TrustBloc Documentation

Release 0.0.1

SecureKey

Jan 15, 2021

Contents

1 Introduction 3

2 Architecture 5

3 Projects 7

4 Verifiable Credential Service (VCS) 9

5 Adapters 21

6 Direct Wallet/CHAPI Interactions 27

7 Blinded Routing 29

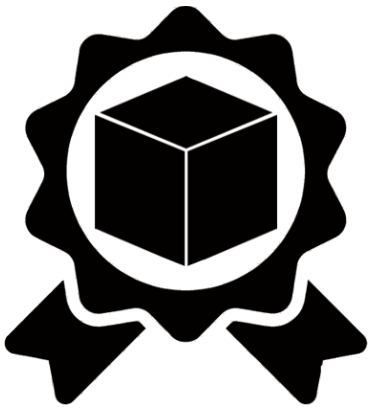
8 Message Routing and Storage 33

9 Privacy-Enhanced OAuth 2.0 39

10 How to Contribute! 43

11 Have Questions? 45

Bibliography 47



CHAPTER 1

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 2

Architecture

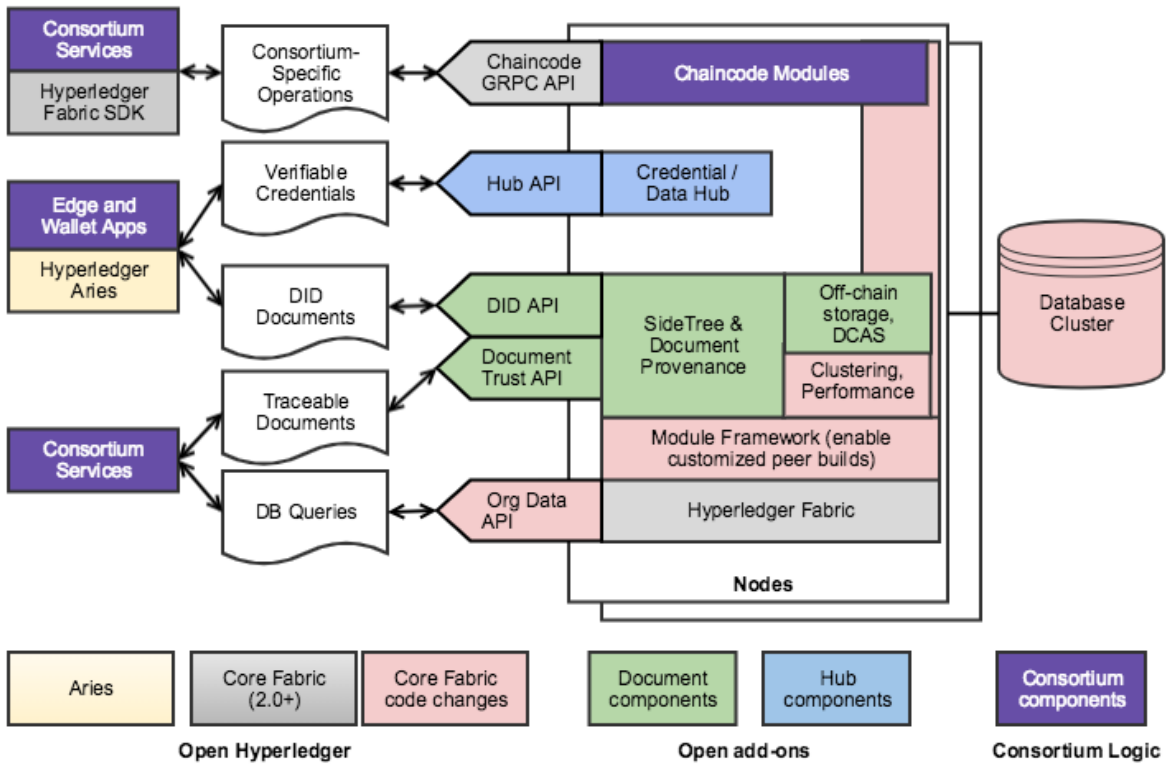


Diagram represents an overview of the components related to the TrustBloc initiative.

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 3

Projects

3.1 Edge

- Edge-Adapter
- Edge-Agent
- Edge-Core
- Edge-Sandbox
- Edge-Service

3.2 Agent SDK

- Agent SDK

3.3 Bloc Hub

- Bloc-Hub

3.4 Sidetree

- Sidetree-Node

3.5 Hyperledger Fabric Extensions

- Fabric-Mod

- Fabric-Peer-Ext
- Fabric-SDK-Go-Ext

3.6 Upstream Project

- Fabric-SDK-Go
- Hyperledger Fabric
- Hyperledger Aries
- DIF

CHAPTER 4

Verifiable Credential Service (VCS)

4.1 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*)

4.2 Edge-Service

TrustBloc's Edge-Service contains servers that handle the issuance and verification of verifiable credentials.

4.2.1 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:
         --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
```

(continues on next page)

(continued from previous page)

```
   -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
   -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:
```

(continues on next page)

(continued from previous page)

```
    - edv.example.com
  networks:
    - 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

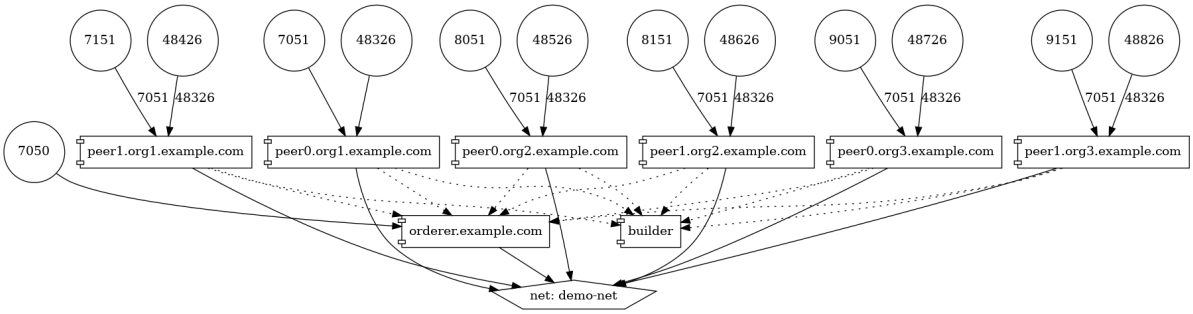
4.2.2 Deploying the service

In order to deploy Edge-Service, the following components are required.

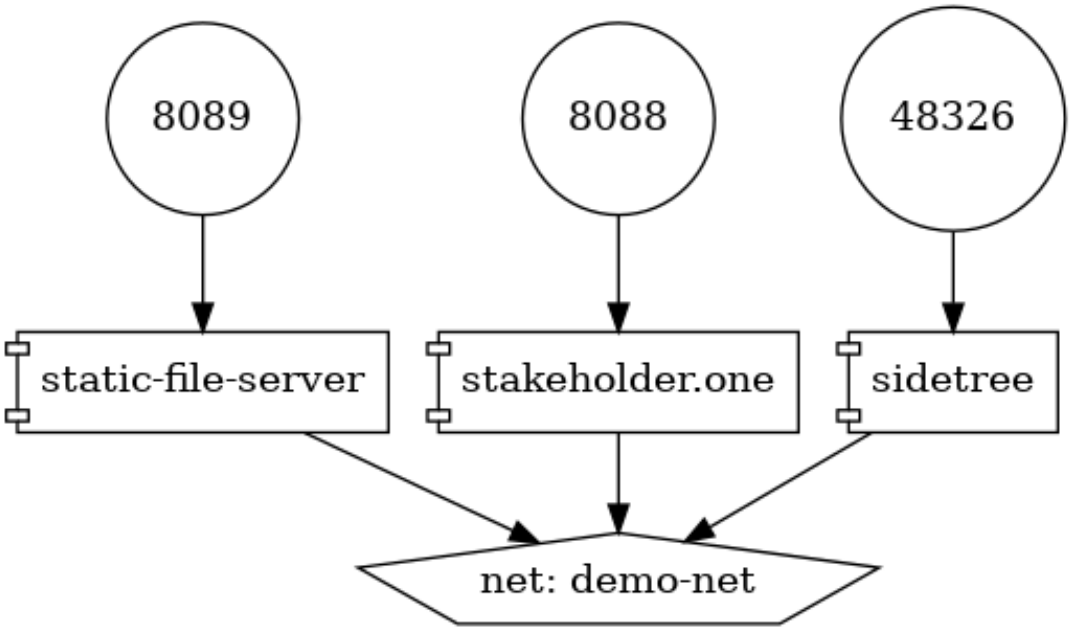
Note: An example of how these components interact together is shown here.

Sidetree

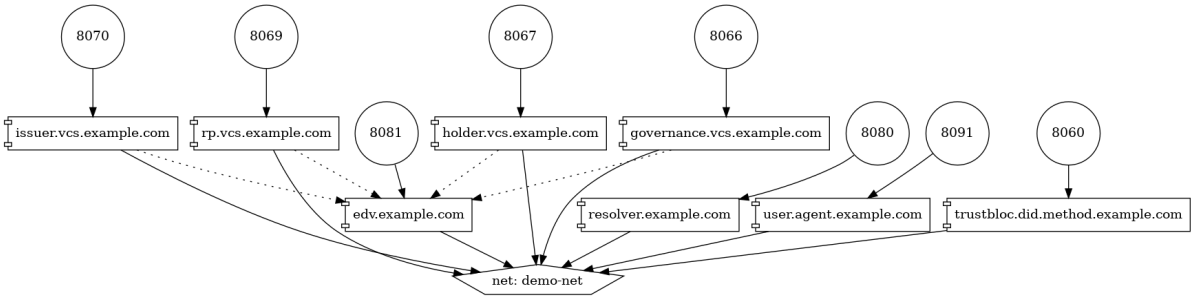
Sidetree Fabric



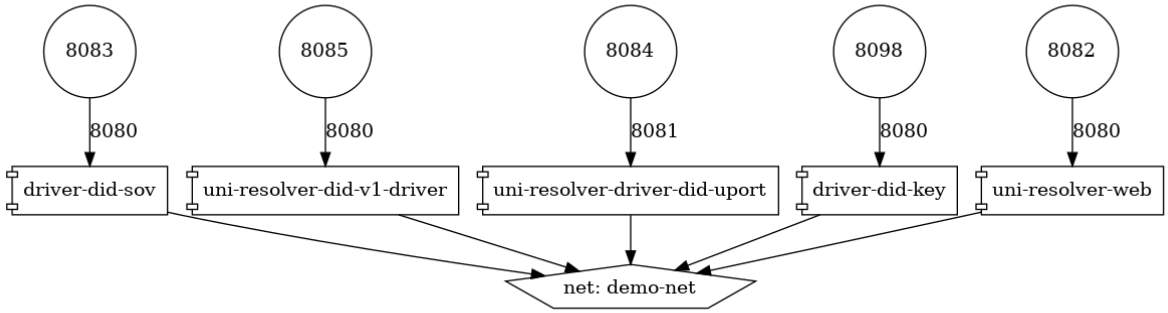
Sidetree Mock



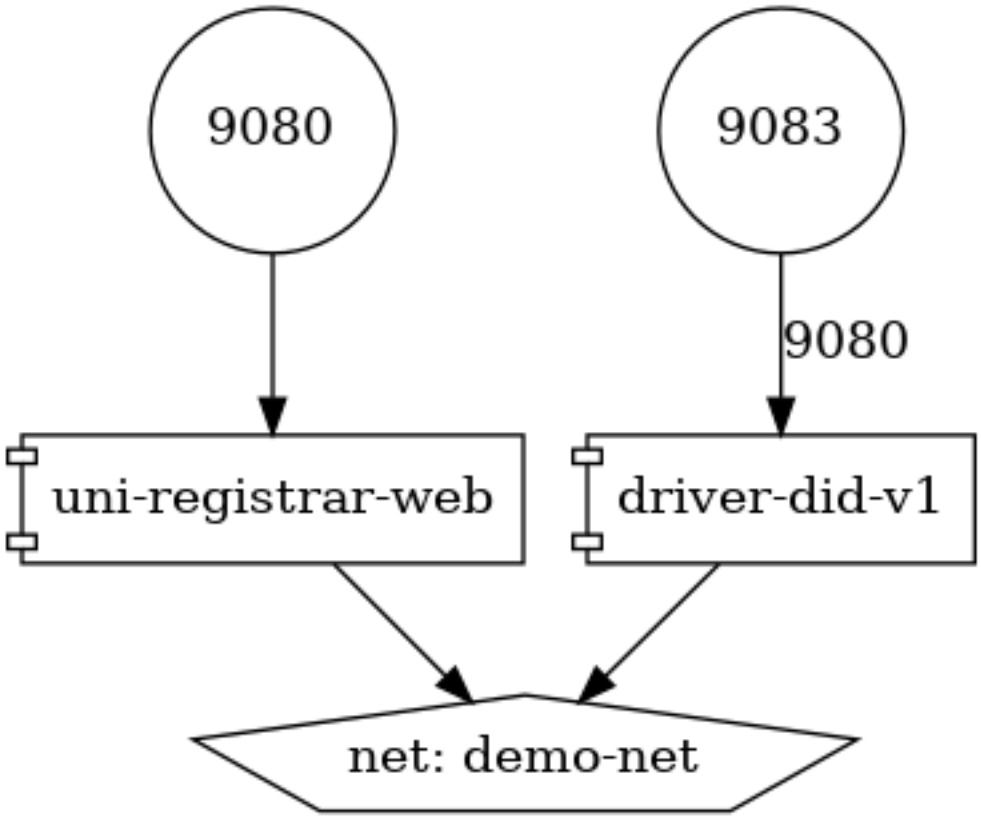
Edge Components



DID Resolvers



DID Registrars



4.2.3 VCS Components (CHAPI + VC Services)

4.2.4 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"
   ],
   "issuanceDate":"2020-03-25T19:38:54.45546Z",
   "expirationDate":"2020-06-25T19:38:54.45546Z",
   "claims":{
```

(continues on next page)

(continued from previous page)


```
      "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.

4.2.5 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":{
         "id":"did:example:oakek12as93mas91220dapop092",
         "name":"University"
      },
      "proof":{
         "created":"2020-04-09T15:35:35Z",
         "jws":"eyJhbGciOiJFZERTQSIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..
↪kN1srfFqoiejHJwxM8Y0Y9yIonAvFeF2Aoiv6_LTkPqcNXc2rXwT94-uO_
↪PQJbxWJgTD78MvpfCJWsUSRvgCBw",
```


(continues on next page)

(continued from previous page)

```
         "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.

4.3 Connecting to the TestNet

4.3.1 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:

- trustbloc@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- v1@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- elem@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- sov@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- web@https://resolver.sandbox.trustbloc.dev/1.0/identifiers
- key@https://resolver.sandbox.trustbloc.dev/1.0/identifiers

4.4 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.

4.4.1 Register A Wallet

Be sure to register your wallet as in the video below:

4.4.2 Issue a Credit Score Report

4.4.3 Issue a Driver's License

4.5 References

CHAPTER 5

Adapters

5.1 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
```

(continues on next page)

(continued from previous page)

```
       --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␣
↪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.
```

5.2 RP Adapter

The Relying Party (RP) Adapter enables standard OpenID Connect flows on top of DIDComm.

5.2.1 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: ${RP_ADAPTER_REST_IMAGE}:latest
  environment:
    - ADAPTER_REST_HOST_URL=0.0.0.0:8070
    - ADAPTER_REST_TLS_CACERTS=/etc/tls/ec-cacert.pem
    - ADAPTER_REST_GOVERNANCE_VCS_URL=http://governance.vcs.example.com:8066
    - ADAPTER_REST_TLS_SYSTEMCERTPOOL=true
    - ADAPTER_REST_TLS_SERVE_CERT=/etc/tls/ec-pubCert.pem
    - ADAPTER_REST_TLS_SERVE_KEY=/etc/tls/ec-key.pem
    - ADAPTER_REST_DSN=mysql://rpadapter:rpadapter-secret-pw@tcp(mysql:3306)/
    - ADAPTER_REST_OP_URL=http://PUT-SOMETHING-HERE.com
    - ADAPTER_REST_PRESENTATION_DEFINITIONS_FILE=/etc/testdata/
↪presentationdefinitions.json
    - ADAPTER_REST_DIDCOMM_INBOUND_HOST=0.0.0.0:8071
    - ADAPTER_REST_DIDCOMM_INBOUND_HOST_EXTERNAL=http://rp.adapter.rest.example.
↪com:8071
    - ADAPTER_REST_TRUSTBLOC_DOMAIN=${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.

5.2.2 Deploying the RP Adapter

To learn about integrating your OIDC client to a TrustBloc RP Adapter, read our integration guide.

5.3 Issuer Adapter

This component is an intermediary to act on behalf of an Issuer to perform DIDComm related use cases.

5.3.1 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
```

(continues on next page)

(continued from previous page)

```
  image: ${ISSUER_ADAPTER_REST_IMAGE}:latest
  environment:
    - ADAPTER_REST_HOST_URL=0.0.0.0:9070
    - ADAPTER_REST_GOVERNANCE_VCS_URL=http://governance.vcs.example.com:8066
    - ADAPTER_REST_TLS_CACERTS=/etc/tls/ec-cacert.pem
    - ADAPTER_REST_TLS_SYSTEMCERTPOOL=true
    - ADAPTER_REST_TLS_SERVE_CERT=/etc/tls/ec-pubCert.pem
    - ADAPTER_REST_TLS_SERVE_KEY=/etc/tls/ec-key.pem
    - ADAPTER_REST_DIDCOMM_INBOUND_HOST=0.0.0.0:9071
    - ADAPTER_REST_DIDCOMM_INBOUND_HOST_EXTERNAL=http://issuer.adapter.rest.example.
↪com:9071
    - ADAPTER_REST_TRUSTBLOC_DOMAIN=${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.

5.3.2 Deploying the Issuer Adapter

Integration guide

5.4 Adapter Components (CHAPI + DIDComm)

5.5 Flows

5.5.1 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

5.5.2 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 6

Direct Wallet/CHAPI Interactions

6.1 VCS

- Integration guide

6.2 Wallet

- Integration guide

CHAPTER 7

Blinded Routing

7.1 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.

7.2 Flow Diagram

7.3 Components and Configurations

- Wallet
- Router
- Issuer Adapter
- RP Adapter

7.4 DIDComm Messages

7.4.1 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>
          }
   }
}
```

7.4.2 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>
```

(continues on next page)

(continued from previous page)

```
            }
    }
}
```

7.4.3 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 8

Message Routing and Storage

8.1 Summary

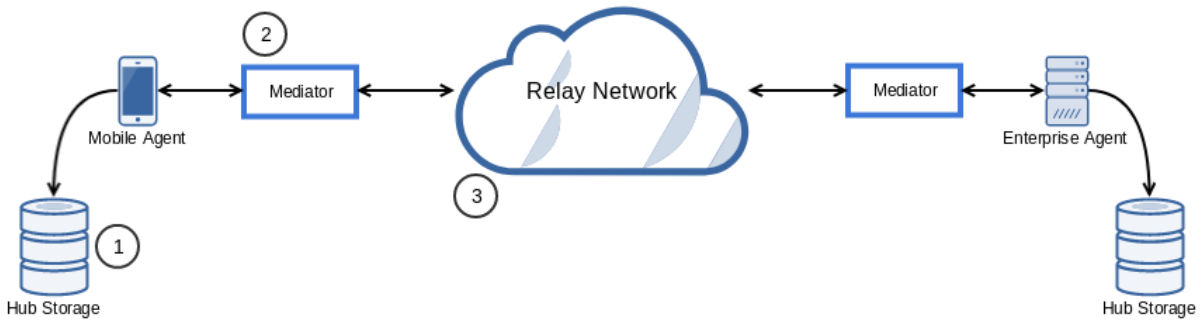
This proposal reuses, modifies, and adapts several proposals from the Hyperledger Aries/Indy, and the DIF communities to in order to enable:

- 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:



8.1.1 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.

8.1.2 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.

8.1.3 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.

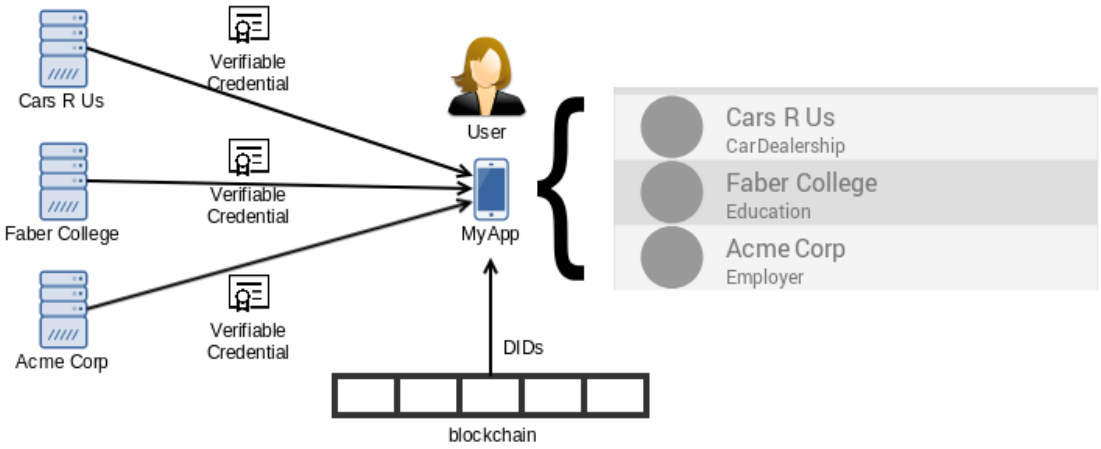
8.2 Trusted Contexts

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.

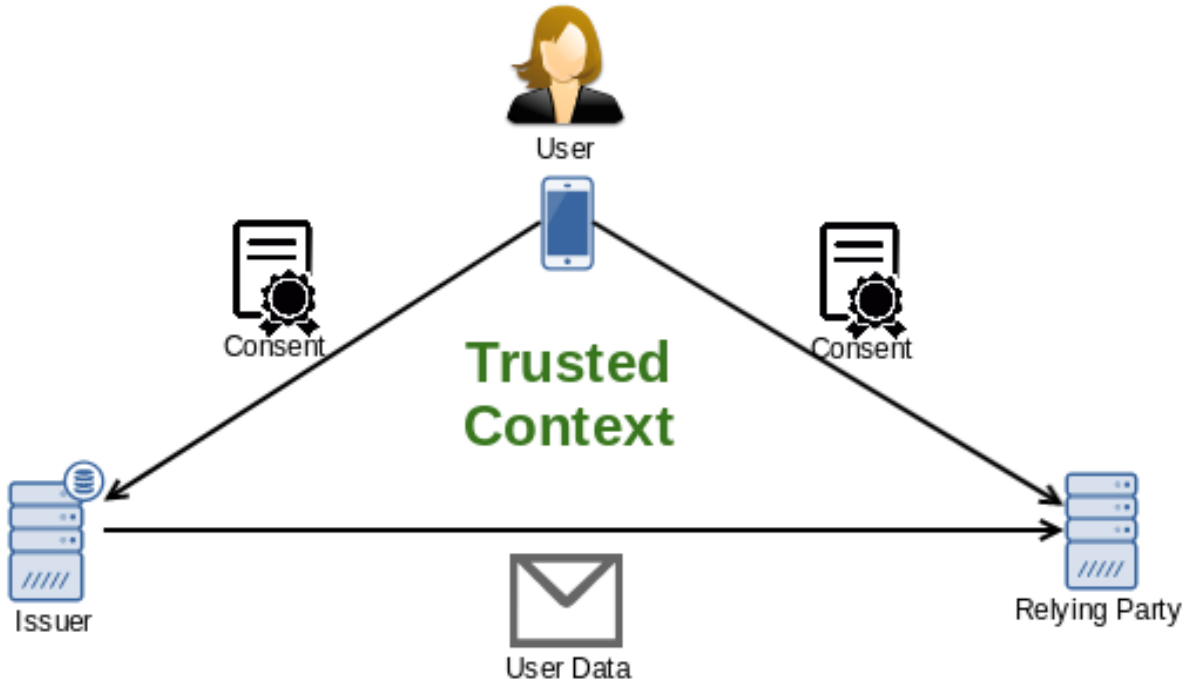
The User may introduce themselves directly to the other parties by sharing Peer DIDs, or they may discover these other peers through an app that displays the public, well-known, blockchain-anchored DIDs of recognized institutions.

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.

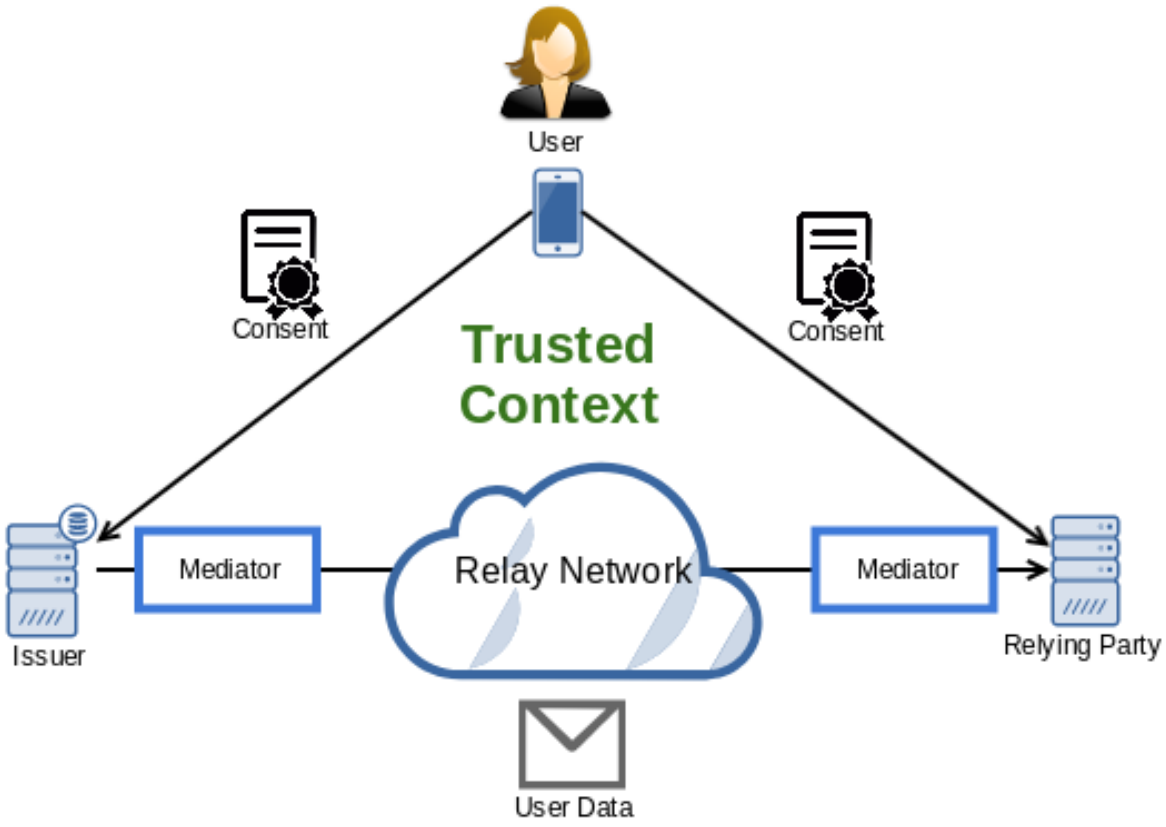


Setup of the Trusted Context ends with the User providing the other parties a consent receipt.



8.2.1 Relays based on Trusted Context

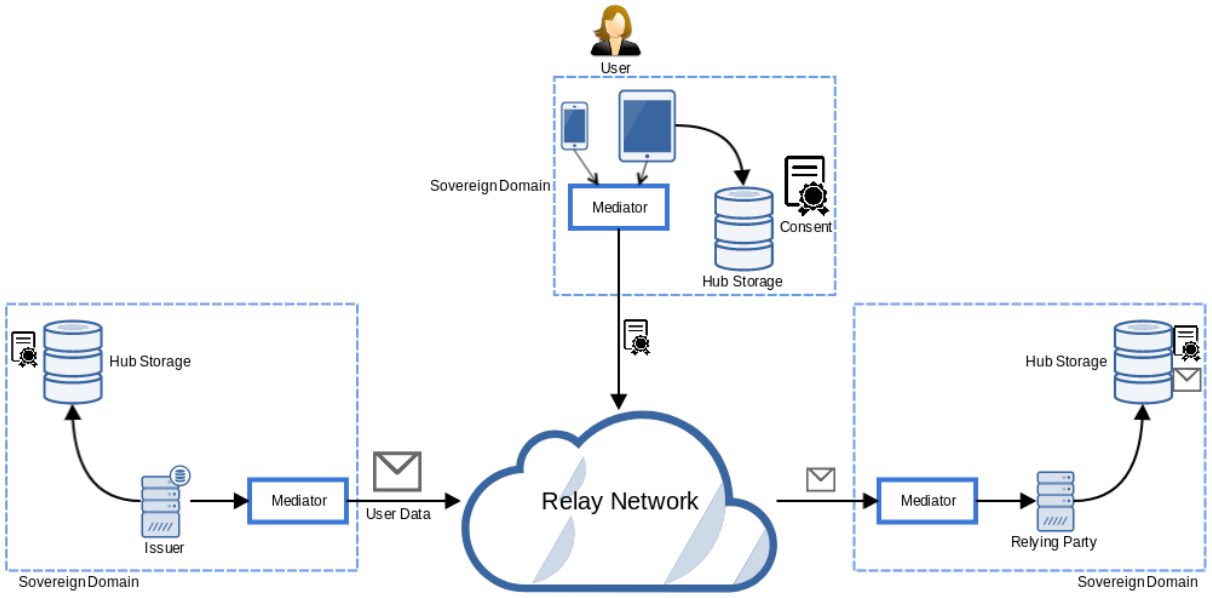
The previous diagram shows the logical construction of a trusted context. For greater clarity, there are Mediators and Relay Networks between the participants that route based on the trusted context. In particular, the User Data traverses the mediators and relay network:



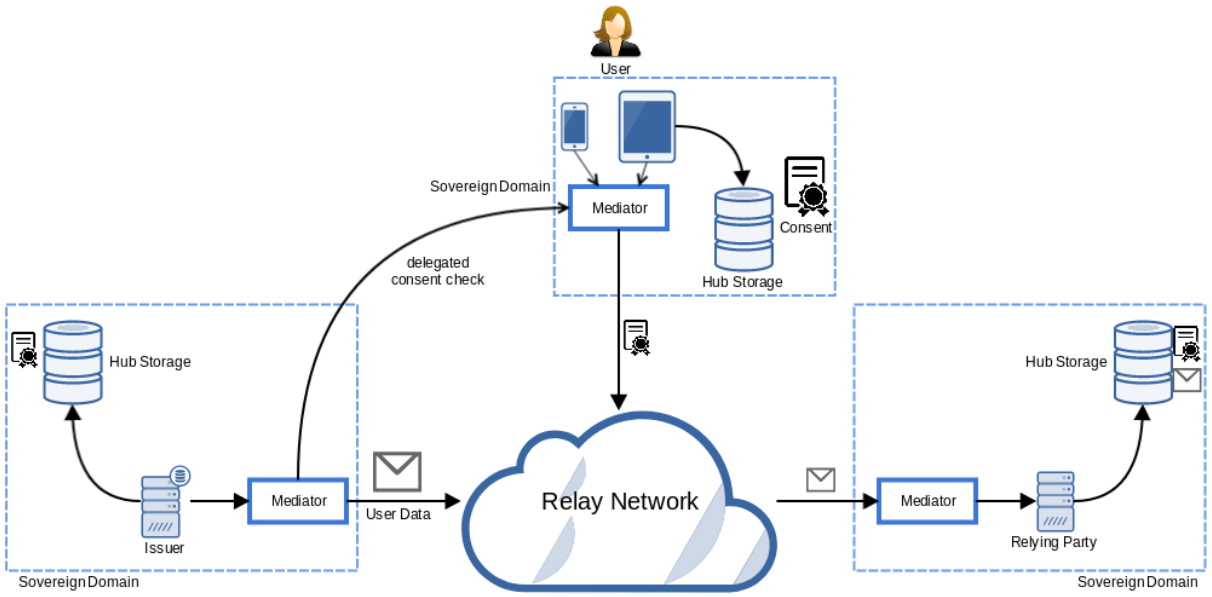
8.3 Putting it all together

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:



CHAPTER 9

Privacy-Enhanced OAuth 2.0

Status: DRAFT

Table of Contents

- *Contributors*
- *Introduction*
  - *Purpose of this document*
  - *Motivation*
  - *Objectives*
  - *Constraints*
- *System Overview*
- *References*
  - *Normative References*
  - *Informative References*

9.1 Contributors

- George Aristy (SecureKey Techologies)

9.2 Introduction

9.2.1 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.

9.2.2 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 ([RFC6749]), 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 ([PRIV-DESIGN]) in mind.

First published in 2009, the principles of privacy by design became widely known after the GDPR adopted them ([GDPR-PRIV]) 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) ([DID-CORE]) 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.

9.2.3 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.

9.2.4 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.

9.3 System Overview

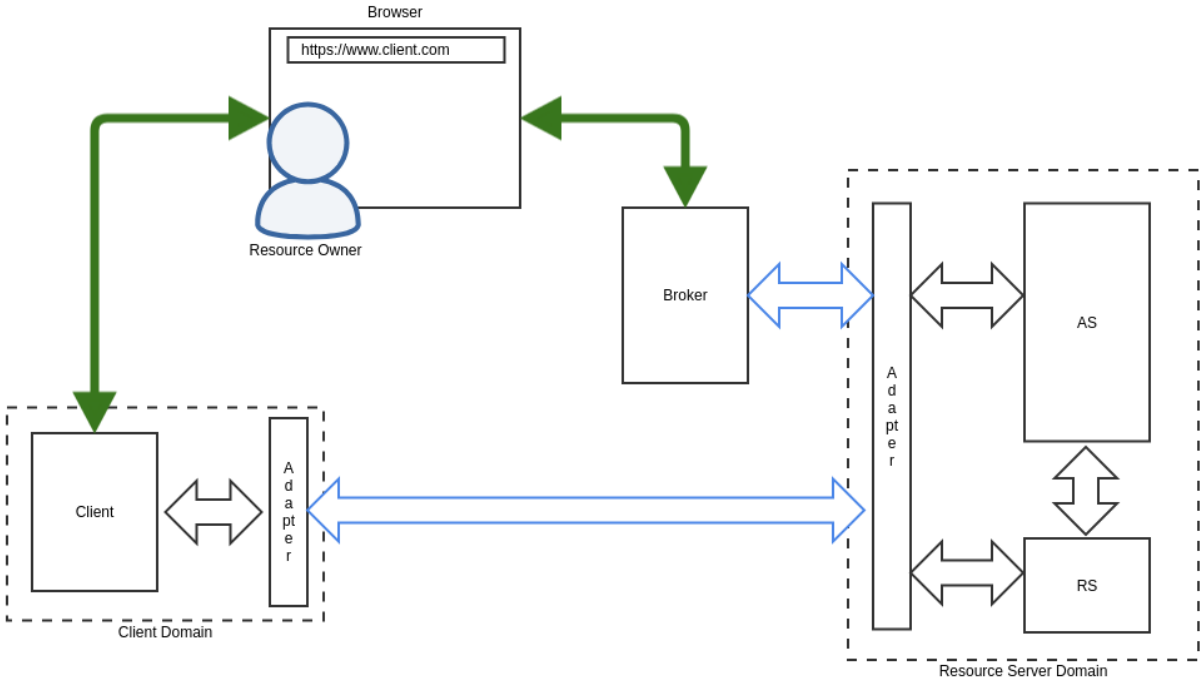


Fig. 1: System Overview

- 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[1] 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.

[1] As per section 2.4 of [RFC6749] , 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 [O2-BCP]).

9.4 References

9.4.1 Normative References

9.4.2 Informative References

CHAPTER 10

How to Contribute!

Thank you for showing interest to contribute to TrustBloc. Visit Contribution Guideline.

10.1 Setup

10.2 Fork on Github

Before you do anything else, login/signup on GitHub and fork TrustBlock Projects from the GitHub project.

10.3 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
```

10.4 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.

10.5 Issues

TODO

10.6 Tips

TODO: how to define issues

10.7 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.

10.8 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

10.9 How pull requests are checked, tested, and done

TODO

10.10 Contributing Organizations

- SecureKey Technologies.

CHAPTER 11

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.

Bibliography

[VC-DEF] Manu Sporny; Grant Noble; Dave Longley; David Chadwick, "Verifiable Credentials Data Model 1.0", November 2019

[VC-TERM] Manu Sporny; Grant Noble; Dave Longley; David Chadwick, "Verifiable Credentials Data Model 1.0", November 2019

[RFC6749] D. Hardt (Microsoft), "IETF RFC6749 - The OAuth 2.0 Authorization Framework", October 2012

[O2-BCP] T. Lodderstedt (yes.com), J. Bradley (Yubico), A. Labunets (Facebook), D. Fett (yes.com), "OAuth 2.0 Security Best Current Practice", version 13.

[DID-CORE] 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

[PRIV-DESIGN] Ann Cavoukian (Information & Privacy Commissioner of Ontario, Canada), "Privacy by Design - The 7 Foundational Principles", Retrieved December 10 2019

[GDPR-PRIV] 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>"‘_**, May 31 2018