GNAP Meeting
IETF 110

draft-ietf-gnap-core-protocol-04

March 9, 2021

Justin Richer • Aaron Parecki • Fabien Imbault

Agenda

- Core draft update: changes since IETF109 (from -02 to -04)
  - Editorial Changes
  - Functional Changes
- Core draft roadmap: overview of next big topics
  - Subject identifier
  - On-device use cases and component definitions
  - Message signing mechanisms
  - Key rotation
- Topic: Subject identifiers
- Topic: On-device use cases and component definitions (SSI and GNAP)

Differences since IETF109 (-02 to -04)

https://www.ietf.org/rfcdiff

?url2=draft-ietf-gnap-core-protocol-04

&url1=draft-ietf-gnap-core-protocol-02

25 Merged Pull Requests

https://github.com/ietf-wg-gnap/gnap-core-protocol/pulls

?q=is%3Apr+is%3Aclosed+merged%3A2020-11-18..2021-02-22

Functional Changes

- Dropped redirect to a short URL (#139, #121, #53)
- Dropped OpenID Connect “claims” parameter (#140)
- Made access tokens mandatory for continuation request (#129, #67)
- Changed access token request and response (#40, #39, #10, #13, #162)
- Refactor “key” information to new section (#152)
- Group interaction modes in the request (#122, #163)
- Dropped reading grant and token (#98, #99)

Access Token: New Request Syntax

Interaction Modes: New Request Syntax

```
"interact": {
    "start": ["redirect", "user_code"],
    "finish": {
        "method": "redirect",
        "uri": "https://client.example.net/return/123455",
        "nonce": "LKLTI25DK82FX4T4QFZC"
    },
    "hints": {
      "ui_locales": ["en-US"]
    }
}
```

Start Modes

Finish Mode

UI Hints

Editorial Changes

- Removed closed issues from draft text (#150, #172)
- Updated subject identifier info (#153, #177)
- Minor typo fixes (#126, #179, #181)
- Updated acknowledgements (#157)
- Updated terminology (#29, #155)
- Also done:
  - Fixes to the preview/render/tag system in GitHub
  - Lots of post -04 typo/format fixes are pending (thank you!)

Discussion Items

Core Draft Roadmap

- Subject identifier
- On-device use cases and component definitions (SSI and GNAP)
- Message signing mechanisms
- Key rotation

Message Signing

- HTTP Message Signature Draft is progressing in HTTP WG
- New ideas on using JOSE
- DPoP profile in FAPI
- Open questions:
  - Is anything MTI?
  - What are the failure states when negotiating signature methods?

Key Rotation

- For client instances
  - Client management API?
- For ongoing grants
  - Grant update API?
  - Does this also rotate key for client/token?
- For access tokens
  - Part of token management API?
  - Could client instance use different keys for different tokens?

Subject Identifiers

Open issues related to subject identifiers

178 sub_ids and different contexts -> subject_types/subject_types_supported
171 subject identifiers as portable identifiers -> aliases could provide the same functionality
75 scope of subject identifiers -> local/AS scope
43 user information request items -> assertion related to same user only
42 use of identifiers as communication channels -> avoid email
41 assertion format names -> suggest an array of assertions
51 User reference as an assertion -> opaque ref
49 AS Validation of RC-presented assertions -> should or must?
50 user identification items -> current structure seems fine?
16 bad use of subject identifier -> avoid email
197 requesting User Information -> clarify when RO != end-user
198 terminology -> use of subject seems fine?

~ 10% of all open issues

PR pending merge - subject_types

#184 `subject_types (array of strings)` in the request
`subject_types_supported (array of strings)` in the discovery

```
"subject": { // request.subject is what is requested to the AS

        "subject_types": ["iss_sub", "email"],      // draft-ietf-secevent-subject-identifiers-06

        "assertions": ["id_token"]                  // refers to OIDC

}
```

Note that we also have a subject_type (singular) in the request.user and in response.subject
request.user = what the client tells the AS about the current user (later in the slides, I propose request.subject.hints.self)
request.subject = what the client asks the AS about the current user
response.subject = what the AS tells the client about the current user

Fix open issues : email examples

- Change examples - current based on email attribute #16 #42
  - email is still a valid SECEVENT attribute, but don’t overly encourage its use
  - what to use in the GNAP examples? *(I propose as_ref, see later)*

```
"user": {

   "sub_ids": [ {

     "subject_type": "email",

     "email": "user@example.com"

   } ]

}
```

1. No assurance it's unique (ex: admin@example.com)
2. No idea on email policy (could be transfered to someone else)
3. Ties to a delivery method (which shouldn't be our goal here)

Note: current SECEVENT types are account / email / phone / issuer / alias.

Fix open issues: scope and portability

- scope of subject identifiers #75
  - subject information: statement asserted locally by an AS about a subject (current terminology)
  - a global unique identifier doesn’t seem enforceable anyway (+ privacy considerations)

- subject identifiers as portable identifiers #171
  - alias or assertion? (depends on what we support as types)
  - portability might be beyond scope (more transactional), but GNAP can help support various types of identity systems (interoperability)

Fix open issues: opaque reference

- user reference as an assertion #51
  - label to be confirmed: “as a reference” (generic) or “AS reference” (gnap specific)
  - this is unique locally to the AS, random looking string (e.g. nanoid), private by design

```
"user": { // option 1: extension SECEVENTS
   "sub_ids": [ {
     "subject_type": "as_ref",
     "as_ref": "XUT2MFM1XBIKJKSDU8QM"
   } ]
}
```

assuming we can define additional types in the registry. This should be prefered if other SECEVENT types are considered useful anyways (easier to bootstrap with existing systems)

```
"user": { // option 2: alternative
   "as_ref": "XUT2MFM1XBIKJKSDU8QM"
}
```

do we need to explicit as/ref? (similar to iss_sub, decomposed in iss/sub)

Fix open issues: assertions

- issue #43: assertions
  - Array

- issue #41 : assertions_supported?
  - id_token (OIDC)
  - Add
    - jwkthumb?
    - DID?
    - VerifiableCredential?
  - What's in the core? extension?
    - Remove samlv2 from core (XML security) - could be an extension

```
// example response assertions, the rest is not provided
"subject": {
   "assertions": [ {
     "assertion_label": "assertion1",
     "assertion_type": "id_token",
     "id_token": {
        "iss": "https://oidc.example.com",
        "sub": "24400320",
        "aud": "s6BhdRkqt3",
        "nonce": "n-0S6_WzA2Mj",
        "exp": 1311281970,
        "iat": 1311280970,
        "auth_time": 1311280969,
     },
     {
     "assertion_label": "assertion2",
     "assertion_type": "verifiableCredential",
     "verifiableCredential":
        {
            @context: [ ],
            id: "http://example.cred.com/3732",
            type: [
                "VerifiableCredential",
                "WorkingCredential"
            ],
            issuer: { id: "did:example:cityhospital"},
            issuanceDate: "2020-03-10T04:24:12.164Z",
            credentialSubject: {
                id: "did:example:ebfebec21",
                job: {type: "Doctor", specialty: "Surgeon"}
            },
            proof: { }
        }
     },
    }]
}
```

sub_ids vs assertions

- DID / jwkthumb
  - subject_type or assertion_type? (analogy with id_token -> assertion seems logical)

```
// example DID
"subject": {
   "assertions": [ {
     "assertion type": "did",
     "did": "did:example:123456"
   } ]
}
```

```
// example JWK thumbprint
"subject": {
   "assertions": [ {
     "assertion type": "jwkthumb",
     "jwkthumb": {
        "e":"AQAB",
        "kty":"RSA",
        "n":"0vx7agcQ...LqDw"
        }
   } ]
  }
```

Is there a security issue in having a mapping between sub_ids and assertions? *Risk of making an association between a weak identifier (e.g. email, used elsewhere) and stronger identity assertions. An association between a local opaque as_ref and a VerifiableCredential would be more generic and safer (still depends on the issuer of the VC, to be checked by the AS).*

Fix open issues: RO vs end-user

- Requesting User Information #197
  - Titles = “2.4. Identifying the User” / “3.4. Returning User Information”
  - Is user the RO or the end-user?
  - The underlying assumption of the current text: RO = end-user = User
    - “If the identified end-user does not match the RO present at the AS during an interaction step, the AS SHOULD reject the request with an error.” (section 2.4)

- To solve this issue, we need an in-depth discussion about RO vs end-user
  - Careful: this has far reaching implications, I will present my personal thoughts

end-user vs RO: what exists today

- OAuth2 assumptions
  - OAuth2 considers RO = end-user
  - UMA2 considers RO != end-user (ex: doctor asks access to patient’s medical record)
  - This creates a fragmented ecosystem

- GNAP aims to solve this issue too
  - Sequence diagram 1.4.3 : asynchronous consent request to remote RO
  - Sequence diagram 1.4.4 : software-only authorization (i.e. automated rule engine)
  - But section 4 only covers sequences 1.4.1 and 1.4.2 (UI interactions)
    - Btw, we don’t have a sequence related to interaction start = app

end-user vs RO: why it’s not trivial

- When/how do we know if RO = end-user or not?
  - Hardcoded assumption in OAuth2 and UMA2
  - Decision is the AS’s responsability
  - A complex application might require the user’s authentication to decide at run-time
  - How much do we expose that information to the client?
    - RO could be “anyone who knows a specific secret password”
    - The AS shouldn’t send any sensitive information about the RO
- More hints to help the AS? (to be reviewed by WG)
  - Map to sequences 1.4.x
  - Declarative or asserted
    - request.interact.start is already a hint, but for a specific sequence
    - request.user is a hint, that can be easily validated if there’s an assertion proof
    - the client could provide more hints as to its intent

end-user vs RO: map to sequences

- Intuition:
  - the end-user generally knows whether it's "his data" or "someone else's"
  - the client often implements that very logic explicitly (ahead-of-time)
    - when aggregating my banking accounts, I access to my own bank services
    - an information system for doctors will ask for consent to patients

- Proposal: an optional principal (hint for the AS to determine the RO)
  - Principal: first in order of importance for granting privileges
    - If subject only is defined, we're in sequences 1.4.1 or 1.4.2
    - If principal is also defined, we're in sequences 1.4.3 or 1.4.4

A discussion on principal is already provided in SECEVENT 3.1 -> a complement to subject types
Note that we say first in order (Cambridge Dictionary), since RO decides.
If we further support delegatable tokens (cf issues #169 and #189), even the user agent could grant its own subset of privileges (from the token issued by the AS), but that's only second in order delegation (and it impacts the trust model).

end-user vs RO: request with hints (optional)

```
"subject": {  // request.subject is what we ask to the AS

   "subject_types": [ "as_ref" ], // for instance, I ask for an opaque ref

   "assertions": [ ],

   "hints": {      // these are optional

       "self": {}, // request.user would be transfered here?

       "principal": {}

   }

}
```

clarify it's not a role (user is ambiguous)
self = "the set of someone's characteristics, that make that person different from other people" (adapted from Cambridge Dictionary)

The AS `should/must` validate the hints before using them (issue #49)

end-user vs RO: principal for sequence 1.4.3

```
// example 1: sequence 1.4.3 (principal + async)

// proposal: parental control app, a child will ask approval of his dad



"principal": {

    "async": {

        "RO": {

            "subject": { "assertions": [ // DID ] } // dad

        }

    }

}
```

DIDComm can then be used by the AS to get consent from the remote RO
How do we know it's a legitimate DID?
The AS needs to avoid spam, phishing, etc.
(this issue exists as soon as one needs to reach out to a remote RO, irrespective of the syntax/delivery method proposed here) - probably needs some sort of management API for the RO

end-user vs RO: principal for sequence 1.4.4

```
// example 2: sequence 1.4.4 (principal + automated)

// proposal: automated rule engine

// use case: child wants to watch a film, but could be too young (will be checked by the engine).

"self": { // this corresponds to the child

   “sub_ids”: { },

   "assertions": [ // VC on DOB or ZKP on age ]

}

"principal": {

    "automated": true  // here the right would be denied by the engine if too young

}
```

end-user vs RO: composability

- async could fail
  - Remote RO is unavailable / Remote RO doesn’t respond quickly enough
- automated rule engine could be limiting
  - What if the child still wants to request special permission?

```
"principal": {

    "automated": true, // first check the automated rule (only if set explicitly to true)

    "async": {         // but if needed, special permission can be asked to dad

            "fallback": "none",  // to be defined (ex: retry) but no infinite loop (ex: max_retry)

    }

  }
```

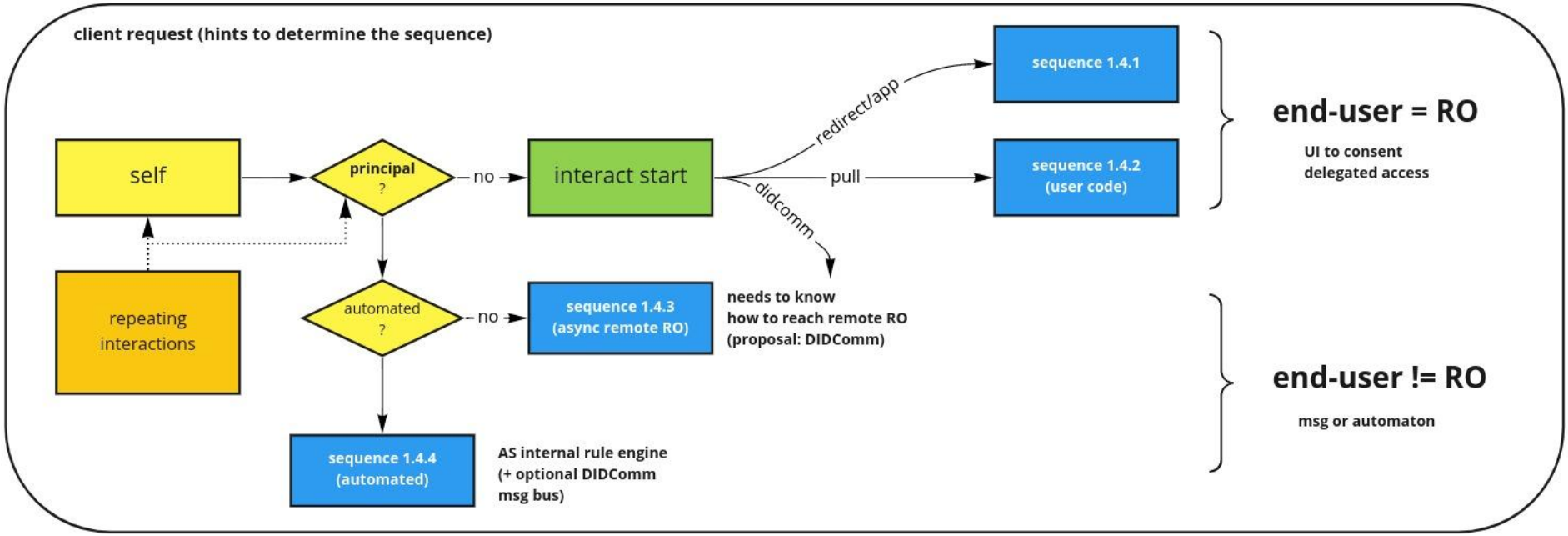
end-user vs RO: multiple ROs?

RO could be an array (ROs)

```
"principal": {
    "async": {
        "consent_query": "quickest",   // manage multi-parties (quickest/all/quorum). Ignored if only one.
        "ROs": [
            { "subject": { "assertions": [ // DID ] } }, // dad
            { "subject": { "assertions": [ // DID ] } }  // mum is maybe nicer? If she answers first, that's cool
        ]
    }
}
```

Need to keep it simple: if there is a complex consent decision (and/or), how does that work when multiple parties are involved (all/quorum)?
Need to be careful: the consent_query is only a request, the AS will decide.

end-user vs RO: map to sequences summary



end-user vs RO: DIDComm (delivery method)

- assumption: AS has its own DID
  - Needs to be managed (key rotation, etc.)
- DIDComm for sequence 1.4.3 (remote RO)
  - Async pattern to reach remote owners
  - Enables integration with SSI (incl. support beyond HTTP)
  - Client has to know the RO's DID, discovery of DIDs considered out of scope
- DIDComm for sequence 1.4.4 (rule engine)
  - Not mandatory but could be useful as a message bus (decouple policy from the actual interaction - cf Interact Server idea)

Back on terminology #197 #198: subject seems fine, because it defines who is involved (self + principal). It may well be that it involves a machine from another individual or company.

end-user vs RO: DIDComm interact

- Still need to decide how the interact would work (message format / query)
  - didcomm/didcomm_query: was discussed in XYZ, see issue #168

```
"interact": {
        "start": ["didcomm"],
        “finish”: { } // way to notify the client back
        "hints": {
            "timeout": "10min” // for async requests (need to define the format)
        }
}
```

Benefits/Downsides

- Role identification (RO / end-user)
  - The proposed approach doesn't remove the need for runtime evaluation by the AS
  - It provides a dynamic configuration covering all sequences (draft-04 currently assumes they're the same, as in OAuth2)
    - Syntax remains light and composable, and allows to mix sync / async
    - Clarifies when there's a web UI interaction and when not
    - Clarifies the who (subject) / how (interact) / what (client)

- Security and privacy considerations
  - Hints vs Assertions (the AS should/must validate). What happens if the client sends misleading hints? (issue exists already in draft-04, but additional hints further emphasize this)
  - DIDComm is still early, will need further developments

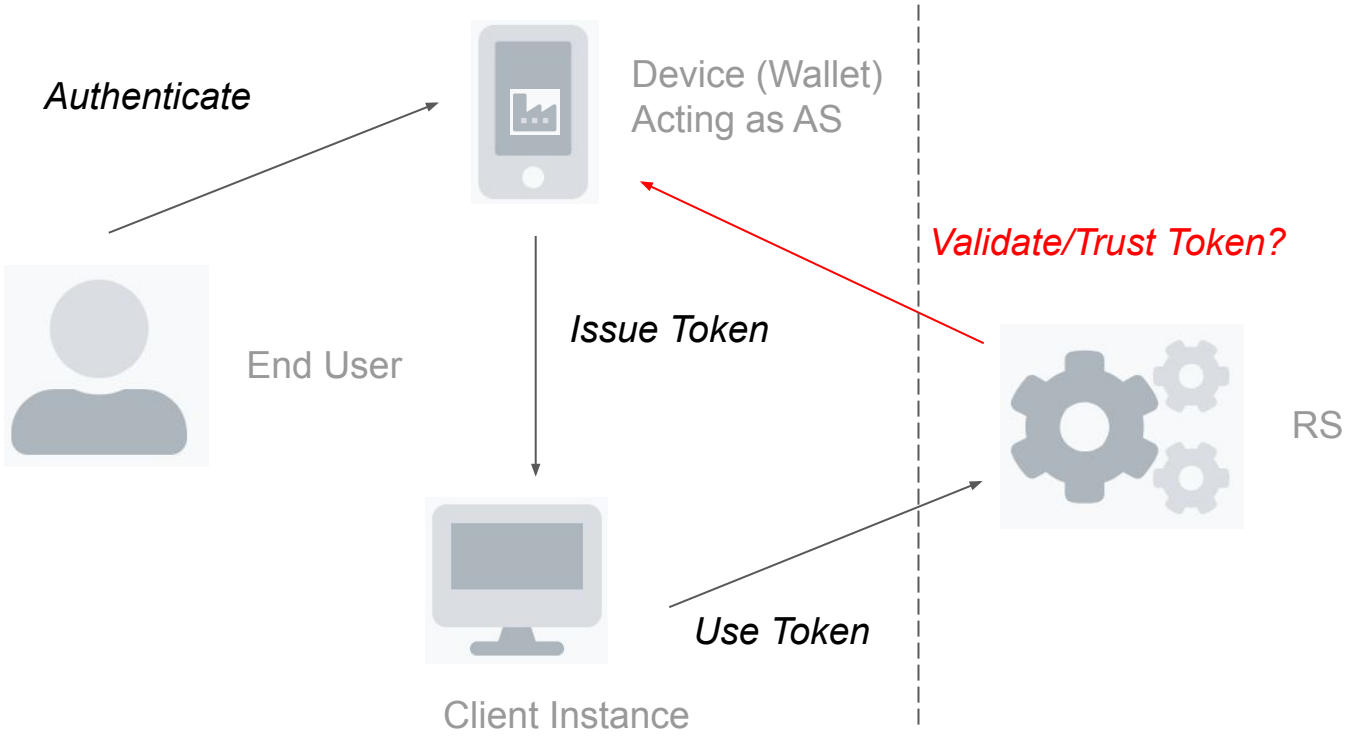
Subject identifiers: further work

- PRs : feedback welcome, now and during the PRs.

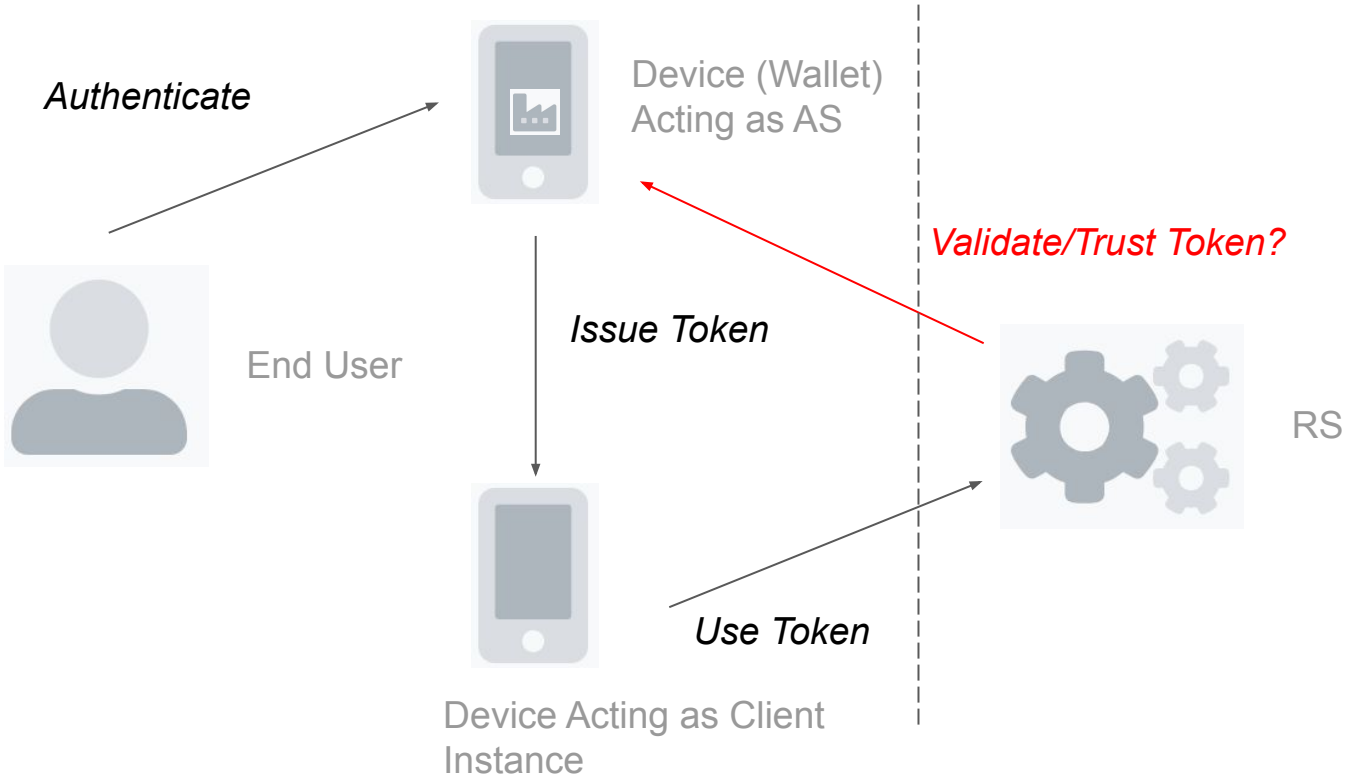
```
// summary of my personal preference (suggestion only, not as an editor)
"subject": {
   "as ref": {                        // response only (wouldn’t require SECEVENT)
      “as”: “https://ex1.as.com”,
      “ref”: “XUT2MFM1XBIKJKSDU8QM”
   },
   "assertions": [ ],                 // request and response (id_token/jwkthumb/DID + VC)

   "hints": {                         // request only (optional)
      "self": {                       // replaces request.user (support SECEVENT here)
         “sub ids”: { },              // can we extend SECEVENT with as_ref?
         “assertions”: [ ]
      },
      "principal": {                  // new proposal presented today
         “automated”: true,           // rule engine
         “async”: { }                 // remote ROs
      }
   }
}
```

SSI and GNAP:
System Architectures

“Bring your own AS”



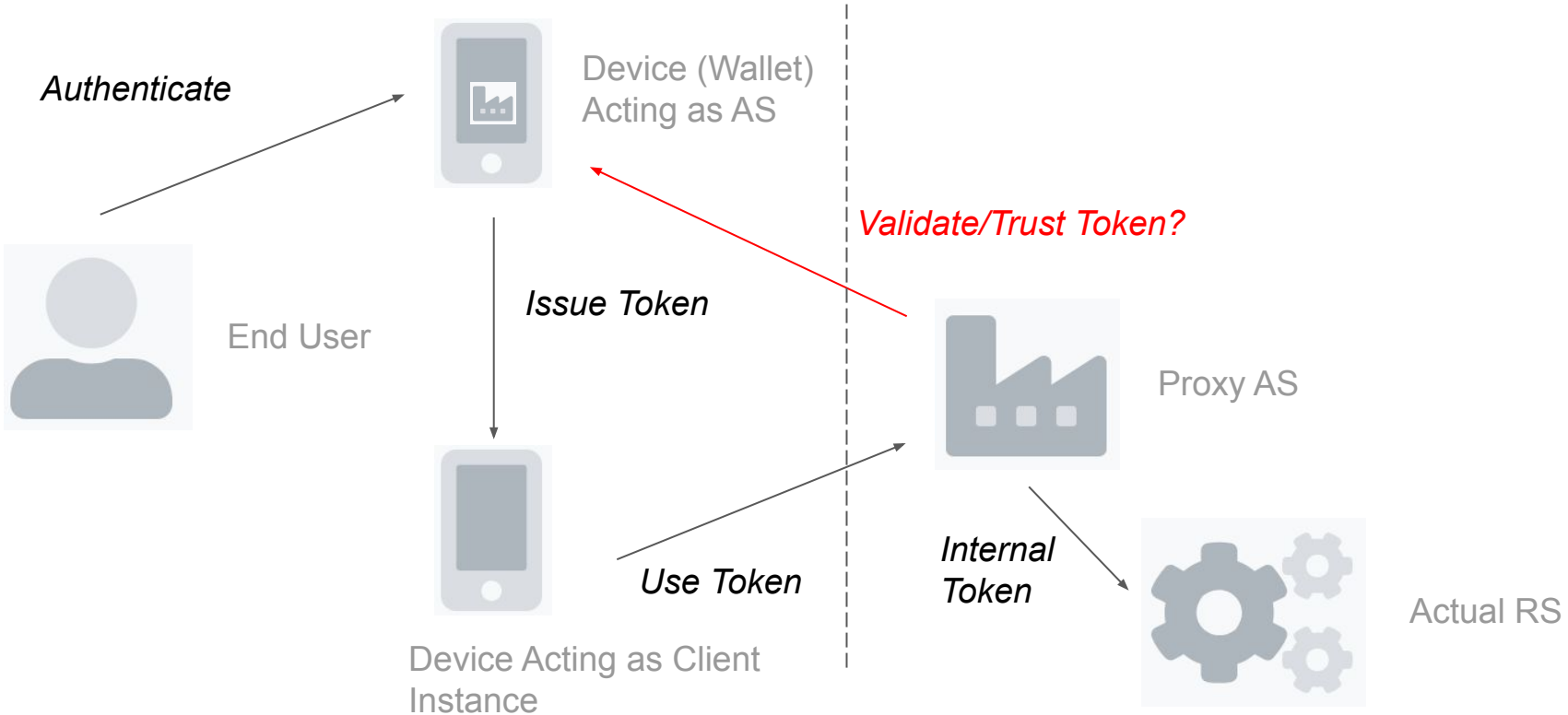
“Bring your own AS”



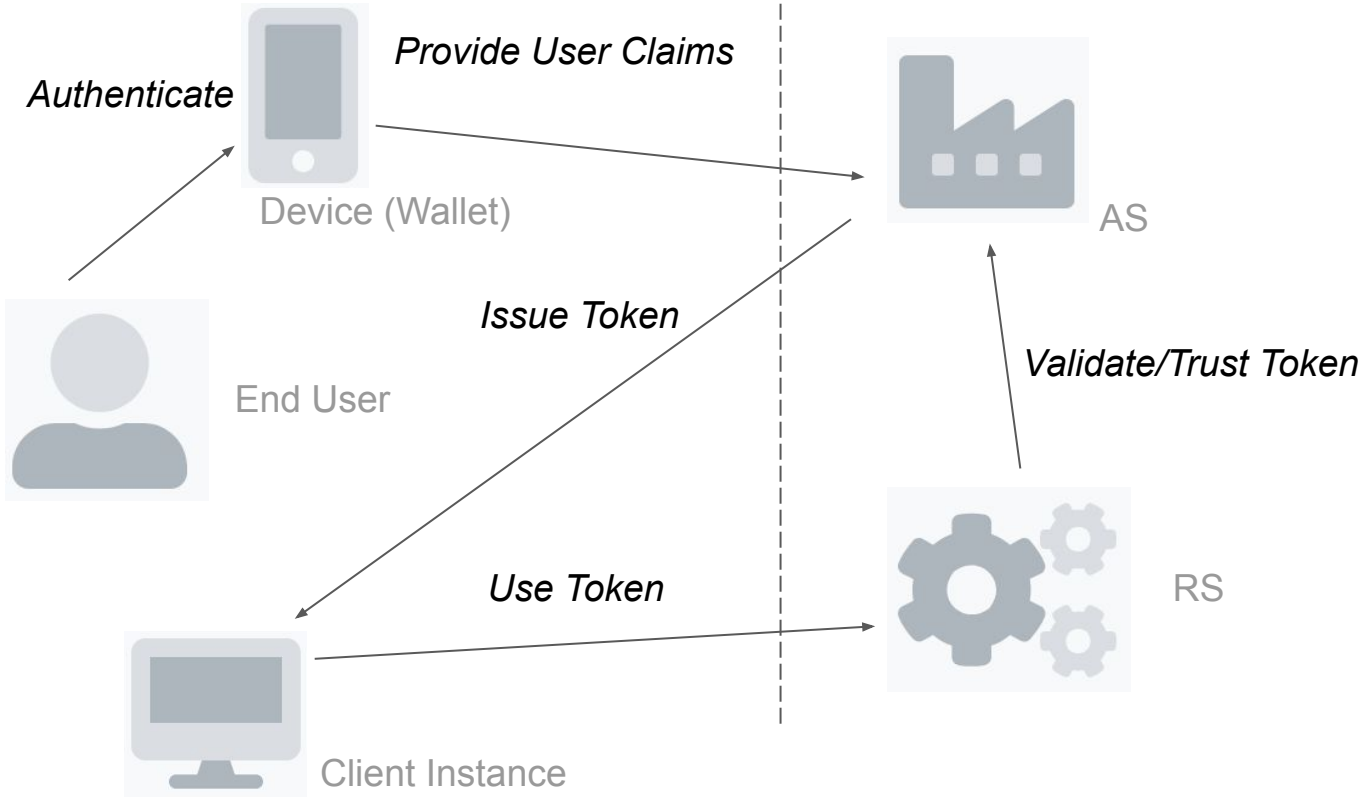
Downsides of this approach

- Difficult to convince an RS to accept arbitrary token
- RS needs to be able to dereference all required information
  - Pushes towards non-opaque token formats
- Difficult for off-device client instances to start flow
  - Logging in from a website
  - Requires non-HTTP communication to start

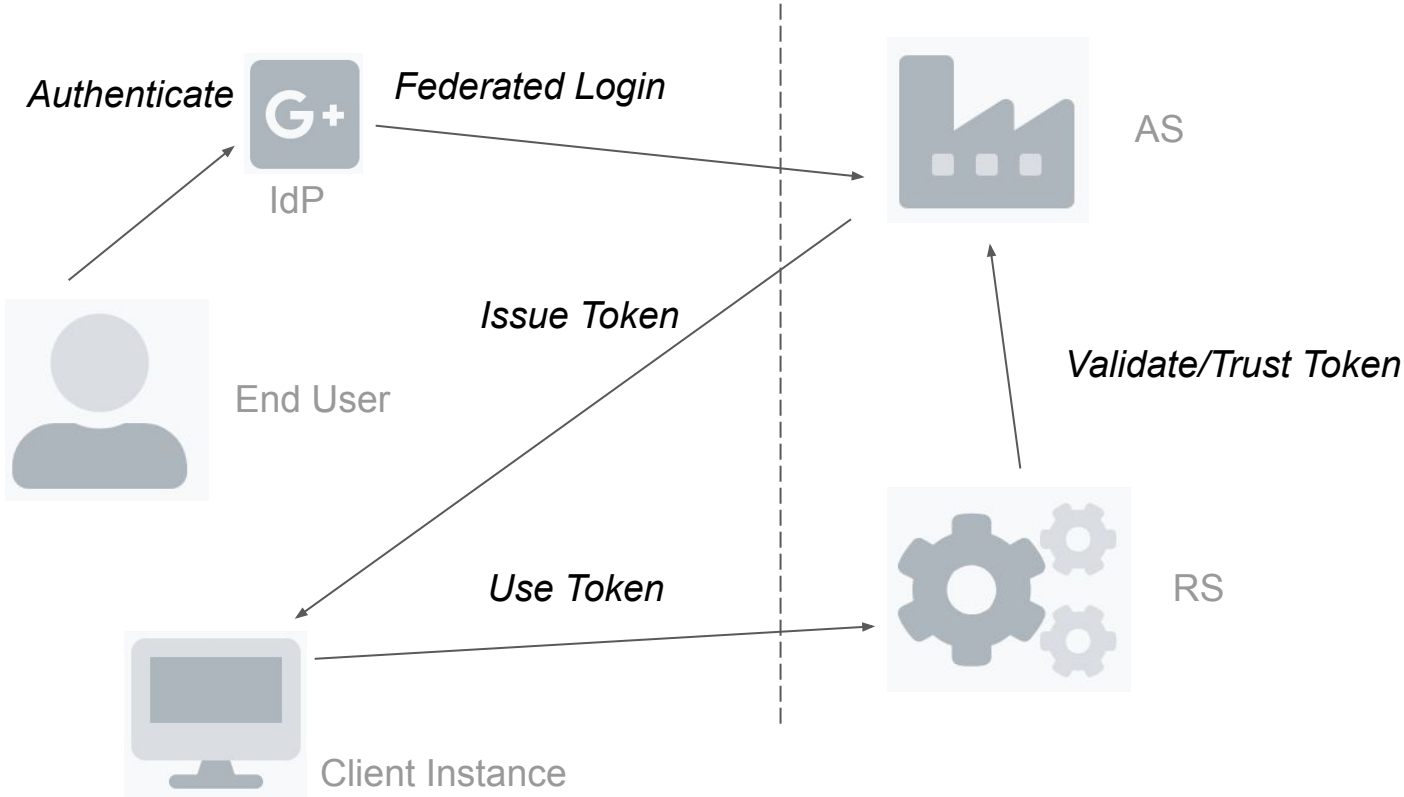
What happens when you “Bring your own AS”



“AS as Token Factory”



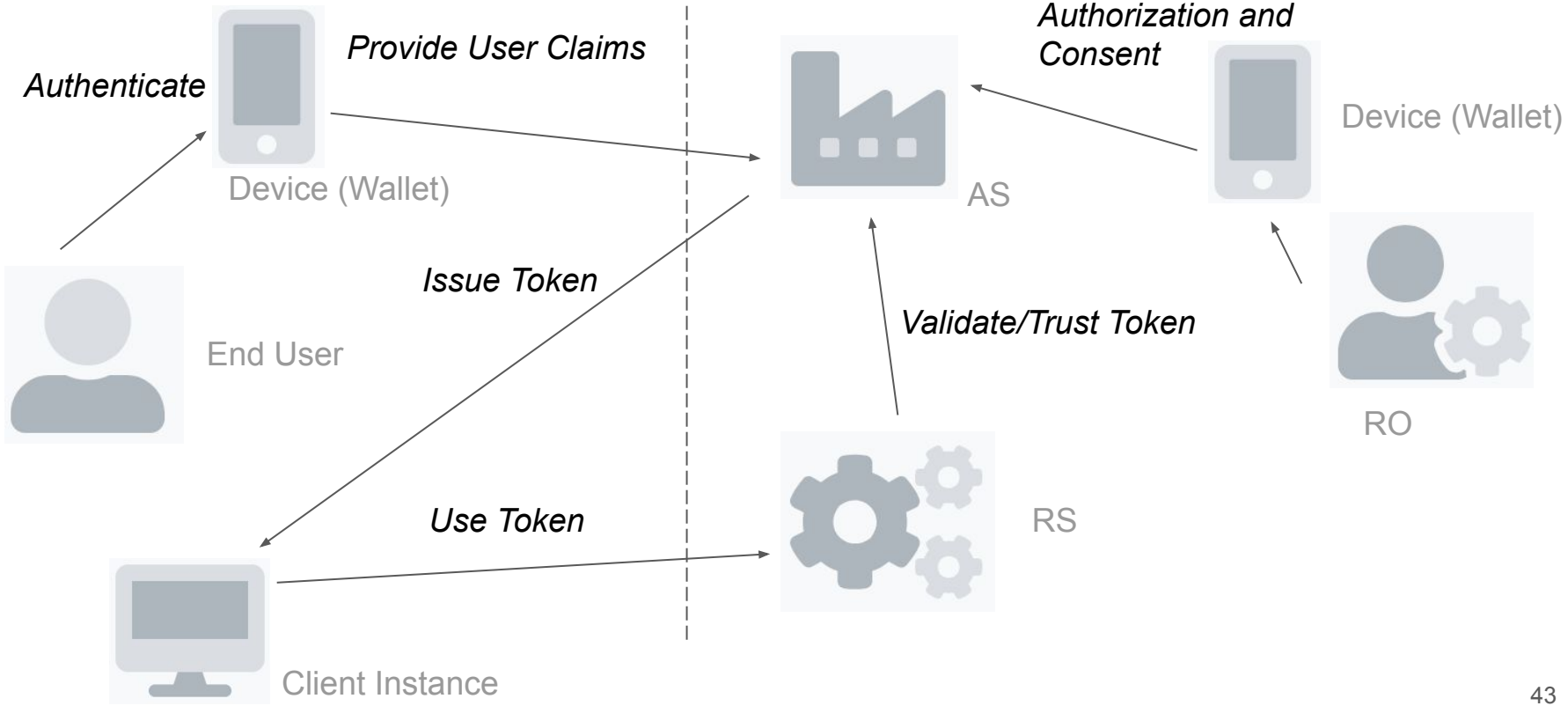
Precedent: Social Login at AS



Benefits of this approach

- RS/AS trust boundary is well-understood
- GNAP doesn’t have to assume user logs in to AS
  - User might not even interact with AS during request
- Extension points for interaction and claim formats
- AS figures out who needs to approve based on what’s being asked for

Remote RO


Authenticate
Provide User Claims
Device (Wallet)
Authorization and Consent
Device (Wallet)
AS
End User
Issue Token
Validate/Trust Token
RO
Use Token
RS
Client Instance