OAuth2 Scopes and Presentation Definition Mapping

Scope design

When designing a system that uses OAuth2, you have to decide how scopes map to resources that the client will attempt to access. "Resource access" is typically a specific REST-style HTTP operation on a specific URL, e.g. `POST /products/staplers/1`. Things to consider when designing scopes are discussed in this section.

Broad v.s. narrow scopes

Broad scopes are generally high level e.g., a scope that gives access to a certain use case or larger group of resources. Narrow scopes are often low-level e.g., a scope that gives read access to a specific resource, limited set of resources or operations. Examples scopes for an employee that's authorized to buy supplies for their employer:

- Very broad: `buyer`
- Broad: `buyer office` (office supplies only)
- Broad: `buyer lt-1000` (orders less than 1000 euros)
- Narrower: `buyer office:staplers` (staplers only)
- Very narrow: `buyer office:staplers:red lt-10` (red staplers only, less than 10 euros)

How scopes are mapped to operations on resources influences:

- How often clients need to request a new access token, if the previous token does not give access to a required resource.
- When access to a specific resource is authorized: when the access token is issued, or when it's used.

Broad, high-level Scopes

High-level, broad scopes typically give access to an entire use case, service, or group of resources. Checks that are executed before an access token is issued are limited to the Verifiable Credentials the client can present.

- Identification and authentication (user/client identity)
- General user access to the functionality (e.g. is admin, can buy supplies, etc.)

A real-life example of a broad scope is the Nuts eOverdracht use case, which specifies the following scopes:

- `eOverdracht-sender` which gives access to the receiver's services required by a care organization that wants to transfer a patient to another organization.
- `eOverdracht-receiver` which gives access to the sender's services to the transfer receiver.

However, when a resource is accessed, the system needs to verify that the scope gives access to the specific resource operation.

This type of scope is supported by the Nuts node.

Narrow, low-level Scopes

Narrow, low-level scopes typically give access to specific operations on specific resources, e.g., reading a specific patient's medical summary.

This type of scope is **not** supported by the Nuts node, because:

- narrow scopes often contain resource identifiers, which requires wildcards/regexes in policy mapping (more on that below), which is currently not supported by the Nuts node.
- this leads to more access tokens, since each access token has a more limited use. If user authentication involves manual input (e.g., presenting a credential using a mobile wallet), user experience will deteriorate.

Another consideration is that using low-level scopes, moves most authorization decisions to the access token issuance. This is viable and supported by the Nuts node, but complicated: it requires the vendor to implement a REST API that understands Presentation Definitions.

Policies: Mapping Scope to Authentication Subject

Nuts differentiates data exchanges that contain PII (Personally Identifiable Information) and/or medical data (e.g., Social Security Number or EHR records) and non-PII (e.g., medical condition without correlatable information, or technical identifiers). These types require different levels of authentication:

- Data exchanges containing non-PII may be exchanged by a service, with only the legal organization being authenticated.
- In data exchanges that do contain PII, both legal organization and human user must be authenticated. This means PII information cannot be exchanged without a human being present.

The authentication subject, meaning whether an organization or user must be authenticated, depends on the requested scope. This mapping is to be specified by the use case, distributed as JSON document and loaded into participating Nuts nodes.

This mapping also determines which protocol is used by the requesting party;

- exchanges with a human user always authenticate using OpenID4VP
- exchanges without human user (only the organization is authenticated) can authenticate using the OAuth2 `vp_token` grant for backend services or OpenID4VP.

Mapping document

This section contains an example of a presentation definition mapping document as it could be specified by a use case. The Presentation Definition is described more in detail in AuthN using Verifiable Credentials.

```
{
 "zorgtoepassing": {
  "organization": {
   "format": {
    "ldp_vc": {
     "proof_type": [
      "JsonWebSignature2020"
     ]
    },
    "ldp_vp": {
     "proof_type": [
      "JsonWebSignature2020"
     ]
    },
    "jwt_vc": {
     "alg": [
      "ES256"
     ]
    },
    "jwt_vp": {
     "alg": [
```

```
      "ES256"
    ]
  }
},
"id": "pd_any_care_organization",
"name": "Care organization",
"purpose": "Finding a care organization for authorizing access to medical metadata",
"input_descriptors": [
  {
    "id": "id_nuts_care_organization_cred",
    "constraints": {
      "fields": [
        {
          "path": [
            "$.type"
          ],
          "filter": {
            "type": "string",
            "const": "NutsOrganizationCredential"
          }
        },
        {
          "id": "organization_name",
          "path": [
            "$.credentialSubject.organization.name",
            "$.credentialSubject[0].organization.name"
          ],
          "filter": {
            "type": "string"
          }
        },
        {
          "id": "organization_city",
          "path": [
            "$.credentialSubject.organization.city",
            "$.credentialSubject[0].organization.city"
          ],
          "filter": {
            "type": "string"
          }
        }
```

```
        ]
       }
     }
   ]
},
"user": {
  "format": {
    "ldp_vc": {
      "proof_type": [
        "JsonWebSignature2020"
      ]
    },
    "ldp_vp": {
      "proof_type": [
        "JsonWebSignature2020"
      ]
    },
    "jwt_vc": {
      "alg": [
        "ES256"
      ]
    },
    "jwt_vp": {
      "alg": [
        "ES256"
      ]
    }
  },
  "id": "pd_any_employee_credential",
  "name": "Employee",
  "purpose": "Finding an employee for authorizing access to medical metadata",
  "input_descriptors": [
    {
      "id": "id_employee_credential_cred",
      "constraints": {
        "fields": [
          {
            "path": [
              "$.type"
            ],
            "filter": {
```

```
            "type": "string",
            "const": "EmployeeCredential"
          }
        },
        {
          "id": "employee_identifier",
          "path": [
            "$.credentialSubject.identifier",
            "$.credentialSubject[0].identifier"
          ],
          "filter": {
            "type": "string"
          }
        },
        {
          "id": "employee_name",
          "path": [
            "$.credentialSubject.name",
            "$.credentialSubject[0].name"
          ],
          "filter": {
            "type": "string"
          }
        },
        {
          "id": "employee_role",
          "path": [
            "$.credentialSubject.roleName",
            "$.credentialSubject[0].roleName"
          ],
          "filter": {
            "type": "string"
          }
        }
      ]
    }
  }
}
```

Revision #8
Created 24 April 2024 07:08:03 by Rein Krul
Updated 14 May 2024 12:23:56 by Rein Krul