Service Discovery

A party often exchanges data through its API endpoints. If a client only has the party's name or identifier, it needs to find the API endpoints. Service Discovery via the Nuts node let clients find involved parties and additional information, like API endpoints.

Technical details can be found on [Read the Docs]

Service definition

Every use case MUST define a service definition. The service definition specifies which Verifiable Credentials are required, which issuers can be trusted and which registration parameters are required. Because the service definition defines the trust, this is extremely important to get right. A wrong definition may render all security measures ineffective!

Service identifier

Use cases must specify a service identifier. It's recommended to include the use case name in the identifier, to avoid clashes with other use cases. It's also recommended to add a version number or year of publication. Since the service identifiers are used in URLs, it's recommended to use lower-case letters, numbers and a limited set of special characters: `_-.:`

Examples:

- name2021
- long_namev1.0
- main:sub:v1

Discovery server

For each use case there can be only one discovery server. Each node can act as discovery server. Choose one to start with, it can be changed later on. The `endpoint` for a service definition is constructed as `https://<host>/discovery/<service_id>`.

DID methods

A service definition may limit the supported DID methods. This can be used to prevent incompatibilities when updating Nuts node versions that introduce new DID methods.

Max validity

A service definition contains a `presentation_max_validity` parameter. This parameter defines how long a registration is valid for. Clients will automatically refresh their registration if needed. This parameter can be used to tweak when registrations are removed automatically. Automatic removal is added to make sure abandoned registrations can no longer be found.

Presentation definition

The presentation definition is the most important part of the service definition. It defines which Verifiable Credentials are required by a client to present and, maybe even more important, who the issuer of those credentials may be. This defines the trust anchors for the use case.

A special credential is added for each registration in which the authorization server URL is added and where there's options for additional parameters like endpoints or feature flags.

A presentation definition uses the Presentation Exchange (or PEX for short) standard.

Required field

All constraints added in a presentation definition will become searchable by default. If a use case requires to find a participant based on a credential field then it's wise to add it as an required field.

```
"presentation_definition": {
  "id": "coffeecorner2024",
  "format": ...,
  "input_descriptors": [
    {
      "id": "NutsOrganizationCredential",
      "constraints": {
        "fields": [
          {
            "path": [
              "$.type"
            ],
            "filter": {
              "type": "string",
              "const": "NutsOrganizationCredential"
            }
          },
          {
            "id": "organization_name",
            "path": [
              "$.credentialSubject.organization.name",
              "$.credentialSubject[*].organization.name",
            ],
            "filter": {
              "type": "string"
            }
          },
          {
            "id": "organization_city",
            "path": [
              "$.credentialSubject.organization.city",
              "$.credentialSubject[*].organization.city",
            ],
            "filter": {
              "type": "string"
            }
          }
        ]
      }
    }
  ]
}
```

The `NutsOrganizationCredential` constraint example from above requires a credential to have 3 fields: `type`, `credentialSubject.organization.name` and `credentialSubject.organization.city`. The constraint on type also requires the value to be a string equal to `NutsOrganizationCredential`. The latter 2 field constraints also specify an `id`. The value of this identifier can be used in a search query.

note: the double path definitions have to do with the fact a VC/VP can be in JWT or JSON format.

Require registration parameters

As discussed earlier, it's possible for a client to add additinal information to a registration. These registration parameters can be made required in the presentation definition. This is done the same way as for other required credential fields. The required credential type is `DiscoveryRegistrationCredential`. Below is an example of an input descriptor making the `auth_server_url` required.

```
...
{
  "id": "DiscoveryRegistrationCredential",
  "constraints": {
    "fields": [
      {
        "path": ["$.type"],
        "filter": {
          "type": "string",
          "const": "DiscoveryRegistrationCredential"
        }
      },
      {
        "id":   "auth_server_url",
        "path": [
          "$.credentialSubject.authServerURL",
          "$.credentialSubject[*].authServerURL"
        ]
      }
    ]
  }
}
...
```

Any key/value passed under `registrationParameters` in the API is transfered to the `credentialSubject` of the `DiscoveryRegistrationCredential`.

Limit issuer

Most credentials will be issued by authentic sources. These sources will have a single issuer identifier. You can limit the issuer of a credential with the following snippet:

```
"constraints": {
  "fields": [
    {
      "path": ["$.issuer"],
      "purpose": "We can only accept credentials from a trusted issuer",
      "filter": {
        "type": "string",
        "pattern": "^did:web:example.com:iam:[\\w-]$"
      }
    }
  ]
}
```

The example above uses a `regex` to limit the issuer of a credential to any subject hosted by example.com. It's possible to use a `const` instead of a regex to pin a single issuer.

Full Example

A full example can be found here

Registration parameters

Registration paramaters should be used to register endpoints that are required for a service. The authorization server URL is added by default. Most services require a data endpoint, eg: a FHIR endpoint. Such an endpoint can be made required in the service definition.

Registration parameters can also be used to define a limited set of roles a party fulfills within the use case, eg: `sender` vs `receiver`. Since these values could be used to search on, a use case MUST specify the allowed values.

Definition distribution

Since the service definition defines the trust anchors, it's important to setup a secure distribution channel where clients can download the definition.

Revision #8
Created 13 May 2024 11:57:43 by Rein Krul
Updated 25 October 2024 09:35:29 by Wout Slakhorst