Rich Authorization Requests


IETF-106, 21.11.2019, Singapore
Brian Campbell, Justin Richer, Torsten Lodderstedt
How does the Payment API know that the user authorized the payment of that amount to this account?
Payment Authorization

1. Merchant

2. amount: 124,34€
to: DE02100100109307118603
reference: purchase 123456

3. www.example.com/auth

4. amount: 124,34€
from: DE40100100103307118608
to: DE02100100109307118603
reference: purchase 123456

5. payment

Pls. transfer 124,34€ to DE02100100109307118603
Pls. transfer 124,34€ to DE02100100109307118603
Pls. transfer 123,45€ from my account
DE40100100103307118608 to Merchant Co. (account: DE02100100109307118603) Reference: purchase 123456
Use Cases with similar characteristics

● Access to Account Information
● Creation of Electronic Signatures
● Access to Health Data
● Access to Tax Data
● Strong Identity Attestation
Commonalities

- Privileges very narrowly defined (and must also be enforced)
- Authorization data fine grained & structured (voluminous)
- Sometimes transaction authorization (one time & transaction specific values)
- Integrity and authenticity of authorization request data needed
- Authorization data may contain PII - confidentiality might be important
Challenges

- Expressiveness of scopes is not sufficient for the scenarios just explained
  - No structure, no dynamic values - made for simple static access requests
  - Ambiguous (“openid email read”)
- Allocation of requested permissions to resource server specific access tokens is hard (despite resource indicators)
Rich Authorization Requests

- **draft-lodderstedt-oauth-rar** specifies new parameter "authorization_details"
- "authorization_details" contains, in JSON notation, an array of objects
- Each JSON object contains the data to specify the authorization requirements for a certain type of resource.
- The type of resource or access requirement is determined by the "type" field.
- Note: same structure is used in OAuth.xyz

```json
[
  {
    "type": "payment_initiation",
    "locations": [
      "https://example.com/payments"
    ],
    "actions": ["initiate", "status", "cancel"],
    "instructedAmount": {
      "currency": "EUR",
      "amount": "123.50"
    },
    "creditorName": "Merchant123",
    "creditorAccount": {
      "iban": "DE02100100109307118603"
    },
    "remittanceInformationUnstructured": "purchase 123456"
  }
]
```
Combination

- Authorization requirements for a multiple resources can be combined
- “locations” field allows assignment to particular resource (server)
- “resource” parameter used to select authorization details for RS-specific access tokens

```json
[
  {
    "type":"payment_initiation",
    "locations": ["https://example.com/payments"],
    "actions": ["initiate","status","cancel"],
    "instructedAmount": {
      "currency":"EUR",
      "amount":"123.50"
    },
    "creditorName":"Merchant123",
    "creditorAccount": {
      "iban": "DE02100100109307118603"
    },
    "remittanceInformationUnstructured": "purchase 123456"
  },
  {
    "type": "account_information",
    "locations": ["https://example.com/accounts"],
    "actions": ["list_accounts","read_balances","read_transactions"]
  }
]```
authorization_details can be used ...

- where “scope” can be used
- in combination with or instead of “scope”
- Example: pushed authorization request

```text
POST /as/par HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnlxS3REUmJuZ
response_type=code
&client_id=s6BhdRkqt3
&state=af0lfjslkdj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
&code_challenge_method=S256
&code_challenge=K2-ltc83acc4h0c9w6ESC_rEMTJ3bww-uCHaoeK1t8U
&authorization_details=%5B%7B%22type%22%3A%22account%5Finformation%22%2C%22actions%22%3A%5B%22list%5Faccounts%22%2C%22read%5Fbalances%22%2C%22read%5Ftransactions%22%5D%7D%5D
```
Advantages

- Flexible and type safe way to represent rich authorization data
- Allows definition of API-specific authorization data structures
  - no “one size fits all”
- Common data set elements to address common use cases
- Interoperable and easy way to issue RS-specific Access Tokens and Token Introspections Responses (Data Minimization and Disambiguation)
Status

- -03 revision (based on previous work at the FAPI WG)
- Positive feedback on the list, also from people new to our community

Thanks for working on this! It's an interesting and timely read for me as we're starting to bump up against some limitations of the `scope` parameter. I have some initial feedback below, and will hopefully be able to send through more as I dig in in more detail.

- implementations/prototypes exist (authlete, yes.com)

Would the WG consider to adopt this draft?