OAuth Meta

In a nut shell

A framework for returning the meta-data associated with the response from each endpoints:
- Resource Endpoint
- Authorization Endpoint
- Token Endpoint

Uses RFC5988 as the framework.

Defines 4 rels (ruri, turi, auri, duri) but extensible.
- Just define a new rel in RFC5988 IANA registry

Dynamic
- Sometimes, these values are not going to be determined until after the authorization grant is given by the user. E.g., medical record location may only be given after the user consent as the location itself may be a sensitive information.
Token EP Response Example

HTTP/1.1 200 OK
Link: <https://example.com/userinfo>; rel="ruri", <https://example.com/disco>; rel="duri"
Content-Type: application/JSON; charset=utf-8

{
    "access_token":"aCeSsToKen"
}
Use case 1: The client starts at the resource

- The Client finds the resource location somehow but does not know the authorization server for the user.

  1. The user accesses the client to receive some service.
  2. The Client accesses the resource but gets 401 Unauthorized error.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example"
```

3. The Client needs to find where it can get the authorization. OAuth Meta can help.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example"
Link: <https://example.com/authz/>; rel="auri",
     <https://example.net/tenants/cooltools/az/>; rel="auri",
     <https://example.com/payment-upon-trial-expiry>; rel="payments"
```
Use case 2: The client starts at the resource type

The Client does not know the concrete resource location, but knows the resource type it needs.

1. The user access the client to receive some service.
2. The client asks the user to click on the NASCAR to select the Authz server.
3. The client sends an Authz request with scope=x-ray to the Authz EP.
4. Authz EP authenticates the user and asks the user to give permission to the access to his x-ray stored at a hospital.
5. The user authorizes it, and the Authz response with code is returned to the client through the browser redirect.
6. The client redeems access token from the token endpoint using the code, and gets the concrete resource endpoint location etc. with it using OAuth Meta.

HTTP/1.1 200 OK
Link: <https://example.com/users/nat/x-rays>; rel="ruri"
    <https://specs.example.net/rt_types#tb>; rel="rt_type"
Content-Type: application/JSON; charset=utf-8

```json
{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"sender_cnf_tb",
    "expires_in":3600,
    "refresh_token":"tGzv3J0kF0XG5Qx2TlKWIA",
}
```
Discussion Points

- Do we want a pattern match on the links? (e.g., on the ruri)
  - URI template?

- Do we want JSON representation instead?
  - Earlier version of the draft was using a JSON representation.
    - Pro:
      a. It can be parsed with the response easily,
      b. The metadata can be stored as part of the JSON response,
      c. Can be much more expressive that it can actually express how to use the response just like Swagger.
    - Con:
      - It needs to be programmed in the endpoint; the static link header can be done by the http server configuration instead easily.

- Authz Response through query parameter is not integrity protected and thus cannot be trusted.
  - Do we want to drop it? Do we want to integrity protect the authz response? Do we want to ignore the issue?