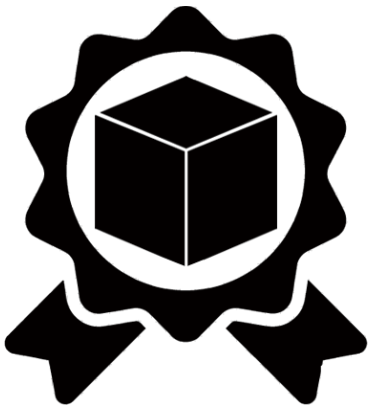
TrustBloc Documentation

Release 0.0.1

SecureKey

Mar 17, 2023

CONTENTS



CHAPTER

ONE

INTRODUCTION

This is a guide for installing and deploying TrustBloc projects. If you'd like to contribute, fork us on GitHub!.

1.1 What is TrustBloc?

An interoperability initiative aimed at establishing common standards and development frameworks for next generation digital identity networks.

1.2 Why use TrustBloc?

Several projects exist today with the goal of creating digital identities that would allow everyday users to have self-governance over the storage, distribution and control of their identity. However many of these solutions are only offered partially and may not integrate well with each other.

TrustBloc provides the end-to-end architecture that enables the deployment of a production-ready digital identity platform.

CHAPTER

TWO

ARCHITECTURE

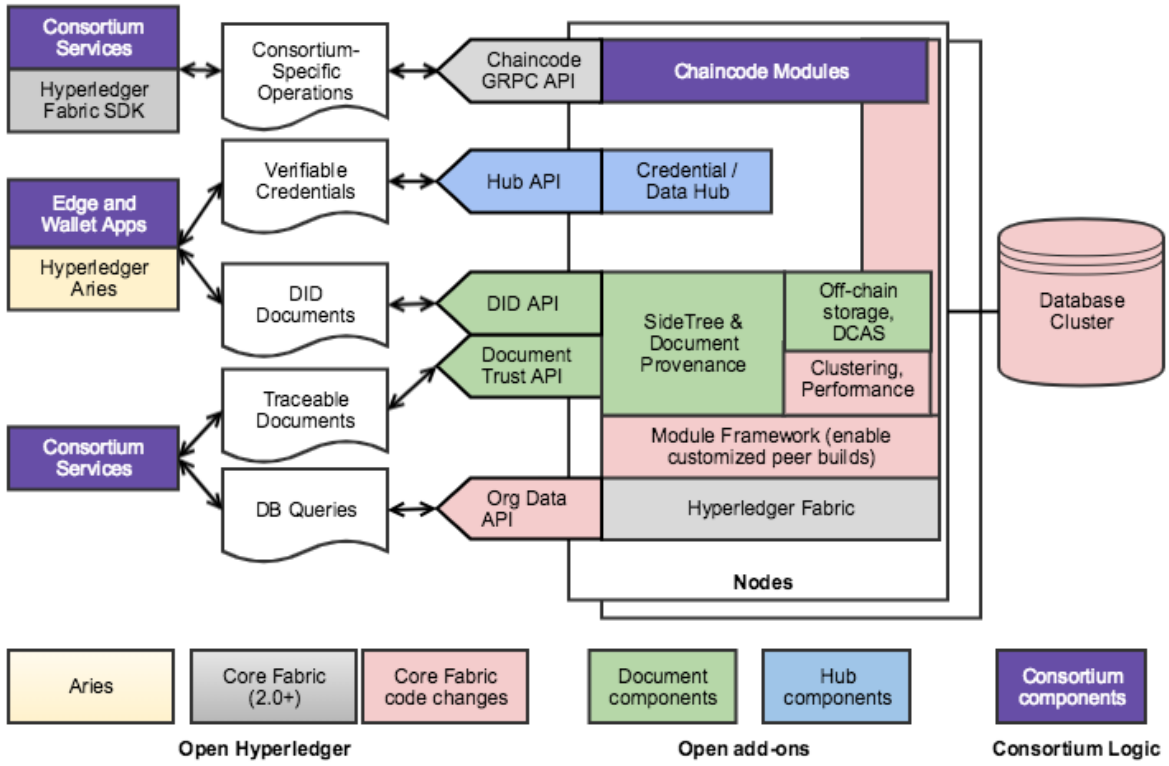


Diagram represents an overview of the components related to the TrustBloc initiative.

The TrustBloc initiative provides a common infrastructure for consortium to enable document provenance and data exchange, while still allowing these consortiums to differentiate based on their service offerings. We are building on top of existing community efforts from Hyperledger Fabric along with the proposed Hyperledger Aries project.

The TrustBloc initiative aims to provide consortium-based document provenance and services to reduce trust obstacles. We are enabling DIDs to be managed and exposed from Fabric, and more generically enabling document provenance. The following features are early goals of the initiative:

- Provide out-of-the-box capabilities for document provenance (including DIDs).
- Enhance Hyperledger Fabric private collections to support SideTrees – allowing identifiers and documents to be anchored to a channel without performing individual Fabric transaction for each identifier or document.
- Enable off-chain distributed storage model to support transactions (e.g., transient storage, content-addressable storage).

- Support a storage and query model for data scoped to a particular organization.

Data Exchange Components

TrustBloc will also provide a verifiable credential exchange flow. We are enabling a model for digital identity exchange based on DIDs, Verifiable Credentials, and Hubs. This will be achieved by:

- Leveraging emerging community specifications (e.g., DIDs and Verifiable Credentials).
- Enabling usage of Peer DIDs to establish data exchange transactions while anchoring to ledger-based DIDs and verifiable credentials.
- Supporting a Hub-based model for efficient, real-time data exchange.

CHAPTER

THREE

ORB

3.1 Introduction

Orb implements the following specifications: did:orb, Activity Anchors. The did:orb method is based on the Sidetree specification and Activity Anchors is based on the ActivityPub, ActivityStreams, and Linkset specifications.

3.1.1 Services

A typical Orb domain consists of the following services:

1) Orb instance (multiple instances may be running for redundancy scalability)
2) Document database (AWS DocumentDB or MongoDB)
3) AMQP message broker (RabbitMQ)
4) Key Management Service (Aries KMS)
5) Verifiable Credential Transparency (VCT) (Google Trillian)
6) IPFS (optional)

3.1.2 Components

The diagram below displays the components that make up the Orb server.

The section below describes a typical DID creation flow that gives an overview of the component interactions. More detailed descriptions are provided in the various sections of this document.

DID Creation Flow

Creation and resolution of DIDs involve two REST endpoints: Operation Writer and DID Resolver. The Operation Writer endpoint accepts Sidetree operations to create, update, deactivate and recover DIDs. The DID Resolver endpoint reads the Sidetree operations for a DID suffix and returns a DID document.

When an operation is posted to the Operation Writer, a message is posted to the AMQP operation queue which is consumed by the Batch Writer. The Batch Writer stores the operation and cuts a batch when the maximum batch size is reached, or the batch times out. The batch contains the DID operations that were posted since the last batch was cut. The batch is written to the Content Addressable Storage (CAS) (which can be in a local storage or IPFS) and an anchor linkset is created. The anchor linkset contains the DIDs from the batch and a verifiable credential containing a proof

from the local server. The verifiable credential is added to the local VCT. An AnchorEvent (which wraps the linkset) is created and posted via an Offer activity to the Outbox. The Offer activity is stored in the Activities database and an HTTP request (signed by KMS) is sent to the Inbox of one or more Orb servers in the witnesses collection.

Proofs from witnessing servers are received in the Inbox as Accept activities. After the Inbox verifies the HTTP signature of the request, the Accept activity is stored in the Activities database and the proof is sent to the Witness Proof Handler. When this handler determines that enough proofs have been received (according to the Witness Policy), a new anchor linkset is constructed with the gathered proofs. The anchor linkset is written to CAS and the hash of the anchor linkset is posted to the *anchor* queue for processing.

The Observer consumes the anchor linkset hash from the queue and reads the anchor linkset (along with the corresponding Sidetree operations) from CAS, and processes/stores each operation to the *operations* database.

A client sends a request to the DID Resolver endpoint to retrieve the DID document for the created/updated operation. The DID Resolver retrieves all operations related to the provided DID suffix from the Operations database, applies the operations, and returns the DID document.

3.2 Getting Started Tutorial

In this tutorial you'll start up two stand-alone Orb nodes (with in-memory database and message queue) and create, update and resolve DIDs. Further in the tutorial you'll start up the Orb nodes along with a VCT node, and you'll verify that the proofs in an anchor linkset are in the VCT log.

3.2.1 Setup

Install Docker Desktop from here.

Clone the orb project:

```
git clone git@github.com:trustbloc/orb.git
```

Build Orb and the CLI:

```
cd orb

make clean orb-docker build-orb-cli-binaries extract-orb-cli-binaries
```

3.2.2 Orb with no VCT

Start two Orb instances (orb1.local and orb2.local) and a command-line container:

```
cd orb/samples/tutorial

docker-compose -f docker-compose-cli.yml -f ../docker/docker-compose-dev.yml up&
```

In another terminal, open an interactive Docker shell:

```
docker exec -ti cli /bin/bash

cd ./orb
```

Query non-existing DID:

```
orb-cli did resolve --did-uri=did:orb:http:orb1.local:uAAA:EiBFejklGvpC6hn--
↪gZEoiDnEaeineV8xP7p0AcH1-N33A --verify-resolution-result-type=all
```

Response:

```
Error: failed to resolve did: failed to resolve did: DID does not exist
 [orb-cli] 2022/05/24 16:06:31 UTC - main.main -> CRITICAL Failed to run orb-cli: failed␣
↪to resolve did: failed to resolve did: DID does not exist
```

Create a DID at orb1:

```
orb-cli did create --domain=http://orb1.local --publickey-file=./create_publickeys.json -
↪-service-file=./create_services.json --recoverykey-file=./recover_publickey.pem --
↪updatekey-file=./update_publickey.pem --did-anchor-origin=http://orb1.local | jq
```

Response:


```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2018/v1",
    "https://w3id.org/security/suites/jws-2020/v1"
  ],
  "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
  "verificationMethod": [
    {
      "controller": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
      "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key1",
      "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
      "type": "Ed25519VerificationKey2018"
    },
    {
      "controller": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
      "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key2",
      "publicKeyJwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
        "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
      },
      "type": "JsonWebKey2020"
    }
  ],
  "service": [
    {
      "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#svc1",
      "priority": 1,
      "recipientKeys": [
        "key1"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
```


(continues on next page)

(continued from previous page)


```
            "key1"
          ],
          "uri": "https://example.com"
        }
      ],
      "type": "type1"
    },
    {
      "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#svc2",
      "priority": 2,
      "recipientKeys": [
        "key2"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
            "key2"
          ],
          "uri": "https://example.com"
        }
      ],
      "type": "type2"
    }
  ],
  "authentication": [
    "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key1",
    "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key2"
  ]
}
```


Resolve the un-anchored DID at orb1:

NOTE: Set DID_SUFFIX to the DID suffix in the *id* field of the *did create* response. For example, did:orb:uAAA:EiDEfq8bA3CqrN3_s76O5uPhm3cGnV3D7oNloVvHfHTg3w.

```
export DID_SUFFIX=EiDEfq8bA3CqrN3_s76O5uPhm3cGnV3D7oNloVvHfHTg3w

orb-cli did resolve --domain=http://orb1.local --did-uri=did:orb:uAAA:${DID_SUFFIX} --
↪verify-resolution-result-type=all | jq
```

Response:


```
{
  "@context": [
    "https://w3id.org/did-resolution/v1"
  ],
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1",
      "https://w3id.org/security/suites/jws-2020/v1"
    ],
    "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
```


(continues on next page)

(continued from previous page)


```
  "verificationMethod": [
    {
      "controller": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
      "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key1",
      "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
      "type": "Ed25519VerificationKey2018"
    },
    {
      "controller": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
      "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key2",
      "publicKeyJwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
        "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
      },
      "type": "JsonWebKey2020"
    }
  ],
  "service": [
    {
      "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#svc1",
      "priority": 1,
      "recipientKeys": [
        "key1"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
            "key1"
          ],
          "uri": "https://example.com"
        }
      ],
      "type": "type1"
    },
    {
      "id": "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#svc2",
      "priority": 2,
      "recipientKeys": [
        "key2"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
            "key2"
          ],
          "uri": "https://example.com"
        }
      ],
      "type": "type2"
    }
```


(continues on next page)

(continued from previous page)


```
    ],
    "authentication": [
      "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key1",
      "did:orb:uAAA:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key2"
    ]
  },
  "didDocumentMetadata": {
    "versionId": "uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_0eltldPYy4g",
    "canonicalId": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
    "equivalentId": [
      "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
      "did:orb:hl:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_0eltldPYy4g:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1vaDg2ZlFwOEZXT05NdmpSYXNyOGxpQzU1NE4tbzJfMGVsdGxkUFl5NGc:EiBq
↪"
    ],
    "method": {
      "updateCommitment": "EiDsq7uyQDTYJss8JAV68EcVb1sJHXgUl8knESbzvN4RWw",
      "recoveryCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "published": true,
      "anchorOrigin": "http://orb1.local",
      "publishedOperations": [
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtc2VydmljZXMiLCJzZXJ2aWNlcyI6W3siaWQiOiJzdmMxIiwicHJp
↪",
          "transactionTime": 1655130057,
          "type": "create",
          "anchorOrigin": "http://orb1.local",
          "canonicalReference": "uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_0eltldPYy4g",
          "equivalentReferences": [
            "hl:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_0eltldPYy4g:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1vaDg2ZlFwOEZXT05NdmpSYXNyOGxpQzU1NE4tbzJfMGVsdGxkUFl5NGc
↪"
          ]
        }
      ]
    }
  }
}
```


Get the hash of the anchor from the *resolve* response (from the *canonicalId* field in the metadata) and set the ANCHOR_HASH environment variable. For example did:orb:uEiB_V_lRcrutfhF2LvEIvRqsR2uO4j_1nbx6T6X-pUZQPQ:EiDEfq8bA3CqrN3_s76O5uPhm3cGnV3D7oNloVvHfHTg3w:

```
export ANCHOR_HASH=uEiB_V_lRcrutfhF2LvEIvRqsR2uO4j_1nbx6T6X-pUZQPQ
```

Resolve the canonical DID at orb1:

```
orb-cli did resolve --domain=http://orb1.local --did-uri=did:orb:${ANCHOR_HASH}:${DID_
↪SUFFIX} --verify-resolution-result-type=all | jq
```

Response:


```
{
  "@context": [
    "https://w3id.org/did-resolution/v1"
  ],
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1",
      "https://w3id.org/security/suites/jws-2020/v1"
    ],
    "id": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
    "verificationMethod": [
      {
        "controller": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
        "id": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key1",
        "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
        "type": "Ed25519VerificationKey2018"
      },
      {
        "controller": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
        "id": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key2",
        "publicKeyJwk": {
          "kty": "EC",
          "crv": "P-256",
          "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
          "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
        },
        "type": "JsonWebKey2020"
      }
    ],
    "service": [
      {
        "id": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#svc1",
        "priority": 1,
        "recipientKeys": [
          "key1"
        ],
        "serviceEndpoint": [
          {
            "routingKeys": [
              "key1"
            ],
            "uri": "https://example.com"
          }
        ],
```


(continues on next page)

(continued from previous page)


```
        "type": "type1"
      },
      {
        "id": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#svc2",
        "priority": 2,
        "recipientKeys": [
          "key2"
        ],
        "serviceEndpoint": [
          {
            "routingKeys": [
              "key2"
            ],
            "uri": "https://example.com"
          }
        ],
        "type": "type2"
      }
    ],
    "authentication": [
      "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key1",
      "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA#key2"
    ]
  },
  "didDocumentMetadata": {
    "versionId": "uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_0eltldPYy4g",
    "canonicalId": "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
    "equivalentId": [
      "did:orb:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_
↪0eltldPYy4g:EiBqyfvEMLPdGl7tADKOMV6uP2ub3PW0RemzFrMAulDxWA",
      "did:orb:hl:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_0eltldPYy4g:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1vaDg2ZlFwOEZXT05NdmpSYXNyOGxpQzU1NE4tbzJfMGVsdGxkUFl5NGc:EiBq
↪"
    ],
    "method": {
      "updateCommitment": "EiDsq7uyQDTYJss8JAV68EcVb1sJHXgUl8knESbzvN4RWw",
      "recoveryCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "published": true,
      "anchorOrigin": "http://orb1.local",
      "publishedOperations": [
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtc2VydmljZXMiLCJzZXJ2aWNlcyI6W3siaWQiOiJzdmMxIiwicHJp
↪",
          "transactionTime": 1655130057,
          "type": "create",
          "anchorOrigin": "http://orb1.local",
          "canonicalReference": "uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_0eltldPYy4g",
```


(continues on next page)

(continued from previous page)

```
          "equivalentReferences": [
            "hl:uEiBmoh86fQp8FWONMvjRasr8liC554N-o2_0eltldPYy4g:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1vaDg2ZlFwOEZXT05NdmpSYXNyOGxpQzU1NE4tbzJfMGVsdGxkUFl5NGc
↪"
          ]
        }
      ]
    }
  }
}
```

Resolve the DID at orb2. These should return 'not found' since orb2 is not following orb1:

```
orb-cli did resolve --sidetree-url-resolution=http://orb2.local/sidetree/v1/identifiers -
↪-did-uri=did:orb:uAAA:${DID_SUFFIX} --verify-resolution-result-type=all
orb-cli did resolve --sidetree-url-resolution=http://orb2.local/sidetree/v1/identifiers -
↪-did-uri=did:orb:${ANCHOR_HASH}:${DID_SUFFIX} --verify-resolution-result-type=all
```

Response:

```
Error: failed to resolve did: failed to resolve did: DID does not exist
 [orb-cli] 2022/05/24 16:39:50 UTC - main.main -> CRITICAL Failed to run orb-cli: failed␣
↪to resolve did: failed to resolve did: DID does not exist
```

Resolve a discoverable DID at orb2. This should return 'not found' but when we wait a while it should be available at orb2 since replication should have been triggered:

```
orb-cli did resolve --sidetree-url-resolution=http://orb2.local/sidetree/v1/identifiers -
↪-did-uri=did:orb:http:orb1.local:${ANCHOR_HASH}:${DID_SUFFIX}  --verify-resolution-
↪result-type=all
```

Wait a few seconds and resolve the same DID at orb2. This should return the DID document:

```
orb-cli did resolve --sidetree-url-resolution=http://orb2.local/sidetree/v1/identifiers -
↪-did-uri=did:orb:${ANCHOR_HASH}:${DID_SUFFIX}  --verify-resolution-result-type=all | jq
```

Have orb2 be a follower of orb1:

```
orb-cli follower --outbox-url=http://orb2.local/services/orb/outbox --actor=http://orb2.
↪local/services/orb --to http://orb1.local/services/orb --action=Follow
```

Response:

```
success Follow id: "http://orb2.local/services/orb/activities/17a3f004-305a-44b2-84c6-
↪7f5312ab007f"
```

Query the servers that orb2 is following:

```
curl -s "http://orb2.local/services/orb/following?page=true&page-num=0" | jq
```

Response:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "http://orb2.local/services/orb/following?page=true&page-num=0",
  "items": [
    "http://orb1.local/services/orb"
  ],
  "totalItems": 1,
  "type": "CollectionPage"
}
```


Query the servers that are following orb1:

```
curl -s "http://orb1.local/services/orb/followers?page=true&page-num=0" | jq
```

Response:


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "http://orb1.local/services/orb/followers?page=true&page-num=0",
  "items": [
    "http://orb2.local/services/orb"
  ],
  "totalItems": 1,
  "type": "CollectionPage"
}
```


Create another DID at orb1:

```
orb-cli did create --domain=http://orb1.local --publickey-file=./create_publickeys.json -
↪-service-file=./create_services2.json --recoverykey-file=./recover_publickey.pem --
↪updatekey-file=./update_publickey.pem --did-anchor-origin=http://orb1.local | jq
```

Response:


```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2018/v1",
    "https://w3id.org/security/suites/jws-2020/v1"
  ],
  "id": "did:orb:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
  "verificationMethod": [
    {
      "controller": "did:orb:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
      "id": "did:orb:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key1",
      "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
      "type": "Ed25519VerificationKey2018"
    },
    {
      "controller": "did:orb:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
      "id": "did:orb:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key2",
      "publicKeyJwk": {
        "kty": "EC",
```


(continues on next page)

(continued from previous page)


```
        "crv": "P-256",
        "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
        "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
      },
      "type": "JsonWebKey2020"
    }
  ],
  "service": [
    {
      "id": "did:orb:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#svc3",
      "priority": 1,
      "recipientKeys": [
        "key3"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
            "key3"
          ],
          "uri": "https://example.com"
        }
      ],
      "type": "type3"
    }
  ],
  "authentication": [
    "did:orb:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key1",
    "did:orb:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key2"
  ]
}
```


NOTE: Set DID_SUFFIX to the DID suffix in the *id* field of the *did create* response. For example, did:orb:uAAA:EiAEjRECXpwFaqFYwoJT5XuuUTqwigSNrOOlq4sZKlrnJg.

```
export DID_SUFFIX=EiAEjRECXpwFaqFYwoJT5XuuUTqwigSNrOOlq4sZKlrnJg

orb-cli did resolve --did-uri=did:orb:http:orb1.local:uAAA:${DID_SUFFIX} --verify-
↪resolution-result-type=all | jq
```

Response:


```
{
  "@context": [
    "https://w3id.org/did-resolution/v1"
  ],
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1",
      "https://w3id.org/security/suites/jws-2020/v1"
    ],
    "id": "did:orb:http:orb1.local:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
```


(continues on next page)

(continued from previous page)


```
    "verificationMethod": [
      {
        "controller": "did:orb:http:orb1.
↪local:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
        "id": "did:orb:http:orb1.
↪local:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key1",
        "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
        "type": "Ed25519VerificationKey2018"
      },
      {
        "controller": "did:orb:http:orb1.
↪local:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
        "id": "did:orb:http:orb1.
↪local:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key2",
        "publicKeyJwk": {
          "kty": "EC",
          "crv": "P-256",
          "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
          "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
        },
        "type": "JsonWebKey2020"
      }
    ],
    "service": [
      {
        "id": "did:orb:http:orb1.
↪local:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#svc3",
        "priority": 1,
        "recipientKeys": [
          "key3"
        ],
        "serviceEndpoint": [
          {
            "routingKeys": [
              "key3"
            ],
            "uri": "https://example.com"
          }
        ],
        "type": "type3"
      }
    ],
    "authentication": [
      "did:orb:http:orb1.local:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key1",
      "did:orb:http:orb1.local:uAAA:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key2"
    ]
  },
  "didDocumentMetadata": {
    "versionId": "uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg",
    "canonicalId": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
    "equivalentId": [
```


(continues on next page)

(continued from previous page)

```
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
      "did:orb:hl:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1sUTg5LThuamtFc1BBN3FDRFZRZjJqbWFxSXFXcjg2X1RKek5nRHo0aGc:EiDZ
↪"
    ],
    "method": {
      "updateCommitment": "EiDsq7uyQDTYJss8JAV68EcVb1sJHXgUl8knESbzvN4RWw",
      "recoveryCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "published": true,
      "anchorOrigin": "http://orb1.local",
      "publishedOperations": [
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtcHVibGljLWtleXMiLCJwdWJsaWNLZXlzIjpbeyJpZCI6ImtleTEi
↪",
          "transactionTime": 1655130300,
          "type": "create",
          "anchorOrigin": "http://orb1.local",
          "canonicalReference": "uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg",
          "equivalentReferences": [
            "hl:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1sUTg5LThuamtFc1BBN3FDRFZRZjJqbWFxSXFXcjg2X1RKek5nRHo0aGc
↪"
          ]
        }
      ]
    }
  }
}
```

NOTE: Set ANCHOR_HASH to the anchor hash of the DID in the *did resolve* response. For example, did:orb:uEiAISqEWoGwXgKEtx7Hgt1GXJkh0jPpq3w6bZ6_8FIo5nA:EiAEjRECXpwFaqFYwoJT…

```
export ANCHOR_HASH=uEiAISqEWoGwXgKEtx7Hgt1GXJkh0jPpq3w6bZ6_8FIo5nA
```

Resolve the canonical DID at orb2:

```
orb-cli did resolve --domain=http://orb2.local --did-uri=did:orb:${ANCHOR_HASH}:${DID_
↪SUFFIX} --verify-resolution-result-type=all | jq
```

Response:

```
{
  "@context": [
    "https://w3id.org/did-resolution/v1"
  ],
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1",
      "https://w3id.org/security/suites/jws-2020/v1"
```

(continues on next page)

(continued from previous page)


```
    ],
    "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
    "verificationMethod": [
      {
        "controller": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
        "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key1",
        "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
        "type": "Ed25519VerificationKey2018"
      },
      {
        "controller": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
        "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key2",
        "publicKeyJwk": {
          "kty": "EC",
          "crv": "P-256",
          "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
          "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
        },
        "type": "JsonWebKey2020"
      }
    ],
    "service": [
      {
        "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#svc3",
        "priority": 1,
        "recipientKeys": [
          "key3"
        ],
        "serviceEndpoint": [
          {
            "routingKeys": [
              "key3"
            ],
            "uri": "https://example.com"
          }
        ],
        "type": "type3"
      }
    ],
    "authentication": [
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key1",
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key2"
    ]
  },
```


(continues on next page)

(continued from previous page)


```
  "didDocumentMetadata": {
    "versionId": "uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg",
    "canonicalId": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
    "equivalentId": [
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
      "did:orb:hl:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1sUTg5LThuamtFc1BBN3FDRFZRZjJqbWFxSXFXcjg2X1RKek5nRHo0aGc:EiDZ
↪"
    ],
    "method": {
      "updateCommitment": "EiDsq7uyQDTYJss8JAV68EcVb1sJHXgUl8knESbzvN4RWw",
      "recoveryCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "published": true,
      "anchorOrigin": "http://orb1.local",
      "publishedOperations": [
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtcHVibGljLWtleXMiLCJwdWJsaWNLZXlzIjpbeyJpZCI6ImtleTEi
↪",
          "transactionTime": 1655130300,
          "type": "create",
          "anchorOrigin": "http://orb1.local",
          "canonicalReference": "uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg",
          "equivalentReferences": [
            "hl:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1sUTg5LThuamtFc1BBN3FDRFZRZjJqbWFxSXFXcjg2X1RKek5nRHo0aGc
↪"
          ]
        }
      ]
    }
  }
}
```


Look at orb2's inbox:

```
curl -s "http://orb2.local/services/orb/inbox?page=true&page-num=0" | jq
```


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "http://orb2.local/services/orb/inbox?page=true&page-num=0",
  "orderedItems": [
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "http://orb1.local/services/orb",
      "id": "http://orb1.local/services/orb/activities/e9b5b0c0-85dc-4f24-8077-
↪64c846b0578b",
      "object": {
        "@context": "https://www.w3.org/ns/activitystreams",
        "actor": "http://orb2.local/services/orb",
```


(continues on next page)

(continued from previous page)


```
        "id": "http://orb2.local/services/orb/activities/eab55426-ddc8-42b3-98c9-
↪9011a3e46199",
        "object": "http://orb1.local/services/orb",
        "to": "http://orb1.local/services/orb",
        "type": "Follow"
      },
      "to": "http://orb2.local/services/orb",
      "type": "Accept"
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "http://orb1.local/services/orb",
      "id": "http://orb1.local/services/orb/activities/217b2908-f605-4c6f-b483-
↪6359e1db3398",
      "object": {
        "@context": "https://w3id.org/activityanchors/v1",
        "object": {
          "linkset": [
            {
              "anchor": "hl:uEiC7l-eOSKfFgvBAj8P5tDS8uJLMgqyP5FuDT_8lRs7xZQ",
              "author": [
                {
                  "href": "http://orb1.local/services/orb"
                }
              ],
              "original": [
                {
                  "href": "data:application/gzip;base64,H4sIAAAAAAAA/
↪1yOzU7CQBRG3+W6BQZSEJ3dGEABJRITjCFdTDsDvTBwYX46QNN3N2Vl3H7JOd+pwOBx77QHvq5AHvOCLHAoDA9jFH192e2y4Uj0g0
↪fy0eDkqsRKxAmtbp/zj/Yy3n0nSxs0+n+F44zFBFWH7Lb5fii7UKd1Wv8GAAD///xpexUBAQAA",
                  "type": "application/linkset+json"
                }
              ],
              "profile": [
                {
                  "href": "https://w3id.org/orb#v0"
                }
              ],
              "related": [
                {
                  "href": "data:application/gzip;base64,H4sIAAAAAAAA/1TNTW+CMACA4f/
↪SXXUqlYDc/MApGyiIH7iYhdoKLY3FAgU1/Pdl3nZ/87xPwOk1K0gJrO8niK/nVEhggZRblU2nBu+S1ebzMk/
↪UZMzMtV7ONmblfLnJ7b7W59Us/DF5UBjN0QcdkEtxoZy8pFSSy59Tlnlh9Xo1pPhdyKQnJHpTfdCeOkDR+F/
↪6Wo6HpGEMGbPxsJKwoGYT7ScQLgSm96UxGOHs1k2Qip2+XluV8LsTYu8EXgT1ipoKQYe6G71Ampei6YhFB1fh7S73w4CTj/
↪wWMWz7YTA4M+8Rh+lwv8NT9+EtkRbkZJtBN+RZDO3yqDlatBW1dziD9tSe2t8AAAD//48HdpUnAQAA",
                  "type": "application/linkset+json"
                }
              ],
              "replies": [
                {
                  "href": "data:application/gzip;base64,H4sIAAAAAAAA/4ySXXObPBSE/
↪8vJLRgQn+bqtY0zftO49VdsJ51MR0gCCwg4kgA7mfz3DklqZzpNp7fS7uo5R/sM/
```


(continues on next page)

(continued from previous page)


```
↪5GqVOygIPwOO6X2MjSMtm17rd2rRGog0woMIhhlpeK4kEZjgXYW2py+yjBRvOHqiEuyq8RnKslILbg6GrLmikkja6WOTGT+o5xR5L
↪2S51+BMtqzjjBEF4TO8QUAIuyKsx3zgsEOWxX40cGphSx4cbjdD255UlB//960+zR/
↪1NG7wlem2HYZgyck78gudfVt+SS7TZjjIgpmromVQX11P08fjzL2so9WPoFhI/
↪3A3Bw32okp4wTr778NUIr5oTNBAsAJCKHiZS6ZAA3Xcs275g1fqa17mcP+iAafvKaHRea1eURFcGA0xnBihwLQS3UVerDu47+gYUa
↪z+u7qG+5XSCtHjAv/
↪xY4q8W+kh0clpIJxatyytSuor8Ea1zU3fWT46+vM4JXfLK6XOVB5q9anK2z9OjNN+3G5dtR5VKlUDsJHlMcIGvnr+10lm0fHLUYXa
↪ph/
↪NatJU9LrGrBuo6BBg0TPOEEf2ALgXIatiwOzxNdjCdfJ3xr3ZIbPJ16C3MYRdZ467V8saebqX8zmT+xeh1Lcz6Fl/
↪Ofr1/jcVyw0anIoL1X4cPR/cvPAAAA//+M9CZ8qQMAAA==",
                    "type": "application/ld+json"
                  }
                ]
              }
            ]
          },
          "type": "AnchorEvent",
          "url": "hl:uEiAEtDeVURdVG6-0vyyyaQERtMrwr8-7PbKqv-JIjZuBKg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQUV0RGVWVVJkVkc2LTB2eXl5YVFFUnRNcndyOC03UGJLcXYtSklqWnVCS2c
↪"
        },
        "published": "2022-06-13T14:25:00.0312114Z",
        "to": [
          "http://orb1.local/services/orb/followers",
          "https://www.w3.org/ns/activitystreams#Public"
        ],
        "type": "Create"
      }
    ],
    "totalItems": 2,
    "type": "OrderedCollectionPage"
}
```


Update the DID:

```
orb-cli did update --domain=http://orb1.local --did-uri=did:orb:${ANCHOR_HASH}:${DID_
↪SUFFIX} --add-publickey-file=./update_publickeys.json --signingkey-file=./update_
↪privatekey.pem --nextupdatekey-file=./nextupdate_publickey.pem --tls-
↪systemcertpool=true
```

Response:

```
successfully updated DID did:orb:uEiAISqEWoGwXgKEtx7Hgt1GXJkh0jPpq3w6bZ6_
↪8FIo5nA:EiAEjRECXpwFaqFYwoJT5XuuUTqwigSNrOOlq4sZKlrnJgroot@af0b0d3c551a
```

Resolve the DID on orb1:

```
orb-cli did resolve --domain=http://orb1.local --did-uri=did:orb:${ANCHOR_HASH}:${DID_
↪SUFFIX} --verify-resolution-result-type=all | jq
```

Response:


```
{
```


(continues on next page)

(continued from previous page)


```
  "@context": [
    "https://w3id.org/did-resolution/v1"
  ],
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1",
      "https://w3id.org/security/suites/jws-2020/v1"
    ],
    "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
    "verificationMethod": [
      {
        "controller": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
        "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key3",
        "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
        "type": "Ed25519VerificationKey2018"
      },
      {
        "controller": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
        "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key4",
        "publicKeyJwk": {
          "kty": "EC",
          "crv": "P-256",
          "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
          "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
        },
        "type": "JsonWebKey2020"
      }
    ],
    "authentication": [
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key3",
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key4"
    ]
  },
  "didDocumentMetadata": {
    "versionId": "uEiD6XU_tO_8n1PmJneefWZlmoHVFG8huAZZeLP3xTX7kYg",
    "canonicalId": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
    "equivalentId": [
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
      "did:orb:hl:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1sUTg5LThuamtFc1BBN3FDRFZRZjJqbWFxSXFXcjg2X1RKek5nRHo0aGc:EiDZ
↪"
    ],
```


(continues on next page)

(continued from previous page)


```
    "method": {
      "updateCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "recoveryCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "published": true,
      "anchorOrigin": "http://orb1.local",
      "publishedOperations": [
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtcHVibGljLWtleXMiLCJwdWJsaWNLZXlzIjpbeyJpZCI6ImtleTEi
↪",
          "transactionTime": 1655130300,
          "type": "create",
          "anchorOrigin": "http://orb1.local",
          "canonicalReference": "uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg",
          "equivalentReferences": [
            "hl:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1sUTg5LThuamtFc1BBN3FDRFZRZjJqbWFxSXFXcjg2X1RKek5nRHo0aGc
↪"
          ]
        },
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJyZW1vdmUtcHVibGljLWtleXMiLCJpZHMiOlsia2V5MSIsImtleTIiXX0s
↪",
          "transactionTime": 1655130434,
          "type": "update",
          "canonicalReference": "uEiD6XU_tO_8n1PmJneefWZlmoHVFG8huAZZeLP3xTX7kYg",
          "equivalentReferences": [
            "hl:uEiD6XU_tO_8n1PmJneefWZlmoHVFG8huAZZeLP3xTX7kYg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpRDZYVV90T184bjFQbUpuZWVmV1psbW9IVkZHOGh1QVpaZUxQM3hUWDdrWWc
↪"
          ]
        }
      ]
    }
  }
}
```


Resolve the DID on orb2:

```
orb-cli did resolve --domain=http://orb1.local --did-uri=did:orb:${ANCHOR_HASH}:${DID_
↪SUFFIX} --verify-resolution-result-type=all | jq
```

Response:


```
{
  "@context": [
    "https://w3id.org/did-resolution/v1"
  ],
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
```


(continues on next page)

(continued from previous page)


```
      "https://w3id.org/security/suites/ed25519-2018/v1",
      "https://w3id.org/security/suites/jws-2020/v1"
    ],
    "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
    "verificationMethod": [
      {
        "controller": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
        "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key3",
        "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
        "type": "Ed25519VerificationKey2018"
      },
      {
        "controller": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
        "id": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key4",
        "publicKeyJwk": {
          "kty": "EC",
          "crv": "P-256",
          "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
          "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
        },
        "type": "JsonWebKey2020"
      }
    ],
    "authentication": [
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key3",
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ#key4"
    ]
  },
  "didDocumentMetadata": {
    "versionId": "uEiD6XU_tO_8n1PmJneefWZlmoHVFG8huAZZeLP3xTX7kYg",
    "canonicalId": "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
    "equivalentId": [
      "did:orb:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_
↪TJzNgDz4hg:EiDZqFqs4RzlwAbkOQhuabqA6QD0KSTkanBGcxdfbFQ5hQ",
      "did:orb:hl:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1sUTg5LThuamtFc1BBN3FDRFZRZjJqbWFxSXFXcjg2X1RKek5nRHo0aGc:EiDZ
↪"
    ],
    "method": {
      "updateCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "recoveryCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "published": true,
      "anchorOrigin": "http://orb1.local",
      "publishedOperations": [
```


(continues on next page)

(continued from previous page)


```
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtcHVibGljLWtleXMiLCJwdWJsaWNLZXlzIjpbeyJpZCI6ImtleTEi
↪",
          "transactionTime": 1655130300,
          "type": "create",
          "anchorOrigin": "http://orb1.local",
          "canonicalReference": "uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg",
          "equivalentReferences": [
            "hl:uEiBmlQ89-8njkEsPA7qCDVQf2jmaqIqWr86_TJzNgDz4hg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQm1sUTg5LThuamtFc1BBN3FDRFZRZjJqbWFxSXFXcjg2X1RKek5nRHo0aGc
↪"
          ]
        },
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJyZW1vdmUtcHVibGljLWtleXMiLCJpZHMiOlsia2V5MSIsImtleTIiXX0s
↪",
          "transactionTime": 1655130434,
          "type": "update",
          "canonicalReference": "uEiD6XU_tO_8n1PmJneefWZlmoHVFG8huAZZeLP3xTX7kYg",
          "equivalentReferences": [
            "hl:uEiD6XU_tO_8n1PmJneefWZlmoHVFG8huAZZeLP3xTX7kYg:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpRDZYVV90T184bjFQbUpuZWVmV1psbW9IVkZHOGh1QVpaZUxQM3hUWDdrWWc
↪"
          ]
        }
      ]
    }
  }
}
```


Shut down the environment:

```
docker-compose -f docker-compose-cli.yml -f ../docker/docker-compose-dev.yml down
```

3.2.3 Orb with VCT

Start two Orb instances (orb1.local and orb2.local), VCT (includes the Google Trillian server and signer) and a command-line container:

```
cd orb/samples/tutorial

docker-compose -f docker-compose-cli.yml -f ../docker/docker-compose-dev-vct.yml up&
```

In another terminal, open an interactive Docker shell:

```
docker exec -ti cli /bin/bash

cd ./orb
```

Add a VCT log to orb1:

```
orb-cli log update --url http://orb1.local/log --log http://orb.vct:8077/maple2022
```

Response:

```
Domain log has successfully been updated.
```

Create a DID at orb1:

```
orb-cli did create --domain=http://orb1.local --publickey-file=./create_publickeys.json --
↪-service-file=./create_services.json --recoverykey-file=./recover_publickey.pem --
↪updatekey-file=./update_publickey.pem --did-anchor-origin=http://orb1.local | jq
```

Response:


```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2018/v1",
    "https://w3id.org/security/suites/jws-2020/v1"
  ],
  "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA",
  "verificationMethod": [
    {
      "controller": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA",
      "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#key1",
      "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
      "type": "Ed25519VerificationKey2018"
    },
    {
      "controller": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA",
      "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#key2",
      "publicKeyJwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
        "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
      },
      "type": "JsonWebKey2020"
    }
  ],
  "service": [
    {
      "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#svc1",
      "priority": 1,
      "recipientKeys": [
        "key1"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
            "key1"
          ],
          "uri": "https://example.com"
```


(continues on next page)

(continued from previous page)

```
        }
      ],
      "type": "type1"
    },
    {
      "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#svc2",
      "priority": 2,
      "recipientKeys": [
        "key2"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
            "key2"
          ],
          "uri": "https://example.com"
        }
      ],
      "type": "type2"
    }
  ],
  "authentication": [
    "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#key1",
    "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#key2"
  ]
}
```

Resolve the DID at orb1 to get the anchor hash:

NOTE: Set DID_SUFFIX to the DID suffix in the *id* field of the *did create* response. For example, did:orb:uAAA:EiCGIu4PrFTVEZRb8SOZt4nbZRF7Wp2_qX4Zt0czROKHxg.

```
export DID_SUFFIX=EiCGIu4PrFTVEZRb8SOZt4nbZRF7Wp2_qX4Zt0czROKHxg

orb-cli did resolve --domain=http://orb1.local --did-uri=did:orb:uAAA:${DID_SUFFIX} --
↪verify-resolution-result-type=all | jq
```

Response:

```
{
  "@context": [
    "https://w3id.org/did-resolution/v1"
  ],
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1",
      "https://w3id.org/security/suites/jws-2020/v1"
    ],
    "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA",
    "verificationMethod": [
      {
        "controller": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA",
```

(continues on next page)

(continued from previous page)


```
      "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#key1",
      "publicKeyBase58": "BzcCdRP41BvfyYUq2aC5U5RXdp4zXjYfduubF6EuE79R",
      "type": "Ed25519VerificationKey2018"
    },
    {
      "controller": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA",
      "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#key2",
      "publicKeyJwk": {
        "kty": "EC",
        "crv": "P-256",
        "x": "bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
        "y": "PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
      },
      "type": "JsonWebKey2020"
    }
  ],
  "service": [
    {
      "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#svc1",
      "priority": 1,
      "recipientKeys": [
        "key1"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
            "key1"
          ],
          "uri": "https://example.com"
        }
      ],
      "type": "type1"
    },
    {
      "id": "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#svc2",
      "priority": 2,
      "recipientKeys": [
        "key2"
      ],
      "serviceEndpoint": [
        {
          "routingKeys": [
            "key2"
          ],
          "uri": "https://example.com"
        }
      ],
      "type": "type2"
    }
  ],
  "authentication": [
    "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#key1",
```


(continues on next page)

(continued from previous page)

```
      "did:orb:uAAA:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA#key2"
    ]
  },
  "didDocumentMetadata": {
    "versionId": "uEiCVI1cESlp5f_H503m_WZn-Zu3jt6f0j1bqc8xE79hRkw",
    "canonicalId": "did:orb:uEiCVI1cESlp5f_H503m_WZn-
↪Zu3jt6f0j1bqc8xE79hRkw:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA",
    "equivalentId": [
      "did:orb:uEiCVI1cESlp5f_H503m_WZn-
↪Zu3jt6f0j1bqc8xE79hRkw:EiDMMnBARgGo1J7Y6v3p3CFv5Wu5XcNEpQSmisoufLyrVA",
      "did:orb:hl:uEiCVI1cESlp5f_H503m_WZn-Zu3jt6f0j1bqc8xE79hRkw:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQ1ZJMWNFU2xwNWZfSDUwM21fV1puLVp1M2p0NmYwajFicWM4eEU3OWhSa3c:EiDM
↪"
    ],
    "method": {
      "updateCommitment": "EiDsq7uyQDTYJss8JAV68EcVb1sJHXgUl8knESbzvN4RWw",
      "recoveryCommitment": "EiAVtjXx-WHYfWcAzzgbHdcADhF1KVSzpy5Tb_xnu1cbzA",
      "published": true,
      "anchorOrigin": "http://orb1.local",
      "publishedOperations": [
        {
          "operation":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtcHVibGljLWtleXMiLCJwdWJsaWNLZXlzIjpbeyJpZCI6ImtleTEi
↪",
          "transactionTime": 1655130610,
          "type": "create",
          "anchorOrigin": "http://orb1.local",
          "canonicalReference": "uEiCVI1cESlp5f_H503m_WZn-Zu3jt6f0j1bqc8xE79hRkw",
          "equivalentReferences": [
            "hl:uEiCVI1cESlp5f_H503m_WZn-Zu3jt6f0j1bqc8xE79hRkw:uoQ-
↪BeEVodHRwOi8vb3JiMS5sb2NhbC9jYXMvdUVpQ1ZJMWNFU2xwNWZfSDUwM21fV1puLVp1M2p0NmYwajFicWM4eEU3OWhSa3c
↪"
          ]
        }
      ]
    }
  }
}
```

Get hash of anchor from the *resolve* response (from the *canonicalId* field in the metadata) and set the ANCHOR_HASH environment variable. For example, did:orb:uEiBtg3HASdLSUHLYds6givvkjJX_DEb8cqONqpKuq_Y_aA:EiCGIu4PrFTVEZRb8SOZ...

Verify that the proof(s) in the anchor linkset are in the VCT log:

```
export ANCHOR_HASH=uEiBtg3HASdLSUHLYds6givvkjJX_DEb8cqONqpKuq_Y_aA

orb-cli vct verify --cas-url=http://orb1.local/cas --anchor=${ANCHOR_HASH}
```

Response:

```
[
  {
```

(continues on next page)

(continued from previous page)

```
    "domain": "http://orb.vct:8077/maple2022",
    "proofValue":
↪"z5LcS2Aa8zGVwunZKFap9DJZYs3YaRgU2MD5kfMjYyK5oLfBZUZieZBsvkpdqqjUZeZCBJqBQnJ6r6esQUwQXmMhG
↪",
    "found": true,
    "leafIndex": 0,
    "auditPath": null
  }
]
```

You'll notice that *leafIndex* is 0 and *auditPath* is empty since there's only one entry in the Merkle tree.

Create another DID at orb1 and use `orb-cli vct verify` to verify that the proofs are in the log.

```
[
  {
    "domain": "http://orb.vct:8077/maple2022",
    "proofValue": "JnP2nE4jLiHsmR65q1a6Tp-
↪HUKZanjpCstm7qAljIE3Z6Y2mEcNarOESiu1gFlsonz1soQJe69piE7qZnTqrAw",
    "found": true,
    "leafIndex": 1,
    "auditPath": [
      "rgWgpjyv6227zFbcpyRv3YR1vbLbXaOuCeYoF7uDaY4="
    ]
  }
]
```

You'll now notice that *leafIndex* is 1 and *auditPath* has one entry.

Shut down the environment:

```
docker-compose -f docker-compose-cli.yml -f ../docker/docker-compose-dev-vct.yml down
```

3.3 Client Libraries and Utilities

3.3.1 Command-Line Interface (CLI)

The Orb Command Line Interface (Orb CLI) is a unified tool that provides a consistent interface for interacting with all parts of Orb.

Create DID

This command used for creating DID.

Usage

```
did create [flags]
```

Flags

- `domain` *[string]* - URL to the Orb domain.
- `sidetree-url` *[array|string]* - Array of one or more Sidetree URLs.
- `sidetree-write-token` *[string]* - The Sidetree write token.
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.
- `publickey-file` *[string]* - The file contains the DID public keys.
- `service-file` *[string]* - The file contains the DID services.
- `recoverykey` *[string]* - The public key PEM used for recovery of the document.
- `recoverykey-file` *[string]* - The file that contains the public key PEM used for recovery of the document.
- `updatekey` *[string]* - The public key PEM used for validating the signature of the next update of the document.
- `updatekey-file` *[string]* - The file that contains the public key PEM used for validating the signature of the next update of the document.

Example

create cmd

```
did create --domain https://orb1.com --publickey-file ./publickeys.json --service-file ./
↪services.json
--recoverykey-file ./keys/recover/public.pem --updatekey-file ./keys/update/public.pem
```

publickeys.json

```
[
 {
  "id": "key1",
  "type": "Ed25519VerificationKey2018",
  "purposes": ["authentication"],
  "jwkPath": "./key1_jwk.json"
 },
 {
  "id": "key2",
  "type": "JwsVerificationKey2020",
  "purposes": ["capabilityInvocation"],
  "jwkPath": "./key2_jwk.json"
 }
]
```

key1_jwk.json

```
{
  "kty":"OKP",
  "crv":"Ed25519",
  "x":"o1bG1U7G3CNbtALMafUiFOq8ODraTyVTmPtRDO1QUWg",
  "y":""
}
```

key2_jwk.json

```
{
  "kty":"EC",
  "crv":"P-256",
  "x":"bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
  "y":"PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
}
```

services.json

```
[
  {
    "id": "svc1",
    "type": "type1",
    "priority": 1,
    "routingKeys": ["key1"],
    "recipientKeys": ["key1"],
    "serviceEndpoint": "http://www.example.com"
  },
  {
    "id": "svc2",
    "type": "type2",
    "priority": 2,
    "routingKeys": ["key2"],
    "recipientKeys": ["key2"],
    "serviceEndpoint": "http://www.example.com"
  }
]
```

Resolve

This command used for resolving DID.

Usage

```
did resolve [flags]
```

Flags

- `domain` *[string]* - URL to the Orb domain.
- `sidetree-url-resolution` *[array|string]* - Array of one or more Sidetree URLs resolution.
- `auth-token` *[string]* - The Auth token.
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.
- `did-uri` *[string]* - DID URI.
- `verify-resolution-result-type` *[string]* - Verify resolution result type. Values [all, none, unpublished].

Difference between domain and sidetree-url-resolution

Domain will be used to discover Orb resolution endpoint if you have Orb DID without discovery information. sidetree-url-resolution will not discover any endpoint and hit the resolution directly.

Example

resolve cmd

```
did resolve --domain https://orb1.com --did-uri␣
↪did:orb:3XvwJ:EiDnJwbKHkHdaco4khFeBzvSL1hZ4eBGQq3q1Yjrpi5d4g
--verify-resolution-result-type all
```

resolve cmd DID with discovery information

```
did resolve --did-uri did:orb:https:orb1.com:uAAA:EiAFqEsuKDpwbfFxHfqP-TLdFPjqSrFWwgj8_
↪dU64GMcEQ
--verify-resolution-result-type all
```

Update

This command used for updating DID.

Usage

```
did update [flags]
```

Flags

- `domain` *[string]* - URL to the Orb domain.
- `sidetree-url-operation` *[array|string]* - Array of one or more Sidetree URLs operation.
- `sidetree-url-resolution` *[array|string]* - Array of one or more Sidetree URLs resolution.
- `sidetree-write-token` *[string]* - The Sidetree write token.
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.
- `did-uri` *[string]* - DID URI.
- `add-publickey-file` *[string]* - The file contains the DID public keys to be added or updated.
- `add-service-file` *[string]* - The file contains the DID services to be added or updated.
- `nextupdatekey` *[string]* - The public key PEM used for validating the signature of the next update of the document.
- `nextupdatekey-file` *[string]* - The file that contains the public key PEM used for validating the signature of the next update of the document.
- `signingkey` *[string]* - The private key PEM used for signing the update of the document.
- `signingkey-file` *[string]* - The file that contains the private key PEM used for signing the update of the document.
- `signingkey-password` *[string]* - The Signing key PEM password.

Example

update cmd

```
did update --domain https://orb1.com --did-uri␣
↪did:orb:3XvwJ:EiDnJwbKHkHdaco4khFeBzvSL1hZ4eBGQq3q1Yjrpi5d4g
--add-publickey-file ./publickeys.json --add-service-file ./services.json --signingkey-
↪file ./keys/update/key_encrypted.pem --signingkey-password 123
--nextupdatekey-file ./keys/update2/public.pem
```

publickeys.json

```
[
 {
 "id": "key2",
 "type": "JwsVerificationKey2020",
 "purposes": ["capabilityInvocation"],
 "jwkPath": "./key2_jwk.json"
 },
 {
  "id": "key3",
  "type": "Ed25519VerificationKey2018",
  "purposes": ["authentication"],
  "jwkPath": "./key3_jwk.json"
 }
]
```

key2_jwk.json

```
{
  "kty":"EC",
  "crv":"P-256",
  "x":"bGM9aNufpKNPxlkyacU1hGhQXm_aC8hIzSVeKDpwjBw",
  "y":"PfdmCOtIdVY2B6ucR4oQkt6evQddYhOyHoDYCaI2BJA"
}
```

key3_jwk.json

```
{
  "kty":"OKP",
  "crv":"Ed25519",
  "x":"o1bG1U7G3CNbtALMafUiFOq8ODraTyVTmPtRDO1QUWg",
  "y":""
}
```

services.json

```
[
  {
    "id": "svc3",
    "type": "type3",
    "priority": 3,
    "routingKeys": ["key3"],
    "recipientKeys": ["key3"],
    "serviceEndpoint": "http://www.example.com"
  }
]
```

Recover

This command used for recovering DID.

Usage

```
did recover [flags]
```

Flags

- `domain` *[string]* - URL to the Orb domain.
- `sidetree-url-operation` *[array|string]* - Array of one or more Sidetree URLs operation.
- `sidetree-url-resolution` *[array|string]* - Array of one or more Sidetree URLs resolution.
- `sidetree-write-token` *[string]* - The Sidetree write token.
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.
- `did-uri` *[string]* - DID URI.
- `publickey-file` *[string]* - The file contains the DID public keys to be recovered.
- `service-file` *[string]* - The file contains the DID services to be recovered.
- `nextupdatekey` *[string]* - The public key PEM used for validating the signature of the next update of the document.
- `nextupdatekey-file` *[string]* - The file that contains the public key PEM used for validating the signature of the next update of the document.
- `nextrecoverkey` *[string]* - The public key PEM used for validating the signature of the next recovery of the document.
- `nextrecoverkey-file` *[string]* - The file that contains the public key PEM used for validating the signature of the next recovery of the document.
- `signingkey` *[string]* - The private key PEM used for signing the recovery of the document.
- `signingkey-file` *[string]* - The file that contains the private key PEM used for signing the recovery of the document.
- `signingkey-password` *[string]* - The Signing key PEM password.

Example

recover cmd

```
did recover --domain https://orb1.com --did-uri␣
↪did:orb:3XvwJ:EiDZTmh3BNBzhwSlOdh3FwAdjzu4BkWly2MoTVNHoNdJpw
--publickey-file ./publickeys.json  --service-file ./services.json
--nextrecoverkey-file ./keys/recover2/public.pem --nextupdatekey-file ./keys/update3/
↪public.pem
--signingkey-file ./keys/recover/key_encrypted.pem --signingkey-password 123
```

publickeys.json

```
[
 {
  "id": "key-recover-id",
  "type": "Ed25519VerificationKey2018",
  "purposes": ["authentication"],
  "jwkPath": "./fixtures/did-keys/recover/key1_jwk.json"
 }
]
```

key1_jwk.json

```
{
  "kty":"OKP",
  "crv":"Ed25519",
  "x":"o1bG1U7G3CNbtALMafUiFOq8ODraTyVTmPtRDO1QUWg",
  "y":""
}
```

services.json

```
[
  {
    "id": "svc-recover-id",
    "type": "type1",
    "priority": 1,
    "routingKeys": ["key1"],
    "recipientKeys": ["key1"],
    "serviceEndpoint": "http://www.example.com"
  }
]
```

Deactivate

This command used for Deactivating DID.

Usage

```
did deactivate [flags]
```

Flags

- `domain` *[string]* - URL to the Orb domain.
- `sidetree-url-operation` *[array|string]* - Array of one or more Sidetree URLs operation.
- `sidetree-url-resolution` *[array|string]* - Array of one or more Sidetree URLs resolution.
- `sidetree-write-token` *[string]* - The Sidetree write token.
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.
- `did-uri` *[string]* - DID URI.
- `signingkey` *[string]* - The private key PEM used for signing deactivate of the document.
- `signingkey-file` *[string]* - The file that contains the private key PEM used for signing deactivate of the document.
- `signingkey-password` *[string]* - The Signing key PEM password.

Example

deactivate cmd

```
deactivate-did --domain https://orb1.com --did-uri␣
↪did:orb:3XvwJ:EiDnJwbKHkHdaco4khFeBzvSL1hZ4eBGQq3q1Yjrpi5d4g
--signingkey-file ./keys/recover2/key_encrypted.pem --signingkey-password 123
```

Follow

This command used for manage followers.

Usage

```
follower [flags]
```

Flags

- `outbox-url` *[string]* - Outbox url.
- `actor` *[string]* - Actor IRI.
- `to` *[string]* - To IRI.
- `action` *[string]* - Follower action (Follow, Undo).
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.
- `auth-token` *[string]* - Auth token.
- `follow-id` *[string]* - Follow id required for undo action.

Example

follow cmd (orb-2 server follows orb-1 server)

```
follower --outbox-url=https://orb-1.com/services/orb/outbox --actor=https://orb-2/
↪services/orb
--to=https://orb-1/services/orb --action=Follow --auth-token=token123
```

undo cmd (orb-2 server unfollows orb-1 server)

```
follower --outbox-url=https://orb-1.com/services/orb/outbox --actor=https://orb-2/
↪services/orb
--to=https://orb-1/services/orb --action=Undo --auth-token=token123 --follow-id=r232DD##
```

Witness

This command used for manage witness.

Usage

```
follower [flags]
```

Flags

- `outbox-url` *[string]* - Outbox url.
- `actor` *[string]* - Actor IRI.
- `to` *[string]* - To IRI.
- `action` *[string]* - Follower action (Follow, Undo).
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.
- `auth-token` *[string]* - Auth token.
- `follow-id` *[string]* - Follow id required for undo action.

Example

witness cmd (orb-1 invites orb-2 to be a witness)

```
witness --outbox-url=https://orb-1.com/services/orb/outbox --actor=https://orb-1.com/
↪services/orb
--to=https://orb-2.com/services/orb --action=InviteWitness --auth-token=token123
```

undo cmd (orb-1 uninvite orb-2 to be a witness)

```
witness --outbox-url=https://orb-1.com/services/orb/outbox --actor=https://orb-1.com/
↪services/orb
--to=https://orb-2.com/services/orb --action=InviteWitness --auth-token=token123 --
↪invite-witness-id=r2WW##
```

Accept List

The *acceptlist* command adds and removes actors from the *follow* and *witness* accept-lists.

Usage

```
acceptlist [command] [flags]
```

Flags

- `url` *[string]* - Accept-list endpoint.
- `actor` *[array|string]* - Array of one or more actors to add to the accept-list.
- `type` *[string]* - Accept-list type - either *follow* or *witness*.

Add Command

Adds one or more actors to an accept-list of a given type.

Example

The orb domain, orb-1.com, adds orb-2.com and orb-3.com to the *follow* accept list:

```
acceptlist add --url https://orb-1.com/services/orb/acceptlist --actor https://orb-2.com/
↪services/orb --actor https://orb-3.com/services/orb --type follow
```

Remove Command

Removes one or more actors from an accept-list of a given type:

Example

The orb domain, orb-1.com, removes orb-2.com from the *witness* accept list:

```
acceptlist remove --url https://orb-1.com/services/orb/acceptlist --actor https://orb-2.
↪com/services/orb --type witness
```

Get Command

Retrieves all accept-lists or an accept-list of the given type.

Example

Retrieve orb-1.com's *witness* accept-list:

```
acceptlist get --url https://orb-1.com/services/orb/acceptlist --type witness
```

Retrieve all of orb-1's accept-lists:

```
acceptlist get --url https://orb-1.com/services/orb/acceptlist
```

Policy

The *policy* command updates and retrieves the witness policy.

Usage

```
policy [command] [flags]
```

Flags

- `url` *[string]* - Witness policy endpoint.
- `policy` *[string]* - The policy. For example "MinPercent(100,batch) AND OutOf(1,system)".

Update Command

The *update* command updates a witness policy.

Example

```
policy update --url https://orb-1.com/policy --policy "MinPercent(100,batch) AND OutOf(1,
↪system)"
```

Get Command

The *get* command retrieves the witness policy.

Example

```
policy get --url https://orb-1.com/policy
```

Log

The *log* command updates and retrieves the VCT log URL.

Usage

```
log [command] [flags]
```

Flags

- `url` *[string]* - Log endpoint.
- `log` *[string]* - The log URL. For example "https://vct.com/log".

Update Command

The *update* command updates a log URL.

Example

```
log update --url https://orb-1.com/log --log https://vct.com/log
```

Get Command

The *get* command retrieves the log URL.

Example

```
log get --url https://orb-1.com/log
```

Log Monitor

The *logmonitor* command activates and deactivates log from the list of logs that are observed by log monitoring service. log-monitor.

Usage

```
logmonitor [command] [flags]
```

Flags

- `url` *[string]* - Log monitor endpoint.
- `log` *[array|string]* - Array of one or more logs to activate for log monitoring.

Activate Command

Adds one or more logs to the list of logs that are monitored by log monitoring service.

Example

```
logmonitor activate --url https://orb-1.com/log-monitor --url https://vct.com/log --url␣
↪https://other-vct.com/some-log
```

Deactivate Command

Removes one or more logs from the list of logs that are monitored by log monitoring service.

Example

```
logmonitor deactivate --url https://orb-1.com/log-monitor --url https://vct.com/log --
↪url https://other-vct.com/some-log
```

Get Command

Retrieves all logs of the given status that are observed by monitoring service. Status can be either active or inactive. If not supplied it defaults to active.

Example

Retrieve orb-1.com's inactive logs:

```
logmonitor get --url https://orb-1.com/log-monitor --status inactive
```

Retrieve all of orb-1's active logs:

```
logmonitor get --url https://orb-1.com/log-monitor --status active
```

VCT

The *vct* command interacts with the VCT server to verify that an anchor has been added to a log.

Usage

```
vct [command] [flags]
```

Flags

- `anchor` *[string]* - The hash of the anchor linkset.
- `cas-url` *[string]* - The CAS URL.
- `vct-auth-token` *[string]* - The authorization bearer token for the VCT server (optional).

Verify Command

The *verify* command verifies that the anchor linkset (given by the anchor hash) has been added to the VCT log(s) that are specified in the proofs of the verifiable credential in the linkset.

Example

```
vct verify --anchor uEiDuIicNljP8PoHJk6_aA7w1d4U3FAvDMfF7Dsh7fkw3Wg --cas-url https://
↪orb.domain1.com/cas
```

Response:

```
[
  {
    "domain": "https://vct.domain1.com/maple2020",
    "proofValue": "JnP2nE4jLiHsmR65q1a6Tp-
↪HUKZanjpCstm7qAljIE3Z6Y2mEcNarOESiu1gFlsonz1soQJe69piE7qZnTqrAw",
    "found": true,
    "leafIndex": 12,
    "auditPath": [
      "rgWgpjyv6227zFbcpyRv3YR1vbLbXaOuCeYoF7uDaY4=",
      "lg4iaTTfHNrH41C8CBuSlo+VIK+f+2dA+5i5N8NzsIM="
    ]
  },
  {
    "domain": "https://orb.domain2.com",
    "proofValue": "6H8dEtYndINR48g4mFJ9wAxXMVYu7FdOEzpwRap5AuOTPPpd9E7LYjME-
↪pH2kvqiFLe9ZmKDJa2YzqOoLjHJAQ",
    "found": false,
    "leafIndex": 0,
    "auditPath": null,
    "error": "get STH: 404 page not found\n"
  }
]
```

3.3.2 Universal Resolver Driver

Orb Driver is used for Universal Resolver

API

`https://driver-url/1.0/identifiers/<your-did>`

Driver will receive an `Accept` header with the value `application/ld+json`, and it should return either a valid DID Document or a DID Resolution Result in the HTTP body.Driver should also return an appropriate value in the `Content-Type` header, such as `application/did+ld+json`.

Startup Parameters

ORB_DRIVER_HOST_URL

URL to run the orb-driver instance on. Format: HostName:Port.

ORB_DRIVER_TLS_CACERTS

Comma-Separated list of ca certs path.

ORB_DRIVER_TLS_CERTIFICATE

TLS certificate for ORB driver.

ORB_DRIVER_TLS_KEY

TLS key for ORB driver.

ORB_DRIVER_DOMAIN

Discovery endpoint domain.

ORB_DRIVER_SIDETREE_TOKEN

Authorization token for driver REST endpoints.

ORB_DRIVER_VERIFY_RESOLUTION_RESULT_TYPE

verify resolution result type. Values [all, none, unpublished].

3.3.3 Verifiable Data Registry (VDR)

The VDR is a Go library that may be used by clients to manage DID operation.

New

New will return new instance of Orb VDR.

```
func New(keyRetriever KeyRetriever, opts ...Option) (*VDR, error)
```

Key Retriever interface

Key Retriever is used to manage operation keys.

GetNextRecoveryPublicKey

GetNextRecoveryPublicKey is called in recover DID to get the recover next public key. This public key will be used to verify next recover request (the client need to use the private key to sign next recover request).

```
GetNextRecoveryPublicKey(didID, commitment string) (crypto.PublicKey, error)
```

GetNextUpdatePublicKey

GetNextUpdatePublicKey is called in update and recover DID to get the update next public key. This public key will be used to verify next update request (the client need to use the private key to sign next update request).

```
GetNextUpdatePublicKey(didID, commitment string) (crypto.PublicKey, error)
```

GetSigningKey

GetSigningKey is called in update,recover and deactivate DID to get the private key. This private key will be used to sign update,recover and deactivate request.

OperationType update need private key for update DID request. OperationType recover need private key for recover or deactivate DID request.

```
GetSigningKey(didID string, ot OperationType, commitment string) (crypto.PrivateKey,
error)
```

Options

New options.

WithHTTPClient

WithHTTPClient option is for custom http client.

```
WithHTTPClient(httpClient *http.Client) Option
```

WithTLSConfig

WithTLSConfig option is for definition of secured HTTP transport using a tls.Config instance.

```
func WithTLSConfig(tlsConfig *tls.Config) Option
```

WithUnanchoredMaxLifeTime

WithUnanchoredMaxLifeTime option is max time for unanchored to be trusted.

```
func WithUnanchoredMaxLifeTime(duration time.Duration) Option
```

WithVerifyResolutionResultType

WithVerifyResolutionResultType option is set verify resolution result type.

VerifyResolutionResultType Types:

- ***All***: Will not trust server and verify provided resolution result from server against resolution result that is assembled from published (DID anchored) and unpublished (DID not anchored yet) operations.
- ***Unpublished***: Will not trust server and verify provided resolution result from server against resolution result that is assembled from unpublished operations (DID not anchored yet).
- ***None***: Will trust server and not verify document.

```
func WithVerifyResolutionResultType(v VerifyResolutionResultType) Option
```

WithAuthToken

WithAuthToken option add auth token.

```
func WithAuthToken(authToken string) Option
```

WithDomain

WithDomain option add Orb domains that vdr will them to communicate.

To add multiple domains you need to call this option once for each domain.

```
func WithDomain(domain string) Option
```

WithDocumentLoader

WithDocumentLoader option overrides the default JSONLD document loader used when processing JSONLD DID documents.

```
func WithDocumentLoader(l jsonld.DocumentLoader) Option
```

Example

```
import (
        "crypto"
        "github.com/hyperledger/aries-framework-go-ext/component/vdr/orb"
)

type keyRetrieverImpl struct {
        nextRecoveryPublicKey crypto.PublicKey
        nextUpdatePublicKey   crypto.PublicKey
        updateKey             crypto.PrivateKey
        recoverKey            crypto.PrivateKey
}

func (k *keyRetrieverImpl) GetNextRecoveryPublicKey(didID string) (crypto.PublicKey,␣
↪error) {
        return k.nextRecoveryPublicKey, nil
}

func (k *keyRetrieverImpl) GetNextUpdatePublicKey(didID string) (crypto.PublicKey,␣
↪error) {
        return k.nextUpdatePublicKey, nil
}

func (k *keyRetrieverImpl) GetSigningKey(didID string, ot orb.OperationType) (crypto.
↪PrivateKey, error) {
        if ot == orb.Update {
                return k.updateKey, nil
        }

        return k.recoverKey, nil
}


keyRetrieverImpl := &keyRetrieverImpl{}
```

(continues on next page)

(continued from previous page)

```
vdr, err := orb.New(keyRetrieverImpl, orb.WithDomain("https://testnet.devel.trustbloc.dev
↪"))
        if err != nil {
                return err
}
```

Create

Create used to create new Orb DID.

`func Create(did *docdid.Doc, opts ...vdrapi.DIDMethodOption) (*docdid.DocResolution, error)`

Options

Create options.

RecoveryPublicKeyOpt

This option is mandatory. Will be used to set recovery key private key for create.

UpdatePublicKeyOpt

This option is mandatory. Will be used to set update key private key for create.

OperationEndpointsOpt

This option is mandatory when domain not set. Will be used to set operation endpoint.

AnchorOriginOpt

This option is mandatory when domain not set. Will be used to set anchor origin for create request.

CheckDIDAnchored

This option is not mandatory. Will be used check if DID is anchored.

Value of CheckDIDAnchored option:

```
type ResolveDIDRetry struct {
        MaxNumber int
        SleepTime *time.Duration
}
```

Example

```
import (
"crypto"
"crypto/ed25519"
"crypto/rand"
"fmt"

ariesdid "github.com/hyperledger/aries-framework-go/pkg/doc/did"
"github.com/hyperledger/aries-framework-go/pkg/doc/jose"
vdrapi "github.com/hyperledger/aries-framework-go/pkg/framework/aries/api/vdr"

"github.com/hyperledger/aries-framework-go-ext/component/vdr/orb"
)

recoveryKey, recoveryKeyPrivateKey, err := ed25519.GenerateKey(rand.Reader)
if err != nil {
        return err
}

updateKey, updateKeyPrivateKey, err := ed25519.GenerateKey(rand.Reader)
if err != nil {
        return err
}

didPublicKey, _, err := ed25519.GenerateKey(rand.Reader)
if err != nil {
        return err
}

jwk, err := jose.JWKFromKey(didPublicKey)
if err != nil {
        return err
}

vm,err:=ariesdid.NewVerificationMethodFromJWK("key1", "Ed25519VerificationKey2018", "",␣
↪jwk)
if err != nil {
        return err
}

didDoc := &ariesdid.Doc{}

// add did keys
didDoc.Authentication = append(didDoc.Authentication, *ariesdid.
↪NewReferencedVerification(vm,
                ariesdid.Authentication))

// add did services
didDoc.Service = []ariesdid.Service{{ID: "svc1", Type: "type", ServiceEndpoint: "http://
↪www.example.com/"}}
```

(continues on next page)

(continued from previous page)

```
// create did
createdDocResolution, err := vdr.Create(didDoc,
                vdrapi.WithOption(orb.RecoveryPublicKeyOpt, recoveryKey),
                vdrapi.WithOption(orb.UpdatePublicKeyOpt, updateKey),
                // No need to use this option because we already use domain
                // vdrapi.WithOption(orb.OperationEndpointsOpt, []string{"https://orb-1.
↪devel.trustbloc.dev/sidetree/v1/operations"}),
                vdrapi.WithOption(orb.AnchorOriginOpt, "https://orb-2.devel.trustbloc.
↪dev/services/orb"))
if err != nil {
        return err
}

fmt.Println(createdDocResolution.DIDDocument.ID)
```

Read

Read used to resolve Orb DID.

```
func Read(did string, opts ...vdrapi.DIDMethodOption) (*docdid.DocResolution, error)
```

Options

Read options.

ResolutionEndpointsOpt

This option is mandatory when domain not set. Will be used to set resolution endpoint.

Example

```
docResolution, err := vdr.Read(didID)
if err != nil {
        return err
}

fmt.Println(docResolution.DIDDocument.ID)
```

Update

Update used to update or recover Orb DID.

`func Update(didDoc *docdid.Doc, opts ...vdrapi.DIDMethodOption) error`

Options

Update options.

RecoverOpt

This option is mandatory. Will be used to signal that it's recover request [true, false].

AnchorOriginOpt

This option is not mandatory. Will be used to set anchor origin for recover request.

OperationEndpointsOpt

This option is mandatory when domain not set. Will be used to set operation endpoint for recover request.

ResolutionEndpointsOpt

This option is mandatory when domain not set. Will be used to set resolution endpoint.

CheckDIDUpdated

This option is not mandatory. Will be used check if DID is updated.

Value of CheckDIDUpdated option:

```
type ResolveDIDRetry struct {
        MaxNumber int
        SleepTime *time.Duration
}
```

Example Update

```
updateKey, updateKeyPrivateKey, err := ed25519.GenerateKey(rand.Reader)
if err != nil {
        return err
}

// this key will used for next update request
keyRetrieverImpl.nextUpdatePublicKey = updateKey
```

(continues on next page)

(continued from previous page)

```
didPublicKey, _, err := ed25519.GenerateKey(rand.Reader)
if err != nil {
        return err
}

jwk, err := jose.JWKFromKey(didPublicKey)
if err != nil {
        return err
}

vm,err:=ariesdid.NewVerificationMethodFromJWK("key1", "Ed25519VerificationKey2018", "",␣
↪jwk)
if err != nil {
        return err
}


didDoc := &ariesdid.Doc{ID: didID}

didDoc.Authentication = append(didDoc.Authentication, *ariesdid.
↪NewReferencedVerification(vm,
                ariesdid.Authentication))

didDoc.CapabilityInvocation = append(didDoc.CapabilityInvocation, *ariesdid.
↪NewReferencedVerification(vm,
                ariesdid.CapabilityInvocation))

didDoc.Service = []ariesdid.Service{
                {
                        ID:              "svc1",
                        Type:            "typeUpdated",
                        ServiceEndpoint: "http://www.example.com/",
                },
                {
                        ID:              "svc2",
                        Type:            "type",
                        ServiceEndpoint: "http://www.example.com/",
                },
}

if err := vdr.Update(didDoc); err != nil {
        return err
}

// update private key will be used to sign next update request
keyRetrieverImpl.updateKey = updateKeyPrivateKey
```

Example Recover

```
recoveryKey, recoveryKeyPrivateKey, err := ed25519.GenerateKey(rand.Reader)
if err != nil {
        return err
}

// this key will used for next recover request
keyRetriever.nextRecoveryPublicKey = recoveryKey

didDoc := &ariesdid.Doc{ID: didID}

didPublicKey, _, err := ed25519.GenerateKey(rand.Reader)
if err != nil {
        return err
}

jwk, err := jose.JWKFromKey(didPublicKey)
if err != nil {
        return err
}

vm,err:=ariesdid.NewVerificationMethodFromJWK("key1", "Ed25519VerificationKey2018", "",␣
↪jwk)
if err != nil {
        return err
}


didDoc.CapabilityInvocation = append(didDoc.CapabilityInvocation, *ariesdid.
↪NewReferencedVerification(vm,
        ariesdid.CapabilityDelegation))

didDoc.Service = []ariesdid.Service{{ID: "svc1", Type: "type", ServiceEndpoint: "http://
↪www.example.com/"}}

if err := e.vdr.Update(didDoc,
        vdrapi.WithOption(orb.RecoverOpt, true),
        vdrapi.WithOption(orb.AnchorOriginOpt, "https://orb-2.devel.trustbloc.dev/
↪services/orb")); err != nil {
        return err
}

// recover private key will be used to sign next recover request
keyRetrieverImpl.recoverKey = recoveryKeyPrivateKey
```

Deactivate

Deactivate used to deactivate Orb DID.

`func Deactivate(didID string, opts ...vdrapi.DIDMethodOption) error`

Options

Deactivate options.

OperationEndpointsOpt

This option is mandatory when domain not set. Will be used to set operation endpoint.

Example

```
if err:=vdr.Deactivate(discoverableDID);err!=nil{
 return err
}
```

3.4 System Services

3.4.1 ActivityPub

Orb implements the ActivityPub spec for server to server communication. Communication is based on posting an activity (JSON document) to the server's local outbox which then gets delivered to one or more server inboxes. HTTP signatures are used for authentication and authorization.

Outbox/Inbox

Inter-server communication is performed by posting an Activity to the Orb server's Outbox. The Outbox handler delivers the activity to one or more targets by resolving the URIs of the target inboxes (from the "to" field of the activity) and posting the activity to each of the URIs. The target Inbox authorizes the message and invokes the appropriate handler.

Outbox

Messages are published to the AMQP *orb.activity.outbox* queue and are handled asynchronously by a single instance in the Orb domain. A message contains two fields:

1) Type: One of 'broadcast', 'deliver', or 'resolve-and-deliver'
2) Activity: The posted ActivityPub activity

When an activity is posted to the Outbox, it is first validated and then a message of type, 'broadcast', is published to the AMQP *orb.activity.outbox* queue. The message is handled by a single instance in the Orb domain.

Broadcast Message Handler

The 'broadcast' message handler performs the following steps:

1) Stores the activity (which is contained in the message) to the local *outbox* database.
2) Invokes an activity handler (this handler may include additional steps, depending on the type of activity).
3) Resolves the inboxes of the URIs in the "to" field of the activity. This may involve retrieving URIs from the *followers* or *witnesses* collections. The ActivityPub service (actor) of each recipient URI is resolved via .well-known and a result containing the resolved URI (and potentially an error) is returned for each resolved URI.

Each result returned from the *Outbox Resolver* contains a URI and potentially an error. For each result that does not contain an error, a message of type, 'deliver', is published to the *orb.activity.outbox* queue with the URI from the result. For each result containing an error, a message of type, 'resolve-and-deliver', is published to the *orb.activity.outbox queue* so that the URI may be retried.

Deliver Message Handler

The 'deliver' handler posts the activity to the inbox URI (contained in the message). (Note that the appropriate HTTP signatures are added to the HTTP request.) If a transient error occurs (such as HTTP 500) then the message is NACK'ed and the 'deliver' message will be retried (according to the message redelivery mechanism).

Resolve-and-Deliver Message Handler

The 'resolve-and-deliver' handler resolves the inboxes of the URI contained in the message. Each result returned from the *Outbox Resolver* contains a URI and potentially an error. For each result that does not contain an error, a 'deliver' message is published to the *orb.activity.outbox* queue with the URI from the result. If the result contains a transient error (e.g. HTTP 500) then the message is NACK'ed and the 'resolve-and-deliver' message is retried (according to the message redelivery mechanism). If a persistent error occurs (e.g. 400) then the URI is skipped.

Inbox

The inbox is a REST endpoint in Orb which accepts activities. When an activity is received, the HTTP signature in the header of the request is verified using the public key of the actor that sent the activity. After the actor is authenticated, a message is posted to the *orb.activity.inbox* queue and is processed by one of the server instances in the domain.

The inbox handler first stores the activity in the 'inbox' database and authorizes the activity. (Each activity could have different authorization criteria which are explained in each of the activities below.) The appropriate activity handler is then invoked.

Activities

Follow

A Follow activity is posted by a server to another server so that activities may be synchronized between them. For example, domain1 posts a Follow activity to domain2 indicating that it wants notifications of objects created by domain2 (via the Create activity). Domain1 is also notified of any objects created by servers that domain2 is following via the Announce activity. When a server receives a Follow activity in its inbox, it authorizes the 'actor' (which is the originating server) in the Follow request using a configurable *Follow Authorization Policy*. If authorized, the actor is added to the list of *followers* and an Accept activity is posted to the actor's inbox. When an Accept activity is received

in the inbox, the activity is first validated against a previously posted Follow activity and then the 'actor' in the Accept activity (which was the target server in Follow) is added to the *following* collection.

If the actor posting the Follow activity is not authorized then a Reject activity is posted to the actor's inbox. A Reject of a Follow activity simply logs the fact that the request was rejected.

Undo Follow

An Undo activity is posted to 'unfollow' a server. The Undo activity is posted to the server to which the previous Follow activity was posted and the target server is removed from the *following* list. The target server handles the Undo activity by removing the originating server from the *followers* list.

Invite Witness

An Invite activity is posted to another server in order to invite that server to be a witness of Anchor Events. For example, domain1 posts an Invite activity to domain2 indicating that it wants domain2 to be a witness of Anchor Events produced by domain1 (using the *Offer* activity).

When a server receives an Invite witness activity in its inbox, it authorizes the actor (which is the originating server) in the Invite request using an *Invite Witness Authorization Policy*. If authorized, the actor is added to the *witnessing* collection and an Accept activity is posted to the actor's inbox. When an Accept activity is received in the inbox, the activity is first validated against a previously posted Invite witness activity and then the 'actor' in the Accept activity is added to the *witnesses* collection.

If the actor is not authorized, then a Reject activity is posted to the actor's inbox. A Reject of an Invite witness activity simply logs the fact that the request was rejected.

Undo Invite Witness

An Undo activity is posted to remove a server as a witness. The Undo activity is posted to the server to which the previous Invite activity was posted and the target server is removed from the *witnesses* list. The target server handles the Undo activity by removing the originating server from the *witnessing* list.

Offer/Accept

An Offer activity is posted to one or more servers that are contained in the *witnesses* collection in order to collect proofs. (The selection of witnesses is dictated by a Witness Policy which determines the minimum number of proofs required.) When a server receives an Offer activity in its inbox, the request is first authorized by ensuring that the actor of the Offer is in the *witnessing* collection. Once authorized, the Anchor Linkset which is embedded in the Offer activity is added to the ledger (VCT) and an Accept anchor activity (which contains the proof) is posted back to the originator of the offer. When an Accept anchor activity is received in the inbox, the activity is first validated against a previously posted Offer activity and then the embedded proof for the Anchor Linkset is added to the collection of existing proofs. Once a sufficient number of proofs is received (according to the Witness Policy) then a complete anchor linkset (containing all proofs) is constructed and the anchor linkset is posted to the queue, *orb.anchor_linkset*, to be processed by the Batch Writer.

If the actor of the Offer is not in the *witnessing* collection then a Reject activity is posted to the originating actor. The Reject of an Offer activity simply logs the fact that the request was rejected.

Create/Announce

A Create activity is posted by the Batch Writer to one or more servers that are contained in the *followers* collection.

When a server receives a Create activity in its inbox, the Anchor Linkset which is embedded in the Create activity is first stored to CAS and then the hashlink of the anchor linkset is posted to the *orb.anchor* queue so that it may be processed by the Observer. After the Observer processes the anchor linkset, it posts a Like activity back to the "actor" of the Create activity. The Create activity is also forwarded to the servers in the *followers* collection using the Announce activity.

When a server receives an Announce activity in its inbox the Anchor Linkset which is embedded in the Announce activity is first stored to CAS and then the hashlink of the anchor linkset is posted to the *orb.anchor* queue so that it may be processed by the Observer. After the Observer processes the anchor linkset, it posts a Like activity back to the "actor" of the Announce activity as well as to the originator of the anchor linkset.

Like

A Like activity is posted by the Observer after having processed an anchor linkset. The Like activity includes a "url" field which is a hashlink. This hashlink contains metadata of one or more locations where the anchor is replicated. (The alternate locations may be retrieved using WebFinger.) When a server receives the Like activity in its inbox, it adds the additional URIs to the *anchor link* database.

Undo Like

An Undo activity is posted to inform other servers that it no longer replicates an anchor. The Undo activity is posted to the server to which the previous Like activity was posted. The target server handles the Undo activity by removing the link from the *anchor link* database.

Authorization Policy

Authorization policies are applied when handling the Follow and Invite activities. If authorization fails then a Reject activity is posted to the originating server.

Follow Authorization Policy

An authorization policy may be configured for the *Follow* activity using the configuration parameters, follow-auth-policy. The possible values are *accept-all* and *accept-list*. If accept-all is used then all Follow requests are accepted. If accept-list is used then the actor in the Follow activity must be in the *accept-list* database.

Invite Witness Authorization Policy

An authorization policy may be configured for the *Invite Witness* activity using the configuration parameters, invite-witness-auth-policy. The possible values are *accept-all* and *accept-list*. If accept-all is used then all Invite witness requests are accepted. If accept-list is used then the actor in the Invite activity must be in the *accept-list* database.

Accept List

An accept-list is simply a database of server URLs that are authorized for a particular type of operation. The two types of operations are *follow* and *invite-witness*. If the actor URL is in the *follow* accept list then a Follow request from that server is authorized. If the actor URL is in the *invite-witness* accept list then an Invite witness request from that server is authorized.

The accept-list is managed via the REST endpoint /services/orb/acceptlist

3.4.2 Authorization

Authorization in Orb is typically performed using bearer tokens for client-to-server communication. For server-to-server communication, authentication is first performed using HTTP signatures and then each endpoint performs its own authorization.

Bearer Tokens

For endpoints that require authorization, a client must add a bearer token to the HTTP request header as follows:

```
Authorization:[Bearer mytoken]
```

The server matches the bearer token in the request header against the required token(s) for the particular endpoint. Each REST endpoint may be configured to require tokens for both read (GET) and write (POST) requests. If no token is defined for an endpoint then no authorization is performed. Multiple tokens may be defined for read and write requests. If more than one token is defined then authorization succeeds if any of the tokens is found in the request header. If a token for the request is required but not found in the request header then *HTTP signature* verification is performed.

HTTP Signatures

A common HTTP client within Orb is used for all server-to-server communications. If HTTP signatures are enabled then the HTTP client sets additional headers on the HTTP request.

Headers

The following headers are added to the request before it is sent: *Date*, *Digest*, and *Signature*. For example:

```
Date:[Thu, 17 Feb 2022 14:29:24 GMT]
Digest:[SHA-512=vXXy/lk6D+XE8fYRTL7OS7izBUn6ntk60Rn97/
↪gOSnbSxHZuaPvPTw1FW427qLqYpA0xGcXyPDQh4ujwret4aw==]
Signature:[keyId="https://orb.domain1.com/services/orb/keys/main-key",algorithm="Ed25519
↪",headers="(request-target) Date Digest",signature=
↪"8AGVZi+xaDQ2kgD4sZUd4e2c2oOIkxzou2MoSSvQv72QJeSsoLa8+qJ1A+w2xkTDHNDfBTG8T/
↪mNmtmYouv9Ag=="]
```

Date Header

The *Date* header is set to the current date/time.

Digest Header

The *Digest* header contains the hash of the request body, prepended by the algorithm. If a body was not included in the request then *Digest* will be empty.

Signature Header

The *Signature* header contains a comma-separated string of field-values (i.e. field1=value1,field2=value2,...) where the fields are defined as follows:

keyId

The value of the *keyId* field is the URI of the public key that may be used to verify the signature. The value of *keyId* must be resolvable via HTTP or another protocol.

algorithm

The *algorithm* field contains the algorithm used to sign the request. Orb uses KMS to sign the request using the *Ed25519* algorithm.

headers

The *headers* field declares the set of headers that should be signed. The value of this field is a space-separated string that contains the following set of fields:

1) **(request target)** - Includes the request method (GET, POST) and the request URI. For example: POST https://orb.domain1.com.
2) **Date** - Points to the *Date* header.
3) **Digest** - Points to the *Digest* header (this value is not included if the *Digest* header is empty).

signature

The *signature* field contains the base64-encoded signature of the headers that are declared in the *headers* field.

Signature Verification

On receiving a request, the Orb server retrieves the URI specified in the value of the *keyId* field from the *Signature* header and then sends a request to this URI. The response of the request is in the following format:

```
{
  "id": "https://orb.domain1.com/services/orb/keys/main-key",
  "owner": "https://orb.domain1.com/services/orb",
  "publicKeyPem": "-----BEGIN PUBLIC KEY-----\
↪nMCowBQYDK2VwAyEArK46BYVBHCM1Th+kKCFzabVmbTmTXRL5SwH+m2WvKKY=\n-----END PUBLIC KEY-----
↪\n"
}
```

The value in the *signature* field is then verified using the public key. If not valid then an *HTTP 401 Unauthorized* response is sent to the client. If valid, the *owner* of the key is retrieved. (The owner is an ActivityPub service (actor)). Following is a sample response:

```
{
  "@context": [
    "https://www.w3.org/ns/activitystreams",
    "https://w3id.org/security/v1",
    "https://w3id.org/activityanchors/v1"
  ],
  "followers": "https://orb.domain1.com/services/orb/followers",
  "following": "https://orb.domain1.com/services/orb/following",
  "id": "https://orb.domain1.com/services/orb",
  "inbox": "https://orb.domain1.com/services/orb/inbox",
  "liked": "https://orb.domain1.com/services/orb/liked",
  "likes": "https://orb.domain1.com/services/orb/likes",
  "outbox": "https://orb.domain1.com/services/orb/outbox",
  "publicKey": {
    "id": "https://orb.domain1.com/services/orb/keys/main-key",
    "owner": "https://orb.domain1.com/services/orb",
    "publicKeyPem": "-----BEGIN PUBLIC KEY-----\
↪nMCowBQYDK2VwAyEArK46BYVBHCM1Th+kKCFzabVmbTmTXRL5SwH+m2WvKKY=\n-----END PUBLIC KEY-----
↪\n"
  },
  "shares": "https://orb.domain1.com/services/orb/shares",
  "type": "Service",
  "witnesses": "https://orb.domain1.com/services/orb/witnesses",
  "witnessing": "https://orb.domain1.com/services/orb/witnessing"
}
```

The value of the *publicKey.id* field in the ActivityPub service (actor) is then validated against the ID of the public key to ensure that they match. If they don't match then an *HTTP 401 Unauthorized* response is sent to the client. If they *do* match then authentication has succeeded and the request, along with the actor, is forwarded to the appropriate handler. (The actor may be used by the handler to perform authorization.)

Note that public keys and actors are cached (with an expiry) so that remote calls aren't required each time a signature verification is performed.

3.4.3 Batch Writer

A Sidetree operation (Create, Update, Recover and Deactivate) is posted by a client over the operations REST endpoint. The operation is validated according to Sidetree DID Operations and added to the *Operation Queue*.

The Batch Writer:

1) Drains operations from the Operation Queue
2) Batches multiple Sidetree operations together into Sidetree batch files as per Sidetree file structure spec
3) Stores the Sidetree batch files into Content Addressable Storage (CAS)
4) Anchors a reference to the main Sidetree batch file (core index file) on the anchoring system as a Sidetree transaction

The number of operations that can be stored in the Sidetree batch files is limited by Sidetree protocol parameter MAX_OPERATION_COUNT. The Batch Writer cuts Sidetree batches if the number of operations in the batch writer queue reaches MAX_OPERATION_COUNT or if the batch writer reaches the batch writer timeout batch-writer-timeout.

Operation Queue

The *Operation Queue* in Orb is an implementation of the sidetree-core-go operation queue interface. The implemented functions are:

1) **Add**: Adds an operation to the queue
2) **Remove**: Returns and removes up to N operations from the queue
3) **Peek**: Returns (but does not remove) up to N operations from the queue
4) **Len**: Returns the current length of the queue

Orb's implementation of the *Operation Queue* is backed by an AMQP message broker and a database. Each Orb instance of a domain stores a *task* entry in the *op-queue* database on startup. The *task* entry contains an *ID* and an *update time*. The ID is simply the unique ID of the Orb instance (which is a GUID generated by the Task Manager on startup) and the *update time* contains the timestamp of when the task entry was last updated. (This timestamp is used to check the aliveness of the Orb instance.)

Add Operation

A Sidetree operation is posted by a client to the operations REST endpoint (which is exposed by the sidetree-core-go library). After the operation is validated, the *Add* function of Orb's *Operation Queue* is invoked. The *Operation Queue* then publishes a message containing the operation to the AMQP *orb.operation* queue. Each Orb instance has a pool of subscribers for the *orb.operation* queue. The number of subscribers in the pool is determined by startup parameter op-queue-pool. One of the subscribers on an Orb instance handles the message by adding the operation to the *op-queue* database and also to an in-memory queue. Operations are stored to the database for recovery purposes, i.e. if the Orb instance goes down then another instance will repost the operations. (See *Recovery* for details.) Each database entry contains the contents of the operation and is also tagged (indexed) by:

1) **Task ID**: Associates an operation with a specific *task* entry (as described above)
2) **Expiration Time**: Tells the Database Expiry service when this entry may be deleted. This value is set to (*current time*) + (batch writer timeout) + (1 minute).

Remove Operations

The sidetree-core-go library queries the *Operation Queue* to see if there are enough operations to cut a batch (according to the Sidetree protocol parameter MAX_OPERATION_COUNT, or if the batch has timed out (according to startup parameter batch-writer-timeout). When the batch is cut then the sidetree-core-go library calls the *Remove* function on the *Operation Queue* to remove up to N operations from the queue. The *Remove* function is quasi-transactional such that it returns an *Ack* and a *Nack* function along with the operations.

Ack Function

When the sidetree-core-go library has successfully processed the operations then the *Ack* function is called. The *Ack* function deletes the removed operations from the *op-queue* database.

Nack Function

If the sidetree-core-go library has failed to successfully process the operations then the *Nack* function is called. The *Nack* function reposts the operations to the AMQP *orb.operation* queue so they may be retried (potentially by another server instance) and the operations are deleted from the database. Each operation message is reposted with a delay to give the server a chance to recover from whatever caused processing to fail in the first place. The delay is calculated according to the number of failed retries along with parameters, mq-redelivery-initial-interval, mq-redelivery-max-interval, and mq-redelivery-multiplier. The *retries* header value is also set on the message. The value is first incremented before it is reposted. Once the maximum number of retries for an operation has been reached, the operation is discarded.

Recovery

An Orb server may go down with pending operations in the queue. The *Operation Queue Monitor Task* is registered with the Task Manager on startup to periodically run on one server instance in the domain. The period is specified by startup parameter op-queue-task-monitor-interval. This task monitors the operation queue tasks of other servers to ensure that if a server goes down then the operations associated with that server are reposted to the AMQP *orb.operation* queue.

Each Orb instance periodically (also using the period specified by op-queue-task-monitor-interval) updates the *update time* of its own *task* entry in the database in order to indicate to other servers that the instance is still alive.

When the monitor task runs, it queries the *op-queue* database for all *task* entries (excluding its own) and checks the *update time* of the entry. If the update time is older than the expiration time configured with startup parameter op-queue-task-expiration then the server that owns the task is considered to be down. At this point, the *op-queue* database is queried for the operations associated with the task and each operation is reposted to the queue. (As described in the *section* above, each operation message reposted to the *orb.operation* queue has a *retries* header value which is incremented before it is reposted. Once the maximum number of retries for an operation has been reached, the operation is discarded.) All operations associated with this *task* are then deleted from the database and the *task* entry itself is also deleted from the database. (The *task* entry is deleted from the database since, when the dead server comes back online, it will generate a new task ID.)

Anchor Writer

When a batch of operations is cut from the *operation queue*, the sidetree-core-go library creates a Sidetree anchor object and invokes *WriteAnchor*. The Orb implementation of *WriteAnchor* performs the following steps:

1) Retrieves previous anchors for all DIDs in the batch
2) Resolves the witnesses for the batch
3) Creates an Anchor Linkset containing the operations
4) Posts an Offer activity (containing the anchor linkset) to each of the witnesses

Witness Proof Handler

A witness accepts an Offer of an anchor linkset by posting an Accept activity to the inbox. The proof is extracted from the anchor linkset and the proof handler is invoked.

The current status of the anchor is retrieved from the *anchor status* database. If the status of the anchor is still not *complete* then the provided proof from the Accept activity is added to the existing set of proofs. The witness policy is then evaluated in order to determine if the anchor has a sufficient number of proofs. If the witness policy is not satisfied then nothing else is done, otherwise:

1) The status of the anchor is marked as *complete*
2) The anchor linkset is updated with all witness proofs
3) The anchor linkset is published to the *orb.anchor_linkset* queue so that it may be processed by the
4) *Anchor Linkset Handler*

Anchor Linkset Handler

On startup, the Anchor Linkset Handler subscribes to the *orb.anchor_linkset* queue in order to receive witnessed anchor linksets. Upon receiving an anchor linkset from the queue:

1) The verifiable credential is extracted from the anchor linkset and saved to the verifiable credential database
2) The anchor linkset is saved to the Content Addressable Storage (CAS)
3) The anchor linkset is published to the *orb.anchor* queue so that it may be processed by the Observer
4) A Create activity (containing the anchor linkset) is posted to all followers

3.4.4 Observer

The Observer subscribes to the *orb.anchor* queue and handles witnessed anchor linksets. Upon receiving an anchor linkset, the Observer:

1) Loads the anchor linkset from the Content Addressable Storage (CAS)
2) Validates the witness signatures
3) Retrieves Sidetree batch files from CAS
4) Validates Sidetree batch files:

- Ensures that the number of operations does not exceed the maximum allowed limit
- Ensures that the size of each operation does not exceed the maximum allowed limit
- Ensures the batch meets proof-of-work requirements

5) Stores each DID operation into the *Operation* store

6) Deletes the corresponding unpublished DID operations from the *Unpublished Operation* store

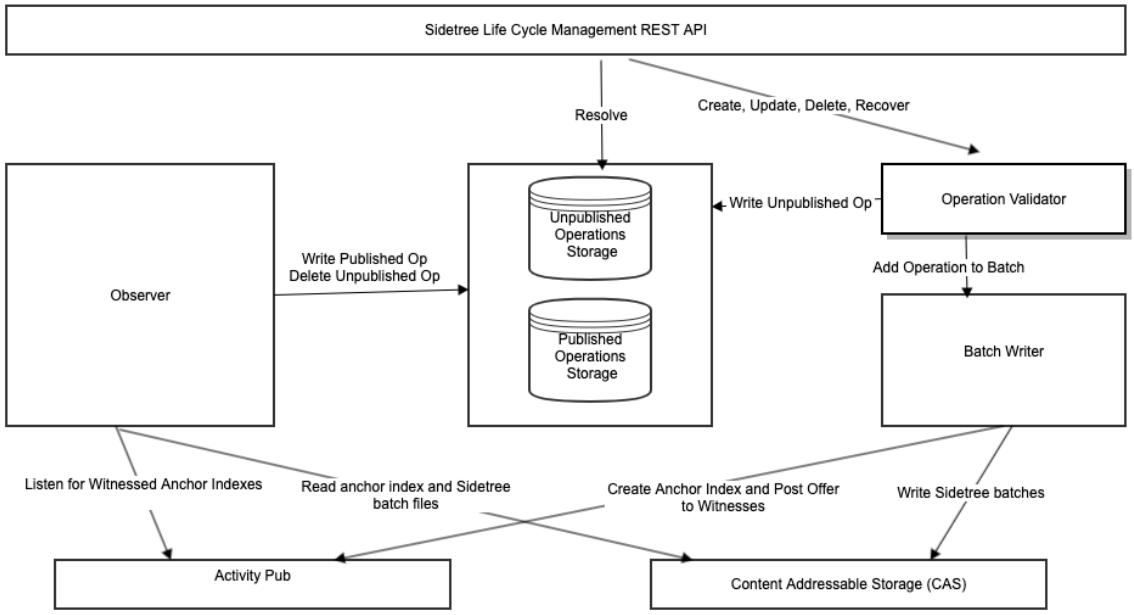
3.4.5 Sidetree

Protocol

Sidetree is a protocol for creating scalable Decentralized Identifier networks that can run atop any existing decentralized anchoring system (e.g. Bitcoin, Ethereum, distributed ledgers, witness-based approaches). The protocol allows users to create globally unique, user-controlled identifiers and manage their associated PKI metadata, all without the need for centralized authorities or trusted third parties. The syntax of the identifier and accompanying data model used by the protocol is conformant with the W3C Decentralized Identifiers specification. Implementations of the protocol can be codified as their own distinct DID Methods and registered in the W3C DID Method Registry.

See the latest spec for the full Sidetree specification.

Sidetree Interactions



Orb node will validate each Sidetree operation as per specification. Valid operation will then be added to the batch writer queue and unpublished operations store. Batch writer will then batch multiple Sidetree operations together and store them in Sidetree batch files as per Sidetree file structure spec.

Next, Sidetree batch file information will be stored into anchor index and anchor index will be witnessed as per witness policy. Observer will process witnessed anchor index and Sidetree batches as per Sidetree transaction operation

processing and store DID operations into operations store. It will delete observed DID operation from unpublished operation store. DID will be resolved from stored DID operations.

DID Operations

Sidetree supports Create, Update, Recover and Deactivate(CRUD) operations. Check our Sidetree did operations and Sidetree API spec to learn how to construct JSON for Sidetree CRUD operations.

DID Resolution

Upon invocation of resolution, DID resolver will first retrieve all published (observed) and unpublished operations for the DID Unique Suffix of the DID URI being resolved.

DID document processing starts by iterating over all Create operations (ordered by anchoring time) and applying first valid Create operation. Next step in the processing is applying Recover and Deactivate operations based on previous operation Next Recovery Commitment. Once Recover and Deactivate operations have been applied to the document (or if no Recovery and Deactivate operations are present) the resolver will proceed to Update operation processing based on previous operation Next Update Commitment only if the Next Update Commitment value is present and no Deactivate operations were successfully processed earlier.

The resolver will check each operation for validity (including validating signature for update, recover and deactivate operations) before it applies that operation patch and commitments to DID Document.

See Sidetree resolution spec for more details about resolution.

REST Endpoints

Sidetree resolution endpoint:

/sidetree/v1/identifiers

Sidetree operations endpoint:

/sidetree/v1/operations

See the API spec for the full API specification to interact with a Sidetree node.

Versioning

Sidetree References

See the latest spec for the full Sidetree specification.

See the API spec for the full API specification to interact with a Sidetree node.

See the reference implementation document for description of the reference implementation.

See the reference implementation for Sidetree reference implementation.

Operation/Resolution Caching

Orb node provides two unique caching features for resolving DIDs:

1. resolve DIDs using unpublished operations
2. resolve DIDs using published/unpublished operations from anchor origin

Operation/Resolution Caching Parameters

The following section enumerates the startup parameters that have to be configured in order to enable operation caching and resolution from anchor origin domain.

Parameter enable-unpublished-operation-store should be set to true to enable un-published operation store. If set to true Orb node will upon validating Sidetree operation insert this operation into unpublished operations store and use it immediately for DID resolution. Hence, Orb clients will be able to immediately see their change (without waiting for anchoring/observing process to complete). This unpublished Sidetree operation will be removed from unpublished operation store when Orb node is done observing Sidetree operation. At this point Sidetree operation has been stored into published operation store and will be used for DID resolution.

Parameter unpublished-operation-store-operation-types determines the operation types that will be caches. Default value is "create,update" which enables storing unpublished 'create' and 'update' operations into unpublished store and using those unpublished 'create' and 'update' operations for resolving document.

Parameter resolve-from-anchor-origin should be set to "true" in order to resolve DID from anchor origin. Orb node will resolve DID document locally, determine DID's anchor origin and then resolve DID document from anchor origin Orb node. If anchor origin Orb node has additional published/unpublished operations then Orb node will add those operations for DID resolution. This feature allows non origin servers to immediately resolve updated DIDs.

Parameter include-unpublished-operations-in-metadata should be set to "true" to include unpublished operations in metadata. This setting is required on anchor origin node if non anchor origin nodes wish to resolve from anchor origin.

Parameter include-published-operations-in-metadata should be set to "true" to include published operations in metadata. This setting is required on anchor origin node if non anchor origin nodes wish to resolve from anchor origin.

Operation Update Parameters

In order to ensure consistent operation caching, Orb client will normally issue document update against anchor origin Orb node. However, if Orb node receives an update operation for different anchor origin it will resolve DID against anchor origin Orb node and make sure it has processed the latest operations from anchor origin Orb node before accepting an update operation.

Parameter verify-latest-from-anchor-origin should be set to "true" to verify if Orb node has already processed the latest operations from anchor origin before accepting an update operation. If Orb node is behind in processing DID operations (comparing to anchor origin) then Sidetree operation will be rejected. This feature also guards against DID document branching.

3.4.6 Content Addressable Storage (CAS)

Content Addressable Storage (CAS) is a persistent storage of content where the primary key is the hash of the content. This ensures that content for a given key is immutable, i.e. if the content changes then the primary key changes.

Orb supports two types of CAS: local storage (in a database) and IPFS. The type of CAS storage that an Orb instance uses is dictated by the startup parameter, cas-type. Additionally, if cas-type is set to *local* and the startup parameter, replicate-local-cas-writes-in-ipfs, is set to *true*, then CAS data is stored in the local database and also replicated to IPFS.

Local

A local CAS storage is simply a database that stores content using the sha-256 multihash of the content as the primary key. The content is *not* automatically replicated (as is the case with IPFS). A *local* CAS storage relies on ActivityPub activities (Create and Announce) to replicate data.

IPFS

If the cas-type is set to *ipfs* then an additional startup parameter, ipfs-url is required to point to the IPFS node. The IPFS node must have permissions to read and write content. This node replicates the content to other IPFS nodes in the network.

The version of the content ID (CID) that is used as the key for content retrieval is specified with startup parameter cid-version.

3.4.7 AMQP Publisher/Subscriber

The publisher/subscriber subsystem in Orb is configurable to either use an AMQP message broker or an in-memory implementation. (The in-memory implementation should only be used during development and for demos.)

The AMQP URL is specified by the startup parameter mq-url. If this parameter is not set then the in-memory implementation is used.

Publisher and Subscriber

A subscriber is a handler for a message posted to a queue. On startup, each Orb instance subscribes to various queues. The queues are global to the domain, i.e. each Orb instance subscribes to the same queues.

A publisher posts messages to the queues. The posted message is handled by one (and only one) of the subscribers in one of the Orb instances. If the handler replies with an *ack*, then the message is considered to be processed.

It is up to the AMQP implementation to direct messages to subscribers using a load-balancing algorithm.

Message Redelivery

When a message is published to a queue, one of the subscribers in the Orb domain handles the message. If a processing error occurs (such as the DB is temporarily unavailable) then the handler replies with a *nack*. In this case the message is sent to a *orb.redelivery* queue so that it may be retried at a later time.

The *orb.redelivery* queue is configured as the *dead-letter-queue* for all queues in Orb. When a message is rejected (*nacked*) by a subscriber, it is automatically sent to the *orb.redelivery* queue. The first time a message is rejected, the redelivery handler immediately redelivers the message to the original destination queue. If the message is rejected again, an *expiration* header value is set on the message, and the message is posted to a *orb.wait* queue. The expiration value is calculated by a backoff algorithm using the following parameters:

1) mq-redelivery-initial-interval
2) mq-redelivery-multiplier
3) mq-redelivery-max-interval

The backoff algorithm increases the expiration with each redelivery attempt. For example, if the initial interval is set to 2s and the multiplier is set to 1.5 then the expiration is set 3s. The next time a redelivery of the message occurs, the expiration will be set to 4.5s. Expiration time is limited by parameter mq-redelivery-max-interval.

The *orb.wait* queue has no subscribers, so the message sits there until it expires. The *orb.redelivery* queue is also configured as the *dead-letter-queue* for the *orb.wait* queue, so when the message expires it is automatically sent back to the *orb.redelivery* queue and the redelivery handler processes the message again.

The redelivery handler looks at the *reason* field in the message header. If *reason* is set to 'expired' then the message is posted to the original destination queue, otherwise (if reason is 'rejected') it is posted to the *orb.wait* queue with an even greater expiration value. This process repeats until the maximum number of redelivery attempts has been reached, at which point redelivery for the message is aborted.

Publisher Pool

The Publisher publishes messages over an AMQP channel to an AMQP server. A single publisher channel publishes requests synchronously and therefore (for performance reasons) a pool of channels is used so that requests may be published concurrently. The channels in a pool are opened at startup over a single connection, and are reused over the lifetime of the server (since there is a performance penalty for creating and closing channels). The size of the channel pool is specified by parameter mq-publisher-channel-pool-size. If the value of this paramter is zero then a publisher pool is not used and a channel is opened/closed for each publish request.

An AMQP server may have a limit to the number of channels that can be opened for a single connection. This limit is specified by parameter mq-max-connection-channels. If the value of mq-publisher-channel-pool-size is greater than the value of mq-max-connection-channels then multiple publisher pools are created (each with its own dedicated connection) and the requests are balanced across the pools.

Subscriber Pool

Each AMQP subscription is handled synchronously. If the handler takes a long time then subsequent messages in the queue need to wait until the previous message is processed. A subscriber pool may be configured for a given queue such that multiple subscribers concurrently process messages from the same queue. This setting is available for the following queues:

1) op-queue-pool
2) mq-observer-pool

Typically, all subscriber channels are created on the same AMQP connection, although an AMQP server may have a limit to the number of channels that can be opened for a single connection. Therefore, the limit for the number of channels for a single connection is specified by parameter mq-max-connection-channels. If the size of the subscriber pool reaches this limit then a new connection is automatically opened for any new subscriber channel.

Queues

When an Orb instance starts up it creates a number of queues (if they aren't already created) and subscribes to receive messages from these queues. Following is a description of the queues:

orb.activity.outbox

ActivityPub activities posted to the outbox are published to the orb.activity.outbox queue which is consumed by the outbox handler.

orb.activity.inbox

ActivityPub activities posted to the inbox are published to the orb.activity.inbox queue which is consumed by the inbox handler.

orb.operation

Sidetree DID operations posted to the operations endpoint are published to the orb.operation queue which is consumed by the operation queue handler.

orb.anchor_linkset

The witness proof handler publishes anchor linksets to the orb.anchor_linkset queue which is consumed by the anchor linkset handler.

orb.anchor

The anchor linkset handler publishes messages to the orb.anchor queue which is consumed by the observer.

orb.did

DID messages are published to the orb.did queue which is consumed by the observer.

orb.redelivery

A message that has been NACK'ed is published to the orb.redelivery queue so that it may be redelivered.

orb.wait

A message that has been NACK'ed and has a backoff time is published to the orb.wait queue. The message sits in this queue for the duration of the specified backoff time, then it is automatically sent to the orb.redeivery queue for redelivery.

3.4.8 Task Manager

The Task Manager is an Orb service that periodically runs tasks on one Orb instance in the domain. A task is registered on startup with a unique ID, a run interval, and a function to invoke at the registered interval. The Task Manager stores a *permit* for each task in the *orb-config* database. The permit contains:

1) **Task ID**: The unique ID of the task
2) **Permit Holder**: The unique ID of the Orb instance that is currently responsible for running the task (a GUID that's generated on startup)
3) **Status**: Either *idle* or *running*
4) **Update Time**: The last time the permit was updated (used to check the last time the task was run and the *aliveness* of the permit holder)

The Task Manager on each Orb instance periodically checks the permit for each task. Different actions are taken depending on whether the Orb instance holds the permit, as described below.

This Orb instance holds the permit

When it is determined that this instance is the permit holder for the task, the permit is checked for its status: either *idle* or *running*.

Idle status

If the status of the task is *idle* then a check is made to see if it is time to run the task using the registered interval for the task and the last update time. If it is time to run the task then:

1) The status of the permit is set to *running*
2) The last update time is set to the current time
3) The task is started

Running status

If the status of the task is *running* then the permit is updated with the current time so that other instances still see that this instance is alive, otherwise (for long-running tasks) other instances may think that this instance is down and attempt to acquire the permit.

Another Orb instance holds the permit

When it is determined that this instance is NOT the permit holder for the task, the health of the permit holder is checked. If the time since the permit was updated is greater than the Task Manager's check interval plus the interval of the task itself, then it is assumed that the permit holder is down and this instance attempts to take over the permit. There is a window in which multiple instances may attempt to take over the permit at the same time, in which case multiple instances may run the task concurrently. This should only happen once during a *takeover* event, since the last instance to update the permit is responsible for all future runs.

Tasks

Tasks registered with the Task Manager must be tolerant of the fact that multiple server instances may run the task concurrently. This may happen when a permit holder goes down and other instances attempt to take over the permit (as described in the section above). Following is a description of the registered tasks:

Database Expiry

The database expiry task periodically deletes expired data from multiple databases. The database expiry service allows multiple databases to be registered for expired data checks. A database is registered with a database name and a tag that contains the expiration time of the document. The expiry task queries for all documents from the registered databases where the expiry time has been reached, and deletes these documents. The scheduled run period is set with the startup parameter data-expiry-check-interval.

Activity Sync

The Activity Sync task periodically synchronizes anchor activities from the inboxes and outboxes of its followers and the domains that it's following. The scheduled period is set with the startup parameter sync-interval.

Anchor Status Monitor

This task monitors the status of an anchor event that is waiting for proofs from witnesses. The task checks if enough proofs have been received for an anchor event and, if not, a new set of witnesses is solicited for proofs. The scheduled period for this task is set with parameter, anchor-status-monitoring-interval.

VCT Monitor

The VCT monitor task checks if a proof that was returned by a witness (in an Accept activity) was actually added to the witness's ledger. The scheduled period for this task is set with parameter, vct-monitoring-interval.

Operation Queue Monitor

The operation queue monitor task monitors the operation queue to ensure that if a server instance goes down then another instance processes the queued operations (see the operation queue recovery page for details). The scheduled period is set with parameter, task-manager-check-interval.

3.4.9 Onboarding and Recovery

Onboarding and recovery of a domain are closely related. Onboarding takes place when a new domain comes online. This means that it starts to follow other domains, asks other domains to be its witness, and potentially other domains will want to follow it and ask it to be a witness. Recovery is the process that takes place after a domain has been offline for a while and needs to catch up with the activities posted by the domains it's following while it was down. Recovery could also mean that a domain's database had to be restored from a backup and it needs to recover data that was lost and also catch up with the missed activities from the domains that it is following.

Activity Sync Task

When an Orb server starts up, an *activity sync* task is registered with the Task Manager. This task periodically synchronizes anchor activities (Create and Announce) with the domains that the local domain is following and also processes any missing anchor activities (Creates that were posted to the domains followers).

The *activity sync* task is run on one server instance in a domain. It first reads from the Inbox of each of its followers to ensure that the Create activities that it had posted to its followers are actually stored locally. If not, it pulls the data from its followers and processes the anchor event. The task then reads the Outbox of the domains that it is following and processes all Create and Announce activities.

When processing an activity, the timestamp of when the activity was published is used to determine the *age* of the activity. The activity is not processed unless its age has reached the configured minimum activity age. This is done to minimize the possibility of both the Inbox and the activity sync task concurrently processing the anchor event. (Even though the system is tolerant of this situation, it still has an impact on performance.)

Once processing has finished, the *activity-sync* database is updated with the last page and index of each domain that it processed so that when the task runs again, processing starts from where it left off.

3.4.10 Databases

Permanent Data

The following databases store data permanently, i.e. the data (for now) is never deleted.

activity

The *activity* database stores ActivityPub activities that are posted to the outbox or received in the inbox.

activity-ref

The *activity-ref* database stores references to the *activity* database using a *RefType* tag so that a query may be performed for activities referenced by a certain type. For example, setting the *RefType* tag to OUTBOX ensures that the activity is included in the result set for a query of activities in the outbox.

Valid values for *RefType* are:

1) INBOX
2) OUTBOX
3) PUBLIC_OUTBOX
4) FOLLOWER
5) FOLLOWING
6) WITNESS
7) WITNESSING
8) LIKE
9) LIKED
10) SHARE
11) ANCHOR_LINKSET

anchor-ref

The *anchor_ref* database contains the hashlinks (with metadata) of where an anchor (tagged with *anchorHash*) may be resolved. This includes the local domain, remote domains, and IPFS.

cas

The *cas* database stores content addressable objects. This database is only used if Orb is configured with the local CAS type.

didanchor

The *didanchor* database stores the latest anchor hash of a DID suffix.

ldcontexts

The *ldcontexts* database stores JSON linked-data (JSON-LD) contexts.

log-monitor

The *log-monitor* database is used to manage the list of logs that domain is following.

log-entry

The *log-entry* database is used to store the entries of observed logs.

operation

The *operation* database stores Sidetree operations.

orb-config

The *orb-config* database stores configuration data.

remoteproviders

TBD

verifiable

The *verifiable* database store verifiable credentials.

Temporary Data

The following databases (for the most part) contain temporary data that is used only during processing of a batch of operations. The data is deleted after the batch has been processed.

anchor-link

The *anchor-link* database stores anchor links (of an anchor Linkset) that have not yet been witnessed. After a sufficient number of proofs have been received for the anchor (according to witness policy) the anchor link is deleted.

anchor-status

The *anchor-status* database stores the status of an anchor while it is waiting for witnesses. The status is either *in-process* or *completed*. After a sufficient number of proofs have been received for the anchor (according to witness policy) the anchor entry is deleted.

activity-sync

The *activity-sync* database stores the page number and index of the last activity that was synchronized for each of the domains that are followed. This information is used by the Activity Sync task.

proof-monitor

The *proof-monitor* database keeps track of the proofs that were received by other domains and ensures that the proofs were added to their VCTs.

operation-queue

The *operation-queue* database contains the operations posted to the operation queue. Operations are deleted after they have been processed.

unpublished-operation

The *unpublished-operation* database contains the operations that were posted by a client via the operations endpoint but have not yet been anchored. Operations are deleted from this database after they have been anchored.

witness

The *witness* database stores the URIs of the domains that were asked for proofs for a given anchor. The witness URIs are deleted from this database after the anchor has been processed.

public-key

The *public-key* database stores the public keys of witnesses that are used to verify the proofs in anchor credentials signed by the witnesses. When the Observer verifies a proof, it first looks in this database for a public key. If the public key is not found then it is retrieved from the witness and then stored in this database.

3.4.11 Witness Policy

An administrator can define and configure witness policy per domain. If configured the witness policy is stored into configuration database "orb-config" under "witness-policy" key. The witness policy is cached on the server with periodic cache updates from the database. Default witness policy cache expiry period has been set to 30 seconds.

Witness Policy Rules

OutOf

OutOf rule defines minimum number of witnesses required to provide proof in order to satisfy witness policy.

Syntax: OutOf(minimumWitnesses,witnessType)

First parameter must be zero or positive integer and it denotes minimum number of witnesses required to provide proof in order to satisfy witness policy.

Second parameter denotes type of witness. Supported witness types are batch and system.

Example: OutOf(2,system)

This rule means that proofs from at least 2 system witnesses are required in order to satisfy witness policy.

MinPercent

MinPercent rule defines minimum percent of witnesses required to provide proof in order to satisfy witness policy.

Syntax: MinPercent(minimumWitnessesPercent,witnessType)

First parameter must be an integer between 0 and 100 and it denotes minimum percent of witnesses required to provide proof in order to satisfy witness policy.

Second parameter denotes type of witness. Supported witness types are batch and system.

Example: MinPercent(50,batch)

This rule means that proofs from at least 50% of batch witnesses are required in order to satisfy witness policy.

LogRequired

LogRequired rule means that all witnesses have to be configured with supported witness log.

Syntax: LogRequired

Combining Rules

Rules can be combined by using operators. Supported operators: AND, OR.

Example: MinPercent(50,batch) AND MinPercent(50,system)

This rule means that proofs from at least 50% of batch witnesses and proofs from at least 50% of system witnesses are required in order to satisfy witness policy.

Additional examples: MinPercent(50,batch) OR MinPercent(50,system) OutOf(3,system) AND OutOf(1,batch) LogRequired

Default Policy

If the witness policy has not been configured the system will default to 100% batch and 100% system witnesses policy.

Configuring Witness Policy

Witness policy can be configured by posting witness configuration rule to domain endpoint "/policy".

3.4.12 Key Management

Orb uses two crypto keys to do the following operations:

- Sign HTTP requests between orb servers.
- Sign VC.

Establish orb key

There are 2 ways to establish the orb key:

- Import a pre-existing private key and ID into KMS at Orb startup via Orb configuration
- Prior to orb startup, create a new key within KMS and establish orb configuration

Add pre-existing private key to orb configuration

When the Orb server starts, it will import the private key and ID into KMS sever.

Steps to create VC sign key

- Create ed25519 private key.
- Configure ORB_VC_SIGN_PRIVATE_KEYS with value of ed25519 private key as base64.

Example VC sign key

```
ORB_KMS_ENDPOINT=https://orb.kms
ORB_VC_SIGN_PRIVATE_KEYS=orbkey1=9kRTh70Ut0MKPeHY3Gdv/
↪pi8SACx6dFjaEiIHf7JDugPpXBnCHVvRbgdzYbWfCGsXdvh/Zct+AldKG4bExjHXg
ORB_VC_SIGN_ACTIVE_KEY_ID=orbkey1
```

Steps to create HTTP sign key

- Create ed25519 private key.
- Configure ORB_PRIVATE_KEY with value of ed25519 private key as base64.

Example HTTP sign key

```
ORB_KMS_ENDPOINT=https://orb.kms
ORB_HTTP_SIGN_PRIVATE_KEY=orbkey1=9kRTh70Ut0MKPeHY3Gdv/
↪pi8SACx6dFjaEiIHf7JDugPpXBnCHVvRbgdzYbWfCGsXdvh/Zct+AldKG4bExjHXg
ORB_HTTP_SIGN_ACTIVE_KEY_ID=orbkey1
```

Create private key in KMS and configure orb to use

Key need to be created in KMS before orb server starting.

Steps to create VC sign key

- Create KMS keystore using KMS cli.
- Create ed25519 key in KMS using KMS cli.
- Configure ORB_VC_SIGN_KEYS_ID with key id.

Example VC sign key

```
ORB_KMS_STORE_ENDPOINT=https://orb.kms/keystore
ORB_VC_SIGN_KEYS_ID=orbkey1
ORB_VC_SIGN_ACTIVE_KEY_ID=orbkey1
```

Steps to create HTTP sign key

- Create KMS keystore using KMS cli.
- Create ed25519 key in KMS using KMS cli.
- Configure ORB_HTTP_SIGN_ACTIVE_KEY_ID with key id.

Example HTTP sign key

```
ORB_KMS_STORE_ENDPOINT=https://orb.kms/keystore
ORB_HTTP_SIGN_ACTIVE_KEY_ID=orbkey1
```

Storing

The keys will be managed and stored in KMS please refer to this doc for more details. Orb is using Scenario 1 in kms doc.

Rotation

There is two ways to rotate orb key:

- Import a pre-existing private key and id into KMS at Orb startup via Orb configuration
- Prior to orb startup, create a new key within KMS and establish orb configuration

Add pre-existing private key to orb configuration

When you need to rotate the key just add the new private key to orb configuration.

Steps to rotate VC sign key

- Create new ed25519 private key.
- Configure ORB_VC_SIGN_PRIVATE_KEYS with new value of ed25519 private key as base64.
- Change the active key id to new key id.

Example VC sign key

```
ORB_KMS_ENDPOINT=https://orb.kms
ORB_VC_SIGN_PRIVATE_KEYS=orbkey1=9kRTh70Ut0MKPeHY3Gdv/
↪pi8SACx6dFjaEiIHf7JDugPpXBnCHVvRbgdzYbWfCGsXdvh/Zct+AldKG4bExjHXg,
↪orbkey2=bwpFhQXFhhPQCkAt3fmj9t05hnuwVqiUkBjaXV9QBeisrjoFhUEcIzVOH6QoIXNptWZtOZNdEvlLAf6bZa8opg
ORB_VC_SIGN_ACTIVE_KEY_ID=orbkey2
```

Steps to rotate HTTP sign key

- Create new ed25519 private key.
- Replace old key in ORB_HTTP_SIGN_PRIVATE_KEY with new value of ed25519 private key as base64.
- Change the active key id to new key id.

Example HTTP sign key

```
ORB_KMS_ENDPOINT=https://orb.kms
ORB_HTTP_SIGN_PRIVATE_
↪KEYS=orbkey2=bwpFhQXFhhPQCkAt3fmj9t05hnuwVqiUkBjaXV9QBeisrjoFhUEcIzVOH6QoIXNptWZtOZNdEvlLAf6bZa8opg
ORB_HTTP_SIGN_ACTIVE_KEY_ID=orbkey2
```

Create private key in KMS and configure orb to use

When you need to rotate the key just create new key in KMS and add key id to orb configuration.

Steps to rotate VC sign key

- Create new ed25519 key in KMS using KMS cli.
- Configure ORB_VC_SIGN_KEYS_ID with key id.
- Change the active key id to new key id.

Example VC sign key

```
ORB_KMS_STORE_ENDPOINT=https://orb.kms/keystore
ORB_VC_SIGN_KEYS_ID=orbkey1,orbkey2
ORB_VC_SIGN_ACTIVE_KEY_ID=orbkey2
```

Steps to rotate HTTP sign key

- Create new ed25519 key in KMS using KMS cli.
- Change the active key id to new key id.

Example HTTP sign key

```
ORB_KMS_STORE_ENDPOINT=https://orb.kms/keystore
ORB_HTTP_SIGN_ACTIVE_KEY_ID=orbkey2
```

Distribute

The key will be managed and stored in KMS please refer to this doc for more details. Orb is using Scenario 1 in kms doc.

Impact of loss

- Http signatures it's short-lived no impact of loss.
- Can't verify VC signed before loss.

Impact of compromise

- TBD

3.4.13 Metrics

An Orb server records performance metrics at each subsystem if the startup parameter metrics-provider-name is set. Below are the metrics defined at each subsystem.

ActivityPub

The ActivityPub subsystem deals with server to server communications.

activitypub_outbox_post_seconds

The time (in seconds) that it takes to post a message to the outbox.

activitypub_outbox_resolve_inboxes_seconds

The time (in seconds) that it takes to resolve the inboxes of the destination URLs when posting to the outbox.

activitypub_inbox_handler_seconds

The time (in seconds) that it takes to handle an activity posted to the inbox.

activitypub_outbox_count

The number of activities posted to the outbox.

AnchorEvent

The AnchorEvent subsystem is responsible for gathering proofs for an AnchorEvent from multiple witnesses.

anchor_write_seconds

The time (in seconds) that it takes to write an anchor credential and post an 'Offer' activity.

anchor_witness_seconds

The time (in seconds) that it takes for a verifiable credential to gather proofs from all required witnesses (according to witness policy). The start time is when the verifiable credential is issued and the end time is the time that the witness policy is satisfied.

anchor_process_witnessed_seconds

The time (in seconds) that it takes to process a witnessed anchor credential by publishing it to the Observer and posting a *Create* activity.

anchor_write_build_cred_seconds

The time (in seconds) that it takes to build a credential (inside the *write anchor* function).

anchor_write_get_witnesses_seconds

The time (in seconds) that it takes to get witnesses (inside the *write anchor* function).

anchor_write_sign_cred_seconds

The time (in seconds) that it takes to sign the credential (inside the *write anchor* function).

anchor_write_post_offer_activity_seconds

The time (in seconds) that it takes to post an offer activity (inside the *write anchor* function).

anchor_write_get_previous_anchor_get_bulk_seconds

The time (in seconds) that it takes to perform a database 'bulk get' (inside the *get previous anchor* function).

anchor_write_get_previous_anchor_seconds

The time (in seconds) that it takes to get the previous anchor.

anchor_write_sign_with_local_witness_seconds

The time (in seconds) that it takes to sign with the local witness key.

anchor_write_sign_with_server_key_seconds

The time (in seconds) that it takes to sign with a server key.

anchor_write_sign_local_witness_log_seconds

The time (in seconds) that it takes to witness the log (inside the *sign local* function).

anchor_write_sign_local_watch_seconds

The time (in seconds) that it takes to add the verifiable credential to the VCT monitoring service (inside the *sign local* function).

anchor_write_resolve_host_meta_link_seconds

The time (in seconds) that it takes to resolve a host meta link.

anchor_write_store_seconds

The time (in seconds) that it takes to store an anchor event.

Operation Queue

The *Operation Queue* is an AMQP implementation of a Sidetree operation queue. Operations are posted to the queue and a batch is cut when the queue size reaches the maximum batch size (configured in the Sidetree protocol) or whwn a batch timeout occurs.

opqueue_add_operation_seconds

The time (in seconds) that it takes to add an operation to the queue.

opqueue_batch_cut_seconds

The time (in seconds) that it takes to cut an operation batch. The duration is from the time that the first operation was added to the time that the batch was cut.

opqueue_batch_rollback_seconds

The time (in seconds) that it takes to roll back an operation batch (in case of a transient error). The duration is from the time that the first operation was added to the time that the batch was cut.

opqueue_batch_size

The size of a cut batch.

opqueue_process_anchor_seconds

The time (in seconds) that it takes for the Observer to process an anchor credential.

opqueue_process_did_seconds

The time (in seconds) that it takes for the Observer to process a DID.

Content Addressable Storage

The Content Addressable Store (CAS) is either implemented as a local store or using IPFS.

cas_write_seconds

The time (in seconds) that it takes to write a document to CAS.

cas_resolve_seconds

The time (in seconds) that it takes to resolve a document from CAS.

cas_cache_hit_count

The number of times a CAS document was retrieved from the cache.

cas_read_seconds

The time (in seconds) that it takes to read a document from the CAS storage.

Document

The Document metrics measure times for posting *create* and *update* Sidetree operations, as well as resolving DID documents.

document_create_update_seconds

The time (in seconds) it takes the REST handler to process a create/update operation.

document_resolve_seconds

The time (in seconds) it takes the REST handler to resolve a document.

Database

Database metrics record the times for reads, writes, bulk writes, etc.

db_put_seconds

The time (in seconds) it takes the DB to store data.

db_get_seconds

The time (in seconds) it takes the DB to retrieve data by primary key.

db_get_tags_seconds

The time (in seconds) it takes the DB to get tags.

db_get_bulk_seconds

The time (in seconds) it takes the DB to get bulk.

db_query_seconds

The time (in seconds) it takes to query for data.

db_delete_seconds

The time (in seconds) it takes to delete data.

db_batch_seconds

The time (in seconds) it takes to perform a batch update.

Verifiable Credential Transparency

The Verifiable Credential Transparency (VCT) subsystem records verifiable credentials to a ledger (backed by Google Trillian).

vct_witness_add_proof_vct_nil_seconds

The time (in seconds) it takes the add proof when vct is nil in witness.

vct_witness_add_vc_seconds

The time (in seconds) it takes the add vc in witness.

vct_witness_add_proof_seconds

The time (in seconds) it takes the add proof in witness.

vct_witness_webfinger_seconds

The time (in seconds) it takes web finger in witness.

vct_witness_verify_vct_signature_seconds

The time (in seconds) it takes verify vct signature in witness.

vct_witness_add_proof_parse_credential_seconds

The time (in seconds) it takes the parse credential in add proof.

vct_witness_add_proof_sign_seconds

The time (in seconds) it takes the sign in add proof.

Signer

Following are metrics for the signer, which is either using a local key or KMS.

signer_get_key_seconds

The time (in seconds) it takes to get the key.

signer_sign_seconds

The time (in seconds) it takes to sign.

signer_add_linked_data_proof_seconds

The time (in seconds) it takes to add linked data proof.

Resolver

The document resolver resolves a DID document, either from local store, or from the anchor origin.

resolver_resolve_document_locally_seconds

The time (in seconds) it takes to resolve the document locally.

resolver_get_anchor_origin_endpoint_seconds

The time (in seconds) it takes to get endpoint information from the anchor origin.

resolve_document_from_anchor_origin_seconds

The time (in seconds) it takes to resolve a document from the anchor origin.

resolver_resolve_document_from_create_document_store_seconds

The time (in seconds) it takes to resolve a document from the *create* document store.

resolver_delete_document_from_create_document_store_seconds

The time (in seconds) it takes the resolver to delete a document from the *create* document store.

resolver_verify_cid_seconds

The time (in seconds) it takes to verify a CID in the anchor graph.

resolver_request_discovery_seconds

The time (in seconds) it takes to request DID discovery.

Decorator

The decorator is invoked by the core Sidetree library. It verifies (from the anchor origin) that the local domain has the latest operations.

decorator_decorate_seconds

The time (in seconds) it takes the decorator to pre-process the document operation.

decorator_processor_resolve_seconds

The time (in seconds) it takes the processor to resolve a document before accepting the document operation.

decorator_get_ao_endpoint_and_resolve_from_ao_seconds

The time (in seconds) it takes to resolve a document from the anchor origin before accepting the document operation.

Operation Store

The operation store contains published and unpublished operations.

operations_put_unpublished_operation_seconds

The time (in seconds) it takes to store an unpublished operation.

operations_get_unpublished_operations_seconds

The time (in seconds) it takes to get an unpublished operations for a given suffix.

operations_calculate_unpublished_operation_key_seconds

The time (in seconds) it takes to calculate a key for an unpublished operation.

operations_put_published_operations_seconds

The time (in seconds) it takes to store published operations.

operations_get_published_operations_seconds

The time (in seconds) it takes to get published operations for a given suffix.

Sidetree Core

The follow metrics are produced by the core Sidetree library.

core_process_operation_seconds

The time (in seconds) it takes to process a DID operation.

core_get_protocol_version_seconds

The time (in seconds) it takes to get the protocol version (in *process operation*).

core_parse_operation_seconds

The time (in seconds) it takes to parse an operation in (in *process operation*).

core_validate_operation_seconds

The time (in seconds) it takes to validate an operation (in *process operation*).

core_decorate_operation_seconds

The time (in seconds) it takes to decorate an operation (in *process operation*).

core_add_unpublished_operation_seconds

The time (in seconds) it takes to add an unpublished operation to the store (in *process operation*).

core_add_operation_to_batch_seconds

The time (in seconds) it takes to add an operation to the batch (in *process operation*).

core_get_create_operation_result_seconds

The time (in seconds) it takes to get a *create* operation result (in *process operation*).

core_http_create_update_seconds

The time (in seconds) it takes for a *create/update* HTTP call.

core_http_resolve_seconds

The time (in seconds) it takes for a *resolve* HTTP call.

orb_core_cas_write_size

The size of the data written to CAS.

3.5 Verifiable Credential Transparency (VCT)

3.5.1 Introduction

The Verifiable Credential Transparency (VCT) Witness ledger is based on certificate transparency RFC6962. VCT logs are append-only ledgers of anchor credentials. Because they're distributed and independent, anyone can query them to see what Anchor Credentials have been included and when. Because they're append-only, they are verifiable by Monitors. VCT log may be named to enable periodical rollover.

Logs are cryptographically monitored by Orb node. Monitors cryptographically check that Anchor Credential have been included in VCT log.

Orb specification extended RFC6962 to include Verifiable Credential (VC) objects.

Add Verifiable Credential Flow

VCT node will validate verifiable credential and add it to Trillian log.

See add-vc REST endpoint for more information.

VCT Monitoring

Monitors cryptographically check if verifiable credentials have been included in logs. They can also monitor log for consistency. Monitors can be set up and run by anyone.

VCT node supports the following monitoring API:

Get Signed Tree Head (STH)

Retrieve latest Signed Tree Head as described in RFC6962. See get-sth REST endpoint for more information.

Get Signed Tree Head (STH) Consistency

Retrieve Merkle Consistency Proof between Two Signed Tree Heads as per RFC6962.

See get-sth-consistency REST endpoint for more information.

Get Entries

Retrieve entries from log as per RFC6962.

See get-entries REST endpoint for more information.

Retrieve Entry and Proof from Log

Retrieve entry and Merkle audit proof from log as per RFC6962.

See get-entry-and-proof REST endpoint for more information.

Retrieve Merkle Audit Proof from Log by Leaf Hash

Retrieve Merkle audit proof by leaf hash from log as per RFC6962.

See get-proof-by-hash REST endpoint for more information.

VCT Discovery and Administration

WebFinger

Retrieve discovery information about VCT log.

See .well-known/webfinger REST endpoint for more information.

Get Issuers

Retrieves log issuers (if configured). If the log issuer list has been configured for the log then Verifiable Credential issuer ID has to be present in the log's issuer list in order to add Verifiable Credential to the log.

See get-issuers REST endpoint for more information.

3.5.2 Log configuration

An administrator can configure log per domain. When configured the log URL is stored into configuration database "orb-config" under "log-url" key. The log URL is cached on the server with periodic cache updates from the database. Default log URL cache expiry period has been set to 1 minute.

Configuring log URL

Log URL can be configured by posting log URL to domain endpoint "/log".

Retrieve log URL

Log URL can be retrieved by issuing GET to domain endpoint "/log".

3.5.3 Log Monitoring

Orb server will periodically invoke log monitoring service that will watch logs for consistency and check that they behave correctly. In order to watch logs, the monitoring service follows these steps for each log:

- Fetch the current signed tree head(STH).
- Verify the STH signature.

If the service is processing log for the first time and the log is not empty:

- Fetch all the entries in the tree corresponding to the STH.
- Confirm that the tree made from the fetched entries produces the same hash as that in the STH.

If the service has already encountered this log and log's STH has changed since last check:

- Fetch a consistency proof for the new STH with the previous STH.
- Verify the consistency proof.

Upon successful STH signature and consistency verification, log monitoring service will save current STH for each domain.

The service can be configured to retrieve and store all log entries during log monitoring. See Log Entries Store Enabled Parameter.

For more details about log monitoring see: https://datatracker.ietf.org/doc/html/rfc6962#section-5.3.

3.5.4 REST Endpoints

Add Anchor Credential to Log

Endpoint: "/{alias}/v1/add-vc"

Add Anchor Credential to log.

Parameter: Anchor Credential.

Example

```
POST /maple2020/v1/add-vc HTTP/1.1
Host: witness.com
```

Request:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/security/suites/jws-2020/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "credentialSubject": "hl:uEiDKH_3UYP_4_F8BAV7mOPa6IDrTMuMShNNwHVyfam7CBg",
  "id": "https://orb.domain1.com/vc/41ae2e2f-4974-4425-ac0e-a1b14b2cfea3",
  "issuanceDate": "2022-05-19T22:16:19.7471831Z",
  "issuer": "https://orb.domain1.com",
  "type": "VerifiableCredential"
}
```

Output:

sct_version: The version of the SignedCertificateTimestamp structure.

id: The log ID, base64 encoded.

timestamp: The SCT timestamp, in decimal.

extensions: An opaque type for future expansion.

signature: The SCT signature, base64 encoded.

```
{
  "svct_version": 0,
  "id": "c0JZOeGbBoFbJYTJpin68J2IhCHr1muAEi4QCY7cTko=",
  "timestamp": 1652998579753,
  "extensions": "",
  "signature":
↪"eyJhbGdvcml0aG0iOnsic2lnbmF0dXJlIjoiRUNEU0EiLCJ0eXBlIjoiRUNEU0FQMjU2REVSIn0sInNpZ25hdHVyZSI6Ik1FVUNJ
↪"
}
```

Get Signed Tree Head (STH)

Endpoint: "/{alias}/v1/get-sth"

Retrieve latest Signed Tree Head as described in RFC6962

Example

```
GET /maple2020/v1/get-sth HTTP/1.1
Host: witness.com
```

Output:

```
{
  "tree_size": 24,
  "timestamp": 1652906464409,
  "sha256_root_hash": "E2HaBxp1VbZg1Mx/OSMeDSo/94yKXY1+cD0UzoH1kIk=",
  "tree_head_signature":
↪"eyJhbGdvcml0aG0iOnsic2lnbmF0dXJlIjoiRUNEU0EiLCJ0eXBlIjoiRUNEU0FQMjU2REVSIn0sInNpZ25hdHVyZSI6Ik1FVUNJ
↪"
}
```

Get Signed Tree Head (STH) Consistency

Endpoint: "/{alias}/v1/get-sth-consistency?first=5&second=10

Retrieve Merkle Consistency Proof between Two Signed Tree Heads as per RFC6962.

Parameters:

first: The tree_size of the first tree

second: The tree_size of the second tree

Both tree sizes must be from existing v1 STHs (Signed Tree Heads).

Example

http://witness.com/maple2020/v1/get-sth-consistency?first=20&second=23

```
GET /maple2020/v1/get-sth-consistency?first=20&second=23 HTTP/1.1
Host: witness.com
```

Output:

consistency: An array of Merkle Tree nodes, base64 encoded.

```
{
  "consistency": [
    "a7jghQovOhwnucOdJLbPD/I/OoiY+qprT/mk+LVmeL4=",
    "Ml0LmRpcONX42qwxHAb/qw3rFrhqmEH/Bdw5rMGMedw=",
    "M/Qu3X7+iS23AWUrgU+plLBRUi3WDHcodMV2+oUcQbM="
  ]
}
```

Get Entries

Endpoint: "/{alias}/v1/get-entries?start=0&end=3"

Retrieve entries from log as per RFC6962.

Parameters:

start: 0-based index of first entry to retrieve

end: 0-based index of last entry to retrieve

Example

```
GET /maple2020/v1/get-entries?start=0&end=3 HTTP/1.1
Host: witness.com
```

Output:

entries: An array of objects, each consisting of

leaf_input: The base64-encoded MerkleTreeLeaf structure.

extra_data: The base64-encoded unsigned data pertaining to the log entry.


```
{
  "entries": [
    {
      "leaf_input":
↪"eyJ2ZXJzaW9uIjowLCJsZWFmX3R5cGUiOjEwMCwidGltZXN0YW1wZWRfZW50cnkiOnsidGltZXN0YW1wIjoxNjUyOTA2MzY1NTYz
↪",
      "extra_data": "bnVsbA=="
    },
    {
      "leaf_input":
↪"eyJ2ZXJzaW9uIjowLCJsZWFmX3R5cGUiOjEwMCwidGltZXN0YW1wZWRfZW50cnkiOnsidGltZXN0YW1wIjoxNjUyOTA2MzcxNTQx
↪",
      "extra_data": "bnVsbA=="
    },
    {
      "leaf_input":
↪"eyJ2ZXJzaW9uIjowLCJsZWFmX3R5cGUiOjEwMCwidGltZXN0YW1wZWRfZW50cnkiOnsidGltZXN0YW1wIjoxNjUyOTA2MzczNDc1
↪",
      "extra_data": "bnVsbA=="
    },
    {
      "leaf_input":
↪"eyJ2ZXJzaW9uIjowLCJsZWFmX3R5cGUiOjEwMCwidGltZXN0YW1wZWRfZW50cnkiOnsidGltZXN0YW1wIjoxNjUyOTA2MzgwNTA3
↪",
      "extra_data": "bnVsbA=="
    }
  ]
}
```

Retrieve Entry and Proof from Log

Endpoint: "/{alias}/v1/get-entry-and-proof?leaf_index=1&tree_size=23"

Retrieve entry and Merkle audit proof from log as per RFC6962.

Parameters:

leaf_index: The index of the desired entry.

tree_size: The tree_size of the tree for which the proof is desired.

Example

```
GET /maple2020/v1/get-entry-and-proof?leaf_index=1&tree_size=23 HTTP/1.1
Host: witness.com
```

Output:

leaf_input: The base64-encoded MerkleTreeLeaf structure.

extra_data: The base64-encoded unsigned data pertaining to the log entry.

audit_path: An array of base64-encoded Merkle Tree nodes proving the inclusion of entry.


```
{
  "leaf_input":
↪"eyJ2ZXJzaW9uIjowLCJsZWFmX3R5cGUiOjEwMCwidGltZXN0YW1wZWRfZW50cnkiOnsidGltZXN0YW1wIjoxNjUyOTA2MzcxNTQx
↪",
  "extra_data": "bnVsbA==",
  "audit_path": [
    "40IbH54WlW/7XICpmttJmOPv4/DaRsCZDc4osAdzPxg=",
    "aaNL9DnYEGLdCjbCV72u6vZ/yBj/Mo0b5Wt7h5zdKMI=",
    "B4KHZzvpTXVOUtyI2skBE6SO8CZFcZl0hDAybAOhV/M=",
    "3yME5CgWt6xUnb3jlz10puvcZT4vRsQcxp9+dceMdrg=",
    "Verp7o4zRhZPJvIxEzCzPNPtw2ZKYj9HCYKtHpDY6RY="
  ]
}
```


Decoded leaf_input:


```
{
  "version": 0,
  "leaf_type": 100,
  "timestamped_entry": {
    "timestamp": 1652906371541,
    "entry_type": 100,
    "vc_entry":
↪"eyJAY29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkZW50aWFscy92MSIsImh0dHBzOi8vdzNpZC5vcmcvc2Vj
↪",
    "extensions": null
  }
}
```

Retrieve Merkle Audit Proof from Log by Leaf Hash

Endpoint: "/{alias}/v1/get-proof-by-hash?tree_size=24&hash=bsmXlruDXTFrWOfFQ2WRBERc51rMk3hvRdOAROIZmHI="

Retrieve Merkle audit proof by leaf hash from log as per RFC6962.

Parameters:

hash: A base64-encoded v1 leaf hash.

tree_size: The tree_size of the tree on which to base the proof.

Example

```
GET /maple2020/v1/get-proof-by-hash?tree_size=24&
↪hash=bsmXlruDXTFrWOfFQ2WRBERc51rMk3hvRdOAROIZmHI= HTTP/1.1
Host: witness.com
```

Output:

leaf_index: The 0-based index of the end entity corresponding to the "hash" parameter.

audit_path: An array of base64-encoded Merkle Tree nodes proving the inclusion of the entry.

```
{
  "leaf_index": 2,
  "audit_path": [
    "6pKexnNWd+pRegRB+0kUct/yIlPhEP8F4YASINoP3PI=",
    "KB/U4HPHLwE36ZZCcWhBhk9M24242oDPe3kJxAUkfu0=",
    "m+GAt5OxvxxX5kNBDqqHkoHJHeJxcitAzc0wJMbIlQs=",
    "GkRLjI7Sb6pkqjU63baSCyoAiqRYrBbY7Efz+cjlXY4=",
    "XuYjKQt/oygHn96E4tSJrGsT4ZqLAvlsui9TS4+6f3g="
  ]
}
```

VCT Discovery and Administration Endpoints

WebFinger

Endpoint: "/.well-known/webfinger?resource={log-id}"

Retrieve discovery information about VCT log.

Example

```
GET /.well-known/webfinger?resource=https://witness.com/maple2020 HTTP/1.1
Host: witness.com
```

Output:

```
{
  "subject": "https://witness.com/maple2020",
  "properties": {
    "https://trustbloc.dev/ns/ledger-type": "vct-v1",
    "https://trustbloc.dev/ns/public-key": "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEfCc/
↪5CT+K59Dv7+r+MiVX+ARfMeFK9CwdLlicTyjoNJdhFfP4/wnVfXg+vLjrqBYFsYzgokTSTZBSk72WF1RrQ=="
```

(continues on next page)

(continued from previous page)


```
  },
  "links": [
    {
      "rel": "self",
      "href": "https://witness.com/maple2020"
    }
  ]
}
```


Get Issuers

Endpoint: "/{alias}/get-issuers"

Retrieves log issuers (if configured). If the log issuer list has been configured then Anchor Credential issuer ID has to be present in the log's issuer list in order to add Anchor Credential to the log.

Example

```
GET /maple2020/v1/get-issuers HTTP/1.1
Host: witness.com
```

Output:


```
[
↪"did:key:zUC724vuGvHpnCGFG1qqpXb81SiBLu3KLSqVzenwEZNPoY35i2Bscb8DLaVwHvRFs6F2NkNNXRcPWvqnPDUd9ukdjLkj
↪"]
```


Health Check

Endpoint: "/healthcheck"

Returns VCT status.

Example

```
GET /healthcheck HTTP/1.1
Host: witness.com
```

Output:


```
{
  "current_time": "2022-05-19T17:56:54.0594632Z",
  "status": "success"
}
```

Metrics

Endpoint: "/metrics"

Returns VCT metrics.

Example

```
GET /metrics HTTP/1.1
Host: witness.com
```

3.5.5 Startup Parameters

This section enumerates the startup parameters for an VCT server. Parameters in the *Required Parameters* section are required, otherwise the server will not start. Parameters in the *Optional Parameters* section are optional and will use a default value if not specified.

Required Parameters

Following are the required parameters for a VCT server.

api-host

Arg	Env
–api-host	VCT_API_HOST

URL to run the VCT instance on. Format: HostName:Port.

base-url

Arg	Env
–base-url	VCT_BASE_URL

Base URL. e.g (https://vct.com)

kms-type

Arg	Env	Default
–kms-type	VCT_KMS_Type	

KMS type (local,web,aws).

kms-endpoint

Arg	Env	Default
–kms-endpoint	VCT_KMS_ENDPOINT	

Remote KMS URL.

log-active-key-id

Arg	Env	Default
–log-active-key-id	VCT_LOG_SIGN_ACTIVE_KEY_ID	

Active Key ID for signing logs.

logs

Arg	Env
–logs	VCT_LOGS

A list of Trillian logs (comma separated). Format must be :@.

Examples: maple2021:rw@server.com,maple2020:r@server.com:9890

issuers

Arg	Env	Default
–issuers	VCT_ISSUERS	

Comma-Separated list of supported issuers.

Examples: maple2021@did:key:zUC724vuGvHpnCGFG1qqpXb81SiBLu3KLSqVzenwEZNPoY35i2Bscb8DLaVwHvRFs6F2NkNNX
maple2020@did:key:zUC724vuGvHpnCGFG1qqpXb81SiBLu3KLSqVzenwEZNPoY35i2Bscb8DLaVwHvRFs6F2NkNNXRcPWvqn

trillian-db-conn

Arg	Env
–trillian-db-conn	VCT_TRILLIAN_DB_CONN

Trillian db conn.

Example: user=postgres host=trillian.postgres password=password dbname=test port=5432 sslmode=disable

dsn

Arg	Env
–dsn	VCT_DSN

Datasource Name with credentials if required. Format must be :[//]. Supported drivers are [mem, couchdb, postgres, mongodb].

Examples: 'postgres://jack:secret@pg.example.com:5432/mydb', 'mem://test', 'mongodb://mongodb.example.com:27017'

timeout

Arg	Env
–timeout	VCT_TIMEOUT

Total time in seconds to wait until the services are available before giving up.

sync-timeout

Arg	Env
–sync-timeout	VCT_SYNC_TIMEOUT

Total time in seconds to resolve config values.

tls-systemcertpool

Arg	Env	Default
–tls-systemcertpool	VCT_TLS_SYSTEMCERTPOOL	false

Use system certificate pool. Possible values true and false. Defaults to false if not set.

tls-cacerts

Arg	Env	Default
–tls-cacerts	VCT_TLS_CACERTS	

Comma-Separated list of ca certs path.

tls-serve-cert

Arg	Env	Default
–tls-serve-cert	VCT_TLS_SERVE_CERT	

TLS certificate for VCT server. Path to the server certificate to use when serving HTTPS.

tls-serve-key

Arg	Env	Default
–tls-serve-key	VCT_TLS_SERVE_KEY	

TLS key for VCT server. Path to the private key to use when serving HTTPS.

context-provider-url

Arg	Env	Default
–context-provider-url	VCT_CONTEXT_PROVIDER_URL	

Comma-separated list of remote context provider URLs to get JSON-LD contexts from.

Optional Parameters

Below are the optional parameters for an VCT server. If not specified then the default value is used.

api-read-token

Arg	Env	Default
–api-read-token	VCT_API_READ_TOKEN	

Check for bearer token in the authorization header (optional).

api-write-token

Arg	Env	Default
–api-write-token	VCT_API_WRITE_TOKEN	

Check for bearer token in the authorization header (optional).

dev-mode

Arg	Env	Default
–dev-mode	VCT_DEV_MODE	false

database-prefix

Arg	Env	Default
–database-prefix	VCT_DATABASE_PREFIX	

An optional prefix to be used when creating and retrieving underlying databases. This allows a database to be shared by multiple VCT domains. (Mainly used in development environments.)

host-metrics-url

Arg	Env	Default
–metrics-host	VCT_METRICS_HOST	

URL that exposes the metrics endpoint. Format: HostName:Port.

3.5.6 Rotate VCT logs

Rotate VCT logs process requires new configuration for both VCT server and Orb node.

Configure VCT

Add new log to VCT logs parameter

VCT_LOGS=maple2023:rw@orb.trillian.log.server:8090,maple2022:rw@orb.trillian.log.server:8090

In order for changes to take effect an administrator has to re-start VCT server.

Configure Orb

Configuring Orb node with new log requires setting up new VCT log URL for Orb domain, adding that new log URL to log monitoring list and removing old log URL from log monitoring list. An administrator may want to wait some time before deactivating old log in order to allow for all items in the old log to be processed and for log to be verified.

Rotate VCT Log Steps for Orb Node:

- configure Orb node with new VCT log URL
- add new log URL to log monitoring list
- remove old log URL from log monitoring list

Configure Orb node with new VCT log

An administrator can configure new VCT log per Orb domain by posting new log URL to /log endpoint. See log configuration REST endpoint for more information.

```
POST /log HTTP/1.1
Host: orb.domain1.com
Content-Type: application/ld+json

http://orb.vct:8077/maple2023
```

Activate monitoring for new log

Activate monitoring of new log by posting to log-monitor REST endpoint.

```
POST /log-monitor HTTP/1.1
Host: orb.domain1.com

{
  "activate": [
    "http://orb.vct:8077/maple2023"
  ]
}
```

Deactivate monitoring for old log

De-activate monitoring of old log by posting to log-monitor REST endpoint.

POST /log-monitor HTTP/1.1 Host: orb.domain1.com

```
{
  "deactivate": [
    "http://orb.vct:8077/maple2022"
  ]
}
```

List active/inactive logs for log monitoring service

Endpoint: "/log-monitor?status=active"

Retrieve active log list for log monitoring service.

Parameters:

status: active or inactive; it defaults to active if status parameter is not provided

Example

```
GET /log-monitor?status=active HTTP/1.1
Host: orb.domain1.com
```

Output:

```
{
  "active": [
    {
      "log_url": "http://orb.vct:8077/maple2022",
      "sth_response": {
        "tree_size": 24,
        "timestamp": 1654869615262,
        "sha256_root_hash": "GDCyCWRPqGPtrgNtj1iFGxwSg0emoxuq/W1Dc4lEiro=",
        "tree_head_signature":
↪"eyJhbGdvcml0aG0iOnsic2lnbmF0dXJlIjoiRUNEU0EiLCJ0eXBlIjoiRUNEU0FQMjU2REVSIn0sInNpZ25hdHVyZSI6Ik1FVUNJ
↪"
      },
      "pub_key": "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEfCc/
↪5CT+K59Dv7+r+MiVX+ARfMeFK9CwdLlicTyjoNJdhFfP4/wnVfXg+vLjrqBYFsYzgokTSTZBSk72WF1RrQ==",
      "active": true
    },
    {
      "log_url": "http://orb.vct:8077/maple2023",
      "sth_response": {
        "tree_size": 0,
        "timestamp": 1654868145703,
        "sha256_root_hash": "47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=",
        "tree_head_signature":
↪"eyJhbGdvcml0aG0iOnsic2lnbmF0dXJlIjoiRUNEU0EiLCJ0eXBlIjoiRUNEU0FQMjU2REVSIn0sInNpZ25hdHVyZSI6Ik1FWUNJ
↪"
```

(continues on next page)

(continued from previous page)

```
      },
      "pub_key": "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEfCc/
↪5CT+K59Dv7+r+MiVX+ARfMeFK9CwdLlicTyjoNJdhFfP4/wnVfXg+vLjrqBYFsYzgokTSTZBSk72WF1RrQ==",
      "active": true
    }
  ]
}
```

Example

```
GET /log-monitor?status=inactive HTTP/1.1
Host: orb.domain1.com
```

Output:

```
{
  "inactive": [
    {
      "log_url": "http://orb.vct:8077/maple2020",
      "sth_response": {
        "tree_size": 48,
        "timestamp": 1654869871315,
        "sha256_root_hash": "qlODFYrB140S4ZCYY6+ipISvTALA3x2jEs3bpV0UBrI=",
        "tree_head_signature":
↪"eyJhbGdvcml0aG0iOnsic2lnbmF0dXJlIjoiRUNEU0EiLCJ0eXBlIjoiRUNEU0FQMjU2REVSIn0sInNpZ25hdHVyZSI6Ik1FVUNJU
↪"
      },
      "pub_key": "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEfCc/
↪5CT+K59Dv7+r+MiVX+ARfMeFK9CwdLlicTyjoNJdhFfP4/wnVfXg+vLjrqBYFsYzgokTSTZBSk72WF1RrQ==",
      "active": false
    }
  ]
}
```

3.6 REST Endpoints

3.6.1 ActivityPub Endpoints

The ActivityPub spec defines a number of REST endpoints that provide information about a service. The ActivityAnchors spec extends the ActivityPub spec to provide additional endpoints.

Service

The REST endpoints for a service depend on the value of startup parameter service-id. By default, the endpoint is */services/orb*, but if, for example, *service-id* is set to *did:web:orb.domain1.com:services:anchor*, then the service REST endpoint will be */services/anchor*.

Endpoint: /

The returned data is a JSON document that contains REST endpoints that may be queried to return additional information.

GET

Example

Assuming that the default value for service-id is used, then the Orb service is retrieved using the */services/orb* endpoint.

Request:

```
GET /services/orb HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains the service:

```
{
  "@context": [
    "https://www.w3.org/ns/activitystreams",
    "https://w3id.org/security/v1",
    "https://w3id.org/activityanchors/v1"
  ],
  "id": "https://orb.domain1.com/services/orb",
  "type": "Service",
  "followers": "https://orb.domain1.com/services/orb/followers",
  "following": "https://orb.domain1.com/services/orb/following",
  "inbox": "https://orb.domain1.com/services/orb/inbox",
  "liked": "https://orb.domain1.com/services/orb/liked",
  "likes": "https://orb.domain1.com/services/orb/likes",
  "outbox": "https://orb.domain1.com/services/orb/outbox",
  "publicKey": {
    "id": "https://orb.domain1.com/services/orb/keys/main-key",
    "owner": "https://orb.domain1.com/services/orb",
    "publicKeyPem": "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhki....."
  },
  "shares": "https://orb.domain1.com/services/orb/shares",
  "witnesses": "https://orb.domain1.com/services/orb/witnesses",
  "witnessing": "https://orb.domain1.com/services/orb/witnessing"
}
```

Example

Assuming that the value of the service-id parameter is set to *did:web:orb.domain1.com:services:anchor*, then the Orb service is retrieved using the */services/anchor* endpoint.

Request:

```
GET /services/anchor HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains the service:

```
{
  "@context": [
    "https://www.w3.org/ns/activitystreams",
    "https://w3id.org/security/v1",
    "https://w3id.org/activityanchors/v1"
  ],
  "followers": "https://orb.domain1.com/services/anchor/followers",
  "following": "https://orb.domain1.com/services/anchor/following",
  "id": "did:web:orb.domain1.com:services:anchor",
  "publicKey": "did:web:orb.domain1.com:services:anchor
↪#MrWUYMOS4ZGF7LCBVSr8njn98OvG70XaLV31EDFhBH0",
  "inbox": "https://orb.domain1.com/services/anchor/inbox",
  "liked": "https://orb.domain1.com/services/anchor/liked",
  "likes": "https://orb.domain1.com/services/anchor/likes",
  "outbox": "https://orb.domain1.com/services/anchor/outbox",
  "shares": "https://orb.domain1.com/services/anchor/shares",
  "type": "Service",
  "witnesses": "https://orb.domain1.com/services/anchor/witnesses",
  "witnessing": "https://orb.domain1.com/services/anchor/witnessing"
}
```

Keys

Endpoint: /services/orb/keys/[id]

Note: This endpoint is unavailable if a DID is used as the public key, for example,

```
{
  . . .
  "publicKey": "did:web:orb.domain2.com:services:anchor
↪#MrWUYMOS4ZGF7LCBVSr8njn98OvG70XaLV31EDFhBH0",
  . . .
}
```

In this case, a DID resolver is required to resolve the public key.

GET

The public key of an Orb service is retrieved using this endpoint.

Example

Request:

```
GET /services/orb/keys/main-key HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains the public key of the service:

```
{
  "id": "https://orb.domain1.com/services/orb/keys/main-key",
  "owner": "https://orb.domain1.com/services/orb",
  "publicKeyPem": "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhki....."
}
```

did.json

Endpoint: /services/orb/did.json

If the value of startup parameter service-id is set to a DID, for example, *did:web:orb.domain2.com:services:orb*, then the DID document for the service may be resolved at this endpoint.

GET

Example

Request:

```
GET /services/orb/did.json HTTP/1.1
Host: orb.domain2.com
Accept: application/ld+json
Accept-Encoding: gzip, deflate
```

Response contains the DID document for the service, which includes the public key(s) for HTTP signatures and the service endpoint:

```
{
  "@context": [
    "https://w3id.org/did/v1",
    "https://identity.foundation/.well-known/did-configuration/v1"
  ],
  "id": "did:web:orb.domain2.com:services:orb",
  "verificationMethod": [
    {
      "controller": "did:web:orb.domain2.com:services:orb",
      "id": "did:web:orb.domain2.com:services:orb
↪#MrWUYMOS4ZGF7LCBVSr8njn98OvG70XaLV31EDFhBH0",
```

(continues on next page)

(continued from previous page)

```
      "publicKeyMultibase": "z7a7SJkF8LTqGACTxB8r86i2btE7y8qrPnGuCe3M9vk8f",
      "type": "Ed25519VerificationKey2020"
    }
  ],
  "service": [
    {
      "id": "did:web:orb.domain2.com:services:orb#activity-pub",
      "serviceEndpoint": "https://orb.domain2.com",
      "type": "LinkedDomains"
    }
  ]
}
```

Followers

Endpoint: /services/orb/followers

GET

The followers of this Orb service are returned via this endpoint. If no paging parameters are specified in the URL then the response contains information about the collection, i.e. the links to the first and last page, as well as the total number of items in the collection. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request page information:

```
GET /services/orb/followers HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/followers",
  "type": "Collection",
  "totalItems": 19,
  "first": "https://orb.domain1.com/services/orb/followers?page=true",
  "last": "https://orb.domain1.com/services/orb/followers?page=true&page-num=4"
}
```

Request first page:

```
GET /services/orb/followers?page=true&page-num=0 HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items in the first page:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/followers?page=true&page-num=0",
  "type": "CollectionPage",
  "totalItems": 19,
  "next": "https://orb.domain1.com/services/orb/followers?page=true&page-num=1",
  "items": [
    "https://orb.domain2.com/services/orb",
    "https://orb.domain3.com/services/orb",
    "https://orb.domain4.com/services/orb"
  ]
}
```

Following

Endpoint: /services/orb/following

GET

The services following this Orb service are returned via this endpoint. If no paging parameters are specified in the URL then the response contains information about the collection, i.e. the links to the first and last page, as well as the total number of items in the collection. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request:

```
GET /services/orb/following HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/following",
  "type": "Collection",
  "totalItems": 19,
  "first": "https://orb.domain1.com/services/orb/following?page=true",
  "last": "https://orb.domain1.com/services/orb/following?page=true&page-num=4"
}
```

Request first page:

```
GET /services/orb/following?page=true&page-num=0 HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items from the first page:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/following?page=true&page-num=0",
  "type": "CollectionPage",
  "totalItems": 19,
  "next": "https://orb.domain1.com/services/orb/following?page=true&page-num=1",
  "items": [
    "https://orb.domain2.com/services/orb",
    "https://orb.domain3.com/services/orb",
    "https://orb.domain4.com/services/orb"
  ]
}
```

Outbox

Endpoint: /services/orb/outbox

GET

A GET request to the outbox endpoint returns the activities that were posted to a service's Outbox. This endpoint is restricted by authorization rules, i.e. the requester must have a valid authorization bearer token or must be verified using HTTP signatures and also must be in the *following* or *witnesses* collection. Although, any activity sent to a public URI, is returned without authorization.

If no paging parameters are specified in the URL then the response contains information about the outbox collection, i.e. the links to the first and last page, as well as the total number of items in the collection. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request page information:

```
GET /services/orb/outbox HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/outbox",
  "type": "OrderedCollection",
  "totalItems": 3212,
  "first": "https://orb.domain1.com/services/orb/outbox?page=true",
  "last": "https://orb.domain1.com/services/orb/outbox?page=true&page-num=76"
}
```

Request page 41:

```
GET /services/orb/outbox?page=true&page-num=41 HTTP/1.1
Host: orb.domain1.com
```

(continues on next page)

(continued from previous page)

```
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items from page 41:


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/outbox?page=true&page-num=41",
  "type": "OrderedCollectionPage",
  "totalItems": 321,
  "next": "https://orb.domain1.com/services/orb/outbox?page=true&page-num=42",
  "orderedItems": [
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "https://orb.domain1.com/services/orb",
      "id": "https://orb.domain1.com/services/orb/activities/72c49978-0812-4c07-83ac-
↪a39c9faf1a0a",
      "object": "https://orb.domain2.com/services/orb",
      "to": "https://orb.domain2.com/services/orb",
      "type": "Follow"
    },
    {
      "@context": [
        "https://www.w3.org/ns/activitystreams",
        "https://w3id.org/activityanchors/v1"
      ],
      "actor": "https://orb.domain1.com/services/orb",
      "id": "https://orb.domain1.com/services/orb/activities/d5c75aa5-9d19-43ae-92e1-
↪21ae7c6cf67e",
      "object": "https://w3id.org/activityanchors#AnchorWitness",
      "target": "https://orb.domain2.com/services/orb",
      "to": "https://orb.domain2.com/services/orb",
      "type": "Invite"
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "https://orb.domain1.com/services/orb",
      "id": "https://orb.domain1.com/services/orb/activities/db09985e-6d1b-4c9b-870e-
↪bd4c4aa3fec5",
      "object": {
        "@context": "https://www.w3.org/ns/activitystreams",
        "actor": "https://orb.domain2.com/services/orb",
        "id": "https://orb.domain2.com/services/orb/activities/ba65fbd8-ef59-4f69-a3d0-
↪0ba6a115c650",
        "object": "https://orb.domain1.com/services/orb",
        "to": "https://orb.domain1.com/services/orb",
        "type": "Follow"
      },
      "to": "https://orb.domain2.com/services/orb",
      "type": "Accept"
    },
    {
```


(continues on next page)

(continued from previous page)


```
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "https://orb.domain1.com/services/orb",
      "id": "https://orb.domain1.com/services/orb/activities/c3f51db8-fa65-487b-85f5-
↪201f110201b6",
      "object": {
        "@context": "https://w3id.org/activityanchors/v1",
        "object": {
          "linkset": [
            {
              "anchor": "hl:uEiCjjpFjLgWSPtDBJjZLk0oJyrLhcXw3K5SRcG9i5DTqbw",
              "author": [
                {
                  "href": "https://orb.domain1.com/services/orb"
                }
              ],
              "original": [
                {
                  "href": "data:application/gzip;base64,H4sIAAAAAAAA/
↪2yOT0+DMBiHv8vrdY5NF429Vd38s3SBgRpdOBQo8I7S1lKYhPDdDTt58PzL8/
↪yeASSqqhEOyGEArtJSWyBQStKukb6pkLH7sL3reRjIpKsjufu4CfdBxY8qN1fyk9ltf7k0/
↪QlmwFt3pg8DlFbkk8Y50xDP0zaZZ7rmqJbzVNdeI2yHqWimAcZ4BuhE/RfMMCPaJqSllJI10m/DgqqLGH3m/
↪pevdquX/bu/eqKLLRevUXHc/Dw+FJvbREt69hmrc5Tiv5bTNWZzbYvp+6JbwBiP8fgbAAD//9+kd3kHAQAA",
                  "type": "application/linkset+json"
                }
              ],
              "profile": [
                {
                  "href": "https://w3id.org/orb#v0"
                }
              ],
              "related": [
                {
                  "href": "data:application/gzip;base64,H4sIAAAAAAAA/
↪1TNTXOiMACA4f+SvbYrUHUGbrUtOhToIhAgOx4IoATyZeTTDv99Z731/s7zfgNKeHurOmD9/
↪QY5L2qhgAVqavUf5K1ppN24lyT8073vnAa5rSacWbl1kY4vn5vwWOxNsnmPrngET0AqcSa0eki1qs7/
↪na6TN2u1Gl9I+Vuoy0oo/GvQwHJ6AgPJf6SP5WvMQ8/
↪bhb0552FA8cAi6ifb8Bi0ecPP0qCZpz7nZ13Oo9WL4Pmt+uhEeTiOxV0MrmHOGdm02NDrPFlPLvMHHJpNlnpDGUMZQNjHmu4H1Ne/
↪UlrHun3LOOrgfvqCdxTF1FZZInvEX2dsU79ou40b2WOVluuA0RFxb+vOJskSpAoGZZZCrUydqUqcKz7UaxQ5V5RAozrsNBQddY8XOl
↪wIAAP//F6rBvooBAAA=",
                  "type": "application/linkset+json"
                }
              ],
              "replies": [
                {
                  "href": "data:application/gzip;base64,H4sIAAAAAAAA/
↪6xUW3OqSBj8L5NXFRgYQZ5W4m29ZI0YL5xKpYZh1FFgcBjgQCr/fYtoEmu3zm4ezit099dfNx+v4A/CY0l/SmD/
↪AAcpk9RWlKIoWoXe4mKvQFWzFCJoQGPJcJgquQYaX0CdBe8wTCTLmSxxTA5c/
↪AqVUpIJJkslzZikqXIs0iZUofpNOA0gQlrnk/LcAF/O3Mw/UiKB/QouJoANDqGd9Vn3KXZnM8fNOiV2H0M/
↪j5bhw7rtLh5P+BjvEhhuZ2JSNrWkLGobgu4+uffHYzI4Tvdrdy57zvjoTU8qH5dieiCbQp8gd0GGHYZ6y7NfcxPBdyykNf2fy3Dh3
↪vNb5Y0mDXwmjTi15IV8lL4qtnEjbUk1TiXAS0rrRD915JhKe1l5xmlIhGY9nVB548AFY4TCrX1cIOVw8cNMa/
↪1zDxdAbROsSppbra5o/
↪qzgzkXxw2lNreI654XlIJvt+4jytnGyx3kTbwXFGplM0ScfQnUAu0Xq41LbeZP1ZCuhfvjiX7WMsM/
```


(continues on next page)

(continued from previous page)


```
↪HhM6eC7RjBN95sELDALqhv19vdxHXXFsKs5MYZ9kf4z7/
↪kepCOo3JIpBef7zVDNS11kQ307nhVWeCt8X+RWpquIdQ20b+CvXZ1nQ1vq/
↪p+pFAGOUvKxCn19ix116tTnlQjHldPT+UEz0fBvurDylhtxodJe3naeosFYcWod+4bu6QX6ZU3Hm2yop8/
↪8p4h995w03nQArn9XZG+r3WXbNDLkYsRiduR4Rxepk29P5cbs9stHX18nryERjwnSz1PCvD2/HViq/ch2A/p/
↪ed/AzSul3fz6Pnt7wAAAP//2OYItxgFAAA=",
                  "type": "application/ld+json"
                }
              ]
            }
          ]
        },
        "type": "AnchorEvent",
        "url": "hl:uEiCUKchKLxlnv5z9ovQ1_brtSIW4WUxrfalPfszpVPjUIg:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpQ1VLY2hLTHhsbnY1ejlvdlExX2JydFNJVzRXVXhyZmFsUGZzenBWUGpV
↪"
      },
      "published": "2022-08-25T21:32:31.941935474Z",
      "to": [
        "https://orb.domain1.com/services/orb/followers",
        "https://www.w3.org/ns/activitystreams#Public"
      ],
      "type": "Create"
    }
  ]
}
```


POST

A POST request to the outbox endpoint adds the activity contained in the request to the service's Outbox, which will be processed by the ActivityPub Outbox. This endpoint is restricted by authorization rules, i.e. the requester must have a valid authorization bearer token, which is usually an administrator token.

Example

Post a *Follow* activity:

```
POST /services/orb/outbox HTTP/1.1
Host: orb.domain2.com
Content-Type: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate

{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/activities/91dbb2e2-1040-4fd5-bd2e-
↪1bcd67bdda8a",
  "type": "Follow",
  "actor": "https://orb.domain1.com/services/orb",
  "to": "https://orb.domain2.com/services/orb",
  "object": "https://orb.domain2.com/services/orb"
}
```

Inbox

Endpoint: /services/orb/inbox

GET

The activities posted to the inbox of this service are returned via this endpoint. If no paging parameters are specified in the URL then the response contains information about the inbox collection, i.e. the links to the first and last page, as well as the total number of items in the inbox. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request page information:

```
GET /services/orb/inbox HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/inbox",
  "type": "OrderedCollection",
  "totalItems": 19,
  "first": "https://orb.domain1.com/services/orb/inbox?page=true",
  "last": "https://orb.domain1.com/services/orb/inbox?page=true&page-num=4"
}
```

Request first page:

```
GET /services/orb/inbox?page=true HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items from the first page:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/inbox?page=true&page-num=0",
  "type": "OrderedCollectionPage",
  "totalItems": 19,
  "next": "https://orb.domain1.com/services/orb/inbox?page=true&page-num=1",
  "orderedItems": [
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "did:web:orb.domain2.com:services:orb",
      "id": "https://orb.domain2.com/services/orb/activities/ba65fbd8-ef59-4f69-a3d0-
↪0ba6a115c650",
      "object": "https://orb.domain1.com/services/orb",
      "to": "https://orb.domain1.com/services/orb",
```

(continues on next page)

(continued from previous page)


```
      "type": "Follow"
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "did:web:orb.domain2.com:services:orb",
      "id": "https://orb.domain2.com/services/orb/activities/29cdbde2-d82a-4594-8fbc-
↪26292316c0af",
      "object": {
        "@context": "https://www.w3.org/ns/activitystreams",
        "actor": "https://orb.domain1.com/services/orb",
        "id": "https://orb.domain1.com/services/orb/activities/72c49978-0812-4c07-83ac-
↪a39c9faf1a0a",
        "object": "https://orb.domain2.com/services/orb",
        "to": "https://orb.domain2.com/services/orb",
        "type": "Follow"
      },
      "to": "https://orb.domain1.com/services/orb",
      "type": "Accept"
    },
    {
      "@context": [
        "https://www.w3.org/ns/activitystreams",
        "https://w3id.org/activityanchors/v1"
      ],
      "actor": "did:web:orb.domain2.com:services:orb",
      "id": "https://orb.domain2.com/services/orb/activities/30e184f6-f45f-43e2-9cf6-
↪9e2b7c1ed230",
      "object": "https://w3id.org/activityanchors#AnchorWitness",
      "target": "https://orb.domain1.com/services/orb",
      "to": "https://orb.domain1.com/services/orb",
      "type": "Invite"
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "did:web:orb.domain2.com:services:orb",
      "id": "https://orb.domain2.com/services/orb/activities/635fe297-47d5-4374-8278-
↪8befcdea2055",
      "object": {
        "@context": [
          "https://www.w3.org/ns/activitystreams",
          "https://w3id.org/activityanchors/v1"
        ],
        "actor": "https://orb.domain1.com/services/orb",
        "id": "https://orb.domain1.com/services/orb/activities/d5c75aa5-9d19-43ae-92e1-
↪21ae7c6cf67e",
        "object": "https://w3id.org/activityanchors#AnchorWitness",
        "target": "https://orb.domain2.com/services/orb",
        "to": "https://orb.domain2.com/services/orb",
        "type": "Invite"
      },
      "to": "https://orb.domain1.com/services/orb",
      "type": "Accept"
```


(continues on next page)

(continued from previous page)


```
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "did:web:orb.domain2.com:services:orb",
      "endTime": "2022-08-26T15:36:20.174639514Z",
      "id": "https://orb.domain2.com/services/orb/activities/21646981-422c-46db-9bcb-
↪66184aa2dd39",
      "object": {
        "linkset": [
          {
            "anchor": "hl:uEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog",
            "author": [
              {
                "href": "did:web:orb.domain2.com:services:orb"
              }
            ],
            "original": [
              {
                "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A
↪%22hl%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%22%2C%22author%22%3A%5B%7B
↪%22href%22%3A%22did%3Aweb%3Aorb.domain2.com%3Aservices%3Aorb%22%7D%5D%2C%22item%22%3A
↪%5B%7B%22href%22%3A%22did%3Aorb%3AuAAA%3AEiDO4M4BoSmK4wFL-sSmFBhOyjeWfb2IuDovY1hRov5Cgg
↪%22%7D%5D%2C%22profile%22%3A%5B%7B%22href%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22
↪%7D%5D%7D%5D%7D",
                "type": "application/linkset+json"
              }
            ],
            "profile": [
              {
                "href": "https://w3id.org/orb#v0"
              }
            ],
            "related": [
              {
                "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A
↪%22hl%3AuEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog%22%2C%22profile%22%3A%5B%7B
↪%22href%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%2C%22via%22%3A%5B%7B%22href
↪%22%3A%22hl%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ2R4bHpiQ01fcFFvcHljQkY4UV9LQTU4aW9pekVhV05KcWxLeEdyZGxTa
↪%22%7D%5D%7D%5D%7D",
                "type": "application/linkset+json"
              }
            ],
            "replies": [
              {
                "href": "data:application/json,%7B%22%40context%22%3A%5B%22https%3A%2F
↪%2Fwww.w3.org%2F2018%2Fcredentials%2Fv1%22%2C%22https%3A%2F%2Fw3id.org
↪%2Factivityanchors%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fjws-2020
↪%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fed25519-2020%2Fv1%22%5D%2C
↪%22credentialSubject%22%3A%7B%22anchor%22%3A%22hl%3AuEiCdxlzbCM_pQopycBF8Q_
↪KA58ioizEaWNJqlKxGrdlSjg%22%2C%22href%22%3A%22hl%3AuEiBu8_7A4JQ1l9cRkxrInZ-
↪2cxgeuoerMOueXmRI-mmfog%22%2C%22profile%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%2C
```


(continues on next page)

(continued from previous page)


```
↪%22rel%22%3A%22linkset%22%2C%22type%22%3A%5B%22AnchorLink%22%5D%7D%2C%22id%22%3A
↪%22https%3A%2F%2Forb.domain2.com%2Fvc%2F65a306e4-1f86-480c-9fbd-c5e627363f0d%22%2C
↪%22issuanceDate%22%3A%222022-08-26T15%3A26%3A20.114938493Z%22%2C%22issuer%22%3A%22https
↪%3A%2F%2Forb.domain2.com%22%2C%22proof%22%3A%7B%22created%22%3A%222022-08-26T15%3A26
↪%3A20.116657916Z%22%2C%22domain%22%3A%22https%3A%2F%2Forb.domain2.com%22%2C
↪%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22%3A
↪%22z63YhBwGe5h5y3ChrjTCeSXxFNf98krSDCP3FGQDRSFFCFbC7BryMpR5gaboGiaqtDRY4tNqUSZPHHHDr7jT3jSzd
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain2.com%23W51yCsfyP-3uyHi9BhTTd9qBzPM14YaQk2A0bCybRbU%22%7D%2C%22type%22%3A
↪%5B%22VerifiableCredential%22%2C%22AnchorCredential%22%5D%7D",
                "type": "application/ld+json"
              }
            ]
          }
        ]
      },
      "startTime": "2022-08-26T15:26:20.174639514Z",
      "target": "https://w3id.org/activityanchors#AnchorWitness",
      "to": [
        "https://orb.domain1.com/services/orb",
        "https://www.w3.org/ns/activitystreams#Public"
      ],
      "type": "Offer"
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "did:web:orb.domain2.com:services:orb",
      "id": "https://orb.domain2.com/services/orb/activities/d5a83f39-f9e3-4833-b668-
↪134e21f089d3",
      "object": {
        "@context": "https://w3id.org/activityanchors/v1",
        "object": {
          "linkset": [
            {
              "anchor": "hl:uEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog",
              "author": [
                {
                  "href": "did:web:orb.domain2.com:services:orb"
                }
              ],
              "original": [
                {
                  "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A
↪%22hl%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%22%2C%22author%22%3A%5B%7B
↪%22href%22%3A%22did%3Aweb%3Aorb.domain2.com%3Aservices%3Aorb%22%7D%5D%2C%22item%22%3A
↪%5B%7B%22href%22%3A%22did%3Aorb%3AuAAA%3AEiDO4M4BoSmK4wFL-sSmFBhOyjeWfb2IuDovY1hRov5Cgg
↪%22%7D%5D%2C%22profile%22%3A%5B%7B%22href%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22
↪%7D%5D%7D%5D%7D",
                  "type": "application/linkset+json"
                }
              ],
              "profile": [
```


(continues on next page)

(continued from previous page)


```
                {
                  "href": "https://w3id.org/orb#v0"
                }
              ],
              "related": [
                {
                  "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A
↪%22hl%3AuEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog%22%2C%22profile%22%3A%5B%7B
↪%22href%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%2C%22via%22%3A%5B%7B%22href
↪%22%3A%22hl%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ2R4bHpiQ01fcFFvcHljQkY4UV9LQTU4aW9pekVhV05KcWxLeEdyZGxT
↪%22%7D%5D%7D%5D%7D",
                  "type": "application/linkset+json"
                }
              ],
              "replies": [
                {
                  "href": "data:application/json,%7B%22%40context%22%3A%5B%22https%3A%2F
↪%2Fwww.w3.org%2F2018%2Fcredentials%2Fv1%22%2C%22https%3A%2F%2Fw3id.org
↪%2Factivityanchors%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fjws-2020
↪%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fed25519-2020%2Fv1%22%5D%2C
↪%22credentialSubject%22%3A%7B%22anchor%22%3A%22hl%3AuEiCdxlzbCM_pQopycBF8Q_
↪KA58ioizEaWNJqlKxGrdlSjg%22%2C%22href%22%3A%22hl%3AuEiBu8_7A4JQ1l9cRkxrInZ-
↪2cxgeuoerMOueXmRI-mmfog%22%2C%22profile%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%2C
↪%22rel%22%3A%22linkset%22%2C%22type%22%3A%5B%22AnchorLink%22%5D%7D%2C%22id%22%3A
↪%22https%3A%2F%2Forb.domain2.com%2Fvc%2F65a306e4-1f86-480c-9fbd-c5e627363f0d%22%2C
↪%22issuanceDate%22%3A%222022-08-26T15%3A26%3A20.114938493Z%22%2C%22issuer%22%3A%22https
↪%3A%2F%2Forb.domain2.com%22%2C%22proof%22%3A%5B%7B%22created%22%3A%222022-08-26T15%3A26
↪%3A20.116657916Z%22%2C%22domain%22%3A%22https%3A%2F%2Forb.domain2.com%22%2C
↪%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22%3A
↪%22z63YhBwGe5h5y3ChrjTCeSXxFNf98krSDCP3FGQDRSFFCFbC7BryMpR5gaboGiaqtDRY4tNqUSZPHHHDr7jT3jSzd
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain2.com%23W51yCsfyP-3uyHi9BhTTd9qBzPM14YaQk2A0bCybRbU%22%7D%2C%7B%22created
↪%22%3A%222022-08-26T15%3A26%3A20.243Z%22%2C%22domain%22%3A%22http%3A%2F%2Forb.vct
↪%3A8077%2Fmaple2020%22%2C%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22
↪%3A
↪%22zr7L6vWBgVBLwPsbGnd59RWiv2t96wvwHCgRwjvLo3D69mvKpmQx6XE9qL2vR7c92LNE8xL58BAbGLWJqVb9WJBW
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain1.com%23c9lXUEfQVRceUV6FdwzzR19EMv4nZE-eopNnSmxum14%22%7D%5D%2C%22type%22
↪%3A%5B%22VerifiableCredential%22%2C%22AnchorCredential%22%5D%7D",
                  "type": "application/ld+json"
                }
              ]
            }
          ]
        },
        "type": "AnchorEvent",
        "url": "hl:uEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25C
↪"
      },
      "published": "2022-08-26T15:26:20.404295851Z",
```


(continues on next page)

(continued from previous page)

```
      "to": [
        "https://orb.domain2.com/services/orb/followers",
        "https://www.w3.org/ns/activitystreams#Public"
      ],
      "type": "Create"
    }
  ]
}
```

POST

A POST request to the inbox endpoint adds the activity contained in the request to the service's Inbox, which will be processed by the ActivityPub Inbox. This endpoint is restricted by authorization rules, i.e. the requester must sign the HTTP request. Some activities also have authorization rules such that the actor must be in the destination server's *followers* and/or *witnessing* collection.

Example

Post an *Invite* activity:

```
POST /services/orb/inbox HTTP/1.1
Host: orb.domain2.com
Content-Type: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate

{
  "@context": [
    "https://www.w3.org/ns/activitystreams",
    "https://w3id.org/activityanchors/v1"
  ],
  "actor": "https://orb.domain1.com/services/orb",
  "id": "https://orb.domain1.com/services/orb/activities/34e67f25-4832-4c62-a199-
↪5614ec3e5582",
  "object": "https://w3id.org/activityanchors#AnchorWitness",
  "target": "https://orb.domain2.com/services/orb",
  "to": "https://orb.domain2.com/services/orb",
  "type": "Invite"
}
```

Witnesses

Endpoint: /services/orb/witnesses

GET

The witnesses of this service are returned via this endpoint. If no paging parameters are specified in the URL then the response contains information about the *witnesses* collection, i.e. the links to the first and last page, as well as the total number of items in the collection. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request page information:

```
GET /services/orb/witnesses HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/witnesses",
  "type": "Collection",
  "totalItems": 19,
  "first": "https://orb.domain1.com/services/orb/witnesses?page=true",
  "last": "https://orb.domain1.com/services/orb/witnesses?page=true&page-num=4"
}
```

Request page 4:

```
GET /services/orb/witnesses??page=true&page-num=4 HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items from page 4:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/witnesses?page=true&page-num=4",
  "type": "CollectionPage",
  "totalItems": 19,
  "items": [
    "https://orb.domain2.com/services/orb",
    "https://orb.domain3.com/services/orb",
    "https://orb.domain4.com/services/orb"
  ]
}
```

Witnessing

Endpoint: /services/orb/witnessing

GET

The services that are witnessing anchor events for this service are returned via this endpoint. If no paging parameters are specified in the URL then the response contains information about the collection, i.e. the links to the first and last page, as well as the total number of items in the collection. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request page information:

```
GET /services/orb/witnessing HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/witnessing",
  "type": "Collection",
  "totalItems": 19,
  "first": "https://orb.domain1.com/services/orb/witnessing?page=true",
  "last": "https://orb.domain1.com/services/orb/witnessing?page=true&page-num=4"
}
```

Request first page:

```
GET /services/orb/witnessing?page=true HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items from the first page:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/witnessing?page=true&page-num=0",
  "type": "CollectionPage",
  "totalItems": 10,
  "next": "https://orb.domain1.com/services/orb/witnessing?page=true&page-num=1",
  "items": [
    "https://orb.domain2.com/services/orb",
    "https://orb.domain3.com/services/orb",
    "https://orb.domain4.com/services/orb"
  ]
}
```

Liked

Endpoint: /services/orb/liked

GET

The anchor events that are *liked* are returned via this endpoint. (Liked means that the anchors in the response were all added to the ledger.) If no paging parameters are specified in the URL then the response contains information about the collection, i.e. the links to the first and last page, as well as the total number of items in the collection. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request page information:

```
GET /services/orb/liked HTTP/1.1
Host: orb.domain3.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain3.com/services/orb/liked",
  "type": "OrderedCollection",
  "totalItems": 21,
  "first": "https://orb.domain3.com/services/orb/liked?page=true",
  "last": "https://orb.domain3.com/services/orb/liked?page=true&page-num=0"
}
```

Request first page:

```
GET /services/orb/liked?page=true HTTP/1.1
Host: orb.domain3.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items from the first page:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain3.com/services/orb/liked?page=true&page-num=0",
  "type": "OrderedCollectionPage",
  "totalItems": 21,
  "next": "https://orb.domain3.com/services/orb/liked?page=true&page-num=1",
  "orderedItems": [
    "hl:uEiDt0x4uc87k2EHTYavT-SjSkZ6ZjIz1S4iylbrOu5ZDcQ:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpRHQweDR1Yzg3azJFSFRZYXZULVNqU2taNlpqSXoxUzRpeWxick91NVpE
↪",
    "hl:uEiCLO_8LAn_dXKtkCWybunh5Fwhnhf8-zXFcqSvBgyKFnw:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpQ0xPXzhMQW5fZFhLdGtDV3lidW5oNUZ3aG5oZjgtelhGY3FTdkJneUtG
↪",
    "hl:uEiCcU17xqBJOppmw4WQtoPpF1G-asBkNsHAU4HM5bCnd2w:uoQ-
```

(continues on next page)

(continued from previous page)

```
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpQ2NVMTd4cUJKT3BwbXc0V1F0b1BwRjFHLWFzQmtOc0hBVTRITTViQ25k
↪",
    "hl:uEiDAdXUexB5Js-KpRfXV0C9uZq_vpk3nkykYlGB9f8-aYg:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpREFkWFVleEI1SnMtS3BSZlhWMEM5dVpxX3ZwazNua3lrWWxHQjlmOC1h
↪"
  ]
}
```

Likes

Endpoint: /services/orb/likes/[id]

GET

This endpoint returns a collection of *Like* activities for a given anchor. If no paging parameters are specified in the URL then the response contains information about the collection, i.e. the links to the first and last page, as well as the total number of items in the collection. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request page information:

```
GET /services/orb/likes/hl%3AuEiCwSSUKoJxrWj_A80KxPE6rM5XA0fhHRVoPX09uVddgSw%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ3dTU1VLb0p4cldqX0E4MEt4UEU2ck01WEEwZmhIUlZvUFgwOXVWZGRnU
↪HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/likes/hl%3AuEiCwSSUKoJxrWj_
↪A80KxPE6rM5XA0fhHRVoPX09uVddgSw%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ3dTU1VLb0p4cldqX0E4MEt4UEU2ck01WEEwZmhIUlZvUFgwOXVWZGRnU
↪",
  "type": "OrderedCollection",
  "totalItems": 1,
  "first": "https://orb.domain1.com/services/orb/likes/hl%3AuEiCwSSUKoJxrWj_
↪A80KxPE6rM5XA0fhHRVoPX09uVddgSw%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ3dTU1VLb0p4cldqX0E4MEt4UEU2ck01WEEwZmhIUlZvUFgwOXVWZGRnU
↪page=true",
  "last": "https://orb.domain1.com/services/orb/likes/hl%3AuEiCwSSUKoJxrWj_
↪A80KxPE6rM5XA0fhHRVoPX09uVddgSw%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ3dTU1VLb0p4cldqX0E4MEt4UEU2ck01WEEwZmhIUlZvUFgwOXVWZGRnU
↪page=true&page-num=0"
}
```

Request first page:

```
GET /services/orb/likes/hl%3AuEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25C
↪page=true&page-num=0 HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items from the first page:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain2.com/services/orb/likes/hl
↪%3AuEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25C
↪page=true&page-num=0",
  "orderedItems": [
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "https://orb.domain3.com/services/orb",
      "id": "https://orb.domain3.com/services/orb/activities/4880e2fc-972d-46db-bc19-
↪c08dc8668da7",
      "object": {
        "@context": "https://w3id.org/activityanchors/v1",
        "type": "AnchorEvent",
        "url": "hl:uEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25C
↪"
      },
      "published": "2022-08-26T15:26:21.171433692Z",
      "result": {
        "@context": "https://w3id.org/activityanchors/v1",
        "type": "AnchorEvent",
        "url": "hl:uEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW4zLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25C
↪"
      },
      "to": [
        "https://orb.domain1.com/services/orb",
        "https://orb.domain2.com/services/orb",
        "https://www.w3.org/ns/activitystreams#Public"
      ],
      "type": "Like"
    },
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "https://orb.domain1.com/services/orb",
      "id": "https://orb.domain1.com/services/orb/activities/5a053c65-eaed-4004-968b-
↪2ead5f94a238",
      "object": {
        "@context": "https://w3id.org/activityanchors/v1",
        "type": "AnchorEvent",
        "url": "hl:uEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25C
```

(continues on next page)

(continued from previous page)

```
↪"
      },
      "published": "2022-08-26T15:26:21.4337500662",
      "result": {
        "@context": "https://w3id.org/activityanchors/v1",
        "type": "AnchorEvent",
        "url": "hl:uEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25C
↪"
      },
      "to": [
        "https://orb.domain2.com/services/orb",
        "https://www.w3.org/ns/activitystreams#Public"
      ],
      "type": "Like"
    }
  ],
  "totalItems": 2,
  "type": "OrderedCollectionPage"
}
```

Shares

Endpoint: /services/orb/shares/[id]

GET

The *Create* activities that were *Announced* are returned via this endpoint. If no paging parameters are specified in the URL then the response contains information about the collection, i.e. the links to the first and last page, as well as the total number of items in the collection. A subsequent request may be made using parameters that include a specified page number in order to retrieve the actual items.

Example

Request page information:

```
GET /services/orb/shares/hl%3AuEiCwSSUKoJxrWj_A80KxPE6rM5XA0fhHRVoPX09uVddgSw%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ3dTU1VLb0p4cldqX0E4MEt4UEU2ck01WEEwZmhIUlZvUFgwOXVWZGRnU
↪HTTP/1.1
Host: orb.domain3.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains page information:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain3.com/services/orb/shares/hl%3AuEiCwSSUKoJxrWj_
↪A80KxPE6rM5XA0fhHRVoPX09uVddgSw%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ3dTU1VLb0p4cldqX0E4MEt4UEU2ck01WEEwZmhIUlZvUFgwOXVWZGRnU
↪",
```

(continues on next page)

(continued from previous page)


```
  "type": "OrderedCollection",
  "totalItems": 1,
  "first": "https://orb.domain3.com/services/orb/shares/hl%3AuEiCwSSUKoJxrWj_
↪A80KxPE6rM5XA0fhHRVoPX09uVddgSw%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ3dTU1VLb0p4cldqX0E4MEt4UEU2ck01WEEwZmhIUlZvUFgwOXVWZGRnU
↪page=true",
  "last": "https://orb.domain3.com/services/orb/shares/hl%3AuEiCwSSUKoJxrWj_
↪A80KxPE6rM5XA0fhHRVoPX09uVddgSw%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ3dTU1VLb0p4cldqX0E4MEt4UEU2ck01WEEwZmhIUlZvUFgwOXVWZGRnU
↪page=true&page-num=0"
}
```


Request first page:

```
GET /services/orb/shares/hl%3AuEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25CZ
↪page=true HTTP/1.1
Host: orb.domain3.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains items from the first page:


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain3.com/services/orb/shares/hl
↪%3AuEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25CZ
↪page=true&page-num=0",
  "orderedItems": [
    {
      "@context": "https://www.w3.org/ns/activitystreams",
      "actor": "https://orb.domain1.com/services/orb",
      "id": "https://orb.domain1.com/services/orb/activities/12922991-3170-4912-95b9-
↪f077b61eaaec",
      "object": {
        "items": [
          {
            "@context": "https://w3id.org/activityanchors/v1",
            "object": {
              "linkset": [
                {
                  "anchor": "hl:uEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog",
                  "author": [
                    {
                      "href": "did:web:orb.domain2.com:services:orb"
                    }
                  ],
                  "original": [
                    {
                      "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor
↪%22%3A%22hl%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%22%2C%22author%22%3A%5B
```


(continues on next page)

(continued from previous page)


```
↪%7B%22href%22%3A%22did%3Aweb%3Aorb.domain2.com%3Aservices%3Aorb%22%7D%5D%2C%22item%22
↪%3A%5B%7B%22href%22%3A%22did%3Aorb%3AuAAA%3AEiDO4M4BoSmK4wFL-
↪sSmFBhOyjeWfb2IuDovY1hRov5Cgg%22%7D%5D%2C%22profile%22%3A%5B%7B%22href%22%3A%22https%3A
↪%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%7D%5D%7D",
                        "type": "application/linkset+json"
                      }
                    ],
                    "profile": [
                      {
                        "href": "https://w3id.org/orb#v0"
                      }
                    ],
                    "related": [
                      {
                        "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor
↪%22%3A%22hl%3AuEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog%22%2C%22profile%22%3A%5B
↪%7B%22href%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%2C%22via%22%3A%5B%7B
↪%22href%22%3A%22hl%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ2R4bHpiQ01fcFFvcHljQkY4UV9LQTU4aW9pekVhV05KcWxLeEdyZGxT
↪%22%7D%5D%7D%5D%7D",
                        "type": "application/linkset+json"
                      }
                    ],
                    "replies": [
                      {
                        "href": "data:application/json,%7B%22%40context%22%3A%5B%22https%3A
↪%2F%2Fwww.w3.org%2F2018%2Fcredentials%2Fv1%22%2C%22https%3A%2F%2Fw3id.org
↪%2Factivityanchors%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fjws-2020
↪%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fed25519-2020%2Fv1%22%5D%2C
↪%22credentialSubject%22%3A%7B%22anchor%22%3A%22hl%3AuEiCdxlzbCM_pQopycBF8Q_
↪KA58ioizEaWNJqlKxGrdlSjg%22%2C%22href%22%3A%22hl%3AuEiBu8_7A4JQ1l9cRkxrInZ-
↪2cxgeuoerMOueXmRI-mmfog%22%2C%22profile%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%2C
↪%22rel%22%3A%22linkset%22%2C%22type%22%3A%5B%22AnchorLink%22%5D%7D%2C%22id%22%3A
↪%22https%3A%2F%2Forb.domain2.com%2Fvc%2F65a306e4-1f86-480c-9fbd-c5e627363f0d%22%2C
↪%22issuanceDate%22%3A%222022-08-26T15%3A26%3A20.114938493Z%22%2C%22issuer%22%3A%22https
↪%3A%2F%2Forb.domain2.com%22%2C%22proof%22%3A%5B%7B%22created%22%3A%222022-08-26T15%3A26
↪%3A20.116657916Z%22%2C%22domain%22%3A%22https%3A%2F%2Forb.domain2.com%22%2C
↪%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22%3A
↪%22z63YhBwGe5h5y3ChrjTCeSXxFNf98krSDCP3FGQDRSFFCFbC7BryMpR5gaboGiaqtDRY4tNqUSZPHHHDr7jT3jSzd
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain2.com%23W51yCsfyP-3uyHi9BhTTd9qBzPM14YaQk2A0bCybRbU%22%7D%2C%7B%22created
↪%22%3A%222022-08-26T15%3A26%3A20.243Z%22%2C%22domain%22%3A%22http%3A%2F%2Forb.vct
↪%3A8077%2Fmaple2020%22%2C%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22
↪%3A
↪%22zr7L6vWBgVBLwPsbGnd59RWiv2t96wvwHCgRwjvLo3D69mvKpmQx6XE9qL2vR7c92LNE8xL58BAbGLWJqVb9WJBW
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain1.com%23c9lXUEfQVRceUV6FdwzzR19EMv4nZE-eopNnSmxum14%22%7D%5D%2C%22type%22
↪%3A%5B%22VerifiableCredential%22%2C%22AnchorCredential%22%5D%7D",
                        "type": "application/ld+json"
                      }
                    ]
                  }
```


(continues on next page)

(continued from previous page)


```
              ]
            },
            "type": "AnchorEvent",
            "url": "hl:uEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQUZTbWhiWlNNT0NLOFlqcHpvWXJLVFRyUFpvNVBPa01HeTNTNnpPS25C
↪"
          }
        ],
        "totalItems": 1,
        "type": "Collection"
      },
      "published": "2022-08-26T15:26:20.787897406Z",
      "to": [
        "https://orb.domain1.com/services/orb/followers",
        "https://www.w3.org/ns/activitystreams#Public"
      ],
      "type": "Announce"
    }
  ],
  "totalItems": 1,
  "type": "OrderedCollectionPage"
}
```


Activities

Endpoint: /services/orb/activities/[id]

GET

This endpoint returns an activity for the specified ID.

Example

Request:

```
GET /services/orb/activities/91dbb2e2-1040-4fd5-bd2e-1bcd67bdda8a HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json; profile="https://www.w3.org/ns/activitystreams"
Accept-Encoding: gzip, deflate
```

Response contains the activity:


```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "id": "https://orb.domain1.com/services/orb/activities/91dbb2e2-1040-4fd5-bd2e-
↪1bcd67bdda8a",
  "type": "Accept",
  "actor": "https://orb.domain1.com/services/orb",
  "to": "https://orb.domain5.com/services/orb",
  "object": {
    "@context": "https://www.w3.org/ns/activitystreams",
```


(continues on next page)

(continued from previous page)

```
    "actor": "https://orb.domain5.com/services/orb",
    "id": "https://orb.domain5.com/services/orb/activities/b9f63dd7-b3da-414f-8c59-
↪894d964787c4",
    "object": "https://orb.domain2.com/services/orb",
    "to": "https://orb.domain1.com/services/orb",
    "type": "Follow"
  }
}
```

3.6.2 Sidetree Endpoints

Endpoints are defined to post Sidetree operations and to resolve DID documents.

Operations

Endpoint: /sidetree/v1/operations

POST

Post a Sidetree operation as per Sidetree DID Operations

Example

Post a *create* operation:

```
POST /sidetree/v1/operations HTTP/1.1
Host: orb.domain1.com
Content-Type: application/ld+json

{
  "delta": {
    "patches": [
      {
        "action": "add-public-keys",
        "publicKeys": [
          {
            "id": "createKey",
            "publicKeyJwk": {
              "crv": "P-256",
              "kty": "EC",
              "x": "kTpW2qcc66DyPWNnTSmaomtcGC0fOB2XC-OavrtSmOQ",
              "y": "_-2MbdKMjYOTSny4zSHyHU-L2sT9MUoDyQfRr2R_avE"
            },
            "purposes": [
              "authentication"
            ],
            "type": "JsonWebKey2020"
          },
          {
            "id": "auth",
```

(continues on next page)

(continued from previous page)

```
            "publicKeyJwk": {
              "crv": "Ed25519",
              "kty": "OKP",
              "x": "4BpSggpgRRtlQJKpPznhKfEon1OAlmr1MFjaN2sS4Ns",
              "y": ""
            },
            "purposes": [
              "assertionMethod"
            ],
            "type": "Ed25519VerificationKey2018"
          }
        ]
      },
      {
        "action": "add-services",
        "services": [
          {
            "id": "didcomm",
            "priority": 0,
            "recipientKeys": [
              "F1nHc1qea4QNfmcFjHvGxDNRogomARVhNpk8T812eWDD"
            ],
            "routingKeys": [
              "3xWoBwfwuyRH9ax82z6Lm24URNRJUoxg4PV6CcXBFvWr"
            ],
            "serviceEndpoint": "https://hub.example.com/.identity/
↪did:example:0123456789abcdef/",
            "type": "did-communication"
          }
        ]
      }
    ],
    "updateCommitment": "EiB1FFyDuSMNoMVKvSqZjybZxs_NfqA8WpMd0RTivT351Q"
  },
  "suffixData": {
    "anchorOrigin": "https://orb.domain1.com",
    "deltaHash": "EiDt8PTc-7SRaEoGOkkTTAUTUb8E3fbr4AMAw0Oa8lDZfQ",
    "recoveryCommitment": "EiDEBk7ejSyjddtZWyNA37gA0SdMvkSAqw7BxkLUQkpbyw"
  },
  "type": "create"
}
```

Response from a *create* operation:

```
{
  "@context": "https://w3id.org/did-resolution/v1",
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/jws-2020/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1"
    ],
```

(continues on next page)

(continued from previous page)


```
    "assertionMethod": [
      "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#auth"
    ],
    "authentication": [
      "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#createKey"
    ],
    "id": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    "service": [
      {
        "id": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#didcomm",
        "priority": 0,
        "recipientKeys": [
          "F1nHc1qea4QNfmcFjHvGxDNRogomARVhNpk8T812eWDD"
        ],
        "routingKeys": [
          "3xWoBwfwuyRH9ax82z6Lm24URNRJUoxg4PV6CcXBFvWr"
        ],
        "serviceEndpoint": "https://hub.example.com/.identity/
↪did:example:0123456789abcdef/",
        "type": "did-communication"
      }
    ],
    "verificationMethod": [
      {
        "controller": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
        "id": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#createKey",
        "publicKeyJwk": {
          "crv": "P-256",
          "kty": "EC",
          "x": "kTpW2qcc66DyPWNnTSmaomtcGC0fOB2XC-OavrtSmOQ",
          "y": "_-2MbdKMjYOTSny4zSHyHU-L2sT9MUoDyQfRr2R_avE"
        },
        "type": "JsonWebKey2020"
      },
      {
        "controller": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
        "id": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#auth",
        "publicKeyBase58": "G5oc2RXAWqCtF6w9K2qDTtPwPjWpSDk923MxWEKiVT6a",
        "type": "Ed25519VerificationKey2018"
      }
    ]
  },
  "didDocumentMetadata": {
    "equivalentId": [
      "did:orb:https:orb.domain1.com:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ"
    ],
    "method": {
      "anchorOrigin": "https://orb.domain1.com",
      "published": false,
      "recoveryCommitment": "EiDEBk7ejSyjddtZWyNA37gA0SdMvkSAqw7BxkLUQkpbyw",
      "updateCommitment": "EiB1FFyDuSMNoMVKvSqZjybZxs_NfqA8WpMd0RTivT351Q"
    }
```


(continues on next page)

(continued from previous page)

```
  }
}
```

Create document response will contain only one entry in equivalentId list in document metadata. The client can use this equivalentId entry with https hint to resolve document from the domain that is specified in the https hint.

If Orb domain supports unpublished operation caching the client will be able to immediately resolve unpublished DID document. The server response for unpublished document equals the above-mentioned *create* operation response.

If the client queries for DID document upon successful anchoring of create operation the document metadata will contain the following information:

- published flag is set to true
- canonicalId will be present
- multiple equivalent IDs will be present in equivalentId list
- versionId will be present

```
{
  "@context": "https://w3id.org/did-resolution/v1",
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/jws-2020/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1"
    ],
    "assertionMethod": [
      "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#auth"
    ],
    "authentication": [
      "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#createKey"
    ],
    "id": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    "service": [
      {
        "id": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#didcomm",
        "priority": 0,
        "recipientKeys": [
          "F1nHc1qea4QNfmcFjHvGxDNRogomARVhNpk8T812eWDD"
        ],
        "routingKeys": [
          "3xWoBwfwuyRH9ax82z6Lm24URNRJUoxg4PV6CcXBFvWr"
        ],
        "serviceEndpoint": "https://hub.example.com/.identity/
↪did:example:0123456789abcdef/",
        "type": "did-communication"
      }
    ],
    "verificationMethod": [
      {
        "controller": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
        "id": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#createKey",
        "publicKeyJwk": {
```

(continues on next page)

(continued from previous page)


```
          "crv": "P-256",
          "kty": "EC",
          "x": "kTpW2qcc66DyPWNnTSmaomtcGC0fOB2XC-OavrtSmOQ",
          "y": "_-2MbdKMjYOTSny4zSHyHU-L2sT9MUoDyQfRr2R_avE"
        },
        "type": "JsonWebKey2020"
      },
      {
        "controller": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
        "id": "did:orb:uAAA:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#auth",
        "publicKeyBase58": "G5oc2RXAWqCtF6w9K2qDTtPwPjWpSDk923MxWEKiVT6a",
        "type": "Ed25519VerificationKey2018"
      }
    ]
  },
  "didDocumentMetadata": {
    "canonicalId": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    "equivalentId": [
      "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
      "did:orb:hl:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpRGFKR05rRkpTTWxTZUtqRUZBSXZqQnhieWJyN1ZXakxweHBsaFhQeDhG
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    ],
    "method": {
      "anchorOrigin": "https://orb.domain1.com",
      "published": true,
      "recoveryCommitment": "EiDEBk7ejSyjddtZWyNA37gA0SdMvkSAqw7BxkLUQkpbyw",
      "updateCommitment": "EiB1FFyDuSMNoMVKvSqZjybZxs_NfqA8WpMd0RTivT351Q"
    },
    "versionId": "uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw"
  }
}
```


The Orb server will include the anchor-hash segment within the canonicalId property of the returned DID document metadata. The anchor-hash is set to the multihash (with multibase prefix) of the latest known AnchorCredential that contains a Create or Recover operation for the DID.

For more details about canonicalId see Canonical IDs

The Orb server will also include canonicalId as the fist item in equivalent IDs list. The second item in the list will be discoverable Cryptographic Hyperlink

Once the client is able to obtain canonicalId, the client should discontinue using 'un-anchored' identifier (DID that includes uAAA) and use either canonicalId or cryptographic hyperlink from equivalentId list.

Example

Post an *update* operation to add new service:

```
POST /sidetree/v1/operations HTTP/1.1
Host: orb.domain1.com
Content-Type: application/ld+json
```

(continues on next page)

(continued from previous page)


```
{
  "delta": {
    "patches": [
      {
        "action": "add-services",
        "services": [
          {
            "id": "newService",
            "serviceEndpoint": "http://hub.my-personal-server.com",
            "type": "SecureDataStore"
          }
        ]
      }
    ],
    "updateCommitment": "EiCosM6zJoafNLoFcvwMTxjqJymG7GRym56PwlP2Jz3Iqg"
  },
  "didSuffix": "EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
  "revealValue": "EiD4_J1zZ89P4uN8F2_R6a7kgge9s3posjMPMur4Jx0Z4A",
  "signedData": "eyJhbGciOiJFUzI1NiJ9.
↪eyJhbmNob3JGcm9tIjoxNjQ2Njg0MDIzLCJhbmNob3JVbnRpbCI6MTY0NjY4NDMyMywiZGVsdGFIYXNoIjoiRWlDN1B5cGYxSE5xN
↪T9GaRytJSm1KsUoUoQcKX6Pdk3DJ9qX-IyuRI9iwCl9oGDPMteA9M3ngU9XiXhJpiVJ6MhDGGhH7bR0jZ5HcuQ
↪",
  "type": "update"
}
```


Resolution response from Orb server upon successful anchoring of update operation: sidetree/v1/identifiers/did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ server


```
{
  "@context": "https://w3id.org/did-resolution/v1",
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/jws-2020/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1"
    ],
    "assertionMethod": [
      "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#auth"
    ],
    "authentication": [
      "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#createKey"
    ],
    "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    "service": [
      {
        "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#didcomm",
        "priority": 0,
```


(continues on next page)

(continued from previous page)


```
        "recipientKeys": [
          "F1nHc1qea4QNfmcFjHvGxDNRogomARVhNpk8T812eWDD"
        ],
        "routingKeys": [
          "3xWoBwfwuyRH9ax82z6Lm24URNRJUoxg4PV6CcXBFvWr"
        ],
        "serviceEndpoint": "https://hub.example.com/.identity/
↪did:example:0123456789abcdef/",
        "type": "did-communication"
      },
      {
        "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#newService",
        "serviceEndpoint": "http://hub.my-personal-server.com",
        "type": "SecureDataStore"
      }
    ],
    "verificationMethod": [
      {
        "controller":
↪"did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
        "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#createKey",
        "publicKeyJwk": {
          "crv": "P-256",
          "kty": "EC",
          "x": "kTpW2qcc66DyPWNnTSmaomtcGC0fOB2XC-OavrtSmOQ",
          "y": "_-2MbdKMjYOTSny4zSHyHU-L2sT9MUoDyQfRr2R_avE"
        },
        "type": "JsonWebKey2020"
      },
      {
        "controller":
↪"did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
        "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#auth",
        "publicKeyBase58": "G5oc2RXAWqCtF6w9K2qDTtPwPjWpSDk923MxWEKiVT6a",
        "type": "Ed25519VerificationKey2018"
      }
    ]
  },
  "didDocumentMetadata": {
    "canonicalId": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    "equivalentId": [
      "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
      "did:orb:hl:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpRGFKR05rRkpTTWxTZUtqRUZBSXZqQnhieWJyN1ZXakxweHBsaFhQeDhG
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
```


(continues on next page)

(continued from previous page)

```
  ],
   "method": {
     "anchorOrigin": "https://orb.domain1.com",
     "published": true,
     "recoveryCommitment": "EiDEBk7ejSyjddtZWyNA37gA0SdMvkSAqw7BxkLUQkpbyw",
     "updateCommitment": "EiCosM6zJoafNLoFcvwMTxjqJymG7GRym56PwlP2Jz3Iqg"
   },
   "versionId": "uEiB0lC69bPfaIYmRN0wBEYygbEiSFZpJEuIjbqFHnui0ig"
 }
}
```

Note that versionId specifies anchor-hash that contains update operation. You can use versionId to resolve document at specific version.

Example

Post a *recover* operation:

```
POST /sidetree/v1/operations HTTP/1.1
Host: orb.domain1.com
Content-Type: application/ld+json
{
  "delta": {
    "patches": [
      {
        "action": "add-public-keys",
        "publicKeys": [
          {
            "id": "recoveryKey",
            "publicKeyJwk": {
              "crv": "P-256",
              "kty": "EC",
              "x": "-IRdzsImjgsrto6r_MzU9LjecdeBg153ixNSyUYSiwI",
              "y": "q5g_WtYdCxjXyey4ASH9N7rg9CIwY9guVoD269mTq4k"
            },
            "purposes": [
              "authentication"
            ],
            "type": "JsonWebKey2020"
          },
          {
            "id": "auth",
            "publicKeyJwk": {
              "crv": "Ed25519",
              "kty": "OKP",
              "x": "6fEptoI2zIVlR9YQHar_xUyWlhnYH6WjJoI-tTIwopY",
              "y": ""
            },
            "purposes": [
              "assertionMethod"
            ],
            "type": "Ed25519VerificationKey2018"
          }
```

(continues on next page)

(continued from previous page)

```
        ]
      },
      {
        "action": "add-services",
        "services": [
          {
            "id": "didcomm",
            "priority": 0,
            "recipientKeys": [
              "EvwVTTeLzVRNxX9vETVv8xRSsqfUGi2r3a6DzrhSMTGc"
            ],
            "routingKeys": [
              "EaMUBZMAYZU8CWcioQBxT6dLjhDvbRjDyVisttixAzpc"
            ],
            "serviceEndpoint": "https://hub.example.com/.identity/
↪did:example:0123456789abcdef/",
            "type": "did-communication"
          }
        ]
      }
    ],
    "updateCommitment": "EiCZDsrREJOpB49QfIPCU-74VXM760gTJGlkhvQAgwdcMQ"
  },
  "didSuffix": "EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
  "revealValue": "EiAkLgBNtJVpSHGzuWceujPdw0A3p_tK3cXhDndQG3JjPQ",
  "signedData": "eyJhbGciOiJFUzI1NiJ9.
↪eyJhbmNob3JGcm9tIjoxNjQ2Njg0MDI4LCJhbmNob3JPcmlnaW4iOiJodHRwczovL29yYi5kb21haW4xLmNvbSIsImFuY2hvclVud
↪EBbqSNSe6vZHOJMsxXkv7tRk1g6k28qzOWwhqAP8XIB_4mpzJEhoVaVSKT14LIOTQBAaQXad-PV52WnwoyBWvg
↪",
  "type": "recover"
}
```

Upon successful anchoring of recover operation the Orb server will return the following resolution response for the following request: sidetree/v1/identifiers/did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ

```
{
  "@context": "https://w3id.org/did-resolution/v1",
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/jws-2020/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1"
    ],
    "assertionMethod": [
      "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#auth"
    ],
    "authentication": [
      "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#recoveryKey"
    ],
```

(continues on next page)

(continued from previous page)


```
    "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    "service": [
      {
        "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#didcomm",
        "priority": 0,
        "recipientKeys": [
          "EvwVTTeLzVRNxX9vETVv8xRSsqfUGi2r3a6DzrhSMTGc"
        ],
        "routingKeys": [
          "EaMUBZMAYZU8CWcioQBxT6dLjhDvbRjDyVisttixAzpc"
        ],
        "serviceEndpoint": "https://hub.example.com/.identity/
↪did:example:0123456789abcdef/",
        "type": "did-communication"
      }
    ],
    "verificationMethod": [
      {
        "controller":
↪"did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
        "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#recoveryKey",
        "publicKeyJwk": {
          "crv": "P-256",
          "kty": "EC",
          "x": "-IRdzsImjgsrto6r_MzU9LjecdeBg153ixNSyUYSiwI",
          "y": "q5g_WtYdCxjXyey4ASH9N7rg9CIwY9guVoD269mTq4k"
        },
        "type": "JsonWebKey2020"
      },
      {
        "controller":
↪"did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
        "id": "did:orb:uEiDaJGNkFJSMlSeKjEFAIvjBxbybr7VWjLpxplhXPx8FCw:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ#auth",
        "publicKeyBase58": "GkDHCaxe2yyD5SHRZPWTd61rzsrx66aimzJAV1SNYPpH",
        "type": "Ed25519VerificationKey2018"
      }
    ]
  },
  "didDocumentMetadata": {
    "canonicalId": "did:orb:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    "equivalentId": [
      "did:orb:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
      "did:orb:hl:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpRFU1MmViUE9SU2pPV3MyTGg1cFRPQkZWbGlvUE55VmRPZnNyTDJmZUg1U
```


(continues on next page)

(continued from previous page)

```
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    ],
    "method": {
      "anchorOrigin": "https://orb.domain1.com",
      "published": true,
      "recoveryCommitment": "EiDcLuyCK7jVTt4wBV3OiHXb_a8Zcd0nM0KgMyKNreZFww",
      "updateCommitment": "EiCZDsrREJOpB49QfIPCU-74VXM760gTJGlkhvQAgwdcMQ"
    },
    "versionId": "uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg"
  }
}
```

Note that canonicalId has changed - it includes anchor-hash that contains recover operation. At this point the client should discontinue using old canonicalId and use new canonicalId.

For more details about canonicalId see Canonical IDs

Example

Post a *deactivate* operation:

```
POST /sidetree/v1/operations HTTP/1.1
Host: orb.domain1.com
Content-Type: application/ld+json
{
  "didSuffix": "EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
  "revealValue": "EiDdHkE3GeAfTcPQdrVi8ET2_4NbRl9emFigCKdx9hp1yQ",
  "signedData": "eyJhbGciOiJFUzI1NiJ9.
↪eyJhbmNob3JGcm9tIjoxNjQ2Njg0MDM4LCJkaWRTdWZmaXgiOiJFaURJRnU4QmV2LW5Kc3E4dXcyeEdIS3RMS3I2YUxuY0ZNbk9sU
↪8lC5SPQO6FsvXCOCly5WNt33muD5SEAHC3iOLG1pGc0pWyct7R5MCOMD80xZwB_KnJManW6eMuVY4lZI6UmuTw
↪",
  "type": "deactivate"
}
```

Upon successful anchoring of deactivate operation the Orb server will return the following resolution response for the following request: sidetree/v1/identifiers/did:orb:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:EiDIFu8Bev-nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ

```
{
  "@context": "https://w3id.org/did-resolution/v1",
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1"
    ],
    "id": "did:orb:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ"
  },
  "didDocumentMetadata": {
    "canonicalId": "did:orb:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
    "deactivated": true,
    "equivalentId": [
      "did:orb:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
```

(continues on next page)

(continued from previous page)

```
      "did:orb:hl:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpRFU1MmViUE9SU2pPV3MyTGg1cFRPQkZWbGlvUE55VmRPZnNyTDJmZUg1U
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ",
      "did:orb:https:shared.domain.
↪com:uEiDU52ebPORSjOWs2Lh5pTOBFVlioPNyVdOfsrL2feH5Pg:EiDIFu8Bev-
↪nJsq8uw2xGHKtLKr6aLncFMnOlSQ9pfP4VQ"
    ],
    "method": {
      "anchorOrigin": "https://orb.domain1.com",
      "published": true
    },
    "versionId": "uEiB7tOd2S0lyQxQYaR0PcgbEQHhSS1PNplvlP5Qtfu-kjw"
  }
}
```

Note that deactivated flag in the document metadata has been set to true and that DID document is an empty document that contains only id. Once DID document has been deactivated it is no longer possible to recover or modify DID document.

Identifiers

Endpoint: /sidetree/v1/identifiers/[id]

This endpoint supports the resolution of DID documents for two did methods:

- did:orb
- did:web

GET

Resolve a DID document as per Sidetree DID Resolution

DID Orb Method Example

Resolve a DID document using a canonical DID:

```
GET /sidetree/v1/identifiers/did:orb:uEiBwYoY8R0tXjX6G-
↪YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q HTTP/1.1
Host: orb.domain2.com
Accept: application/ld+json
Accept-Encoding: gzip, deflate
```

Response contains DID resolution result:

```
{
  "@context": "https://w3id.org/did-resolution/v1",
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/jws-2020/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1"
    ],
```

(continues on next page)

(continued from previous page)


```
    "assertionMethod": [
      "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_RbZx3RCM4aThZ0-
↪QeP0L9x7eoYwJBK2n_a0Um5Q#auth"
    ],
    "authentication": [
      "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_RbZx3RCM4aThZ0-
↪QeP0L9x7eoYwJBK2n_a0Um5Q#recoveryKey"
    ],
    "id": "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_RbZx3RCM4aThZ0-
↪QeP0L9x7eoYwJBK2n_a0Um5Q",
    "service": [
      {
        "id": "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_
↪RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q#didcomm",
        "priority": 0,
        "recipientKeys": [
          "2qN15Qwk1uEi19BYJ19zshrxLHkKa5QvvXkLjdn56AaX"
        ],
        "routingKeys": [
          "3kRwP6VuwcbNSzevReUpfDWwyVGRS4YtqxB69DAs3VFN"
        ],
        "serviceEndpoint": "https://hub.example.com/.identity/
↪did:example:0123456789abcdef/",
        "type": "did-communication"
      }
    ],
    "verificationMethod": [
      {
        "controller": "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_
↪RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q",
        "id": "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_
↪RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q#recoveryKey",
        "publicKeyJwk": {
          "crv": "P-256",
          "kty": "EC",
          "x": "MhxmsoUl3ACv0aZSJejEp6kxEO7QzSHedcYKN-3o0xw",
          "y": "7-TWvrUbnBkFFrqZ_nqVwFTgQyJE_mg11e3ckUEThvQ"
        },
        "type": "JsonWebKey2020"
      },
      {
        "controller": "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_
↪RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q",
        "id": "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_
↪RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q#auth",
        "publicKeyBase58": "4bkaQc1vU79nTbzUwyBSYtiv2pbb9mTYxHP1A7hGbyU3",
        "type": "Ed25519VerificationKey2018"
      }
    ]
  },
  "didDocumentMetadata": {
    "canonicalId": "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_
```


(continues on next page)

(continued from previous page)


```
↪RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q",
    "equivalentId": [
      "did:orb:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_RbZx3RCM4aThZ0-
↪QeP0L9x7eoYwJBK2n_a0Um5Q",
      "did:orb:hl:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:uoQ-
↪BeEJpcGZzOi8vYmFma3JlaWRxbWtkZHlyMmxrNmd4NWJ4enJneGdraHRqczNzcHJ5bmtkaHd3bGkzN2l4eXhzbHB5bnU:EiAzXL_
↪RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q"
    ],
    "method": {
      "anchorOrigin": "https://orb.domain1.com",
      "published": true,
      "publishedOperations": [
        {
          "anchorOrigin": "https://orb.domain1.com",
          "canonicalReference": "uEiDb2k2i5gYlgTg8GA8gN7apTp_JOkLgUCDY9A5eLOTXsQ",
          "equivalentReferences": [
            "hl:uEiDb2k2i5gYlgTg8GA8gN7apTp_JOkLgUCDY9A5eLOTXsQ:uoQ-
↪BeEJpcGZzOi8vYmFma3JlaWczM2pnMmZ6cWdld2F0cXBheWI0cWRwbnZqajJwNHNvc2M0YmljYndodWJ6cGN6emd4d2U
↪"
          ],
          "operationRequest":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtcHVibGljLWtleXMiLCJwdWJsaWNLZXlzIjpbeyJpZCI6ImNyZWF0
↪",
          "protocolVersion": 0,
          "transactionNumber": 0,
          "transactionTime": 1645800397,
          "type": "create"
        },
        {
          "anchorOrigin": "https://orb.domain1.com",
          "canonicalReference": "uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ",
          "equivalentReferences": [
            "hl:uEiBwYoY8R0tXjX6G-YmuZR5pluT44aoZ7WWjf0XxeS34bQ:uoQ-
↪BeEJpcGZzOi8vYmFma3JlaWRxbWtkZHlyMmxrNmd4NWJ4enJneGdraHRqczNzcHJ5bmtkaHd3bGkzN2l4eXhzbHB5bnU
↪"
          ],
          "operationRequest":
↪"eyJkZWx0YSI6eyJwYXRjaGVzIjpbeyJhY3Rpb24iOiJhZGQtcHVibGljLWtleXMiLCJwdWJsaWNLZXlzIjpbeyJpZCI6InJlY292Z
↪",
          "protocolVersion": 0,
          "transactionNumber": 0,
          "transactionTime": 1645800399,
          "type": "recover"
        }
      ],
      "recoveryCommitment": "EiBNlrtr2FzPwvAPQ1shC8lENaNkmsHepC3FDdvo3CL8YA",
      "updateCommitment": "EiAls4yWlFEb0XpdbUbV5i864LOU_NV0lEozrbah7k21lg"
    }
  }
}
```

DID Parameters

Orb supports the following DID query parameters:

versionId

Identifies a specific version of a DID document to be resolved. In orb the version ID specifies anchor hash that includes relevant DID operation. If present, the associated value MUST be an ASCII string.

```
GET /sidetree/v1/identifiers/did:orb:uEiBwYoY8R0tXjX6G-
↪YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q?
↪versionId=uEiCRI0FqR6cFVQ8rOPOsD-muNo8k_m7mYU_RPTYLPxdBvg HTTP/1.1
Host: orb.domain2.com
Accept: application/ld+json
Accept-Encoding: gzip, deflate
```

versionTime

Identifies a certain version timestamp of a DID document to be resolved. That is, the DID document that was valid for a DID at a certain time. If present, the associated value MUST be an ASCII string which is a valid XML datetime value. This datetime value MUST be normalized to UTC 00:00:00 and without sub-second decimal precision. For example: 2020-12-20T19:17:47Z.

```
GET /sidetree/v1/identifiers/did:orb:uEiBwYoY8R0tXjX6G-
↪YmuZR5pluT44aoZ7WWjf0XxeS34bQ:EiAzXL_RbZx3RCM4aThZ0-QeP0L9x7eoYwJBK2n_a0Um5Q?
↪versionTime=2022-03-04T17:40:59Z HTTP/1.1
Host: orb.domain2.com
Accept: application/ld+json
Accept-Encoding: gzip, deflate
```

For more details about versionId and versionTime queries see DID Parameters

DID Web Method Example

Request:

```
GET /sidetree/v1/identifiers/did:web:orb.domain3.
↪com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ  HTTP/1.1
Host: orb.domain3.com
Accept: application/json
```

Response:

```
{
  "@context": "https://w3id.org/did-resolution/v1",
  "didDocument": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/jws-2020/v1",
      "https://w3id.org/security/suites/ed25519-2018/v1"
    ],
    "alsoKnownAs": [
```

(continues on next page)

(continued from previous page)

```
↪"did:orb:uEiDDV4Yn9wx0c5dlrYT90TsjB6fg2Gc6x91X1gG3O5oJIA:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3m
↪",
      "did:orb:hl:uEiDDV4Yn9wx0c5dlrYT90TsjB6fg2Gc6x91X1gG3O5oJIA:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW4zLmNvbS9jYXMvdUVpRERWNFluOXd4MGM1ZGxyWVQ5MFRzakI2ZmcyR2M2eDkxWDFnRzNPNW9K
↪"
    ],
    "assertionMethod": [
      "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ#auth"
    ],
    "authentication": [
      "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ
↪#createKey"
    ],
    "id": "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ",
    "service": [
      {
        "id": "did:web:orb.domain3.
↪com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ#didcomm",
        "priority": 0,
        "recipientKeys": [
          "2FE8JwxEhd3apwqQQxpURmN4iiaM8H7cCAvU1K3D1Mmr"
        ],
        "serviceEndpoint": "https://hub.example.com/.identity/
↪did:example:0123456789abcdef/",
        "type": "did-communication"
      }
    ],
    "verificationMethod": [
      {
        "controller": "did:web:orb.domain3.
↪com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ",
        "id": "did:web:orb.domain3.
↪com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ#createKey",
        "publicKeyJwk": {
          "crv": "P-256",
          "kty": "EC",
          "x": "0Di-FS6Y9v8QyNyswEPdHJ6HK_Yx2Ek-OLsfyEcKyLQ",
          "y": "MsQvvYkqcvRn4ndZCMU7JjTq1sUXpt3xpjaldNLiLxQ"
        },
        "type": "JsonWebKey2020"
      },
      {
        "controller": "did:web:orb.domain3.
↪com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ",
        "id": "did:web:orb.domain3.
↪com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ#auth",
        "publicKeyBase58": "XpPrnB4DAiQTXL2dDCoenje8aVtocY3FqMFcy8w6fBK",
        "type": "Ed25519VerificationKey2018"
      }
    ]
  }
```

(continues on next page)

(continued from previous page)

```
}
```

3.6.3 DID Web File Endpoint

The DID Web file endpoint is used for resolution of did:web method documents by universal resolver and other did:web resolvers. This endpoint will serve DID document only not DID resolution result.

Endpoint: /scid/{id}/did.json

GET

Example

Request:

```
GET /scid/EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ/did.json  HTTP/1.1
Host: orb.domain3.com
Accept: application/json
```

Response:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/jws-2020/v1",
    "https://w3id.org/security/suites/ed25519-2018/v1"
  ],
  "alsoKnownAs": [

↪"did:orb:uEiDDV4Yn9wx0c5dlrYT90TsjB6fg2Gc6x91X1gG3O5oJIA:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3m
↪",
    "did:orb:hl:uEiDDV4Yn9wx0c5dlrYT90TsjB6fg2Gc6x91X1gG3O5oJIA:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW4zLmNvbS9jYXMvdUVpRERWNFluOXd4MGM1ZGxyWVQ5MFRzakI2ZmcyR2M2eDkxWDFnRzNPNW9K
↪"
  ],
  "assertionMethod": [
    "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ#auth"
  ],
  "authentication": [
    "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ
↪#createKey"
  ],
  "id": "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ",
  "service": [
    {
      "id": "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ
↪#didcomm",
      "priority": 0,
      "recipientKeys": [
        "2FE8JwxEhd3apwqQQxpURmN4iiaM8H7cCAvU1K3D1Mmr"
      ],
```

(continues on next page)

(continued from previous page)

```
      "serviceEndpoint": "https://hub.example.com/.identity/did:example:0123456789abcdef/
↪",
      "type": "did-communication"
    }
  ],
  "verificationMethod": [
    {
      "controller": "did:web:orb.domain3.
↪com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ",
      "id": "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ
↪#createKey",
      "publicKeyJwk": {
        "crv": "P-256",
        "kty": "EC",
        "x": "0Di-FS6Y9v8QyNyswEPdHJ6HK_Yx2Ek-OLsfyEcKyLQ",
        "y": "MsQvvYkqcvRn4ndZCMU7JjTq1sUXpt3xpjaldNLiLxQ"
      },
      "type": "JsonWebKey2020"
    },
    {
      "controller": "did:web:orb.domain3.
↪com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ",
      "id": "did:web:orb.domain3.com:scid:EiCNuOisUYmxNfT2HugwwMYQAyzLr9FxI7tbT3eb5r3mFQ
↪#auth",
      "publicKeyBase58": "XpPrnB4DAiQTXL2dDCoenje8aVtocY3FqMFcy8w6fBK",
      "type": "Ed25519VerificationKey2018"
    }
  ]
}
```

3.6.4 .well-known Endpoints

The .well-known endpoints are used for discovery of services within an Orb domain.

did-orb

Endpoint: /.well-known/did-orb

GET

Example

Request:

```
GET /.well-known/did-orb HTTP/1.1
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "resolutionEndpoint": "https://orb.domain1.com/sidetree/v1/identifiers",
  "operationEndpoint": "https://orb.domain1.com/sidetree/v1/operations"
}
```

host-meta

Endpoint: /.well-known/host-meta

GET

Example

Request:

```
GET /.well-known/host-meta HTTP/1.1
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "links": [
    {
      "rel": "self",
      "type": "application/jrd+json",
      "template": "https://orb.domain1.com/.well-known/webfinger?resource={uri}"
    },
    {
      "rel": "self",
      "type": "application/activity+json",
      "href": "https://orb.domain1.com/services/orb"
    }
  ]
}
```

host-meta.json

Endpoint: /.well-known/host-meta.json

GET

Example

Request:

```
GET /.well-known/host-meta.json HTTP/1.1
Host: orb.domain1.com
```

Response:

```
{
  "links": [
    {
      "rel": "self",
      "type": "application/jrd+json",
      "template": "https://orb.domain1.com/.well-known/webfinger?resource={uri}"
    },
    {
      "rel": "self",
      "type": "application/activity+json",
      "href": "https://orb.domain1.com/services/orb"
    }
  ]
}
```

did.json

Endpoint: /.well-known/did.json

GET

Example

Request:

```
GET /.well-known/did.json HTTP/1.1
Host: orb.domain1.com
```

Response:

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:web:orb.domain1.com",
  "verificationMethod": [
    {
      "id": "did:web:orb.domain1.com#orb1key2",
      "controller": "did:web:orb.domain1.com",
      "type": "Ed25519VerificationKey2018",
      "publicKeyBase58": "Cd5DEcovdeweWWkMzXL5XmvtL5ov1jgcf7i6itnbVz8m"
```

(continues on next page)

(continued from previous page)

```
    },
    {
      "id": "did:web:orb.domain1.com#orb1key",
      "controller": "did:web:orb.domain1.com",
      "type": "Ed25519VerificationKey2018",
      "publicKeyBase58": "245RycEHD7YAaaELTdUBLGF66iehCzpT3qy1jHLZoTMX"
    }
  ],
  "authentication": [
    "did:web:orb.domain1.com#orb1key2",
    "did:web:orb.domain1.com#orb1key"
  ],
  "assertionMethod": [
    "did:web:orb.domain1.com#orb1key2",
    "did:web:orb.domain1.com#orb1key"
  ],
  "capabilityDelegation": [
    "did:web:orb.domain1.com#orb1key2",
    "did:web:orb.domain1.com#orb1key"
  ],
  "capabilityInvocation": [
    "did:web:orb.domain1.com#orb1key2",
    "did:web:orb.domain1.com#orb1key"
  ]
}
```

webfinger

Endpoint: /.well-known/webfinger?resource={uri}

GET

Example #1

Request information about the domain, including the VCT (if configured):

```
GET /.well-known/webfinger?resource=https://orb.domain1.com
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "subject": "https://orb.domain1.com",
  "properties": {
    "https://trustbloc.dev/ns/ledger-type": "vct-v1"
  },
  "links": [
    {
      "rel": "self",
```

(continues on next page)

(continued from previous page)

```
      "type": "application/jrd+json",
      "href": "https://orb.domain1.com"
    },
    {
      "rel": "vct",
      "type": "application/jrd+json",
      "href": "http://orb.vct:8077/maple2022"
    }
  ]
}
```

Example #2

Request information about the resolution endpoint:

```
GET /.well-known/webfinger?resource=https://orb.domain1.com/sidetree/v1/identifiers
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "subject": "https://orb.domain1.com/sidetree/v1/identifiers",
  "properties": {
    "https://trustbloc.dev/ns/min-resolvers": 1
  },
  "links": [
    {
      "rel": "self",
      "href": "https://orb.domain1.com/sidetree/v1/identifiers"
    }
  ]
}
```

Example #3

Request information about the *operations* endpoint:

```
GET /.well-known/webfinger?resource=https://orb.domain1.com/sidetree/v1/operations
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "subject": "https://orb.domain1.com/sidetree/v1/operations",
  "links": [
    {
      "rel": "self",
```

(continues on next page)

(continued from previous page)

```
      "href": "https://orb.domain1.com/sidetree/v1/operations"
    }
  ]
}
```

Example #4

Request information about a specific DID:

```
GET /.well-known/webfinger?
↪resource=did:orb:uEiDYK49ULtocB0ecZB5kzdo1oCAbJEJmAr65xpoWDq3Y1w:EiAWa2VAwDV3iC7q2nWygTy4b5AuwfTnf1g_
↪ZYPZJFnJRA
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "properties": {
    "https://trustbloc.dev/ns/anchor-origin": "https://orb.domain1.com",
    "https://trustbloc.dev/ns/min-resolvers": 1
  },
  "links": [
    {
      "rel": "self",
      "type": "application/did+ld+json",
      "href": "https://orb.domain1.com/sidetree/v1/identifiers/did:orb:uEiB9vhEm-
↪4i8Vn1NSLcjYH50IsLCuGYDsKHpCEn6T0VHMg:EiAWa2VAwDV3iC7q2nWygTy4b5AuwfTnf1g_ZYPZJFnJRA"
    },
    {
      "rel": "via",
      "type": "application/ld+json",
      "href": "hl:uEiB9vhEm-4i8Vn1NSLcjYH50IsLCuGYDsKHpCEn6T0VHMg:uoQ-
↪BeEJpcGZzOi8vYmFma3JlaWQ1eHlpc242NGl4cmxoMnRraXc0cndhN3R1ZWxibWZvZGdhb3lrZDJpaWpoNWU2cmtoZ2k
↪"
    },
    {
      "rel": "service",
      "type": "application/activity+json",
      "href": "https://orb.domain1.com/services/orb"
    }
  ]
}
```

Example #5

Request information about an Anchor Linkset which also includes alternate links to the file:

```
GET /.well-known/webfinger?resource=https://orb.domain1.com/cas/
↪uEiCYEQhiFYGnf74G6jcOz7tRdJ6xaA_eakBhsHxu0KJJBg
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "subject": "https://orb.domain1.com/cas/uEiCYEQhiFYGnf74G6jcOz7tRdJ6xaA_eakBhsHxu0KJJBg
↪",
  "links": [
    {
      "rel": "self",
      "type": "application/ld+json",
      "href": "https://orb.domain1.com/cas/uEiCYEQhiFYGnf74G6jcOz7tRdJ6xaA_
↪eakBhsHxu0KJJBg"
    },
    {
      "rel": "alternate",
      "type": "application/ld+json",
      "href": "https://orb.domain2.com/cas/uEiCYEQhiFYGnf74G6jcOz7tRdJ6xaA_
↪eakBhsHxu0KJJBg"
    }
  ]
}
```

nodeinfo

Endpoint: /.well-known/nodeinfo

GET

Returns the NodeInfo endpoints that may be queried to provide general information about an Orb server.

Example

Request:

```
GET /.well-known/nodeinfo HTTP/1.1
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "links": [
```

(continues on next page)

(continued from previous page)

```
    {
      "rel": "http://nodeinfo.diaspora.software/ns/schema/2.0",
      "href": "https://orb.domain1.com/nodeinfo/2.0"
    },
    {
      "rel": "http://nodeinfo.diaspora.software/ns/schema/2.1",
      "href": "https://orb.domain1.com/nodeinfo/2.1"
    }
  ]
}
```

3.6.5 CAS Endpoint

Endpoint: /cas/[id]

GET

Returns content stored in the Content Addressable Storage (CAS). The ID is either an IPFS CID or the hash of the content.

Example

Request an Anchor Linkset using its hash as the ID:

```
GET /cas/uEiAFSmhbZSMOCK8YjpzoYrKTTrPZo5POkMGy3S6zOKnBgg HTTP/1.1
Host: orb.domain1.com
Accept-Encoding: gzip, deflate
```

Response contains the Anchor Linkset:

```
{
  "linkset": [
    {
      "anchor": "hl:uEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog",
      "author": [
        {
          "href": "did:web:orb.domain2.com:services:orb"
        }
      ],
      "original": [
        {
          "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A%22hl
↪%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%22%2C%22author%22%3A%5B%7B%22href%22
↪%3A%22did%3Aweb%3Aorb.domain2.com%3Aservices%3Aorb%22%7D%5D%2C%22item%22%3A%5B%7B
↪%22href%22%3A%22did%3Aorb%3AuAAA%3AEiDO4M4BoSmK4wFL-sSmFBhOyjeWfb2IuDovY1hRov5Cgg%22%7D
↪%5D%2C%22profile%22%3A%5B%7B%22href%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D
↪%7D%5D%7D",
          "type": "application/linkset+json"
        }
      ],
      "profile": [
```

(continues on next page)

(continued from previous page)


```
        {
          "href": "https://w3id.org/orb#v0"
        }
      ],
      "related": [
        {
          "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A%22hl
↪%3AuEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog%22%2C%22profile%22%3A%5B%7B%22href
↪%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%2C%22via%22%3A%5B%7B%22href%22%3A
↪%22hl%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW4yLmNvbS9jYXMvdUVpQ2R4bHpiQ01fcFFvcHljQkY4UV9LQTU4aW9pekVhV05KcWxLeEdyZGxT
↪%22%7D%5D%7D%5D%7D",
          "type": "application/linkset+json"
        }
      ],
      "replies": [
        {
          "href": "data:application/json,%7B%22%40context%22%3A%5B%22https%3A%2F%2Fwww.
↪w3.org%2F2018%2Fcredentials%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Factivityanchors%2Fv1
↪%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fjws-2020%2Fv1%22%2C%22https%3A%2F
↪%2Fw3id.org%2Fsecurity%2Fsuites%2Fed25519-2020%2Fv1%22%5D%2C%22credentialSubject%22%3A
↪%7B%22anchor%22%3A%22hl%3AuEiCdxlzbCM_pQopycBF8Q_KA58ioizEaWNJqlKxGrdlSjg%22%2C%22href
↪%22%3A%22hl%3AuEiBu8_7A4JQ1l9cRkxrInZ-2cxgeuoerMOueXmRI-mmfog%22%2C%22profile%22%3A
↪%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%2C%22rel%22%3A%22linkset%22%2C%22type%22%3A%5B
↪%22AnchorLink%22%5D%7D%2C%22id%22%3A%22https%3A%2F%2Forb.domain2.com%2Fvc%2F65a306e4-
↪1f86-480c-9fbd-c5e627363f0d%22%2C%22issuanceDate%22%3A%222022-08-26T15%3A26%3A20.
↪114938493Z%22%2C%22issuer%22%3A%22https%3A%2F%2Forb.domain2.com%22%2C%22proof%22%3A%5B
↪%7B%22created%22%3A%222022-08-26T15%3A26%3A20.116657916Z%22%2C%22domain%22%3A%22https
↪%3A%2F%2Forb.domain2.com%22%2C%22proofPurpose%22%3A%22assertionMethod%22%2C
↪%22proofValue%22%3A
↪%22z63YhBwGe5h5y3ChrjTCeSXxFNf98krSDCP3FGQDRSFFCFbC7BryMpR5gaboGiaqtDRY4tNqUSZPHHHDr7jT3jSzd
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain2.com%23W51yCsfyP-3uyHi9BhTTd9qBzPM14YaQk2A0bCybRbU%22%7D%2C%7B%22created
↪%22%3A%222022-08-26T15%3A26%3A20.243Z%22%2C%22domain%22%3A%22http%3A%2F%2Forb.vct
↪%3A8077%2Fmaple2020%22%2C%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22
↪%3A
↪%22zr7L6vWBgVBLwPsbGnd59RWiv2t96wvwHCgRwjvLo3D69mvKpmQx6XE9qL2vR7c92LNE8xL58BAbGLWJqVb9WJBW
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain1.com%23c9lXUEfQVRceUV6FdwzzR19EMv4nZE-eopNnSmxum14%22%7D%5D%2C%22type%22
↪%3A%5B%22VerifiableCredential%22%2C%22AnchorCredential%22%5D%7D",
          "type": "application/ld+json"
        }
      ]
    }
  ]
}
```

3.6.6 Witness Policy Endpoint

Endpoint: /policy

POST

Configures the witness policy as per Witness Policy.

Example

Request:

```
POST /policy HTTP/1.1
Host: orb.domain1.com
Content-Type: text/plain

MinPercent(100,batch) AND MinPercent(50,system)
```

GET

Retrieves the current witness policy as per Witness Policy.

Example

Request:

```
GET /policy HTTP/1.1
Host: orb.domain1.com
Accept: text/plain
```

Response:

```
MinPercent(100,batch) AND MinPercent(50,system)
```

3.6.7 VC Endpoints

Endpoint: /vc/[id]

GET

Returns the verifiable credential for the given ID.

Example

```
GET /vc/57cf856f-e489-47dd-8f4e-65ea9524d6ea HTTP/1.1
Host: orb.domain2.com
Accept: application/ld+json
Accept-Encoding: gzip, deflate
```

Response contains the verifiable credential:

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/activityanchors/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "credentialSubject": {
    "anchor": "hl:uEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ",
    "href": "hl:uEiAIMnHwbdHbpWbL3OlruU1xtqW-Potpi0bW0ioXFCZ94w",
    "profile": "https://w3id.org/orb#v0",
    "rel": "linkset",
    "type": [
      "AnchorLink"
    ]
  },
  "id": "https://orb.domain5.com/vc/2a649ce3-3412-4ca7-953c-bf89e333c3db",
  "issuanceDate": "2022-09-20T20:57:11.776289881Z",
  "issuer": "did:web:orb.domain5.com:services:anchor",
  "proof": [
    {
      "created": "2022-09-20T20:57:11.782036216Z",
      "domain": "https://orb.domain5.com",
      "proofPurpose": "assertionMethod",
      "proofValue":
↪"z3VUDMo8tksYYD14crKHQx7HZLtJPwi7V4g2az1puHDhRYxTXSFQ3a2Qch7Az8niyQ1TXdKKaWzjvXoYNFUbJJmr9
↪",
      "type": "Ed25519Signature2020",
      "verificationMethod": "did:web:orb.domain5.com#H72rWxdnDTaf69z2OkDLUG0z7XtFkBY_
↪WmG9__U060Y"
    },
    {
      "created": "2022-09-20T20:57:11.966Z",
      "domain": "http://orb.vct:8077/maple2020",
      "proofPurpose": "assertionMethod",
      "proofValue":
↪"z54WJu6r64W6Fq4LfiVzojHYHkwo4aVEMQ1KA15XYUhqwZPFwByqeV9Dwi3UPFPfKUsUsVUh92edAD1nGsd6n2nf6
↪",
      "type": "Ed25519Signature2020",
      "verificationMethod": "did:web:orb.domain1.com#K3CezR1_
↪tXyZSNbgdFbikNxkPCypfZ1bcJFh9NcsJlk"
    }
  ],
  "type": [
    "VerifiableCredential",
    "AnchorCredential"
  ]
}
```

3.6.8 Log Endpoint

Endpoint: /log

GET

Returns the URL of the current VCT log.

Example

```
GET /log HTTP/1.1
Host: orb.domain1.com
```

Response contains the log URL:

```
https://orb.vct:8077/maple2020
```

POST

Sets the URL of the current VCT log.

Example

```
POST /log HTTP/1.1
Host: orb.domain1.com
Content-Type: text/plain

https://orb.vct:8077/maple2020
```

3.6.9 System Endpoints

Various endpoints are defined to configure Orb and to retrieve system information, such as Orb version and metrics.

Accept List

Endpoint: /services/orb/acceptlist

An accept list is a database of server URLs that are authorized for a particular type of operation.

GET

The accept-list is retrieved using a GET request to this endpoint.

Example

Request:

```
GET /services/orb/acceptlist HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json
Accept-Encoding: gzip, deflate
```

Response contains the accept-list for both *follow* and *invite-witness*:

```
[
  {
    "type": "follow",
    "url": [
      "https://orb.domain2.com/services/orb",
      "https://orb.domain3.com/services/orb"
    ]
  },
  {
    "type": "invite-witness",
    "url": [
      "https://orb.domain2.com/services/orb",
      "https://orb.domain3.com/services/orb"
    ]
  }
]
```

POST

The accept-list is updated using a POST request to this endpoint. Services may be added and removed from the accept-list for *Follow* and *Invite* witness activities.

Example

Request to add domain2 and domain3 to the *follow* and *invite-witness* accept-list as well as remove domain4 from the *follow* accept-list:

```
POST /services/orb/acceptlist HTTP/1.1
Host: orb.domain1.com
Content-Type: application/ld+json"
Accept-Encoding: gzip, deflate

[
  {
    "add": [
      "https://orb.domain2.com/services/orb",
      "https://orb.domain3.com/services/orb"
    ],
    "remove": [
      "https://orb.domain4.com/services/orb",
    ],
    "type": "follow"
  },
  {
    "add": [
      "https://orb.domain2.com/services/orb",
      "https://orb.domain3.com/services/orb"
    ],
    "type": "invite-witness"
  }
]
```

Allowed Anchor Origins

Endpoint: /allowedorigins

An allowed anchor origins is a database of server URIs that are authorized for DID create operations.

GET

The allowed anchor origins are retrieved using a GET request to this endpoint.

Example

Request:

```
GET /allowedorigins HTTP/1.1
Host: orb.domain1.com
Accept: application/ld+json
```

Response contains the anchor origins:

```
[
  "https://orb.domain1.com",
  "https://orb.domain2.com"
]
```

POST

The allowed anchor origins are updated using a POST request to this endpoint. URIs may be added and removed.

Example

Request to add domain3 and domain4, and remove domain2 from the allowed anchor origins list:

```
POST /allowedorigins HTTP/1.1
Host: orb.domain1.com
Content-Type: application/json"

{
  "add": [
    "https://orb.domain3.com",
    "https://orb.domain4.com"
  ],
  "remove": [
    "https://orb.domain2.com"
  ]
}
```

Node Info Endpoints

The NodeInfo endpoints provide general information about an Orb server, including the version, the number of posts (Create activities) and the number of comments (Like activities). Two versions of NodeInfo are supported: v2.0 and v2.1.

Node Info version 2.0

Endpoint: /nodeinfo/2.0

GET

Example

Request:

```
GET /nodeinfo/2.0 HTTP/1.1
Host: orb.domain1.com
Accept: application/json
Accept-Encoding: gzip, deflate
```

Response:

```
{
  "version": "2.0",
  "software": {
    "name": "Orb",
    "version": "0.1.2"
  },
  "protocols": [
    "activitypub"
  ],
  "services": {
    "inbound": [],
    "outbound": []
  },
  "openRegistrations": false,
  "usage": {
    "users": {
      "total": 1
    },
    "localPosts": 4,
    "localComments": 6
  }
}
```

Node Info version 2.1

Endpoint: /nodeinfo/2.1

GET

Example

Request:

```
GET /nodeinfo/2.1 HTTP/1.1
Host: orb.domain1.com
Accept: application/json
Accept-Encoding: gzip, deflate
```

Response:

```
{
  "version": "2.1",
  "software": {
    "name": "Orb",
    "version": "0.1.2",
    "repository": "https://github.com/trustbloc/orb"
  },
  "protocols": [
    "activitypub"
  ],
  "services": {
    "inbound": [],
    "outbound": []
  },
  "openRegistrations": false,
  "usage": {
    "users": {
      "total": 1
    },
    "localPosts": 4,
    "localComments": 6
  }
}
```

Metrics Endpoint

Endpoint: /metrics

GET

This endpoint retrieves metrics data that may be consumed by a prometheus server.

Request:

```
GET /metrics HTTP/1.1
Host: orb.domain1.com
Accept: text/plain
Accept-Encoding: gzip, deflate
```

Response:

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection␣
↪cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 6.49e-05
go_gc_duration_seconds{quantile="0.25"} 0.0001778
go_gc_duration_seconds{quantile="0.5"} 0.0002633
go_gc_duration_seconds{quantile="0.75"} 0.0004953
go_gc_duration_seconds{quantile="1"} 0.0111963
go_gc_duration_seconds_sum 0.1513972
go_gc_duration_seconds_count 248
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 104
# HELP orb_activitypub_inbox_handler_seconds The time (in seconds) that it takes to␣
↪handle an activity posted to the inbox.
# TYPE orb_activitypub_inbox_handler_seconds histogram
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="0.005"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="0.01"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="0.025"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="0.05"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="0.1"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="0.25"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="0.5"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="1"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="2.5"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="5"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="10"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Accept",le="+Inf"} 1
orb_activitypub_inbox_handler_seconds_sum{type="Accept"} 0.1140751
orb_activitypub_inbox_handler_seconds_count{type="Accept"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="0.005"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="0.01"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="0.025"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="0.05"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="0.1"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="0.25"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="0.5"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="1"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="2.5"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="5"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="10"} 0
```

(continues on next page)

(continued from previous page)

```
orb_activitypub_inbox_handler_seconds_bucket{type="Announce",le="+Inf"} 0
orb_activitypub_inbox_handler_seconds_sum{type="Announce"} 0
orb_activitypub_inbox_handler_seconds_count{type="Announce"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="0.005"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="0.01"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="0.025"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="0.05"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="0.1"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="0.25"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="0.5"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="1"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="2.5"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="5"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="10"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Create",le="+Inf"} 1
orb_activitypub_inbox_handler_seconds_sum{type="Create"} 0.2653678
orb_activitypub_inbox_handler_seconds_count{type="Create"} 1
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="0.005"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="0.01"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="0.025"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="0.05"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="0.1"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="0.25"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="0.5"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="1"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="2.5"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="5"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="10"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="Follow",le="+Inf"} 0
orb_activitypub_inbox_handler_seconds_sum{type="Follow"} 0
orb_activitypub_inbox_handler_seconds_count{type="Follow"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="InviteWitness",le="0.005"} 0
orb_activitypub_inbox_handler_seconds_bucket{type="InviteWitness",le="0.01"} 0
. . .
```

Health Check Endpoint

Endpoint: /healthcheck

The health check endpoint performs a check on various subsystems. If the health check fails then an HTTP status, *503: Service Unavailable*, is returned along with details of which component(s) failed. An HTTP status, *200: OK*, is returned when Orb is ready to receive requests. Following are the returned statuses:

- **mqStatus** - Message Broker
  - *success* - The message broker is connected.
  - *not connected* = The message broker is not connected.
- **dbStatus** - Database - Contains "success" or an error message.
  - *success* - a Database 'ping' has succeeded.
  - *error message* - The error message received from the database 'ping'.

- **kmsStatus** - Key Management Service - Contains "success" or an error message.
  - *success* - The KMS health check succeeded.
  - *error message* - The error message received from the KMS health check.
- **vctStatus** - Verifiable Credential Transparency
  - *success* - VCT health check succeeded.
  - *disabled* - VCT is disabled for the Orb domain. (This status is not considered to be a failed status.)
  - *log endpoint not configured* - VCT is enabled but the log endpoint has not yet been configured. (This status is not considered to be a failed status.)
  - *error message* - The error message returned from the VCT health check.

GET

Example

Request:

```
GET /healthcheck HTTP/1.1
Host: orb.domain1.com
Accept: application/json
```

Response:

```
{
  "mqStatus": "success",
  "vctStatus": "disabled",
  "dbStatus": "success",
  "kmsStatus": "success",
  "currentTime": "2022-06-24T20:10:43.0115373Z"
}
```

Log Levels Endpoint

Endpoint: /loglevels

The log levels endpoint allows you to dynamically update the log levels of individual modules and/or to change the default log level. The format of the spec is as follows:

```
module1=level1:module2=level2:module3=level3:defaultLevel
```

Valid log levels are: error, warning, info, debug.

GET

Example

Request:

```
GET /loglevels HTTP/1.1
Host: orb.domain1.com
Accept: text/plain
```

Response:

```
activitypub_store=INFO:expiry-service=INFO:task-manager=INFO:watermill=INFO:DEBUG
```

POST

Example

```
POST /loglevels HTTP/1.1
Host: orb.domain1.com
Content-Type: text/plain

activitypub_store=INFO:expiry-service=INFO:task-manager=INFO:watermill=INFO:DEBUG
```

3.7 Startup Parameters

This section enumerates the startup parameters for an Orb server. Parameters in the *Required Parameters* section are required, otherwise the server will not start. Parameters in the *Optional Parameters* section are optional and will use a default value if not specified.

3.7.1 Required Parameters

Following are the required parameters for an Orb server.

host-url

Arg	Env
–host-url	ORB_HOST_URL

URL to run the orb-server instance on. Format: HostName:Port.

external-endpoint

Arg	Env
–external-endpoint	ORB_EXTERNAL_ENDPOINT

External endpoint that clients use to invoke services. This endpoint is used to generate IDs of anchor credentials and ActivityPub objects and should be resolvable by external clients. Format: HostName[:Port].

service-id

Arg	Env
–service-id	ORB_SERVICE_ID

The ID of the ActivityPub service. By default, the ID is composed of the external endpoint appended with /services/orb. For example, if external-endpoint is set to https://alice.example.com then the service ID will be https://alice.example.com/services/orb. The value may be set to a different path, e.g. https://alice.example.com/services/anchor, or it can be set to a DID, e.g. did:web:alice.example.com:services:anchor. NOTE: The host of the ID must be the same as the host specified by external-endpoint.

database-type

Arg	Env
–database-type	DATABASE_TYPE

The type of database to use for everything except key storage. Supported options: mem, couchdb, mongodb.

vc-sign-private-keys

Arg	Env
–vc-sign-private-keys	ORB_VC_SIGN_PRIVATE_KEYS

VC Sign Private Keys base64 (ED25519Type). For example, key1=privatekeyBase64Value,key2=privatekeyBase64Value

vc-sign-active-key-id

Arg	Env -
–vc-sign-active-key-id	ORB_VC_SIGN_ACTIVE_KEY_ID

VC Sign Active Key ID (ED25519Type).

vc-sign-keys-id

Arg	Env -
–vc-sign-keys-id	ORB_VC_SIGN_KEYS_ID

VC Sign Keys id in kms (ED25519Type).

http-sign-private-key

Arg	Env
–http-sign-private-key	ORB_HTTP_SIGN_PRIVATE_KEY

HTTP Sign Private Key base64 (ED25519Type). For example, key1=privatekeyBase64Value

http-sign-active-key-id

Arg	Env -
–http-sign-active-key-id	ORB_HTTP_SIGN_ACTIVE_KEY_ID

HTTP Sign Active Key ID (ED25519Type).

did-namespace

Arg	Env
–did-namespace	DID_NAMESPACE

DID Namespace.

cas-type

Arg	Env
–cas-type	CAS_TYPE

The type of the Content Addressable Storage (CAS). Supported options: local, ipfs.

anchor-credential-signature-suite

Arg	Env
–anchor-credential-signature-suite	ANCHOR_CREDENTIAL_SIGNATURE_SUITE

Anchor credential signature suite (required).

3.7.2 Optional Parameters

Below are the optional parameters for an Orb server. If not specified then the default value is used.

metrics-provider-name

Arg	Env	Default
–metrics-provider-name	ORB_METRICS_PROVIDER_NAME	

The name of the metrics provider. If not set then metrics are not gathered.

Valid values:

- prometheus - Prometheus metrics provider. Parameter *prom-http-url* must be set.

prom-http-url

Arg	Env	Default
–prom-http-url	ORB_PROM_HTTP_URL	

URL that exposes the Prometheus metrics endpoint. Format: host:port. A Prometheus server may be used to periodically read the metrics at this URL. (Note that the metrics endpoint would be at http://host:port/metrics.)

sync-timeout

Arg	Env	Default
–sync-timeout	ORB_SYNC_TIMEOUT	1

Total time in seconds to resolve config values.

vct-enabled

Arg	Env	Default
–vct-enabled	ORB_VCT_ENABLED	false

Enables setting VCT log. If enabled VCT URL has to be configured via cli log command or REST /log endpoint.

vct-proof-monitoring-interval

Arg	Env	Default
–vct-proof-monitoring-interval	VCT_PROOF_MONITORING_INTERVAL	10s

The interval in which VCTs are monitored to ensure that proofs are anchored.

vct-log-monitoring-interval

Arg	Env	Default
–vct-log-monitoring-interval	VCT_LOG_MONITORING_INTERVAL	10s

The interval in which VCT logs are monitored to ensure that they are consistent.

vct-log-monitoring-max-tree-size

Arg	Env	Default
–vct-log-monitoring-max-tree-size	VCT_LOG_MONITORING_MAX_TREE_SIZE	50000

The maximum tree size for which new VCT logs will be re-constructed in order to verify signed tree head.

vct-log-monitoring-get-entries-range

Arg	Env	Default
–vct-log-monitoring-get-entries-range	VCT_LOG_MONITORING_GET_ENTRIES_RANGE	1000

The maximum number of entries to be retrieved from VCT log in one attempt. Has to be less or equal than 1000 due to VCT limitation.

vct-log-entries-store-enabled

Arg	Env	Default
–vct-log-entries-store-enabled	VCT_LOG_ENTRIES_STORE_ENABLED	false

Enables storing of log entries during log monitoring. Defaults to false if not set.

anchor-status-monitoring-interval

Arg	Env	Default
–anchor-status-monitoring-interval	ANCHOR_STATUS_MONITORING_INTERVAL	5s

The interval in which 'in-process' anchors are monitored to ensure that they will be witnessed(completed) as per policy.

anchor-status-in-process-grace-period

Arg	Env	Default
–anchor-status-in-process-grace-period	ANCHOR_STATUS_IN_PROCESS_GRACE_PERIOD	30s

The period in which witnesses will not be re-selected for 'in-process' anchors.

kms-store-endpoint

Arg	Env	Default
–kms-store-endpoint	ORB_KMS_STORE_ENDPOINT	

KMS storage URL. If this parameter is not set then ORB_KMS_ENDPOINT needs to be set.

kms-endpoint

Arg	Env	Default
–kms-endpoint	ORB_KMS_ENDPOINT	

Remote KMS URL. If this parameter is not set then ORB_KMS_STORE_ENDPOINT needs to be set.

secret-lock-key-path

Arg	Env	Default
–secret-lock-key-path	ORB_SECRET_LOCK_KEY_PATH	

The path to the file with key to be used by local secret lock. If missing noop service lock is used.

discovery-domain

Arg	Env	Default
–discovery-domain	ORB_DISCOVERY_DOMAIN	

Discovery domain for this domain. Format: HostName

tls-systemcertpool

Arg	Env	Default
–tls-systemcertpool	ORB_TLS_SYSTEMCERTPOOL	false

Use system certificate pool. Possible values true and false. Defaults to false if not set.

tls-cacerts

Arg	Env	Default
–tls-cacerts	ORB_TLS_CACERTS	

Comma-Separated list of ca certs path.

tls-certificate

Arg	Env	Default
–tls-certificate	ORB_TLS_CERTIFICATE	

TLS certificate for ORB server.

tls-key

Arg	Env	Default
–tls-key	ORB_TLS_KEY	

TLS key for ORB server.

did-aliases

Arg	Env	Default
–did-aliases	DID_ALIASES	

Aliases for this did method.

ipfs-url

Arg	Env	Default
–ipfs-url	IPFS_URL	

Enables IPFS support. If set, this Orb server will use the node at the given URL. To use the public ipfs.io node, set this to https://ipfs.io (or http://ipfs.io). If using ipfs.io, then the CAS type flag must be set to local since the ipfs.io node is read-only. If the URL doesn't include a scheme, then HTTP will be used by default.

replicate-local-cas-writes-in-ipfs

Arg	Env	Default
–replicate-local-cas-writes-in-ipfs	REPLICATE_LOCAL_CAS_WRITES_IN_IPFS	false

If enabled, writes to the local CAS will also be replicated in IPFS. This setting only takes effect if this server has both a local CAS and IPFS enabled. If the IPFS node is set to ipfs.io, then this setting will be disabled since ipfs.io does not support writes. Supported options: false, true. Defaults to false if not set.

mq-url

Arg	Env	Default
–mq-url	MQ_URL	

The URL of the message broker. If not specified then an in-memory message queue is used.

mq-connect-max-retries

Arg	Env	Default
–mq-connect-max-retries	MQ_CONNECT_MAX_RETRIES	25

The maximum number of retries to connect to an AMQP service, after which the server will panic.

mq-observer-pool

Arg	Env	Default
–mq-observer-pool	MQ_OBSERVER_POOL	5

The size of the Observer queue subscriber pool. When a message is posted to the Observer queue, it is handled by a pool of subscribers. If not specified then the default size will be used.

mq-outbox-pool

Arg	Env	Default
–mq-outbox-pool	MQ_OUTBOX_POOL	5

The size of the Outbox queue subscriber pool. When a message is posted to the Outbox queue, it is handled by a pool of subscribers. If not specified then the default size will be used.

mq-inbox-pool

Arg	Env	Default
–mq-inbox-pool	MQ_INBOX_POOL	5

The size of the Inbox queue subscriber pool. When a message is posted to the Inbox queue, it is handled by a pool of subscribers. If not specified then the default size will be used.

mq-max-connection-channels

Arg	Env	Default
–mq-max-connection-channels	MQ_MAX_CONNECTION_CHANNELS	1000

The maximum number of channels per connection.

mq-publisher-channel-pool-size

Arg	Env	Default
–mq-publisher-channel-pool-size	MQ_PUBLISHER_POOL	25

The size of a channel pool for an AMQP publisher (default is 25). If set to 0 then a channel pool is not used and a new channel is opened/closed for every publish to a queue.

mq-publisher-confirm-delivery

Arg	Env	Default
–mq-publisher-confirm-delivery	MQ_PUBLISHER_CONFIRM_DELIVERY	true

Turns on delivery confirmation of published messages. If set to true then the AMQP publisher waits until a confirmation is received from the AMQP server to guarantee that the message is delivered.

mq-redelivery-max-attempts

Arg	Env	Default
–mq-redelivery-max-attempts	MQ_REDELIVERY_MAX_ATTEMPTS	30

The maximum number of redelivery attempts for a failed message.

mq-redelivery-initial-interval

Arg	Env	Default
–mq-redelivery-initial-interval	MQ_REDELIVERY_INITIAL_INTERVAL	2s

The delay for the initial redelivery attempt.

mq-redelivery-multiplier

Arg	Env	Default
–mq-redelivery-multiplier	MQ_REDELIVERY_MULTIPLIER	1.5

The multiplier for a redelivery attempt. For example, if set to 1.5 and the previous redelivery interval was 2s then the next redelivery interval is set 3s.

mq-redelivery-max-interval

Arg	Env	Default
–mq-redelivery-max-interval	MQ_REDELIVERY_MAX_INTERVAL	1m

The maximum delay for a redelivery.

op-queue-pool

Arg	Env	Default
–op-queue-pool	OP_QUEUE_POOL	5

The size of the operation queue subscriber pool. If <=1 then a pool will not be created.

op-queue-task-monitor-interval

Arg	Env	Default
–op-queue-task-monitor-interval	OP_QUEUE_TASK_MONITOR_INTERVAL	10s

The interval (period) in which operation queue tasks from other server instances are monitored.

op-queue-task-expiration

Arg	Env	Default
–op-queue-task-expiration	OP_QUEUE_TASK_EXPIRATION	30s

The maximum time that an operation queue task can exist in the database before it is considered to have expired. Once expired, any other server instance may delete the task and repost operations associated with the task to the queue, so that they will be processed by another Orb instance.

cid-version

Arg	Env	Default
–cid-version	CID_VERSION	1

The version of the CID format to use for generating CIDs. Supported options: 0, 1. If not set, defaults to 1.

batch-writer-timeout

Arg	Env	Default
–batch-writer-timeout	BATCH_WRITER_TIMEOUT	60s

Maximum time (in millisecond) in-between cutting batches.

database-url

Arg	Env	Default
–database-url	DATABASE_URL	

The URL (or connection string) of the database. Not needed if using memstore. For CouchDB, include the username:password@ text if required.

database-prefix

Arg	Env	Default
–database-prefix	DATABASE_PREFIX	

An optional prefix to be used when creating and retrieving underlying databases. This allows a database to be shared by multiple Orb domains. (Mainly used in development environments.)

kms-secrets-database-type

Arg	Env	Default
–kms-secrets-database-type	KMSSECRETS_DATABASE_TYPE	

The type of database to use for storage of KMS secrets. Supported options: mem, couchdb, mongodb.

kms-secrets-database-url

Arg	Env	Default
–kms-secrets-database-url	KMSSECRETS_DATABASE_URL	

The URL (or connection string) of the database. Not needed if using memstore. For CouchDB, include the username:password@ text if required.

kms-secrets-database-prefix

Arg	Env	Default
–kms-secrets-database-prefix	KMSSECRETS_DATABASE_PREFIX	

An optional prefix to be used when creating and retrieving the underlying KMS secrets database. This allows a database to be shared by multiple Orb domains. (Mainly used in development environments.)

database-timeout

Arg	Env	Default
–database-timeout	DATABASE_TIMEOUT	10s

The timeout for database requests. For example, '30s' for a 30 second timeout. Currently this setting only applies if you're using MongoDB.

anchor-credential-domain

Arg	Env	Default
–anchor-credential-domain	ANCHOR_CREDENTIAL_DOMAIN	ORB_EXTERNAL_ENDPOINT

Anchor credential domain.

allowed-origins

Arg	Env	Default
–allowed-origins	ALLOWED_ORIGINS	

Allowed origins are the bootstrap anchor origins for this did method. Anchor origins may be updated using the allowedorigins REST API.

allowed-origins-cache-expiration

Arg	Env	Default
–allowed-origins-cache-expiration	ALLOWED_ORIGINS_CACHE_EXPIRATION	1m

The expiration time(period) of the allowed origins cache.

max-witness-delay

Arg	Env	Default
–max-witness-delay	MAX_WITNESS_DELAY	10m

Maximum witness response time.

sign-with-local-witness

Arg	Env	Default
–sign-with-local-witness	SIGN_WITH_LOCAL_WITNESS	true

Always sign with local witness flag (default true).

discovery-domains

Arg	Env	Default
–discovery-domains	DISCOVERY_DOMAINS	

Discovery domains.

discovery-minimum-resolvers

Arg	Env	Default
–discovery-minimum-resolvers	DISCOVERY_MINIMUM_RESOLVERS	1

Discovery minimum resolvers number.

enable-http-signatures

Arg	Env	Default
–enable-http-signatures	HTTP_SIGNATURES_ENABLED	true

Set to "true" to enable HTTP signatures in ActivityPub.

enable-did-discovery

Arg	Env	Default
–enable-did-discovery	DID_DISCOVERY_ENABLED	false

Set to "true" to enable did discovery.

enable-unpublished-operation-store

Arg	Env	Default
–enable-unpublished-operation-store	UNPUBLISHED_OPERATION_STORE_ENABLED	false

Set to "true" to enable un-published operation store. Used to enable storing unpublished operations and including them when resolving documents.

unpublished-operation-store-operation-types

Arg	Env	Default
–unpublished-operation-store-operation-types	UNPUBLISHED_OPERATION_STORE_OPERATION_	create, update

Comma-separated list of operation types. Used if unpublished operation store is enabled. Default value is "create,update" which enables storing unpublished 'create' and 'update' operations into unpublished store and using those unpublished 'create' and 'update' operations for resolving document.

include-unpublished-operations-in-metadata

Arg	Env	Default
–include-unpublished-operations-in-metadata	INCLUDE_UNPUBLISHED_OPERATIONS_IN_METAD	false

Set to "true" to include unpublished operations in metadata.

include-published-operations-in-metadata

Arg	Env	Default
–include-published-operations-in-metadata	INCLUDE_PUBLISHED_OPERATIONS_IN_METADATA	false

Set to "true" to include published operations in metadata.

resolve-from-anchor-origin

Arg	Env	Default
–resolve-from-anchor-origin	RESOLVE_FROM_ANCHOR_ORIGIN	false

Set to "true" to resolve from anchor origin.

verify-latest-from-anchor-origin

Arg	Env	Default
–verify-latest-from-anchor-origin	VERIFY_LATEST_FROM_ANCHOR_ORIGIN	false

Set to "true" to verify latest operations against anchor origin.

auth-tokens-def

Arg	Env	Default
–auth-tokens-def	ORB_AUTH_TOKENS_DEF	

Contains the authorization definition for each of the REST endpoints. Format:

```
<path-expr>|<read-token1>&<read-token2>&...>|<write-token1>&<write-token2>&...>,<path-
↪expr>
```

Where:

- path-expr contains a regular expression for a path. Path expressions are processed in the order they are specified.
- read-token defines a token for a read (GET) operation. If not specified then authorization is not performed.
- write-token defines a token for a write (POST) operation. If not specified then authorization is not performed.

If no definition is included for an endpoint then authorization is NOT performed for that endpoint.

Example:

```
ORB_AUTH_TOKENS_DEF=/services/orb/outbox|admin&read|admin,/services/orb/.*|read&admin
```

- The client requires a 'read' or 'admin' token in order to view the outbox's contents
- The client requires an 'admin' token in order to post to the outbox
- The client requires a 'read' or 'admin' token in order to perform a GET on any endpoint starting with /services/orb/

auth-tokens

Arg	Env	Default
–auth-tokens	ORB_AUTH_TOKENS	

Specifies the actual values of the tokens defined in ORB_AUTH_TOKENS_DEF.

Example:

admin=ADMIN_PASSWORD,read=READ_PASSWORD

client-auth-tokens-def

Arg	Env	Default
–client-auth-tokens-def	ORB_CLIENT_AUTH_TOKENS_DEF	ORB_AUTH_TOKENS_DEF

Follows the same rules as ORB_AUTH_TOKENS_DEF but is used by the Orb client transport to determine whether an HTTP signature is required for an outbound HTTP request. If not specified then it is assumed to be the same as ORB_AUTH_TOKENS_DEF.

client-auth-tokens

Arg	Env	Default
–client-auth-tokens	ORB_CLIENT_AUTH_TOKENS	ORB_AUTH_TOKENS

Specifies the actual values of the tokens defined in ORB_CLIENT_AUTH_TOKENS_DEF. If not specified then it is assumed to be the same as ORB_AUTH_TOKENS.

activitypub-page-size

Arg	Env	Default
–activitypub-page-size	ACTIVITYPUB_PAGE_SIZE	50

The maximum page size for an ActivityPub collection or ordered collection.

enable-dev-mode

Arg	Env	Default
–enable-dev-mode	DEV_MODE_ENABLED	false

Set to "true" to enable dev mode. When dev mode is enabled, no TLS is used.

enable-maintenance-mode

Arg	Env	Default
–enable-maintenance-mode	MAINTENANCE_MODE_ENABLED	false

Set to "true" to enable maintenance mode.

When maintenance mode is enabled:

1) Health check returns status OK (200) even if errors are detected
2) Sidetree operations and resolution endpoints are not available (503)
3) Activity pub inbox is not available (503)

nodeinfo-refresh-interval

Arg	Env	Default
–nodeinfo-refresh-interval	NODEINFO_REFRESH_INTERVAL	15s

The interval for refreshing NodeInfo data. For example, '30s' for a 30 second interval.

ipfs-timeout

Arg	Env	Default
–ipfs-timeout	IPFS_TIMEOUT	20s

The timeout for IPFS requests. For example, '30s' for a 30 second timeout.

context-provider-url

Arg	Env	Default
–context-provider-url	ORB_CONTEXT_PROVIDER_URL	

Comma-separated list of remote context provider URLs to get JSON-LD contexts from."

unpublished-operation-lifetime

Arg	Env	Default
–unpublished-operation-lifetime	UNPUBLISHED_OPERATION_LIFETIME	5m

How long unpublished operations remain stored before expiring (and thus, being deleted some time later). For example, '1m' for a 1 minute lifespan. Defaults to 1 minute if not set.

task-manager-check-interval

Arg	Env	Default
–task-manager-check-interval	TASK_MANAGER_CHECK_INTERVAL	10s

How frequently to check for scheduled tasks. For example, a setting of '10s' will cause the task manager to check for outstanding tasks every 10s. Defaults to 10 seconds if not set.

data-expiry-check-interval

Arg	Env	Default
–data-expiry-check-interval	DATA_EXPIRY_CHECK_INTERVAL	1m

How frequently to check for (and delete) any expired data. For example, a setting of '1m' will cause the expiry service to run a check every 1 minute. Defaults to 1 minute if not set.

follow-auth-policy

Arg	Env	Default
–follow-auth-policy	FOLLOW_AUTH_POLICY	accept-all

The type of authorization to use when a 'Follow' ActivityPub request is received. Possible values are: 'accept-all' and 'accept-list'. The value, 'accept-all', indicates that this server will accept any 'Follow' request. The value, 'accept-list', indicates that the service sending the 'Follow' request must be included in an 'accept list'. Defaults to 'accept-all' if not set.

invite-witness-auth-policy

Arg	Env	Default
–invite-witness-auth-policy	INVITE_WITNESS_AUTH_POLICY	accept-all

The type of authorization to use when a 'Invite' witness ActivityPub request is received. Possible values are: 'accept-all' and 'accept-list'. The value, 'accept-all', indicates that this server will accept any 'Invite' request for a witness. The value, 'accept-list', indicates that the service sending the 'Invite' witness request must be included in an 'accept list'. Defaults to 'accept-all' if not set.

http-timeout

Arg	Env	Default
–http-timeout	HTTP_TIMEOUT	20s

The timeout for http requests. For example, '30s' for a 30 second timeout.

http-dial-timeout

Arg	Env	Default
–http-dial-timeout	HTTP_DIAL_TIMEOUT	2s

The timeout for HTTP dial. For example, '30s' for a 30 second timeout.

sync-interval

Arg	Env	Default
–sync-interval	ANCHOR_EVENT_SYNC_INTERVAL	1m

The interval in which anchor events are synchronized with other services that this service is following. Defaults to 1m if not set.

sync-min-activity-age

Arg	Env	Default
–sync-min-activity-age	ANCHOR_EVENT_SYNC_MIN_ACTIVITY_AGE	1m

The minimum age of an anchor activity to be synchronized. The activity will be processed only if its age is greater than this value. Defaults to 1m if not set.

apclient-cache-size

Arg	Env	Default
–apclient-cache-size	ACTIVITYPUB_CLIENT_CACHE_SIZE	100

The maximum size of an ActivityPub service and public key cache.

apclient-cache-Expiration

Arg	Env	Default
–apclient-cache-Expiration	ACTIVITYPUB_CLIENT_CACHE_EXPIRATION	10m

The expiration time of an ActivityPub service and public key cache.

apiri-cache-size

Arg	Env	Default
–apiri-cache-size	ACTIVITYPUB_IRI_CACHE_SIZE	100

The maximum size of an ActivityPub actor IRI cache.

apiri-cache-Expiration

Arg	Env	Default
–apiri-cache-Expiration	ACTIVITYPUB_IRI_CACHE_EXPIRATION	1m

The expiration time of an ActivityPub actor IRI cache.

server-idle-timeout

Arg	Env	Default
–server-idle-timeout	SERVER_IDLE_TIMEOUT	20s

The idle timeout for the HTTP server. For example, ’30s’ for a 30 second timeout.

witness-policy-cache-expiration

Arg	Env	Default
–witness-policy-cache-expiration	WITNESS_POLICY_CACHE_EXPIRATION	30s

The expiration time(period) of witness policy cache. Default value is 30s.

anchor-data-uri-media-type

Arg	Env	Default
–anchor-data-uri-media-type	ANCHOR_DATA_URI_MEDIA_TYPE	application/json

The media type for data URIs in an anchor Linkset. Possible values are 'application/json' and 'application/gzip;base64'. If 'application/json' is specified then the content of the data URIs in the anchor Linkset are encoded as an escaped JSON string. If 'application/gzip;base64' is specified then the content is compressed with gzip and base64 encoded.

sidetree-protocol-versions

Arg	Env	Default
–sidetree-protocol-versions	SIDETREE_PROTOCOL_VERSIONS	1.0

Comma-separated list of Sidetree protocol versions.

current-sidetree-protocol-version

Arg	Env	Default
–current-sidetree-protocol-version	CURRENT_SIDETREE_PROTOCOL_VERSION	

One of available Sidetree protocol versions. Defaults to latest Sidetree protocol version.

log-level

Arg	Env	Default
–log-level	LOG_LEVEL	INFO

Sets the log levels for individual modules as well as the default log level. The format of the spec is as follows:

```
module1=level1:module2=level2:module3=level3:defaultLevel
```

Valid log levels are: error, warning, info, debug.

Example:

```
activitypub_store=INFO:expiry-service=INFO:task-manager=INFO:watermill=INFO:DEBUG
```

tracing-provider

Arg	Env	Default
–tracing-provider	ORB_TRACING_PROVIDER	

The name of the tracing provider. If not set then tracing is disabled.

Valid values:

- JAEGER - Jaeger tracing provider. Parameter *tracing-collector-url* must be set.

tracing-collector-url

Arg	Env	Default
–tracing-collector-url	ORB_TRACING_COLLECTOR_URL	

URL to which tracing data is sent.

tracing-service-name

Arg	Env	Default
–tracing-service-name	ORB_TRACING_SERVICE_NAME	orb

The name of the service as will be displayed in the tracing console.

3.8 Data Model

3.8.1 Anchor Linkset

The Anchor linkset is a JSON document that contains a link to a batch of document operations along with a link to a verifiable credential containing proofs from one or more witnesses. This document is stored in Content Addressible Storage using the multihash of the document as the key. The multihash is used in the first segment of the DID (after did:orb) of a canonical Orb DID. This multihash specifies the latest anchor object that contains a *create* or *recover* operation for the Sidetree DID suffix. (See did:orb.)

Document Operation Batch

The batch of document operations is represented as a linkset. This document contains the DIDs that were anchored, as well as the hashlink of the previous anchor of the DID (if any).


```
{
  "linkset": [
    {
      "anchor": "hl:uEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ",
      "author": [
        {
          "href": "did:web:orb.domain5.com:services:anchor"
        }
      ],
      "item": [
        {
          "href": "did:orb:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-
↪LJPRp3Q:EiAISUte9hAoUQFyxGyzZBOLJiVOSo_6NvQI1_KTSrZEuw",
          "previous": [
            "hl:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q"
          ]
        },
        {
          "href": "did:orb:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-
↪LJPRp3Q:EiCA9cT2WkrRbpVqvK9NYnfYL8oAXIGHozjzN_w9cetyJA",
          "previous": [
            "hl:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q"
          ]
        },
        {
          "href": "did:orb:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q:EiALeU9pNA-
↪LPMW5HsmMNhJ_-JgTVP_fYnkA2InI8RtkzA",
          "previous": [
            "hl:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q"
          ]
        },
        {
          "href": "did:orb:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-
↪LJPRp3Q:EiApG3pQlSpclipU8ImLKpllbWlPCWHjobK_ASmGAjTeYA",
          "previous": [
            "hl:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q"
          ]
        },
        {
          "href": "did:orb:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-
↪LJPRp3Q:EiAtOCeMIculDkB6mm8aSj0m928dwKfM8wqLk-iSo81V7w",
          "previous": [
            "hl:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q"
          ]
        }
      ],
      "profile": [
        {
```


(continues on next page)

(continued from previous page)

```
          "href": "https://w3id.org/orb#v0"
        }
      ]
    }
  ]
}
```

The fields of the Document Operation Batch linkset are described below.

Link Context (anchor)

The linkset contains one link context object where the context is specified by the "anchor" member. The "anchor" member contains the multihash of the Sidetree core index file. A number of link relations are defined for the anchor, which are described below.

Author Relation

The *author* relation specifies the author (creator) of the linkset.

Profile Relation

The *profile* relation specifies the version of the application that was used to generate this linkset. For example, the profile, "https://w3id.org/orb#v0", indicates that the linkset was generated by Orb version 0.

Item Relation

The *item* relation contains an array of the DIDs that were created or updated. The "href" member contains the DID and the "previous" member contains the hashlink of the previous anchor (in the case where the DID was updated).

NOTE: If the DID was created, then the "href" member contains a non-canonical DID and the "previous" member is not present. In the case of a DID update, the DID is in the canonical form and the "previous" attribute is also present.

Related Links Document

The Related Links document is represented as a linkset. This document contains full hashlinks of the objects referenced in the *Document Operation Batch*. (The Document Operation Batch file only contains the multihashes of the core index file and previous anchors, whereas this document contains full hashlinks, including resolvable links to the files.) For example:

```
{
  "linkset": [
    {
      "anchor": "hl:uEiAIMnHwbdHbpWbL3OlruU1xtqW-Potpi0bW0ioXFCZ94w",
      "profile": [
        {
          "href": "https://w3id.org/orb#v0"
        }
      ],
```

(continues on next page)

(continued from previous page)


```
      "up": [
        {
          "href": "hl:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpREk0Ty01ck9QbUtZbm1uaHFnaHpLZmNkbHhjeHhWS2phaWp1LUxKUFJw
↪"
        }
      ],
      "via": [
        {
          "href": "hl:uEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW41LmNvbS9jYXMvdUVpQ0pZUzVKaW4tM1pTd0JUX1JUMGMwelowWjZCM19vMGRkRERDTVpYbFV4
↪"
        }
      ]
    }
  ]
}
```


The fields of the Related Links document are described below.

Link Context (anchor)

This linkset contains one link context object where the context is specified by the "anchor" member. The "anchor" member contains the multihash of the contents of the canonicalized *Document Operation Batch*.

A number of link relations are defined for the anchor, which are described below.

Profile Relation

The *profile* relation specifies the version of the application that was used to generate this linkset. For example, the profile, "https://w3id.org/orb#v0", indicates that the linkset was generated by Orb version 0.

Up Relation

The *up* relation contains hashlinks of the previous anchors that were referenced in the *Document Operation Batch*. Each hashlink contains the multihash of the anchor as well as one or more links to the data. For example:


```
hl:uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q:uoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpREk0Ty01ck9QbUtZbm1uaHFnaHpLZmNkbHhjeHhWS2phaWp1LUxKUFJw
```


The above hashlink contains the resource hash, `uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q`, and the following links to the anchor file:


```
https://orb.domain1.com/cas/uEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
ipfs://bafkreigi4dx3tlhd4yuytzu6dkqiomu7ohmxc4y4kuvdniuo56fsj5dj3u
```

Via Relation

The *via* relation contains the hashlink of the Sidetree core index file. For example:

```
hl:uEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW41LmNvbS9jYXMvdUVpQ0pZUzVKaW4tM1pTd0JUX1JUMGMwelowWjZCM19vMGRkRERDTVpYbFV4
```

The hashlink above contains the hash of the core index file, `uEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ`, and links to the core index file:

```
https://orb.domain5.com/cas/uEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ
```

Anchor Credential Document

The Anchor Credential document is a Verifiable Credential containing witness proofs. For example:


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/activityanchors/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "credentialSubject": {
    "anchor": "hl:uEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ",
    "href": "hl:uEiAIMnHwbdHbpWbL3OlruU1xtqW-Potpi0bW0ioXFCZ94w",
    "profile": "https://w3id.org/orb#v0",
    "rel": "linkset",
    "type": [
      "AnchorLink"
    ]
  },
  "id": "https://orb.domain5.com/vc/2a649ce3-3412-4ca7-953c-bf89e333c3db",
  "issuanceDate": "2022-09-20T20:57:11.776289881Z",
  "issuer": "did:web:orb.domain5.com:services:anchor",
  "proof": [
    {
      "created": "2022-09-20T20:57:11.782036216Z",
      "domain": "https://orb.domain5.com",
      "proofPurpose": "assertionMethod",
      "proofValue":
↪"z3VUDMo8tksYYD14crKHQx7HZLtJPwi7V4g2az1puHDhRYxTXSFQ3a2Qch7Az8niyQ1TXdKKaWzjvXoYNFUbJJmr9
↪",
      "type": "Ed25519Signature2020",
      "verificationMethod": "did:web:orb.domain5.com#H72rWxdnDTaf69z2OkDLUG0z7XtFkBY_
↪WmG9__U060Y"
    },
    {
      "created": "2022-09-20T20:57:11.966Z",
      "domain": "http://orb.vct:8077/maple2020",
      "proofPurpose": "assertionMethod",
      "proofValue":
↪"z54WJu6r64W6Fq4LfiVzojHYHkwo4aVEMQ1KA15XYUhqwZPFwByqeV9Dwi3UPFPfKUsUsVUh92edAD1nGsd6n2nf6
```


(continues on next page)

(continued from previous page)

```
↪",
      "type": "Ed25519Signature2020",
      "verificationMethod": "did:web:orb.domain1.com#K3CezR1_
↪tXyZSNbgdFbikNxkPCypfZ1bcJFh9NcsJlk"
    }
  ],
  "type": [
    "VerifiableCredential",
    "AnchorCredential"
  ]
}
```

The "credentialSubject" object is a link object containing the following members:

type

The JSON-LD type, "AnchorLink", which indicates that this object is a link.

Link Context (anchor)

The "anchor" member specifies the context of the link, which is the hash of the Sidetree core index file. Several link relations are defined for the anchor, which are described below.

profile

The *profile* relation specifies the version of the application that was used to generate the credential subject. For example, the profile, "https://w3id.org/orb#v0", indicates that the credential subject was generated by Orb version 0.

href

The *href* member contains the target IRI, which is the hash of the canonicalized *Document Operation Batch*.

rel

The *rel* relation describes the type of the target object. In this case we use "linkset" since the target (href) is the *Document Operation Batch* linkset.

Anchor Linkset Document

The Anchor Linkset is a document that links all of the above documents: *Document Operation Batch*, *Related Links*, and *Anchor Credential*. For example:

```
{
  "linkset": [
    {
      "anchor": "hl:uEiAIMnHwbdHbpWbL3OlruU1xtqW-Potpi0bW0ioXFCZ94w",
```

(continues on next page)

(continued from previous page)


```
      "author": [
        {
          "href": "did:web:orb.domain5.com:services:anchor"
        }
      ],
      "original": [
        {
          "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A%22hl
↪%3AuEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ%22%2C%22author%22%3A%5B%7B%22href%22
↪%3A%22did%3Aweb%3Aorb.domain5.com%3Aservices%3Aanchor%22%7D%5D%2C%22item%22%3A%5B%7B
↪%22href%22%3A%22did%3Aorb%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
↪%3AEiAISUte9hAoUQFyxGyzZBOLJiVOSo_6NvQI1_KTSrZEuw%22%2C%22previous%22%3A%5B%22hl
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%2C%7B%22href%22%3A%22did
↪%3Aorb%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
↪%3AEiCA9cT2WkrRbpVqvK9NYnfYL8oAXIGHozjzN_w9cetyJA%22%2C%22previous%22%3A%5B%22hl
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%2C%7B%22href%22%3A%22did
↪%3Aorb%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%3AEiALeU9pNA-LPMW5HsmMNhJ_-
↪JgTVP_fYnkA2InI8RtkzA%22%2C%22previous%22%3A%5B%22hl%3AuEiDI4O-
↪5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%2C%7B%22href%22%3A%22did%3Aorb
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
↪%3AEiApG3pQlSpclipU8ImLKpllbWlPCWHjobK_ASmGAjTeYA%22%2C%22previous%22%3A%5B%22hl
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%2C%7B%22href%22%3A%22did
↪%3Aorb%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
↪%3AEiAtOCeMIculDkB6mm8aSj0m928dwKfM8wqLk-iSo81V7w%22%2C%22previous%22%3A%5B%22hl
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%5D%2C%22profile%22%3A%5B%7B
↪%22href%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%7D%5D%7D",
          "type": "application/linkset+json"
        }
      ],
      "profile": [
        {
          "href": "https://w3id.org/orb#v0"
        }
      ],
      "related": [
        {
          "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A%22hl
↪%3AuEiAIMnHwbdHbpWbL3OlruU1xtqW-Potpi0bW0ioXFCZ94w%22%2C%22profile%22%3A%5B%7B%22href
↪%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%2C%22up%22%3A%5B%7B%22href%22%3A
↪%22hl%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%3AuoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpREk0Ty01ck9QbUtZbm1uaHFnaHpLZmNkbHhjeHhWS2phaWp1LUxKUFJwI
↪%22%7D%5D%2C%22via%22%3A%5B%7B%22href%22%3A%22hl%3AuEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_
↪o0ddDDCMZXlUxAQ%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW41LmNvbS9jYXMvdUVpQ0pZUzVKaW4tM1pTd0JUX1JUMGMwelowWjZCM19vMGRkRERDTVpYbFV4
↪%22%7D%5D%7D%5D%7D",
          "type": "application/linkset+json"
        }
      ],
      "replies": [
        {
          "href": "data:application/json,%7B%22%40context%22%3A%5B%22https%3A%2F%2Fwww.
↪w3.org%2F2018%2Fcredentials%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Factivityanchors%2Fv1
```


(continues on next page)

(continued from previous page)

```
↪%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fed25519-2020%2Fv1%22%5D%2C
↪%22credentialSubject%22%3A%7B%22anchor%22%3A%22hl%3AuEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_
↪o0ddDDCMZXlUxAQ%22%2C%22href%22%3A%22hl%3AuEiAIMnHwbdHbpWbL3OlruU1xtqW-
↪Potpi0bW0ioXFCZ94w%22%2C%22profile%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%2C
↪%22rel%22%3A%22linkset%22%2C%22type%22%3A%5B%22AnchorLink%22%5D%7D%2C%22id%22%3A
↪%22https%3A%2F%2Forb.domain5.com%2Fvc%2F2a649ce3-3412-4ca7-953c-bf89e333c3db%22%2C
↪%22issuanceDate%22%3A%222022-09-20T20%3A57%3A11.776289881Z%22%2C%22issuer%22%3A%22did
↪%3Aweb%3Aorb.domain5.com%3Aservices%3Aanchor%22%2C%22proof%22%3A%5B%7B%22created%22%3A
↪%222022-09-20T20%3A57%3A11.782036216Z%22%2C%22domain%22%3A%22https%3A%2F%2Forb.domain5.
↪com%22%2C%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22%3A
↪%22z3VUDMo8tksYYD14crKHQx7HZLtJPwi7V4g2az1puHDhRYxTXSFQ3a2Qch7Az8niyQ1TXdKKaWzjvXoYNFUbJJmr9
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain5.com%23H72rWxdnDTaf69z2OkDLUG0z7XtFkBY_WmG9__U060Y%22%7D%2C%7B%22created
↪%22%3A%222022-09-20T20%3A57%3A11.966Z%22%2C%22domain%22%3A%22http%3A%2F%2Forb.vct
↪%3A8077%2Fmaple2020%22%2C%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22
↪%3A
↪%22z54WJu6r64W6Fq4LfiVzojHYHkwo4aVEMQ1KA15XYUhqwZPFwByqeV9Dwi3UPFPfKUsUsVUh92edAD1nGsd6n2nf6
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain1.com%23K3CezR1_tXyZSNbgdFbikNxkPCypfZ1bcJFh9NcsJlk%22%7D%5D%2C%22type%22
↪%3A%5B%22VerifiableCredential%22%2C%22AnchorCredential%22%5D%7D",
          "type": "application/ld+json"
        }
      ]
    }
  ]
}
```

The fields of the anchor linkset are described below.

Link Context (anchor)

The linkset contains one link context object where the context is specified by the "anchor" member. The "anchor" member contains the multihash of the canonicalized *Document Operation Batch* file. A number of link relations are defined for the anchor, which are described below.

Author Relation

The *author* relation specifies the author (creator) of the linkset.

Profile Relation

The *profile* relation specifies the version of the application that was used to generate this linkset. For example, the profile, "https://w3id.org/orb#v0", indicates that the linkset was generated by Orb version 0.

Original Relation

The *original* relation links to the *Document Operation Batch* file. In the above example anchor linkset, a data URL is used for the target of this relation such that the document operation batch is embedded in the href. NOTE: The contents of the data URL may be resolved by copying the href and pasting it into a browser.

Related Relation

The *related* relation links to the *Related Links Document*. In the above example anchor linkset, a data URL is used for the target of this relation such that the Related Links Document is embedded in the href. NOTE: The contents of the data URL may be resolved by copying the href and pasting it into a browser.

Replies Relation

The *replies* relation links to the *Anchor Credential Document*. In the above example anchor linkset, a data URL is used for the target of this relation such that the anchor credential is embedded in the href. NOTE: The contents of the data URL may be resolved by copying the href and pasting it into a browser.

AnchorEvent

An AnchorEvent is an ActivityPub object that contains the Anchor Linkset and is embedded in Create and Announce activities to propagate the Anchor Linkset to other Orb domains. For example:


```
{
  "@context": "https://w3id.org/activityanchors/v1",
  "object": {
    "linkset": [
      {
        "anchor": "hl:uEiAIMnHwbdHbpWbL3OlruU1xtqW-Potpi0bW0ioXFCZ94w",
        "author": [
          {
            "href": "did:web:orb.domain5.com:services:anchor"
          }
        ],
        "original": [
          {
            "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A%22hl
↪%3AuEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_o0ddDDCMZXlUxAQ%22%2C%22author%22%3A%5B%7B%22href%22
↪%3A%22did%3Aweb%3Aorb.domain5.com%3Aservices%3Aanchor%22%7D%5D%2C%22item%22%3A%5B%7B
↪%22href%22%3A%22did%3Aorb%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
↪%3AEiAISUte9hAoUQFyxGyzZBOLJiVOSo_6NvQI1_KTSrZEuw%22%2C%22previous%22%3A%5B%22hl
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%2C%7B%22href%22%3A%22did
↪%3Aorb%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
↪%3AEiCA9cT2WkrRbpVqvK9NYnfYL8oAXIGHozjzN_w9cetyJA%22%2C%22previous%22%3A%5B%22hl
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%2C%7B%22href%22%3A%22did
↪%3Aorb%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%3AEiALeU9pNA-LPMW5HsmMNhJ_-
↪JgTVP_fYnkA2InI8RtkzA%22%2C%22previous%22%3A%5B%22hl%3AuEiDI4O-
↪5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%2C%7B%22href%22%3A%22did%3Aorb
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
↪%3AEiApG3pQlSpclipU8ImLKpllbWlPCWHjobK_ASmGAjTeYA%22%2C%22previous%22%3A%5B%22hl
```


(continues on next page)

(continued from previous page)

```
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%2C%7B%22href%22%3A%22did
↪%3Aorb%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q
↪%3AEiAtOCeMIculDkB6mm8aSj0m928dwKfM8wqLk-iSo81V7w%22%2C%22previous%22%3A%5B%22hl
↪%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%22%5D%7D%5D%2C%22profile%22%3A%5B%7B
↪%22href%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%7D%5D%7D",
            "type": "application/linkset+json"
          }
        ],
        "profile": [
          {
            "href": "https://w3id.org/orb#v0"
          }
        ],
        "related": [
          {
            "href": "data:application/json,%7B%22linkset%22%3A%5B%7B%22anchor%22%3A%22hl
↪%3AuEiAIMnHwbdHbpWbL3OlruU1xtqW-Potpi0bW0ioXFCZ94w%22%2C%22profile%22%3A%5B%7B%22href
↪%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%7D%5D%2C%22up%22%3A%5B%7B%22href%22%3A
↪%22hl%3AuEiDI4O-5rOPmKYnmnhqghzKfcdlxcxxVKjaiju-LJPRp3Q%3AuoQ-
↪CeEtodHRwczovL29yYi5kb21haW4xLmNvbS9jYXMvdUVpREk0Ty01ck9QbUtZbm1uaHFnaHpLZmNkbHhjeHhWS2phaWp1LUxKUFJwk
↪%22%7D%5D%2C%22via%22%3A%5B%7B%22href%22%3A%22hl%3AuEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_
↪o0ddDDCMZXlUxAQ%3AuoQ-
↪BeEtodHRwczovL29yYi5kb21haW41LmNvbS9jYXMvdUVpQ0pZUzVKaW4tM1pTd0JUX1JUMGMwelowWjZCM19vMGRkRERDTVpYbFV4
↪%22%7D%5D%7D%5D%7D",
            "type": "application/linkset+json"
          }
        ],
        "replies": [
          {
            "href": "data:application/json,%7B%22%40context%22%3A%5B%22https%3A%2F%2Fwww.
↪w3.org%2F2018%2Fcredentials%2Fv1%22%2C%22https%3A%2F%2Fw3id.org%2Factivityanchors%2Fv1
↪%22%2C%22https%3A%2F%2Fw3id.org%2Fsecurity%2Fsuites%2Fed25519-2020%2Fv1%22%5D%2C
↪%22credentialSubject%22%3A%7B%22anchor%22%3A%22hl%3AuEiCJYS5Jin-3ZSwBT_RT0c0zZ0Z6B3_
↪o0ddDDCMZXlUxAQ%22%2C%22href%22%3A%22hl%3AuEiAIMnHwbdHbpWbL3OlruU1xtqW-
↪Potpi0bW0ioXFCZ94w%22%2C%22profile%22%3A%22https%3A%2F%2Fw3id.org%2Forb%23v0%22%2C
↪%22rel%22%3A%22linkset%22%2C%22type%22%3A%5B%22AnchorLink%22%5D%7D%2C%22id%22%3A
↪%22https%3A%2F%2Forb.domain5.com%2Fvc%2F2a649ce3-3412-4ca7-953c-bf89e333c3db%22%2C
↪%22issuanceDate%22%3A%222022-09-20T20%3A57%3A11.776289881Z%22%2C%22issuer%22%3A%22did
↪%3Aweb%3Aorb.domain5.com%3Aservices%3Aanchor%22%2C%22proof%22%3A%5B%7B%22created%22%3A
↪%222022-09-20T20%3A57%3A11.782036216Z%22%2C%22domain%22%3A%22https%3A%2F%2Forb.domain5.
↪com%22%2C%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22%3A
↪%22z3VUDMo8tksYYD14crKHQx7HZLtJPwi7V4g2az1puHDhRYxTXSFQ3a2Qch7Az8niyQ1TXdKKaWzjvXoYNFUbJJmr9
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain5.com%23H72rWxdnDTaf69z2OkDLUG0z7XtFkBY_WmG9__U060Y%22%7D%2C%7B%22created
↪%22%3A%222022-09-20T20%3A57%3A11.966Z%22%2C%22domain%22%3A%22http%3A%2F%2Forb.vct
↪%3A8077%2Fmaple2020%22%2C%22proofPurpose%22%3A%22assertionMethod%22%2C%22proofValue%22
↪%3A
↪%22z54WJu6r64W6Fq4LfiVzojHYHkwo4aVEMQ1KA15XYUhqwZPFwByqeV9Dwi3UPFPfKUsUsVUh92edAD1nGsd6n2nf6
↪%22%2C%22type%22%3A%22Ed25519Signature2020%22%2C%22verificationMethod%22%3A%22did%3Aweb
↪%3Aorb.domain1.com%23K3CezR1_tXyZSNbgdFbikNxkPCypfZ1bcJFh9NcsJlk%22%7D%5D%2C%22type%22
↪%3A%5B%22VerifiableCredential%22%2C%22AnchorCredential%22%5D%7D",
            "type": "application/ld+json"
```

(continues on next page)

(continued from previous page)

```
          }
        ]
      }
    ]
  },
  "type": "AnchorEvent",
  "url": "hl:uEiDmX8aJHcpNUyKg8bTuxG7haDCqLWZ49iOXXd-SW3K-3A:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW41LmNvbS9jYXMvdUVpRG1YOGFKSGNwTlV5S2c4YlR1eEc3aGFEQ3FMV1o0OWlPWFhkLVNXM0st
↪"
}
```

The "object" field contains the *Anchor Linkset* and the "url" field contains the hashlink of the Anchor Linkset, which includes the links to the Anchor Linkset. For example:

```
hl:uEiDmX8aJHcpNUyKg8bTuxG7haDCqLWZ49iOXXd-SW3K-3A:uoQ-
↪BeEtodHRwczovL29yYi5kb21haW41LmNvbS9jYXMvdUVpRG1YOGFKSGNwTlV5S2c4YlR1eEc3aGFEQ3FMV1o0OWlPWFhkLVNXM0st
```

The hashlink above contains the multihash of the Anchor Linkset, `uEiDmX8aJHcpNUyKg8bTuxG7haDCqLWZ49iOXXd-SW3K-3A` and a link to the file:

```
https://orb.domain5.com/cas/uEiDmX8aJHcpNUyKg8bTuxG7haDCqLWZ49iOXXd-SW3K-3A
```

CHAPTER

FOUR

PROJECTS

4.1 Edge

- Edge-Adapter
- Edge-Agent
- Edge-Core
- Edge-Sandbox
- Edge-Service

4.2 Agent SDK

- Agent SDK

4.3 Bloc Hub

- Bloc-Hub

4.4 Sidetree

- Orb
- Sidetree-Node

4.5 Upstream Project

- Hyperledger Aries
- DIF

CHAPTER

FIVE

VERIFIABLE CREDENTIAL SERVICE (VCS)

5.1 What is a Verifiable Credential (VC)?

We use credentials everyday. A driver's license issued by the government certify that we are capable of operating a vehicle on the road. A Permanent Residence card shows the immigration status of an individual.

A verifiable credential is then a document whose contents can be cryptographically proven/verified ([VC-TERM]) to be true. A VC could hold the same data that a physical credential does. Within the scope of TrustBloc projects, this act of verifying credentials can be done with the aid of technology such as digital identities and signatures. The use of digital signatures adds to the integrity of a credential when it is presented.

Holders of verifiable credentials can generate verifiable presentations and then share these verifiable presentations with verifiers to prove they possess verifiable credentials with certain characteristics. Both verifiable credentials and verifiable presentations can be transmitted rapidly, making them more convenient than their physical counterparts when trying to establish trust at a distance. ([VC-DEF])

5.2 Edge-Service

TrustBloc's Edge-Service contains servers that handle the issuance and verification of verifiable credentials.

5.2.1 Configuring the service

Edge-Service can be used in the following modes:

- Issuer
- Verifier
- Holder
- Governance

Get vcs-rest from GitHub packages.

Configuration flags for the server:

```
Start vc-rest inside the edge-service

Usage:
   vc-rest start [flags]

Flags:
```

(continues on next page)

(continued from previous page)

```
        --api-token string                    Check for bearer token in the␣
↪authorization header (optional). Alternatively, this can be set with the following␣
↪environment variable: VC_REST_API_TOKEN
   -f, --backoff-factor string                If no VC is found when attempting to␣
↪retrieve a VC from the EDV, this is the factor to increase the time to wait for␣
↪subsequent retries after the first. Alternatively, this can be set with the following␣
↪environment variable: BACKOFF-FACTOR
   -b, --bloc-domain string                   Bloc domain
        --database-prefix string               An optional prefix to be used when␣
↪creating and retrieving underlying databases. Alternatively, this can be set with the␣
↪following environment variable: DATABASE_PREFIX
   -t, --database-type string                 The type of database to use for everything␣
↪except key storage. Supported options: mem, couchdb. Alternatively, this can be set␣
↪with the following environment variable: DATABASE_TYPE
   -v, --database-url string                  The URL of the database. Not needed if␣
↪using memstore. For CouchDB, include the username:password@ text if required.␣
↪Alternatively, this can be set with the following environment variable: DATABASE_URL
   -e, --edv-url string                       URL EDV instance is running on. Format:␣
↪HostName:Port.
        --governance-claims-file string        Path to governance claimsAlternatively,␣
↪this can be set with the following environment variable: VC_REST_GOVERNANCE_CLAIMS_FILE
   -h, --help                                 help for start
   -u, --host-url string                      URL to run the vc-rest instance on.␣
↪Format: HostName:Port.
   -x, --host-url-external string             Host External Name:Port This is the URL␣
↪for the host server as seen externally. If not provided, then the host url will be␣
↪used here. Alternatively, this can be set with the following environment variable: VC_
↪REST_HOST_URL_EXTERNAL
   -i, --initial-backoff-millisec string      If no VC is found when attempting to␣
↪retrieve a VC from the EDV, this is the time to wait (in milliseconds) before the␣
↪first retry attempt. Alternatively, this can be set with the following environment␣
↪variable: INITIAL_BACKOFF_MILLISEC
        --kms-secrets-database-prefix string   An optional prefix to be used when␣
↪creating and retrieving the underlying KMS secrets database. Alternatively, this can␣
↪be set with the following environment variable: KMSSECRETS_DATABASE_PREFIX
   -k, --kms-secrets-database-type string     The type of database to use for storage of␣
↪KMS secrets. Supported options: mem, couchdb. Alternatively, this can be set with the␣
↪following environment variable: KMSSECRETS_DATABASE_TYPE
   -s, --kms-secrets-database-url string      The URL of the database. Not needed if␣
↪using memstore. For CouchDB, include the username:password@ text if required. It's␣
↪recommended to not use the same database as the one set in the database-url flag (or␣
↪the DATABASE_URL env var) since having access to the KMS secrets may allow the host of␣
↪the provider to decrypt EDV encrypted documents. Alternatively, this can be set with␣
↪the following environment variable: DATABASE_URL
   -l, --log-level string                     Logging level to set. Supported options:␣
↪CRITICAL, ERROR, WARNING, INFO, DEBUG.Defaults to info if not set. Setting to debug␣
↪may adversely impact performance. Alternatively, this can be set with the following␣
↪environment variable: LOG_LEVEL
   -a, --max-retries string                   If no VC is found when attempting to␣
↪retrieve a VC from the EDV, this is the maximum number of times to retry retrieval.␣
↪Defaults to 5 if not set. Alternatively, this can be set with the following␣
↪environment variable: MAX-RETRIES
```

(continues on next page)

(continued from previous page)

```
   -m, --mode string                           Mode in which the vc-rest service will run.
↪ Possible values: ['issuer', 'verifier', 'holder', 'combined'] (default: combined).
        --request-tokens stringArray           Tokens used for http request␣
↪Alternatively, this can be set with the following environment variable: VC_REST_
↪REQUEST_TOKENS
        --tls-cacerts stringArray              Comma-Separated list of ca certs path.
↪Alternatively, this can be set with the following environment variable: VC_REST_TLS_
↪CACERTS
        --tls-systemcertpool string            Use system certificate pool. Possible␣
↪values [true] [false]. Defaults to false if not set. Alternatively, this can be set␣
↪with the following environment variable: VC_REST_TLS_SYSTEMCERTPOOL
   -r, --universal-resolver-url string         Universal Resolver instance is running on.␣
↪Format: HostName:Port.
```

Example: Running in Issuer Mode

The following is a snippet of a Docker Compose ™ file showing how Edge Service can be configured. It makes use of environment variables declared here.

```
issuer.vcs.example.com:
  container_name: issuer.vcs.example.com
  image: ${VCS_IMAGE}:${VCS_IMAGE_TAG}
  environment:
    - VC_REST_HOST_URL=0.0.0.0:8070
    - VC_REST_HOST_URL_EXTERNAL=https://issuer-vcs.trustbloc.local
    - EDV_REST_HOST_URL=https://edv.trustbloc.local/encrypted-data-vaults
    - BLOC_DOMAIN=${BLOC_DOMAIN}
    - UNIVERSAL_RESOLVER_HOST_URL=https://did-resolver.trustbloc.local/1.0/identifiers
    - VC_REST_MODE=issuer
    - DATABASE_TYPE=couchdb
    - DATABASE_URL=${COUCHDB_USERNAME}:${COUCHDB_PASSWORD}@shared.couchdb:5984
    - DATABASE_PREFIX=issuer
    - KMSSECRETS_DATABASE_TYPE=couchdb
    - KMSSECRETS_DATABASE_URL=${COUCHDB_USERNAME}:${COUCHDB_PASSWORD}@shared.couchdb:5984
    - KMSSECRETS_DATABASE_PREFIX=issuer
    - VC_REST_TLS_CACERTS=/etc/tls/trustbloc-dev-ca.crt
    - VC_REST_TLS_SYSTEMCERTPOOL=true
    - VC_REST_API_TOKEN=vcs_issuer_rw_token
    - VIRTUAL_HOST=issuer-vcs.trustbloc.local
  ports:
    - 8070:8070
  entrypoint: ""
  # wait 20 seconds for couchdb to start
  command:  /bin/sh -c "sleep 20;/tmp/scripts/vcs_configure.sh& vc-rest start"
  volumes:
    - ../scripts/:/tmp/scripts #https://github.com/trustbloc/edge-sandbox/tree/master/
↪test/bdd/fixtures/scripts
    - ../keys/tls:/etc/tls
  depends_on:
    - edv.example.com
  networks:
```

(continues on next page)

(continued from previous page)

```
    - demo-net

edv.example.com:
  container_name: edv.example.com
  image: ${EDV_IMAGE}:${EDV_IMAGE_TAG}
  environment:
    - EDV_HOST_URL=0.0.0.0:8081
    - EDV_DATABASE_TYPE=couchdb
    - EDV_DATABASE_URL=${COUCHDB_USERNAME}:${COUCHDB_PASSWORD}@shared.couchdb:5984
    - EDV_DATABASE_PREFIX=edv
    - VIRTUAL_HOST=edv.trustbloc.local
  ports:
    - 8081:8081
  command: start
  networks:
    - demo-net
```

Examples of how the other modes can be configures is available in the following repos:

- edge-sandbox
- edge-service

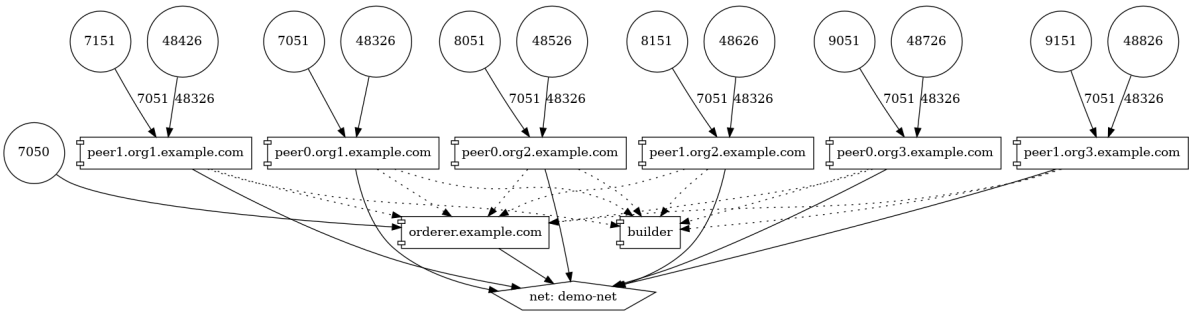
5.2.2 Deploying the service

In order to deploy Edge-Service, the following components are required.

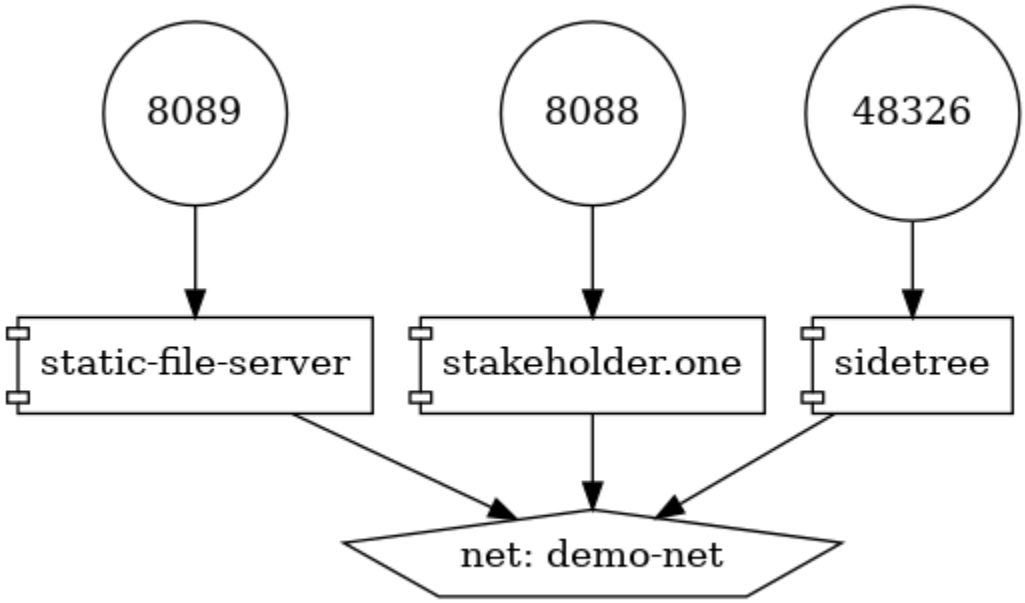
:::{note} An example of how these components interact together is shown here. :::

Sidetree

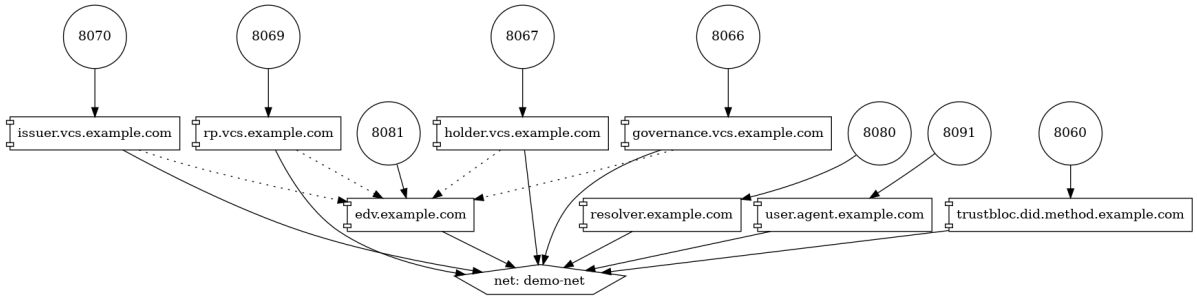
Sidetree Fabric



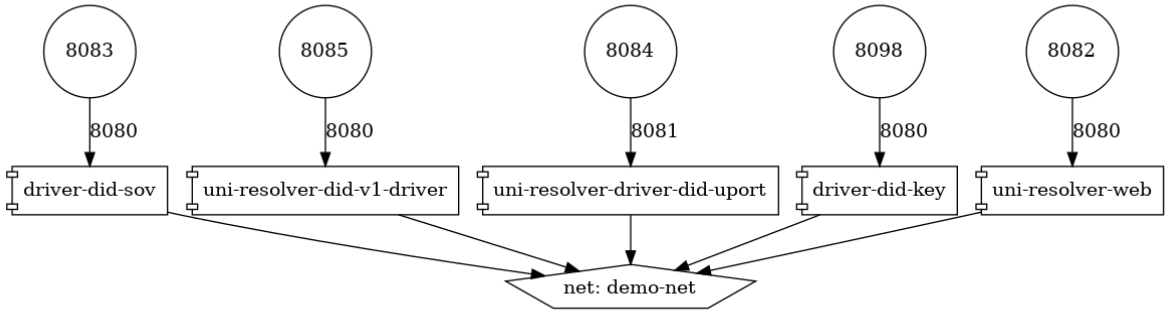
Sidetree Mock



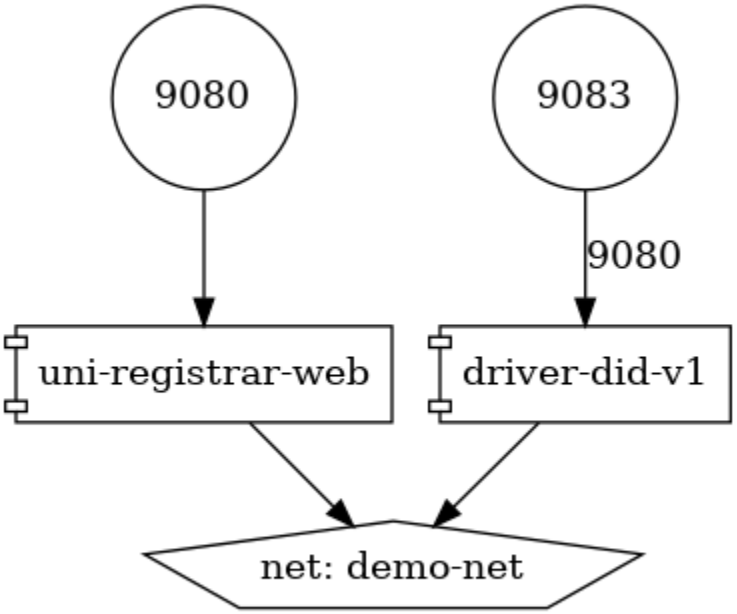
Edge Components



DID Resolvers



DID Registrars



5.2.3 VCS Components (CHAPI + VC Services)

5.2.4 Issuing a VC

In order to issue a Verifiable Credential, you will need to first create a profile.

1. Issue a VC

HTTP POST /{profile}/credentials/issueCredential

```
{
   "credential":{
      "@context":[
         "https://www.w3.org/2018/credentials/v1"
      ],
      "id":"http://example.edu/credentials/1872",
      "type":"VerifiableCredential",
      "credentialSubject":{
         "id":"did:example:ebfeb1f712ebc6f1c276e12ec21"
      },
      "issuer":{
         "id":"did:example:76e12ec712ebc6f1c221ebfeb1f",
         "name":"Example University"
      },
      "issuanceDate":"2010-01-01T19:23:24Z",
      "credentialStatus":{
         "id":"https://example.gov/status/24",
         "type":"CredentialStatusList2017"
      }
   },
   "options":{
      "assertionMethod":"did:trustbloc:testnet.trustbloc.
↪local:EiAiijiRNEAflOr6ZOJN5A7BCFQD1pwFMI1MPzHr3bXezg=="
   }
}
```

More details here.

Try it here.

2. Compose and Issue a VC

HTTP POST /{profile}/credentials/composeAndIssueCredential

```
{
   "issuer":"did:example:uoweu180928901",
   "subject":"did:example:oleh394sqwnlk223823ln",
   "types":[
      "UniversityDegree"
   ],
```

(continues on next page)

(continued from previous page)


```
   "issuanceDate":"2020-03-25T19:38:54.45546Z",
   "expirationDate":"2020-06-25T19:38:54.45546Z",
   "claims":{
      "name":"John Doe"
   },
   "evidence":{
      "id":"http://example.com/policies/credential/4",
      "type":"IssuerPolicy"
   },
   "termsOfUse":{
      "id":"http://example.com/policies/credential/4",
      "type":"IssuerPolicy"
   },
   "proofFormat":"jws",
   "proofFormatOptions":{
      "kid":"did:trustbloc:testnet.trustbloc.local:EiAtPEWAphdPVRxlKpr8N43uyLMhgF-
↪9SFmYfINVpDIzUA==#key-1"
   }
}
```


More details here.

5.2.5 Validating a VC

HTTP POST /verifier/credentials


```
{
   "verifiableCredential":{
      "@context":[
         "https://www.w3.org/2018/credentials/v1",
         "https://www.w3.org/2018/credentials/examples/v1"
      ],
      "credentialSchema":[

      ],
      "credentialStatus":{
         "id":"http://issuer.vc.rest.example.com:8070/status/1",
         "type":"CredentialStatusList2017"
      },
      "credentialSubject":{
         "degree":{
            "degree":"MIT",
            "type":"BachelorDegree"
         },
         "id":"did:example:ebfeb1f712ebc6f1c276e12ec21",
         "name":"Jayden Doe",
         "spouse":"did:example:c276e12ec21ebfeb1f712ebc6f1"
      },
      "id":"http://example.gov/credentials/3732",
      "issuanceDate":"2020-03-16T22:37:26.544Z",
      "issuer":{
```


(continues on next page)

(continued from previous page)


```
        "id":"did:example:oakek12as93mas91220dapop092",
        "name":"University"
      },
      "proof":{
        "created":"2020-04-09T15:35:35Z",
        "jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..
↪kN1srfFqoiejHJwxM8Y0Y9yIonAvFeF2Aoiv6_LTkPqcNXc2rXwT94-uO_PQJbxWJgTD78MvpfCJWsUSRvgCBw
↪",
        "proofPurpose":"assertionMethod",
        "type":"Ed25519Signature2018",
        "verificationMethod":"did:trustbloc:testnet.trustbloc.
↪local:EiD3KVRkHAHt6aLO4Kp5PSO3pNhAY_GPZXuKUekVk1uboQ==#key-1"
      },
      "type":[
        "VerifiableCredential",
        "UniversityDegreeCredential"
      ]
   },
   "options":{
      "checks":[
        "proof"
      ]
   }
}
```


More details here.

Try it here.

5.3 Connecting to the TestNet

5.3.1 Service Endpoints

VC SERVICES

issuer_vcs: https://issuer.sandbox.trustbloc.dev

verifier_vcs: https://verifier.sandbox.trustbloc.dev

holder_vcs: https://holder.sandbox.trustbloc.dev

governance_vcs: https://governance.sandbox.trustbloc.dev

VC Adapters

issuer_adapter:

: rest: https://issuer-adapter.sandbox.trustbloc.dev

didcomm: https://issuer-adapter-didcomm.sandbox.trustbloc.dev

verifier_adapter: #RP/Verifier Adapter

: rest: https://verifier-adapter.sandbox.trustbloc.dev

didcomm: https://verifier-adapter-didcomm.sandbox.trustbloc.dev

hydra: https://verifier-adapter-hydra.sandbox.trustbloc.dev

hydra_admin: https://verifier-adapter-hydra-admin.sandbox.trustbloc.dev

EDV/SDS

sds: https://sds.sandbox.trustbloc.dev

resolver: https://resolver.sandbox.trustbloc.dev

registrar: https://registrar.sandbox.trustbloc.dev

kms: https://kms.sandbox.trustbloc.dev

Wallet Mediator URL

router:

: http: https://router.sandbox.trustbloc.dev

api: https://router-api.sandbox.trustbloc.dev

ws: wss://router-ws.sandbox.trustbloc.dev

router_ws: wss://router-ws.sandbox.trustbloc.dev

router_api: agent: https://agent.sandbox.trustbloc.dev

uni_did: https://uni-did.sandbox.trustbloc.dev

registrar_v1_driver: https://registrar-v1-driver.sandbox.trustbloc.dev

resolver_sov_driver: https://resolver-sov-driver.sandbox.trustbloc.dev

resolver_veresone_driver : https://resolver-veresone-driver.sandbox.trustbloc.dev

resolver_uport_driver: https://resolver-uport-driver.sandbox.trustbloc.dev

resolver_didkey_driver: https://resolver-didkey-driver.sandbox.trustbloc.dev

Demo and 3rd party endpoints

demo_issuer: https://demo-issuer.sandbox.trustbloc.dev

demo_verifier: https://demo-verifier.sandbox.trustbloc.dev

hydra: https://hydra.sandbox.trustbloc.dev

hydra_admin: https://hydra-admin.sandbox.trustbloc.dev

strapi: https://strapi.sandbox.trustbloc.dev

cms: https://cms.sandbox.trustbloc.dev

login: https://login.sandbox.trustbloc.dev

agent_resolver_urls:

- mailto:trustbloc@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- mailto:v1@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- mailto:elem@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- mailto:sov@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- mailto:web@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- mailto:key@https://resolver.sandbox.trustbloc.dev/1.0/identifiers

5.4 Using Edge-Service

To use the demo, navigate to the Demo Issuer homepage.

Then follow the steps in the videos below for their respective demonstrations.

These demos make use of Edge-Sandbox which is a demo environment for edge-service.

5.4.1 Register A Wallet

Be sure to register your wallet as in the video below:

5.4.2 Issue a Credit Score Report

5.4.3 Issue a Driver's License

5.5 References

CHAPTER

SIX

KEY MANAGEMENT SYSTEM (KMS)

6.1 Introduction

6.1.1 What is KMS?

KMS lets to manage cryptographic keys and use them to perform crypto operations. TrustBloc KMS is a server implementation of KMS and Crypto APIs from the Aries Framework Go.

It can be used as a remote KMS and a remote Crypto for the Aries Framework's `webkms` implementation.

6.2 Startup Parameters

KMS server can be run as a standalone binary or a docker container. Refer here for the instructions on how to run a server and supported startup flags and options.

6.3 REST Endpoints

Following is a list of all REST endpoints exposed by KMS:

6.3.1 Server

GET /healthcheck

Returns a health check status.

6.3.2 KMS

POST /v1/keystores

Creates a new key store.

POST /v1/keystores/{key_store_id}/keys

Creates a new key.

PUT /v1/keystores/{key_store_id}/keys

Imports a private key.

GET /v1/keystores/{key_store_id}/keys/{key_id}

Exports a public key.

POST /v1/keystores/{key_store_id}/keys/{key_id}/rotate

Rotates the key.

POST /v1/keystores/did

Creates a DID.

6.3.3 Crypto

POST /v1/keystores/{key_store_id}/easyopen

Unseals a message sealed with Easy.

POST /v1/keystores/{key_store_id}/keys/{key_id}/computemac

Computes message authentication code (MAC) for data.

POST /v1/keystores/{key_store_id}/keys/{key_id}/decrypt

Decrypts a ciphertext with associated authenticated data.

POST /v1/keystores/{key_store_id}/keys/{key_id}/deriveproof

Creates a BBS+ signature proof for a list of revealed messages.

POST /v1/keystores/{key_store_id}/keys/{key_id}/easy

Seals a message.

POST /v1/keystores/{key_store_id}/keys/{key_id}/encrypt

Encrypts a message with associated authenticated data.

POST /v1/keystores/{key_store_id}/keys/{key_id}/sign

Signs a message.

POST /v1/keystores/{key_store_id}/keys/{key_id}/signmulti

Creates a BBS+ signature of messages.

POST /v1/keystores/{key_store_id}/keys/{key_id}/unwrap

Unwraps a wrapped key.

POST /v1/keystores/{key_store_id}/keys/{key_id}/verify

Verifies a signature.

POST /v1/keystores/{key_store_id}/keys/{key_id}/verifymac

Verifies whether MAC is a correct authentication code for data.

POST /v1/keystores/{key_store_id}/keys/{key_id}/verifymulti

Verifies a signature of messages (BBS+).

POST /v1/keystores/{key_store_id}/keys/{key_id}/verifyproof

Verifies a BBS+ signature proof for revealed messages.

POST /v1/keystores/{key_store_id}/keys/{key_id}/wrap

Wraps CEK using ECDH-1PU key wrapping (Authcrypt).

POST /v1/keystores/{key_store_id}/sealopen

Decrypts a payload encrypted with Seal.

POST /v1/keystores/{key_store_id}/wrap

Wraps CEK using ECDH-ES key wrapping (Anoncrypt).

6.4 Keys

6.4.1 Types of the keys in KMS

1) User's working key - for operations initiated by the user (sign, wrap, encrypt, etc.).
2) Secret lock key - for encrypting/decrypting other keys in the server's DB or EDV.
3) Keys for supporting EDV operations (recipient public key, MAC key).

6.4.2 Where keys are stored

Users' working keys are stored either in the server's DB (CouchDB or MongoDB) or in EDV. This depends on whether the `create key store` request contains EDV options (vault URL and authorization capabilities (ZCAPs) to access vault).

When EDV is used, both recipient and MAC keys that are needed for the EDV provider are stored in the server's DB. In EDV data are encrypted so the private key associated with the **recipient public key** can decrypt them. **MAC key** is used for generating deterministic document ID.

Any key that is stored in the server's DB is encrypted with the lock key. There are 2 options for the server's lock key: **local file** or **AWS KMS**.

Users' working keys are protected with the secret lock as well. There are also 2 options for the user's secret lock key: **local key** stored in the server's DB (that key is created when the `create key store` request is processed and is associated with the key store) or key based on **Shamir Secret Sharing** scheme. In this case, a key is generated on a fly from 2 shares - one is stored on Auth server and the other comes in the header with each request.

6.4.3 Use cases

Scenario 1: server's lock is based on AWS key, user's lock uses local key, no EDV

In this scenario, a key for the user's lock is created when the key store is created. That key is encrypted with an AWS key and stored in the server's DB. When a working key is created for the user, it is encrypted with that stored lock key. Before using, user's lock key should be decrypted with an AWS key.

Scenario 2: server's lock is based on local key, user's lock uses Shamir-based key, working keys are stored in EDV

Key for the server's lock is stored in a local file and the path to it is specified in a startup flag or environment variable. When a key store is created, helper recipient and MAC keys for the EDV provider are created as well. They are encrypted with a key from the local file (server's lock) and saved to the server's DB. These keys are associated with a created key store to support EDV operations.

User's lock key is created on a fly using HKDF algorithm that expands the combined secret (from shares using Shamir Secret Sharing) into a symmetric key. That key is used to encrypt/decrypt the user's working keys stored in EDV.

6.4.4 What is the impact of losing/compromising the key

Key	Loss/compromise
server's lock key	no/malicious access to all keys locked with that key
user's lock key	no/malicious access to all user's working keys
user's working key	blocked/malicious operations with that key (signing, encrypting, etc.)
EDV recipient key	no/malicious access to encrypted doc in EDV

6.5 Key Rotation

There are three types of keys in KMS that can be rotated:

- user operational keys
- primary key for the user's key store
- primary key for the server's key store

6.5.1 User operational keys

A user operation key can be rotated by making a request

`POST /v1/keystores/{keystore}/keys/{key}/rotate`.

It will return a new URL for the rotated key.

When the key is rotated, a new key material is added, but old ones are not deleted. New key material is used to encrypt new data. To decrypt data at first the new key material is used. If decryption fails, old key material is used. This allows decrypting data that were encrypted before rotation.

Key material - cryptographic data used for encrypt/decrypt operations. Key - set of key material ordered by date, from the newest to oldest.

Key rotation is implemented using LocalKMS.Rotate from aries-framework-go.

6.5.2 User key store primary key

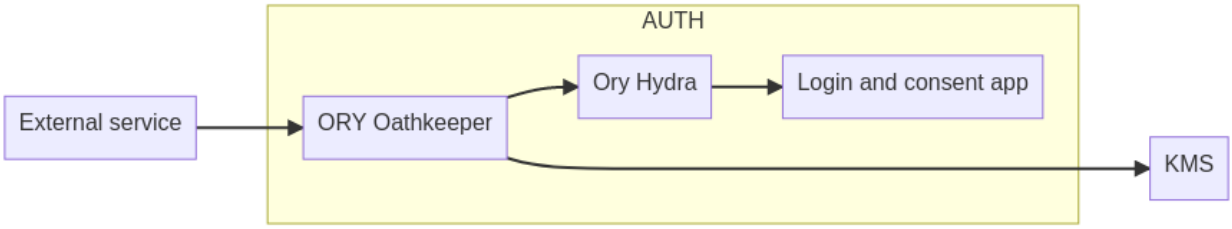
Not implemented yet.

6.5.3 Server key store primary key

The server key store uses AWS KMS to protect keys. AWS KMS supports automatic key rotations.

6.6 Auth

KMS directly doesn't implement auth mechanism. It rely on ORY Oathkeeper.



###ORY Oathkeeper ORY Oathkeeper is an Identity & Access Proxy (IAP) and Access Control Decision API that authorizes HTTP requests based on sets of Access Rules. ORY Oathkeeper is deployed in front of KMS service and is capable of authenticating and optionally authorizing access requests.

6.6.1 Ory Hydra

Ory Hydra is OpenID Certified OAuth 2.0 Server and OpenID Connect Provider. Ory Hydra is not an identity provider (user sign up, user login, password reset flow), but connects to our existing identity provider through a login and consent app.

6.6.2 Login and consent app

Login and consent app can be any OIDC identity provider

6.7 CLI

The KMS Command Line Interface (KMS CLI) is a unified tool that provides a consistent interface for interacting with KMS.

6.7.1 Create Keystore

This command is used for creating a keystore.

Usage

```
keystore create [flags]
```

Flags

- `controller` *[string]* - DID of keystore controller.
- `url` *[string]* - URL of KMS server.
- `auth-token` *[string]* - The Auth token.
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.

Example

create keystore cmd

```
keystore create --controller did:example:123456 --url https://localhost:8078
--tls-cacerts fixtures/keys/tls/ec-cacert.pem
```

6.7.2 Create Key

This command is used for creating keys in the keystore.

Usage

```
key create [flags]
```

Flags

- `keystore` *[string]* - ID of the keystore.
- `type` *[string]* - Type of the key.
- `url` *[string]* - URL of KMS server.
- `auth-token` *[string]* - The Auth token.
- `tls-cacerts` *[array|string]* - Array of one or more CA cert paths.
- `tls-systemcertpool` *[boolean]* - Flag whether to use system certificate pool.

Example

create key cmd

```
key create --keystore {keystoreID} --url https://localhost:8078 --type ED25519
--tls-cacerts fixtures/keys/tls/ec-cacert.pem
```

6.8 Metrics

KMS server records performance metrics at each subsystem. A Prometheus server may be used to periodically read the metrics at the URL `KMS_METRICS_HOST/metrics`. Below are the metrics defined at each subsystem.

6.8.1 Crypto

The Crypto metrics measure times for crypto operations.

crypto_sign_seconds

The time (in seconds) it takes to sign message.

6.8.2 Database

Database metrics record the times for reads, writes, bulk writes, etc.

db_put_seconds

The time (in seconds) it takes the DB to store data.

db_get_seconds

The time (in seconds) it takes the DB to retrieve data by primary key.

db_get_tags_seconds

The time (in seconds) it takes the DB to get tags.

db_get_bulk_seconds

The time (in seconds) it takes the DB to get bulk.

db_query_seconds

The time (in seconds) it takes to query for data.

db_delete_seconds

The time (in seconds) it takes to delete data.

db_batch_seconds

The time (in seconds) it takes to perform a batch update.

6.8.3 Key store.

The Key store metrics measure times for operations with keys in a key store.

key_store_resolve_seconds

The time (in seconds) it takes to resolve a user key store.

key_store_get_key_seconds

The time (in seconds) it takes to get a key from a user key store.

key_store_aws_secret_lock_encrypt_seconds

The time (in seconds) it takes to encrypt a key using AWS secret lock.

key_store_aws_secret_lock_decrypt_seconds

The time (in seconds) it takes to decrypt a key using AWS secret lock.

key_store_key_secret_lock_encrypt_seconds

The time (in seconds) it takes to encrypt a key using a key-based secret lock.

key_store_key_secret_lock_decrypt_seconds

The time (in seconds) it takes to decrypt a key using a key-based secret lock.

6.9 Caching

KMS uses caching to improve the performance of several components (server's database, users' key stores, etc.). Caching functionality is backed by the ristretto implementation.

Caching support is enabled with `KMS_CACHE_ENABLE=true` environment variable (`--enable-cache=true` flag). This turns on caching for operations with server's database. Other caches can be additionally configured with further parameters.

6.9.1 Server's DB cache

The server's DB is used for storing and retrieving

- keys for server's KMS;
- as a default option for users' key stores (EDV is an alternative option);
- JSON-LD contexts and authorization capabilities (ZCAP-LD);
- metadata for users' key stores.

Cache wraps the storage and acts as a proxy, so the operations with the underlying database are cached.

It's important to note that when the server's KMS uses cached storage, key materials are cached in an encrypted form. The secret lock is still required to get usable keys.

6.9.2 Key store cache

Users' key stores can be cached for a limited period of time set by `KMS_KEY_STORE_CACHE_TTL=10m` environment variable (`--key-store-cache-ttl=10m` flag). The default value is 10 minutes.

This cache is especially useful when the key store uses EDV. The keys' materials are always cached in an encrypted form.

6.9.3 Secret lock keys cache

In the case of using AWS KMS as a secret lock for server kms, encrypt and decrypt operations can become expensive. To decrease the number of calls to AWS KMS, decrypted keys can be cached. Caching is enabled by default and default ttl is 10m. To disable caching set KMS_KMS_CACHE_TTL=0s. An appropriate ttl can be set using the same KMS_KMS_CACHE_TTL variable.

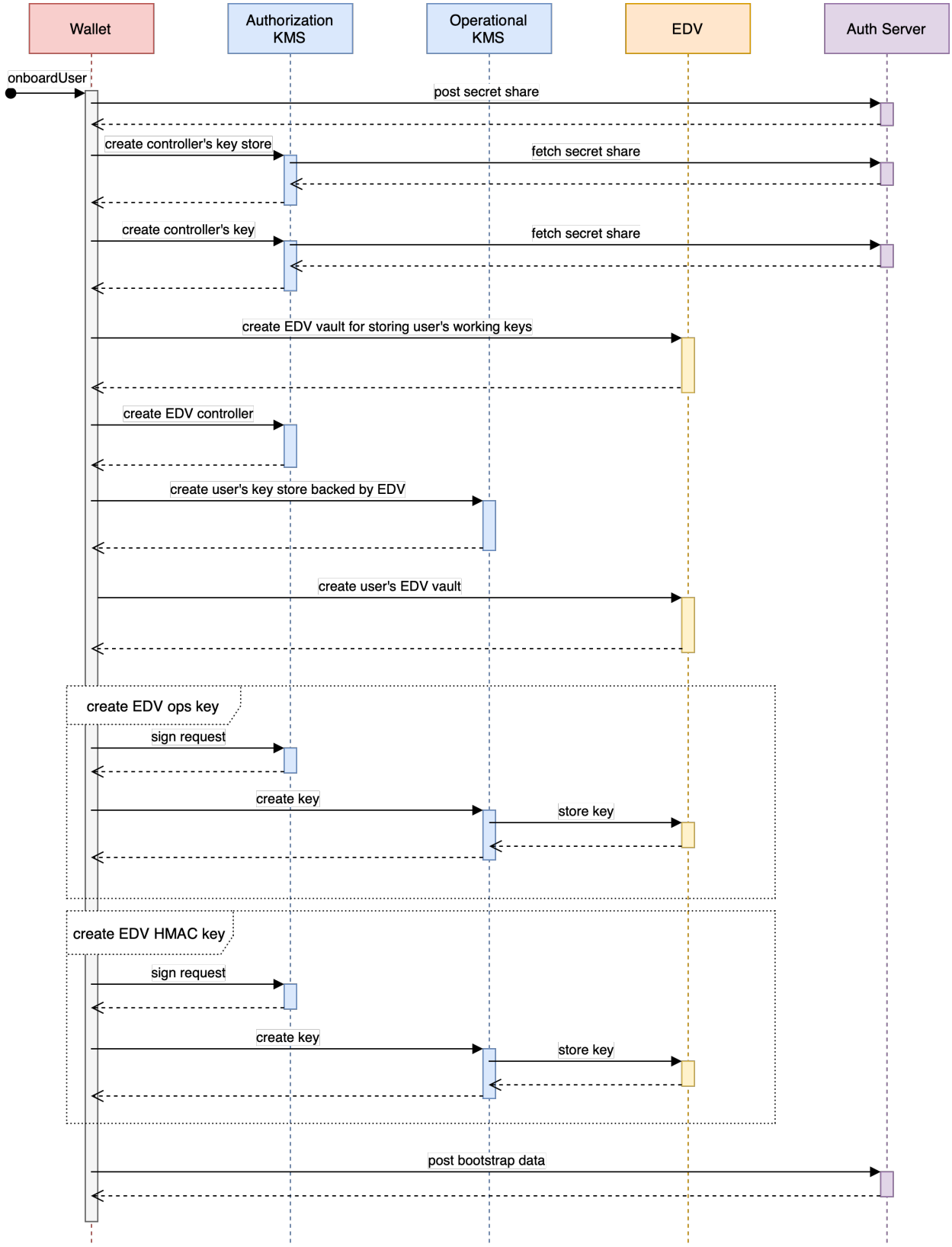
6.9.4 Shamir secret cache

Shamir secrets fetched from Auth Server are cached by default. Default ttl for Shamir secrets is 10m. To disable caching set KMS_SHAMIR_SECRET_CACHE_TTL=0s. An appropriate ttl can be set using the same KMS_SHAMIR_SECRET_CACHE_TTL variable.

6.10 User Onboarding and Recovery

In the TrustBloc environment, KMS server is used as an **authorization KMS** for supporting ZCAP-LD scheme and as an **operational KMS** for regular user's crypto operations.

Both are part of the user onboarding flow in the Wallet.


Wallet
Authorization KMS
Operational KMS
EDV
Auth Server
onboardUser
post secret share
create controller's key store
fetch secret share
create controller's key
fetch secret share
create EDV vault for storing user's working keys
create EDV controller
create user's key store backed by EDV
create user's EDV vault
create EDV ops key
sign request
create key
store key
create EDV HMAC key
sign request
create key
store key
post bootstrap data

6.10.1 Authorization KMS

Authorization KMS allows creating Controller identity (identified by a cryptographic key pair) and then using it for signing requests in a ZCAP-LD authorization model.

The key store for the Controller uses the server's database for storing keys. These keys are encrypted with a Shamir secret lock.

To unlock a key store, the controller's secret share needs to be present in a `Secret-Share` header for every request. The other share comes from the Auth server. The URL of the server is configured with `KMS_AUTH_SERVER_URL` environment variable (or `--auth-server-url` flag).

Server Key Manager is protected with a Local secret lock.

6.10.2 Operational KMS

Operational KMS manages users' working keys and supports crypto operations with those keys.

Keys are protected with a Local secret lock and saved to the EDV server provided by the user. EDV parameters are specified in the request upon creating a key store. Refer to the Storage section for the details.

Recipient and MAC keys for accessing an EDV server are stored in Server Key Manager. A primary key for the user's key store lock is also saved here.

Operational KMS uses ZCAP-LD authorization scheme. Requests are expected to be signed with the Authorization KMS.

Server Key Manager is protected with an AWS secret lock.

6.10.3 Oathkeeper

Certain KMS endpoints are protected with the OAuth scheme. TrustBloc uses Oathkeeper as an authorization proxy in front of KMS services to protect the following endpoints:

- `POST /v1/keystores` for creating a key store;
- `POST /v1/keystores/did` for creating a DID (e.g. controller for EDV).

An example configuration for Authorization KMS can be found here.

6.10.4 Recovery

Planned functionality.

CHAPTER

SEVEN

ADAPTERS

7.1 What is an Adapter?

TrustBloc's Edge-Adapter acts as a go-between for Relying Party (RP) and Issuer components to support DIDComm operations.

TrustBloc's Edge-Adapter can be used to run an Issuer and an RP.

Get the adapter here.

Here are the flags for the server:

```
Start adapter-rest inside the edge-adapter

Usage:
        adapter-rest start [flags]

Flags:
        --didcomm-db-path string                Path to database. Alternatively, this␣
↪can be set with the following environment variable: ADAPTER_REST_DIDCOMM_DB_PATH
        --didcomm-inbound-host string           Inbound Host Name:Port. This is used␣
↪internally to start the didcomm server. Alternatively, this can be set with the␣
↪following environment variable: ADAPTER_REST_DIDCOMM_INBOUND_HOST
        --didcomm-inbound-host-external string  Inbound Host External Name:Port. This␣
↪is the URL for the inbound server as seen externally. If not provided, then the␣
↪internal inbound host will be used here. Alternatively, this can be set with the␣
↪following environment variable: ADAPTER_REST_DIDCOMM_INBOUND_HOST_EXTERNAL
        --dids-trustbloc-domain string          URL to the did:trustbloc consortium's␣
↪domain. Alternatively, this can be set with the following environment variable:␣
↪ADAPTER_REST_TRUSTBLOC_DOMAIN
        --dsn string                            Datasource Name with credentials if␣
↪required. Format must be <driver>:[//]<driver-specific-dsn>. Examples: 'mysql://
↪root:secret@tcp(localhost:3306)/adapter', 'mem://test'. Supported drivers are [mem,␣
↪mysql]. Alternatively, this can be set with the following environment variable:␣
↪ADAPTER_REST_DSN
        --dsn-timeout string                    Total time in seconds to wait until the␣
↪datasource is available before giving up. Default:  seconds. Alternatively, this can␣
↪be set with the following environment variable: ADAPTER_REST_DSN_TIMEOUT
        --governance-vcs-url string             Governance VCS instance is running on.␣
↪Format: HostName:Port.
        -h, --help                                  help for start
        -u, --host-url string                       URL to run the adapter-rest␣
```

(continues on next page)

(continued from previous page)

```
↪instance on. Format: HostName:Port.
       --hydra-url string                        Base URL to the hydra service.
↪Alternatively, this can be set with the following environment variable: ADAPTER_REST_
↪HYDRA_URL
       --log-level string                        Sets the logging level. Possible values␣
↪are [DEBUG, INFO, WARNING, ERROR, CRITICAL] (default is INFO). Alternatively, this can␣
↪be set with the following environment variable: ADAPTER_REST_LOGLEVEL (default "INFO")
       --mode string                             Mode in which the edge-adapter service␣
↪will run. Possible values: ['issuer', 'rp'].
       --op-url string                           URL for the OIDC provider.Alternatively,
↪ this can be set with the following environment variable: ADAPTER_REST_OP_URL
       --presentation-definitions-file string    Path to presentation definitions file␣
↪with input_descriptors.
       --request-tokens stringArray              Tokens used for http request ␣
↪Alternatively, this can be set with the following environment variable: ADAPTER_REST_
↪REQUEST_TOKENS
       --static-path string                      Path to the folder where the static␣
↪files are to be hosted under /ui.Alternatively, this can be set with the following␣
↪environment variable: ADAPTER_REST_STATIC_FILES
       --tls-cacerts stringArray                 Comma-Separated list of ca certs path.␣
↪Alternatively, this can be set with the following environment variable: ADAPTER_REST_
↪TLS_CACERTS
       --tls-serve-cert string                   Path to the server certificate to use␣
↪when serving HTTPS. Alternatively, this can be set with the following environment␣
↪variable: ADAPTER_REST_TLS_SERVE_CERT
       --tls-serve-key string                    Path to the private key to use when␣
↪serving HTTPS. Alternatively, this can be set with the following environment variable:␣
↪ADAPTER_REST_TLS_SERVE_KEY
       --tls-systemcertpool string               Use system certificate pool. Possible␣
↪values [true] [false]. Defaults to false if not set. Alternatively, this can be set␣
↪with the following environment variable: ADAPTER_REST_TLS_SYSTEMCERTPOOL
       -r, --universal-resolver-url string          Universal Resolver instance is␣
↪running on. Format: HostName:Port.
```

7.2 RP Adapter

The Relying Party (RP) Adapter enables standard OpenID Connect flows on top of DIDComm.

7.2.1 Configuring the RP Adapter

The following is a snippet of a Docker Compose ™ file showing how Edge Adapter can be configured for use as an RP.

```
rp.adapter.rest.example.com:
  container_name: rp.adapter.rest.example.com
  image: ${RP_ADAPTER_REST_IMAGE}:latest
  environment:
    - ADAPTER_REST_HOST_URL=0.0.0.0:8070
    - ADAPTER_REST_TLS_CACERTS=/etc/tls/ec-cacert.pem
    - ADAPTER_REST_GOVERNANCE_VCS_URL=http://governance.vcs.example.com:8066
```

(continues on next page)

(continued from previous page)

```
    - ADAPTER_REST_TLS_SYSTEMCERTPOOL=true
    - ADAPTER_REST_TLS_SERVE_CERT=/etc/tls/ec-pubCert.pem
    - ADAPTER_REST_TLS_SERVE_KEY=/etc/tls/ec-key.pem
    - ADAPTER_REST_DSN=mysql://rpadapter:rpadapter-secret-pw@tcp(mysql:3306)/
    - ADAPTER_REST_OP_URL=http://PUT-SOMETHING-HERE.com
    - ADAPTER_REST_PRESENTATION_DEFINITIONS_FILE=/etc/testdata/presentationdefinitions.
↪json
    - ADAPTER_REST_DIDCOMM_INBOUND_HOST=0.0.0.0:8071
    - ADAPTER_REST_DIDCOMM_INBOUND_HOST_EXTERNAL=http://rp.adapter.rest.example.com:8071
    - ADAPTER_REST_TRUSTBLOC_DOMAIN=${BLOC_DOMAIN}
    - ADAPTER_REST_HYDRA_URL=https://hydra.trustbloc.local:4445
    - ADAPTER_REST_UNIVERSAL_RESOLVER_URL=http://did.rest.example.com:8072/1.0/
↪identifiers
    - ADAPTER_REST_DSN_TIMEOUT=45
  ports:
    - 8070:8070
  entrypoint: ""
  command:  /bin/sh -c "adapter-rest start"
  volumes:
    - ../keys/tls:/etc/tls
    - ../testdata:/etc/testdata
  networks:
    - bdd_net
  depends_on:
    - hydra
    - mysql
```

See this example in full here.

7.2.2 Deploying the RP Adapter

To learn about integrating your OIDC client to a TrustBloc RP Adapter, read our integration guide.

7.3 Issuer Adapter

This component is an intermediary to act on behalf of an Issuer to perform DIDComm related use cases.

7.3.1 Configuring the Issuer Adapter

The following is a snippet of a Docker Compose ™ file showing how Edge Adapter can be configured for use as an issuer.

```
issuer.adapter.rest.example.com:
  container_name: issuer.adapter.rest.example.com
  image: ${ISSUER_ADAPTER_REST_IMAGE}:latest
  environment:
    - ADAPTER_REST_HOST_URL=0.0.0.0:9070
    - ADAPTER_REST_GOVERNANCE_VCS_URL=http://governance.vcs.example.com:8066
```

(continues on next page)

(continued from previous page)

```
    - ADAPTER_REST_TLS_CACERTS=/etc/tls/ec-cacert.pem
    - ADAPTER_REST_TLS_SYSTEMCERTPOOL=true
    - ADAPTER_REST_TLS_SERVE_CERT=/etc/tls/ec-pubCert.pem
    - ADAPTER_REST_TLS_SERVE_KEY=/etc/tls/ec-key.pem
    - ADAPTER_REST_DIDCOMM_INBOUND_HOST=0.0.0.0:9071
    - ADAPTER_REST_DIDCOMM_INBOUND_HOST_EXTERNAL=http://issuer.adapter.rest.example.
↪com:9071
    - ADAPTER_REST_TRUSTBLOC_DOMAIN=${BLOC_DOMAIN}
    - ADAPTER_REST_UNIVERSAL_RESOLVER_URL=http://did.rest.example.com:8072/1.0/
↪identifiers
    - ADAPTER_REST_DSN=mysql://issueradapter:issueradapter-secret-pw@tcp(mysql:3306)/
    - ADAPTER_REST_DSN_TIMEOUT=45
  ports:
    - 9070:9070
    - 9071:9071
  entrypoint: ""
  command:  /bin/sh -c "adapter-rest start"
  volumes:
    - ../keys/tls:/etc/tls
  networks:
    - bdd_net
```

See this example in full here.

7.3.2 Deploying the Issuer Adapter

Integration guide

7.4 Adapter Components (CHAPI + DIDComm)

7.5 Flows

7.5.1 The Evidence and Driver's License (DL) Flow

These components allow users to access services with a VC such as a Driver's License. They are:

- Issuer Adapter
- RP Adapter

7.5.2 Combined DL, Evidence & Credit Score Flow

Here is an overfiew of the Bank Account usecase.

This scenario shows how a person can open a bank account using both local and remote credentials. A local credential is stored in a user's wallet while the remote credential is stored with a third-party.

In order to create the bank account, a Drivers License (local credential), Drivers Licence Evidence (remote credential) and Credit Score (remote credential) are required.

These are issued as VCs from a Drivers License Issuer and a Credit Score Issuer.

This uses the Adapter/DIDComm flow.

Watch the demos below.

Creating a New Bank Account

DL, Evidence and Credit Score

CHAPTER

EIGHT

DIRECT WALLET/CHAPI INTERACTIONS

8.1 VCS

- Integration guide

8.2 Wallet

- Integration guide

CHAPTER

NINE

BLINDED ROUTING

9.1 Introduction

The Issuers and RPs can use the TrustBloc Adapters to interact with each other and with the Wallet using DIDComm. The RP Adapter gets data from Issuer Adapter by calling its DIDComm URL and vice versa. In some cases, the Issuer wants to hide its identity from the RP and vice versa. The TrustBloc platforms provide Support for this through the Blinded Routing feature.

In Blinded Routing, the communication between TrustBloc Adapters (Issuer/RP) goes through the Router. The Wallet selects a Router and facilitates the creation of DIDComm connection between the Adapter (Issuer/RP) and the Router. The Adapter registers itself with the Router for that Adapter-Wallet combination. The DIDDocument of the Adapters would include Router's endpoint/keys, which would be shared with other parties to create DIDComm connection with each other.

9.2 Flow Diagram

9.3 Components and Configurations

- Wallet
- Router
- Issuer Adapter
- RP Adapter

9.4 DIDComm Messages

9.4.1 Wallet to Adapter : DID Doc Request

The wallet requests the adapter to provide the a new DID Document, which would be sent to Router.

Request:

```
{
   "@id":"089a0775-7e5f-4b96-912f-25532ec6853d",
   "@type":"https://trustbloc.dev/blinded-routing/1.0/diddoc-req"
}
```

Response:

```
{
   "@id":"a8fb8f8f-4137-4e4e-9168-9b34f8d93fee",
   "@type":"https://trustbloc.dev/blinded-routing/1.0/diddoc-resp",
   "~thread":{
          "thid":"089a0775-7e5f-4b96-912f-25532ec6853d"
   },
   "data":{
          "errorMsg":"<inCaseOfFailure>",
          "didDoc":{
                 <adapterDIDDoc>
          }
   }
}
```

9.4.2 Wallet to Router : Create Connection

The wallet requests the router to create a new connection with the adapter, by passing adapter's DID Document. The router creates a new connection and returns its DID Document to the wallet.

Request:

```
{
   "@id":"1de30277-6849-4797-a9c3-e5f6449c9a17",
   "@type":"https://trustbloc.dev/blinded-routing/1.0/create-conn-req",
   "data":{
          "didDoc":{
                 <adapterDIDDoc>
          }
   }
}
```

Response:

```
{
   "@id":"39aefb3f-562b-410d-b992-ab88e829aae9",
   "@type":"https://trustbloc.dev/blinded-routing/1.0/create-conn-resp",
   "data":{
          "errorMsg":"<inCaseOfFailure>",
          "didDoc":{
                 <routerDIDDoc>
          }
   }
}
```

9.4.3 Wallet to Adapter : Route Registration

The wallet sends the router's DID Document along with Parent threadID. The threadID from earlier DIDDoc req message from wallet to adapter will be used as parentThreadID. The Adapter creates the connection with the router and registers with it.

Request:

```
{
   "@id":"2d8ae926-111d-4970-a8b6-376991750d0f",
   "@type":"https://trustbloc.dev/blinded-routing/1.0/register-route-req",
   "~thread":{
          "pthid":"089a0775-7e5f-4b96-912f-25532ec6853d"
   },
   "data":{
          "didDoc":{
                 <routerDIDDoc>
          }
   }
}
```

Response:

```
{
   "@id":"c3e8dfc0-aa84-420d-87d4-2401e2c41b7b",
   "@type":"https://trustbloc.dev/blinded-routing/1.0/register-route-resp",
   "data":{
          "errorMsg":"<inCaseOfFailure>"
   }
}
```

CHAPTER

TEN

MESSAGE ROUTING AND STORAGE

10.1 Summary

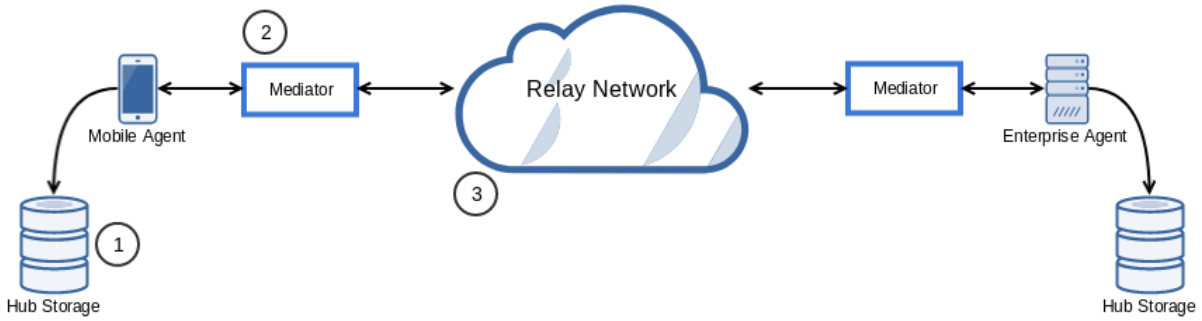
This proposal reuses, modifies, and adapts several proposals from the Hyperledger Aries/Indy, and the DIF communities to in order to enable:

- Advanced use cases for credentials exchange, such as when the Issuer requires the User's prior consent for issuance
- Guaranteed message delivery - even if the Agent is temporarily unavailable
- User-specified routing of messages from mediators to their Agents
- Unified message routing protocols and APIs
- Safe storage of encrypted identities separated from the encryption keys
- Simpler model for synchronization of wallets
- Simplex and duplex messaging paradigms

Specifically, this proposal builds on the foundation laid down by these proposals:

- DIF Identity Hub
- Aries RFC 0046: Mediators and Relays
- Aries RFC 0019: Encryption Envelope
- Aries RFC 0094: Cross Domain Messaging
- Aries RFC 0050: Wallets

Here is a generic, simplified view:



10.1.1 1 - Hub Storage

Corresponds to the DIF Identity Hub's Collections interface and Replication Protocol.

The Permissions API is disregarded because objects stored here are accessible solely by the Agent.

This storage service is plugged into the Agent's wallet as an implementation of the Storage Interface as shown here.

10.1.2 2 - Mediator

The mediator filters messages (authorization) and routes them to the Agent of the Identity Owner's choosing based on user-specified rules stored in a user-specified Hub Storage location.

Mediators buffer undelivered messages sent to the Agents until confirmation of delivery.

Mediators can be extended in many different ways to support many interesting use cases. For example, an Issuer's Agent can request collaboration from other mediators (with prior consent from their respective Agents) in order to fulfill a request.

10.1.3 3 - Relay Network

Messages between sovereign domains are transported via a network of relays.

Mediators may communicate through one or several relay networks as per their requirements.

For example, a mediator might leverage the public TOR relay network to protect the Agent's privacy, or it might simply use the internet.

10.2 Trusted Contexts

The basic exchange implied in the diagram above solves many real-world use cases, but needs to be extended to support scenarios where the Issuer remains the Holder of the User Data - while keeping the User in the locus of control.

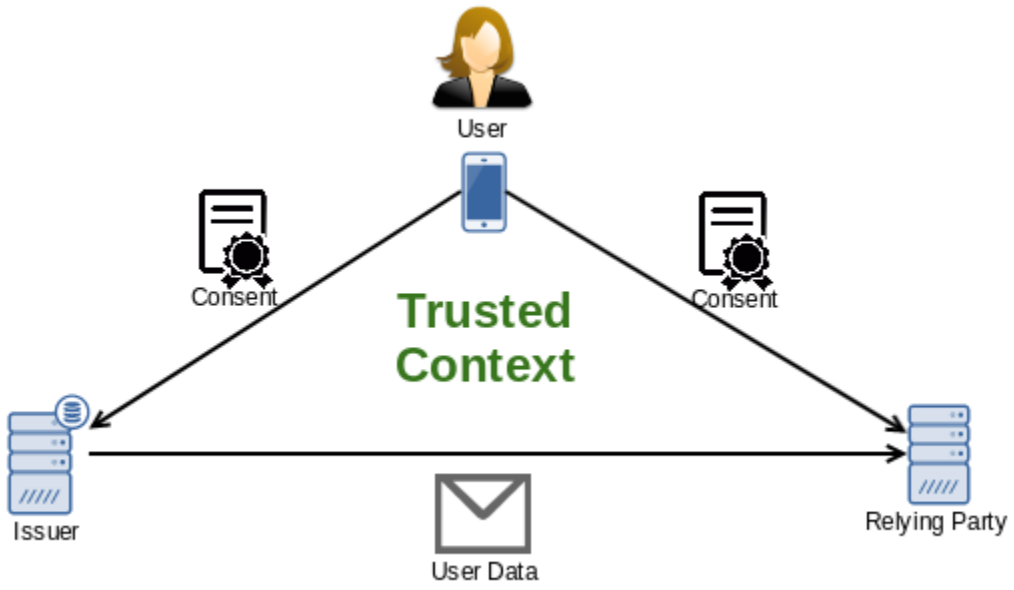
The User may introduce themselves directly to the other parties by sharing Peer DIDs, or they may discover these other peers through an app that displays the public, well-known, blockchain-anchored DIDs of recognized institutions.

:::{figure} _static/ledger-anchored-trust.png :::

Mobile agent displaying parties with well-known DIDs anchored to a blockchain, all of which possess Verifiable Credentials from a trusted Issuer (trusted issuer not shown).

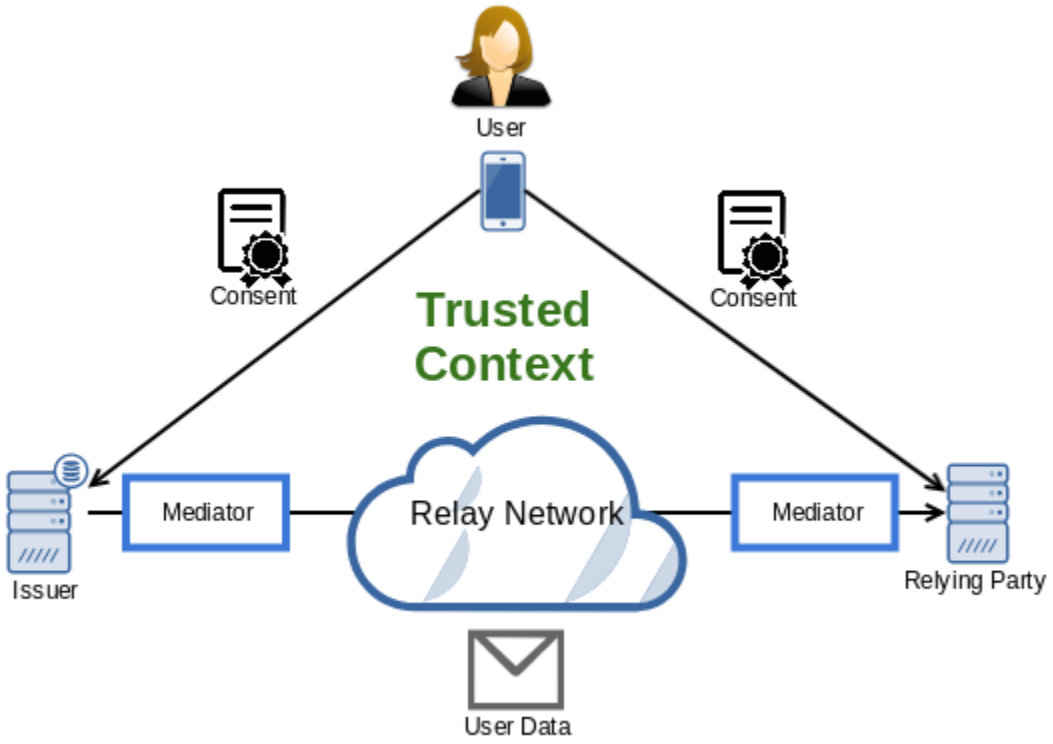
Once introduced to these parties (individually), the User proceeds to create a Trusted Context between all parties by asking them each for a new DID identifier for use in this context, along with any Verifiable Credentials required for membership.

Setup of the Trusted Context ends with the User providing the other parties a consent receipt.



10.2.1 Relays based on Trusted Context

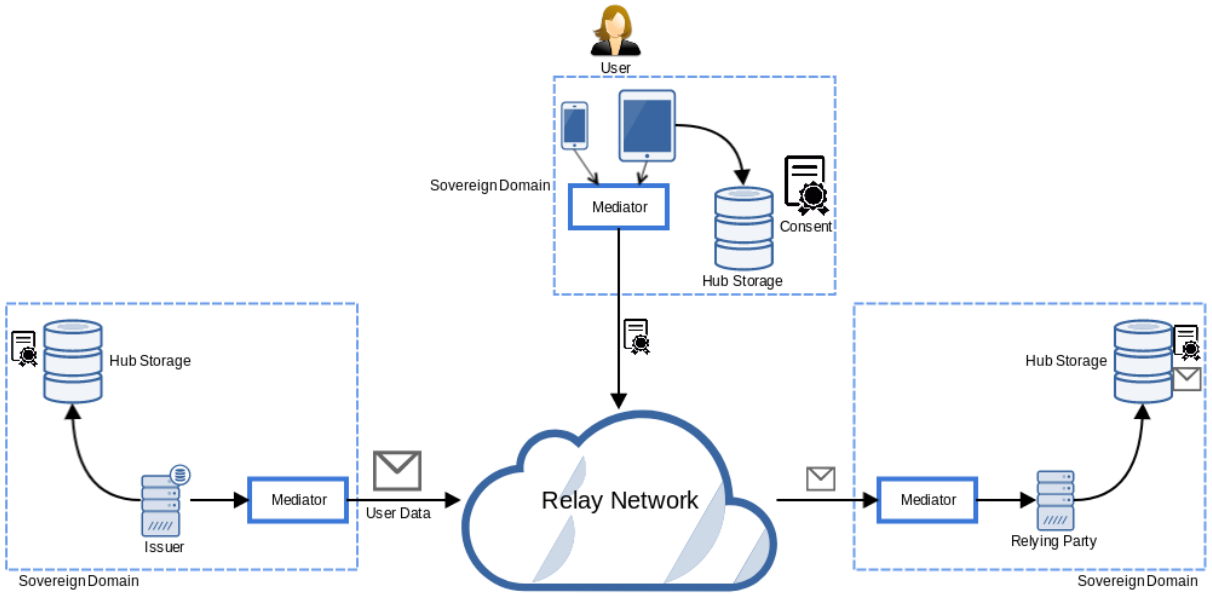
The previous diagram shows the logical construction of a trusted context. For greater clarity, there are Mediators and Relay Networks between the participants that route based on the trusted context. In particular, the User Data traverses the mediators and relay network:



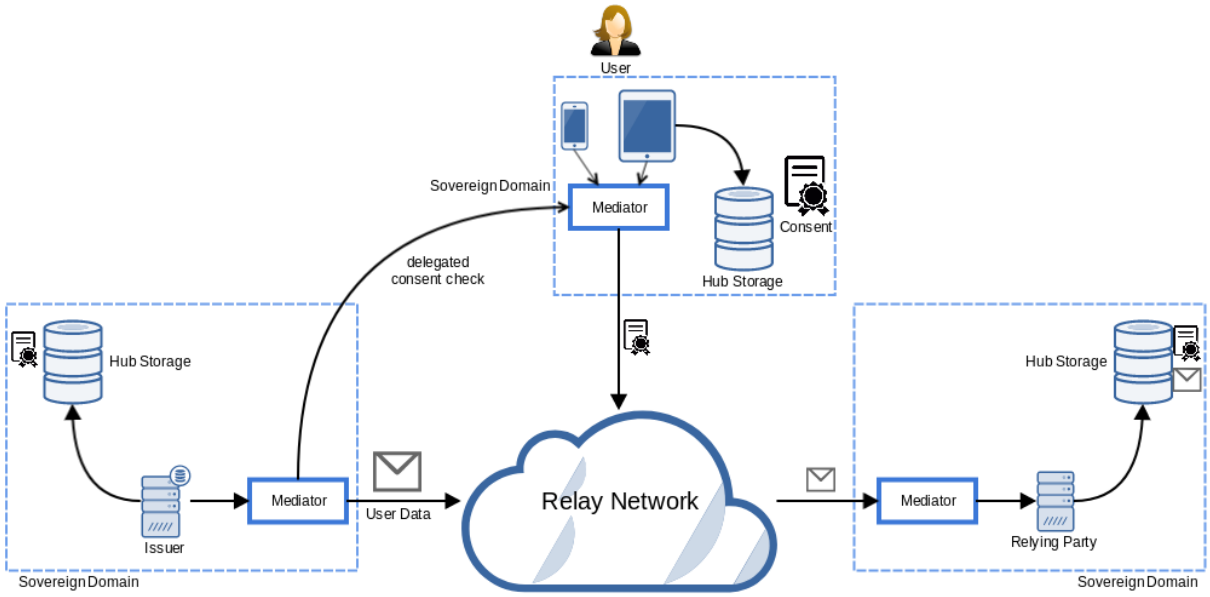
10.3 Putting it all together

Trust contexts are realized by:

- Using DIDs as identifiers within the context
- Using Verifiable Credentials for user data representation * Including the user's consent receipt, which will follow well known standard schemas
- A credential schema negotiated among the parties
- A relay network negotiated among the parties
- Mediators ensuring message delivery to the Agents



Another scenario has the Issuer mediator delegating to the User mediator, in a manner similar to UMA:


User
Sovereign Domain
Mediator
Consent
Hub Storage
delegated
consent check
Hub Storage
Issuer
Mediator
User Data
Relay Network
Mediator
Hub Storage
Relying Party
Sovereign Domain
Sovereign Domain

CHAPTER

ELEVEN

PRIVACY-ENHANCED OAUTH 2.0

Status: DRAFT

Table of Contents

11.1 Contributors

- George Aristy (SecureKey Techologies)

11.2 Introduction

11.2.1 Purpose of this document

This document describes a reference implementation of OAuth 2.0 with unregistered clients communicating and authenticating securely over the backchannel with decentralized identifiers and verifiable credentials.

11.2.2 Motivation

There is a desire to leverage existing OAuth 2.0 infrastructure to build a privacy-enhanced data sharing solution.

Finalized in 2012, OAuth 2.0 ([1]), is an established authorization framework well suited to give a piece of software access to protected resources with the owner's consent. It was not, however, designed with the principles of Privacy by Design ([2]) in mind.

First published in 2009, the principles of privacy by design became widely known after the GDPR adopted them ([3]) and began enforcement in 2018. We seek to address two key principles of privacy by design that OAuth 2.0 does not:

Protect the user's privacy by keeping the solution User-Centric

- Use the authorization grant mechanism of OAuth 2.0 to keep the user in the locus of control.
- Use decentralized identifiers (DIDs) ([4]) so the user (and the other actors) can avoid undesired correlation.

Protect the user's privacy with end-to-end security

- Use end-to-end authenticated encryption of the messages and data while in transit.

11.2.3 Objectives

1. Allow a client to request the user for access to resources hosted on a resource server.
2. Conceal the client's location from the resource server's location (and vice versa).
3. Allow the user to grant the client access to the resources.
4. Allow the user to revoke access to the client.
5. Allow the user to indicate the location of these resources to the client.
6. Minimize exposure of the client's identity from the resource server (and vice versa).
7. Ensure confidentiality in communications through the frontchannel.
8. Ensure confidentiality in communications through the backchannel.

11.2.4 Constraints

1. Use OAuth 2.0 (authorization code grant type).
2. No modification of OAuth components in client nor resource server domains.
3. Use decentralized identifiers.

[1] D. Hardt (Microsoft), "IETF RFC6749 - The OAuth 2.0 Authorization Framework", October 2012

[2] Ann Cavoukian (Information & Privacy Commissioner of Ontario, Canada), "Privacy by Design - The 7 Foundational Principles", Retrieved December 10 2019

[3] European Data Protection Supervisor, ["Preliminary Opinion on privacy by design" <https://edps.europa.eu/sites/edp/files/publication/18-05-31_preliminary_opinion_on_privacy_by_design_en_0.pdf>"]["preliminary opinion on privacy by design" https://edps.europa.eu/sites/edp/files/publication/18-05-31_preliminary_opinion_on_privacy_by_design_en_0.pdf"], May 31 2018

[4] Drummond Reed (Evernym), Manu Sporny (Digital Bazaar), Markus Sabadello (Danube Tech), Dave Longley (Digital Bazaar), Christopher Allen (Blockchain Commons), Ryan Grant, "Decentralized Identifiers (DIDs) v1.0", W3C Working Draft 10 December 2019

11.3 System Overview

:::{figure} _static/oauth_didcomm_highlevel.png :figclass: align-right

System Overview

- Green arrows indicate frontchannel communication.
- Blue arrows indicate backchannel communication over a secure transport.
- Black arrows indicate backchannel communication in their normal (HTTP) form. :::

The figure above shows the main components of the system. It depicts a normal OAuth 2 setup with the client, resource owner, authorization server and resource server roles but adds two new components:

Broker:

OAuth 2.0 requires clients to be registered at the authorization server[5] before sending the authorization request. Our objectives preclude this, therefore the user requires a "broker" component that will relay the authorization request appropriately and to the right location.

Adapter:

The adapters pack and unpack normal authorization requests, token exchange/refresh requests, and requests to the resources to and from HTTP transport and secure, end-to-end encrypted channels between the user, client, and server domains. They also isolate the OAuth 2 components in the client and server domains from the complexity of the network.

11.4 References

11.4.1 Normative References

11.4.2 Informative References

[5] As per section 2.4 of[Page 250, 1] , unregistered clients are out of scope but not precluded by the OAuth2 specification. However, it is difficult to reconcile this in a meaningful way with the fact that authorization servers assign the `client_id` to clients (section 2.2) to ensure their uniqueness so as to avoid impersonation attacks (see section 4.13 of **[^cite_o2-bcp]**).

CHAPTER

TWELVE

HOW TO CONTRIBUTE!

Thank you for showing interest to contribute to TrustBloc. Visit Contribution Guideline.

12.1 Setup

12.2 Fork on Github

Before you do anything else, login/signup on GitHub and fork TrustBlock Projects from the GitHub project.

12.3 Clone your fork locally

If you have git-scm installed, you now clone your git repo using the following command-line argument where <my-github-name> is your account name on GitHub:

For example you fork fabric-mod sub project:

```
git clone git@github.com:<my-github-name>/fabric-mod.git
```

12.4 Installing TrustBloc Projects

Follow our installation instructions defined on each sub projects.Please record any difficulties you have and share them with the TrustBloc community by creating an issue.

12.5 Issues

TODO

12.6 Tips

TODO: how to define issues

12.7 Setting up topic branches and generating pull requests

To create a topic branch, its easiest to use the convenient -b argument to git checkout:

```
git checkout -b fix-update-branch
Switched to a new branch 'fix-update-branch'
```

You should use a verbose enough name for your branch so it is clear what it is about. Now you can commit your changes and regularly merge in the upstream develop as described below. When you are ready to generate a pull request, either for preliminary review, or for consideration of merging into the project you must first push your local topic branch back up to GitHub:

```
git push origin fix-update-branch
```

Now when you go to your fork on GitHub, you will see this branch listed under the "Source" tab where it says "Switch Branches". Go ahead and select your topic branch from this list, and then click the "Pull request" button.

Here you can add a comment about your branch. If this in response to a submitted issue, it is good to put a link to that issue in this initial comment. The repo managers will be notified of your pull request and it will be reviewed (see below for best practices). Note that you can continue to add commits to your topic branch (and push them up to GitHub) either if you see something that needs changing, or in response to a reviewer's comments. If a reviewer asks for changes, you do not need to close the pull and reissue it after making changes. Just make the changes locally, push them to GitHub, then add a comment to the discussion section of the pull request.

12.8 How to get your pull request accepted

- **If you add code/views you need to add tests!** TODO
- **Don't mix code changes with whitespace cleanup** TODO
- **Keep your pull requests limited to a single issue** TODO

12.9 How pull requests are checked, tested, and done

TODO

12.10 Contributing Organizations

- SecureKey Technologies.

CHAPTER

THIRTEEN

HAVE QUESTIONS?

We try to maintain a comprehensive set of documentation for various audiences. However, we realize that often there are questions that remain unanswered. For any technical questions relating to a TrustBloc project not answered here, please use

Gitter (an alternative to Slack) on the #trustbloc-questions channel.

:::{note} Please, when asking about problems you are facing tell us about the environment in which you are experiencing those problems including the OS, which version of Docker you are using, etc. :::

Note: If you have questions not addressed by this documentation, please visit the *Have Questions?* page for some tips on where to find additional help.