FIWARE
Global
Summit

From Data to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

Canis Major

Notarization of digital twin data exchanges using blockchain

Stefan Wiedemann, Technical Lead & Architect, FIWARE Foundation

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds

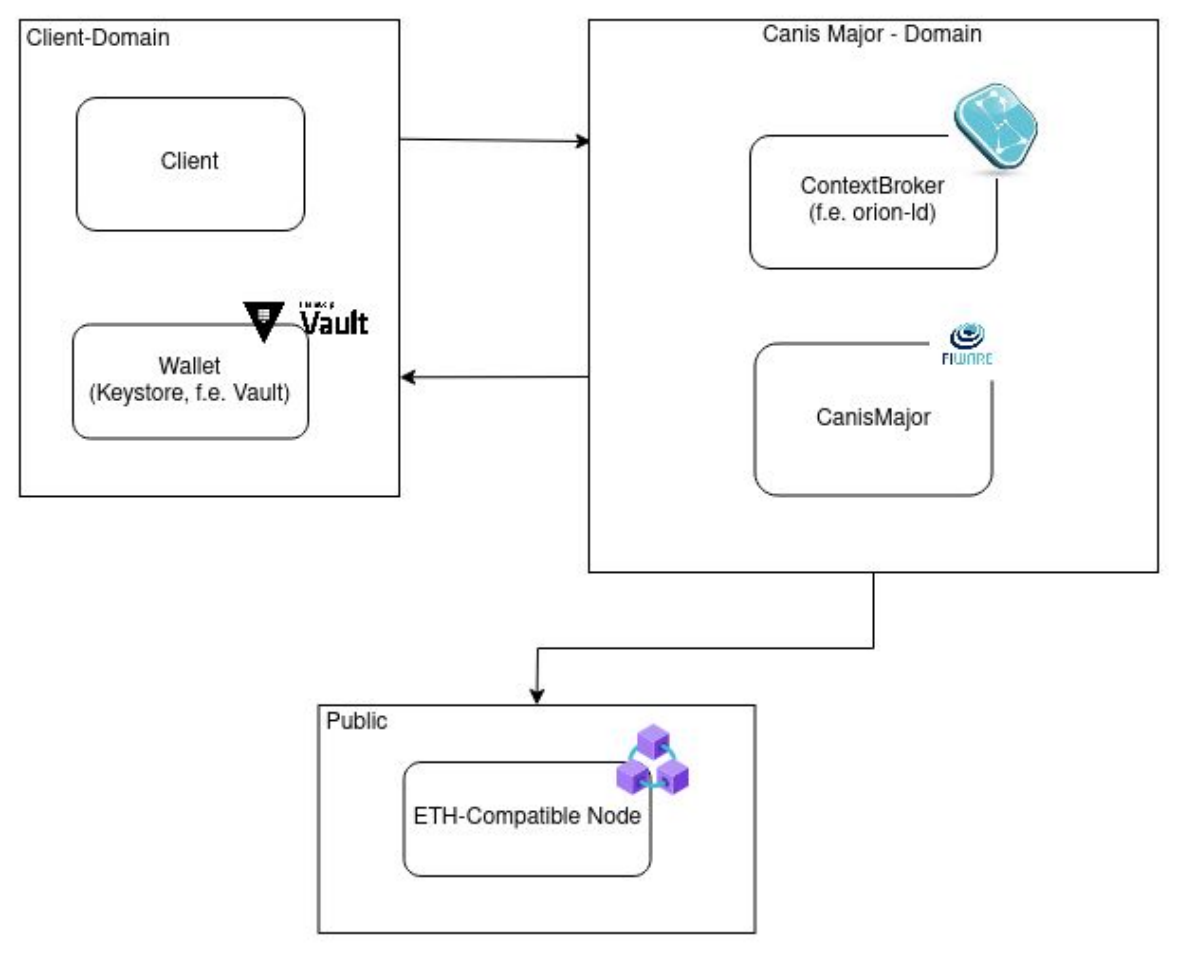
Canis Major

- What it is:
  - an adaptor for notarizing NGSI-LD interactions on Blockchain
  - interface to query notarized transactions for entities
- What it is not:
  - a general connector to Blockchain
  - a storage component for NGSI-LD entities on the Blockchain

Features

- notarization on ETH-compatible blockchains
- notarization for a sub-set of the NGSI-LD API:
  - single entity creation (POST /ngsi-ld/v1/entities/)
  - batch creation (POST /ngsi-ld/v1/entityOperations/upsert)
  - attribute level updates (POST /ngsi-ld/v1/entities/{entityId}/attrs)
  - entity retrieval (GET /ngsi-ld/v1/entities/{entityId})
  - querying (GET /ngsi-ld/v1/entities/)
- storage of the transaction history off-chain in the ContextBroker
- dedicated query-api for transaction data by entities

Overview


Client-Domain
Client
Wallet
(Keystore, f.e. Vault)
Vault
Canis Major - Domain
ContextBroker
(f.e. orion-ld)
CanisMajor
Public
ETH-Compatible Node

Storage

- in order to reduce costs and data leakage, only hashes stored on the Blockchain
- transaction-receipts stored in the broker
  - SmartDatamodel for distributed ledger -> https://github.com/smart-data-models/dataModel.DistributedLedgerTech
  - reference to the original entity
  - allows query by entity
  - change history of an entity can be reproduced and validated through temporal
- Original entity stays untouched

TransactionReceipt Entity


```
{
        "@context": "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld",
        "id": "urn:ngsi-ld:dlttxreceipt:0x3371009d971c0cb2712032d283077e3dff025ce13f8f79bb4f347cfb84e44218",
        "type": "https://smartdatamodels.org/dataModel.DistributedLedgerTech/DLTtxReceipt",
        "https://smartdatamodels.org/dataModel.DistributedLedgerTech/refEntity": {
                "type": "Relationship",
                "object": "urn:ngsi-ld:Packet:packet-1"
        },
        "https://smartdatamodels.org/dataModel.DistributedLedgerTech/TxReceipts": {
                "type": "Property",
                "value": {
                        "blockHash": "0xdc3e80407fa9d3b12268eef903024781dc282a9b92a408a2802921c8d14b84d7",
                        "blockNumber": 4,
                        "blockNumberRaw": "0x4",
                        "cumulativeGasUsed": 23866,
                        "cumulativeGasUsedRaw": "0x5d3a",
                        "from": "0xa508dd875f10c33c52a8abb20e16fc68e981f186", <- address of the signee
                        "gasUsed": 23866,
                        "gasUsedRaw": "0x5d3a",
                        ….
                        "status": "0x1",
                        "statusOK": true,
                        "to": "0x476059cd57800db8eb88f67c2aa38a6fcf8251e0",
                        "transactionHash": "0x3371009d971c0cb2712032d283077e3dff025ce13f8f79bb4f347cfb84e44218",
                        "transactionIndex": 0,
                        "transactionIndexRaw": "0x0"
                }
        }
}
```

Query API

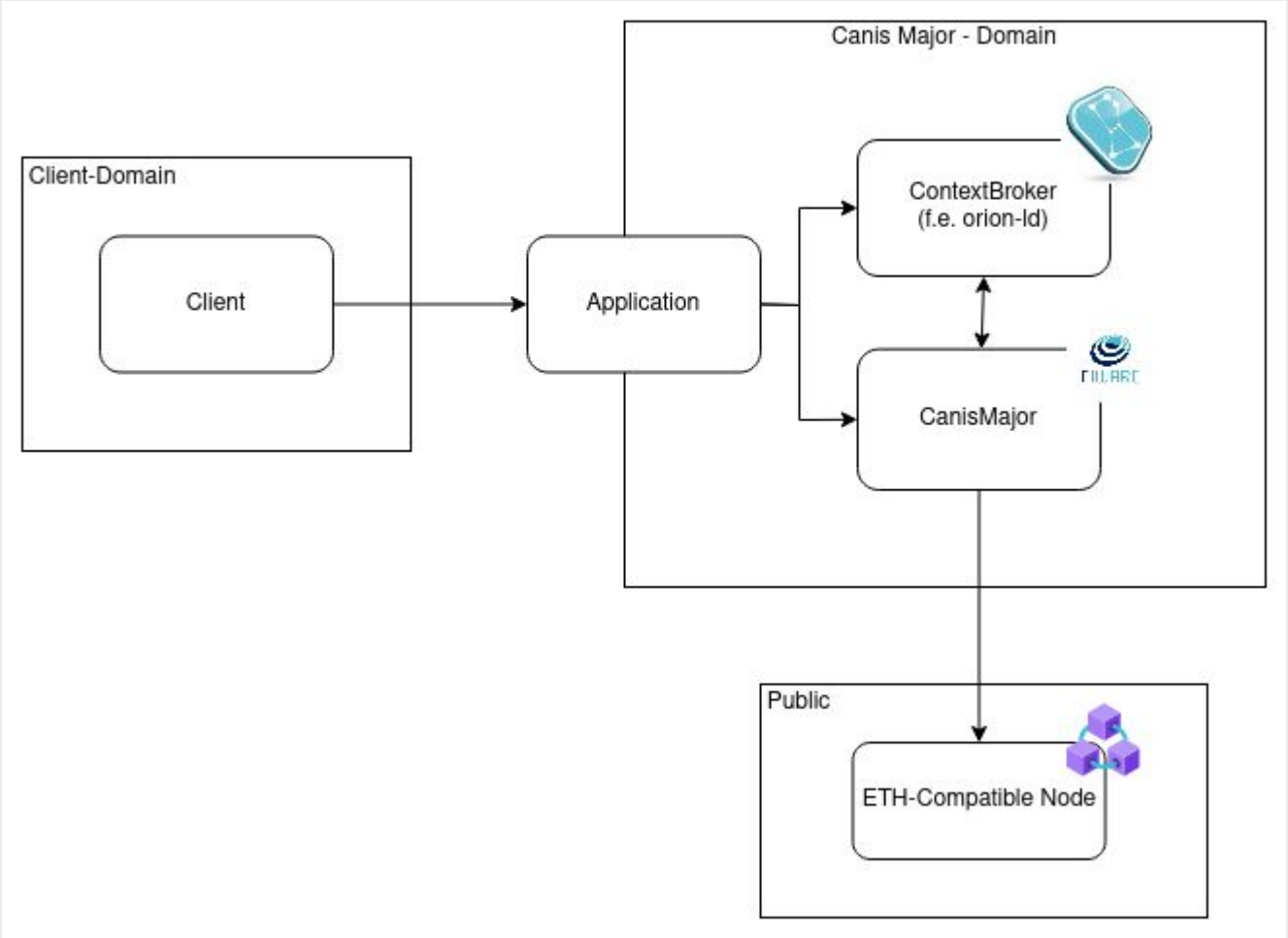
- Canis Major provides two convenience APIs:
  - get all entities with transactions
  - get all transactions for an entity

```
{
	"records": [
		{
		"entityId" : "urn:ngsi-ld:packet:1",
		"txDetails": [
			{
				"transactionHash": "0x3371009d971c0cb2712032d283077e3dff025ce13f8f79bb4f347cfb84e44218",
				"from": "0xa508dd875f10c33c52a8abb20e16fc68e981f186",
				…
			},
			{ another transaction }
		]},
		{
			another entity
		}
	],
	<pagination information>
}
```

Key Management

- every Transaction has to be signed
- 2 modes supported:
  - signed by the requestor
    - more responsibility for the requestor
    - slower
    - no trust in central entity required
  - signed by a “default”-account
    - transparent to the requestor
    - faster
    - trust in the central entity(runnig Canis Major) required, no real benefit from Blockchain integration

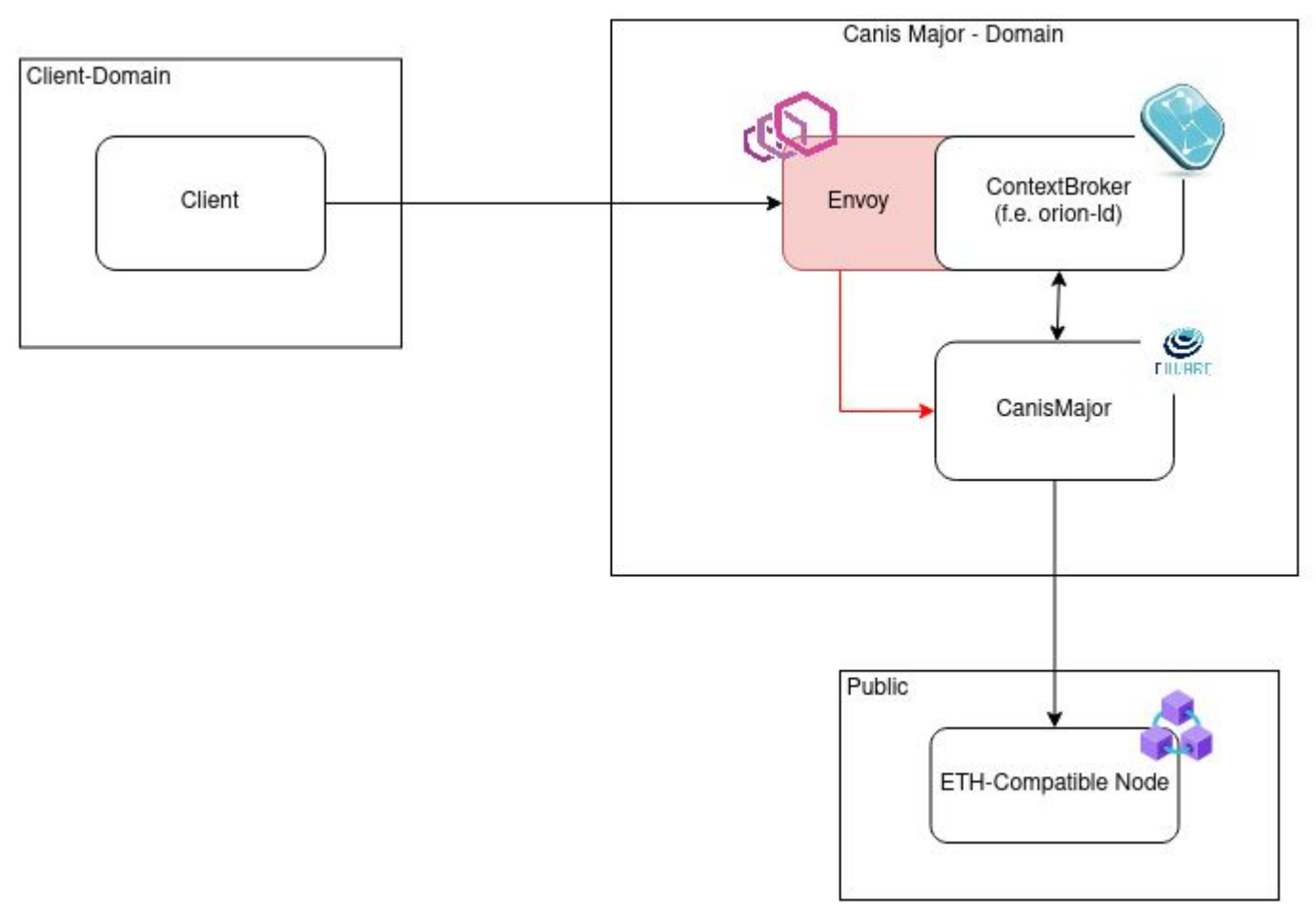
Demo - Simple


Canis Major - Domain
Client-Domain
Client
Application
ContextBroker
(f.e. orion-ld)
CanisMajor
Public
ETH-Compatible Node

Demo - Simple Scenario

- every transaction is signed with the default-account
- the Application has to take care of sending the request to:
  - the broker
  - Canis Major
- Application has to take care of transactionallity between Context Broker and Canis Major updates
- transaction history cannot show “who”

Demo - Proxy Integration



Demo - Proxy Integration

- every transaction is signed with the default-account
- client communicates directly with the Context Broker
- Front-Proxy(envoy) intercepts all incoming requests
- Front-Proxy is responsible for
  - handling communication with Canis Major
  - forwarding the requests to the Context Broker
- transparent to the client

Demo - Proxy Integration

- 2 modes supported:
  - strict:
    - Context Broker request is delayed until Canis Major confirms notarization
    - slows down the request, due to the sequential request
    - guarantees that every change is notarized
  - best-effort:
    - Canis Major and Context Broker request performed in parallel
    - faster, due to parallelism
    - no guarantee that every request is notarized
      - client get informed, needs to handle potential failed notarization itself

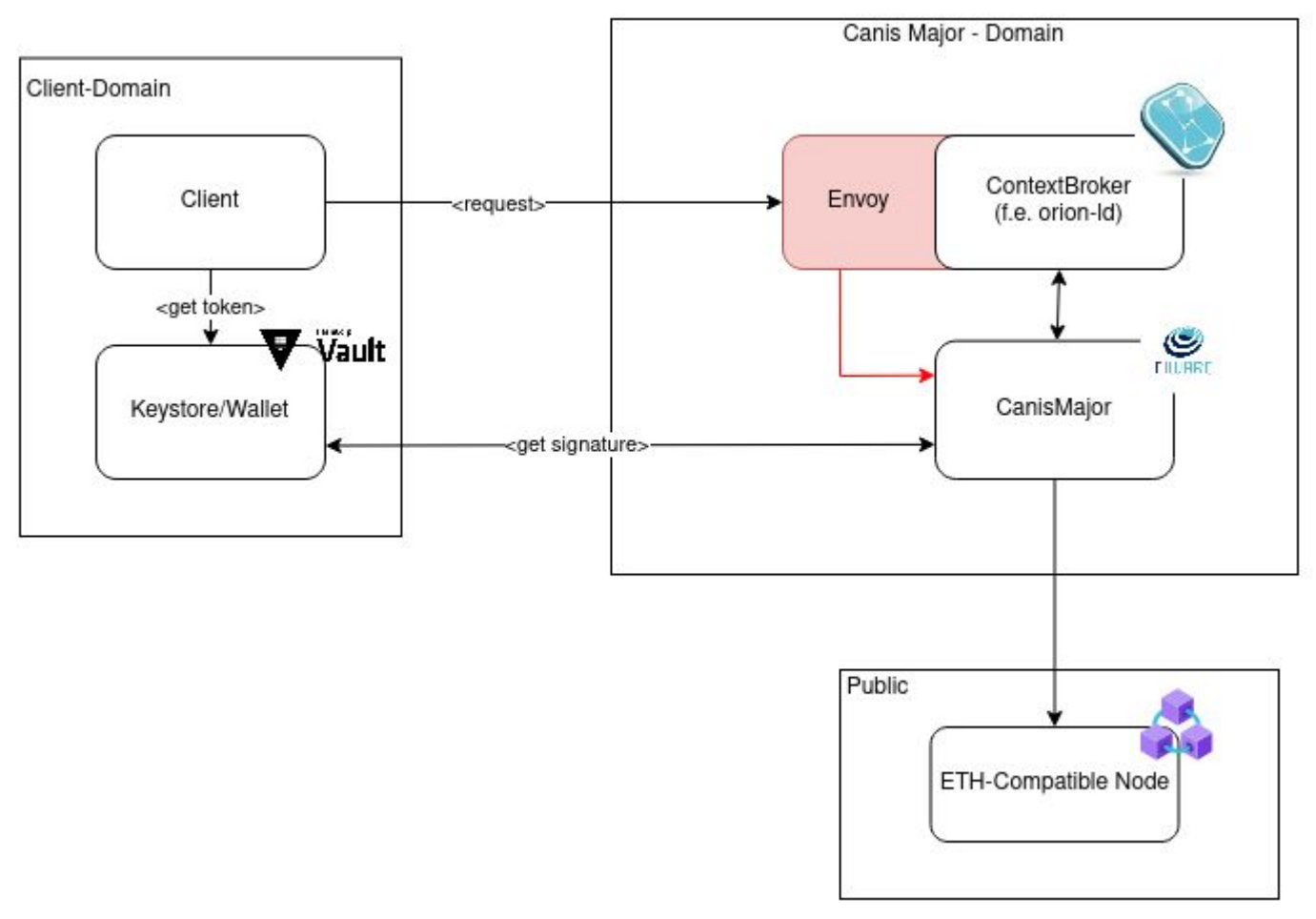
Front-Proxy integration

- Transparent integration of Canis Major into the request chain
- reduces influence on architecture impact for other components

Envoy as Front-Proxy:

- Stable and well maintained OpenSource-Project(CNCF graduated)
- support for proxy-wasm
  - well defined and documented plugin-mechanism
  - potential reuse in other proxies, API-Gateways
- integration with service meshes
  - OSSM and Istio are both Envoy based

Demo - Requestor Signature



Demo - Requestor Signature

- every transaction is signed with the key of the requestor
- 3 additional headers used to provide the wallet-callback:
  - Wallet-Type - currently only “Vault” supported
  - Wallet-Token
  - Wallet-Address
- Canis Major uses the provided information to request a signature for the transaction from the client wallet
  - private key does never leave the client
  - every transaction can be directly verified to the requestor

Demo - Requestor Signature

- Benefits:
  - every change can be connected to the responsible client
  - no single participant to trust, potentially able to add “fake-notarizations”
- Drawback:
  - Client has to provide a compliant endpoint
  - slower than the default-account option, due to the additional call

When to use?

- notarization beyond traditional access-logs is required
- immutability of the transaction history is required
- no single trusted entity exists
- the performance impact can be accepted
  - carefully decide what transactions need to be notarized

- Canis Major on github: https://github.com/FIWARE/CanisMajor
- Demo deployments:
  - https://github.com/FIWARE-Ops/fiware-gitops/tree/master/aws/fiware/canis-major
  - https://github.com/FIWARE-Ops/fiware-gitops/tree/master/aws/token

The slides: https://github.com/wistefan/presentations

FIWARE Global Summit

Find Us On

Stay up to date

JOIN OUR NEWSLETTER

Be certified and featured

FIWAREMarketplace

Hosting Partner

City of Vienna

vienna business agency

Keystone Sponsors

aws

ms.gis

Raseel

Red Hat

Media Partners

bee smart city

CITIES FORUM

CitiesToday

Our Future Water

RM RENEWABLE MATTER

Smart Cities World

the smartcity journal

FIWARE
Global
Summit

Thanks!

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds