Endpoint Discovery

Endpoint resolving is the act of finding endpoints (e.g. the location of a REST API) given a DID.

- Using DID document services
- Using well-known URIs

Using DID document services

A party often exchanges data through its API endpoints. If a client only has the party's DID, services in the DID document can be used to register API endpoints. A service in a DID document describes its type and content (`serviceEndpoint`). A typical example is the FHIR base URL of the FHIR API which is to be accessed by other parties.

Service Type

Use cases should specify the service types when using DID document services for endpoint discovery. It's recommended to include the use case name in the service type, to avoid clashes when multiple use cases use same service type. This might lead to duplication of endpoints, but enables parties to differentiate API endpoints in case multiple use cases specify the same API family.

E.g., given the use cases "Pizza Delivery Service" and "Pasta Delivery Service" both having an "order" API, a bad approach would be:

```
{
  "services": [
    {
      "id": "#order",
      "type": "order-api",
      "serviceEndpoint": "https://example.com/api/pizza/order"
    }
  ]
}
```

An improved use case specification differentiates the service type to avoid clashes:

```
{
  "services": [
    {
      "id": "#order-pizza",
      "type": "order-pizza-api",
```

```
    "serviceEndpoint": "https://example.com/api/pizza/order"
   },
   {
    "id": "#order-pasta",
    "type": "order-pasta-api",
    "serviceEndpoint": "https://example.com/api/pasta/order"
   }
  ]
}
```

Service Endpoint

The `serviceEndpoint` property can contain a JSON string, object, array, or nesting of these types. If a use case requires multiple API endpoints, it could use a JSON object to specify multiple API endpoints:

```
{
 "services": [
  {
   "id": "#order-pizza",
   "type": "pizza-restaurant",
   "serviceEndpoint": {
    "api": "https://example.com/api/pizza/order",
    "menu": "https://example.com/pizza-list.pdf"
   }
  }
 ]
}
```

Note that services in DID documents are not restricted to describing API endpoints only; another example use for services in DID documents could be informing about who is administering the DID document.

Using well-known URIs

Many HTTP-based protocols (e.g. OpenID, OAuth2, SMART on FHIR) use well-known URIs to discover protocol metadata, which in turn contains API endpoints and other protocol-specific information. An alternative to DID document services could be using a well-known URI.

Although Nuts implements well-known URIs for OAuth/OpenID protocol support, it does not support exposing arbitrary data on a well-known URI. Hosting a use case-specific well-known endpoint could then be configured in a reverse proxy. But, as `did:web` DID documents can be resolved just as easily as documents on well-known URIs, it might be preferred to use services in DID documents instead.

Also note, that to be truly well-known as specified by RFC 8615, the URI must be registered at IANA. In any case, a URI must be chosen that does not conflict with existing well-known URIs.