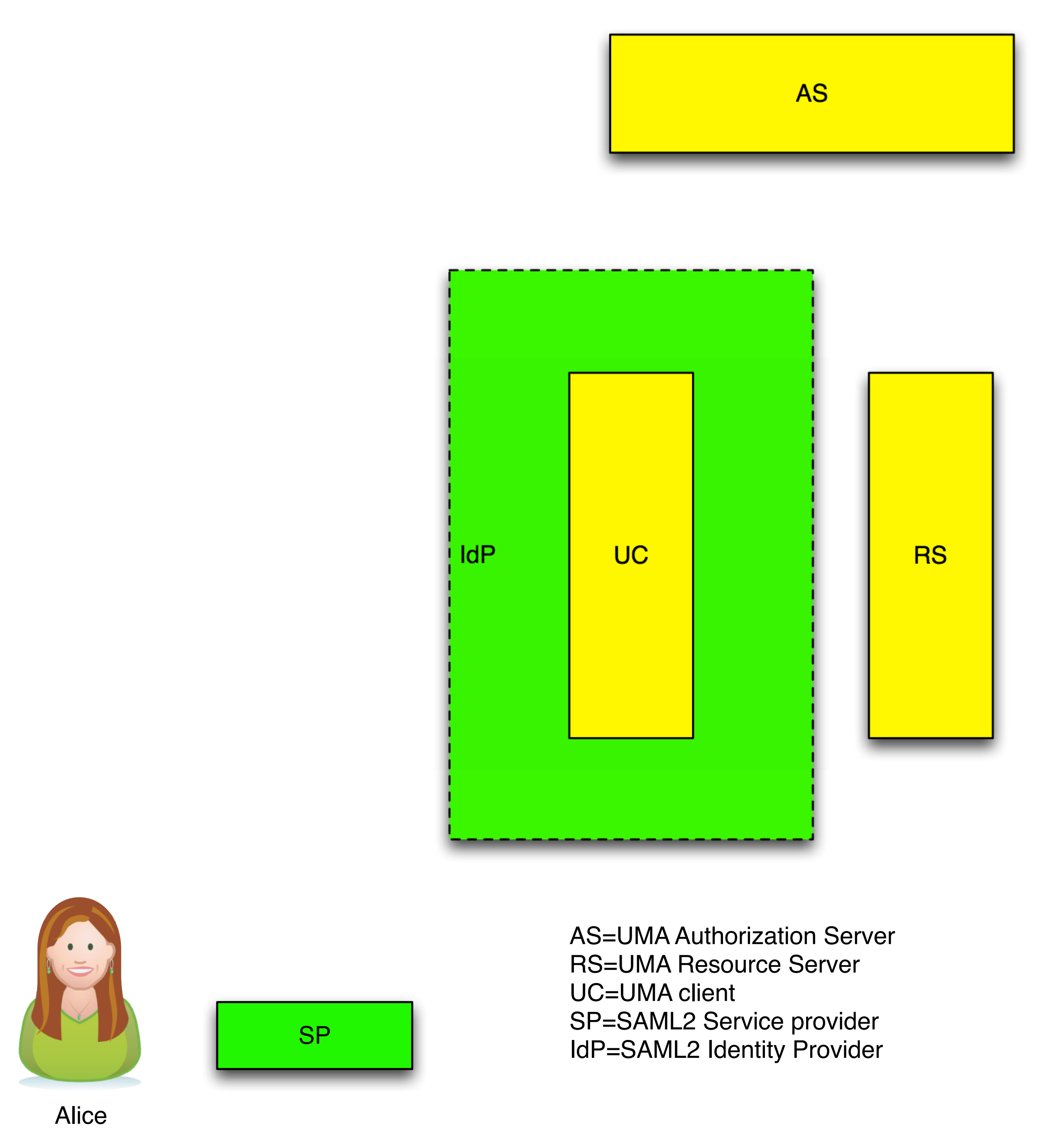
SAML2 + UMA

The pieces



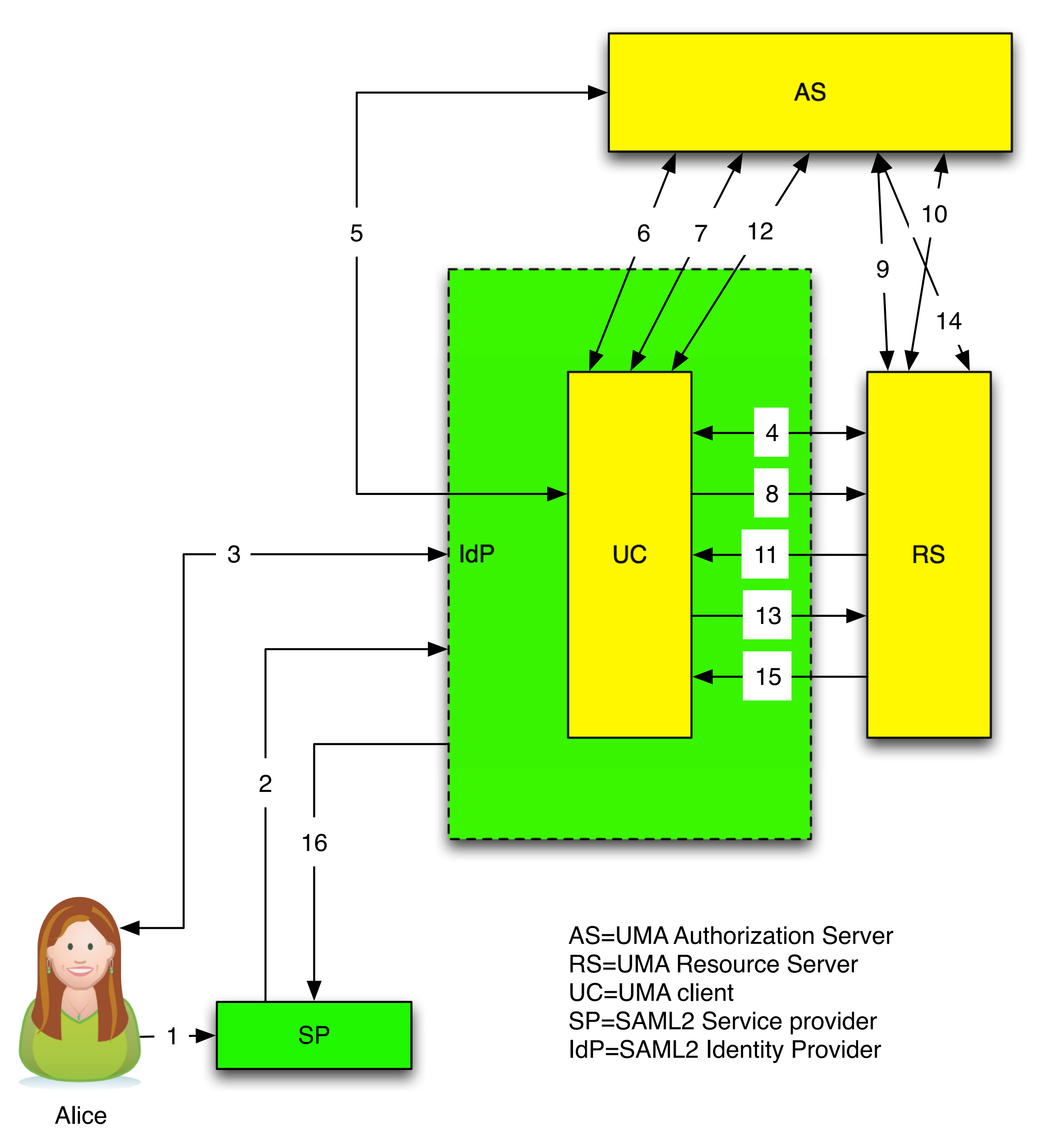
AS=UMA Authorization Server
RS=UMA Resource Server
UC=UMA client
SP=SAML2 Service provider
IdP=SAML2 Identity Provider

The configuration



AS=UMA Authorization Server
RS=UMA Resource Server
UC=UMA client
SP=SAML2 Service provider
IdP=SAML2 Identity Provider

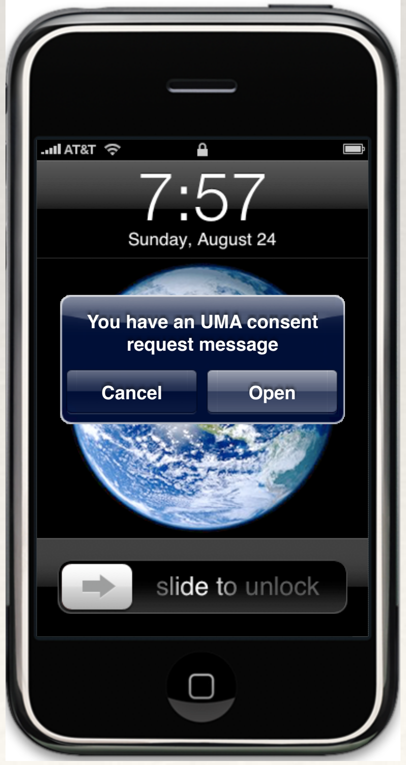
UMA+SAML2



AS=UMA Authorization Server
RS=UMA Resource Server
UC=UMA client
SP=SAML2 Service provider
IdP=SAML2 Identity Provider

Privacy by design

- Proactive not reactive
- Benefits all actors in an online service ecosystem
- Proper testable behavior
- User centric



You have an UMA consent request message

Status

- Version 1.0.1 of the standard published
- Have not started interop yet
- A handful of implementations, growing

Links to documents

- UMA Core V1.0.1 candidate Draft Recommendation
- OAuth RSR V1.0.1 candidate Draft Recommendation