

ODAP

Open Digital Asset Protocol

IETF112 Side Meeting
November 12, 2021

Thomas Hardjono (MIT), Martin Hargreaves (Quant), Ned Smith (Intel), Rama Ramakrishna (IBM)

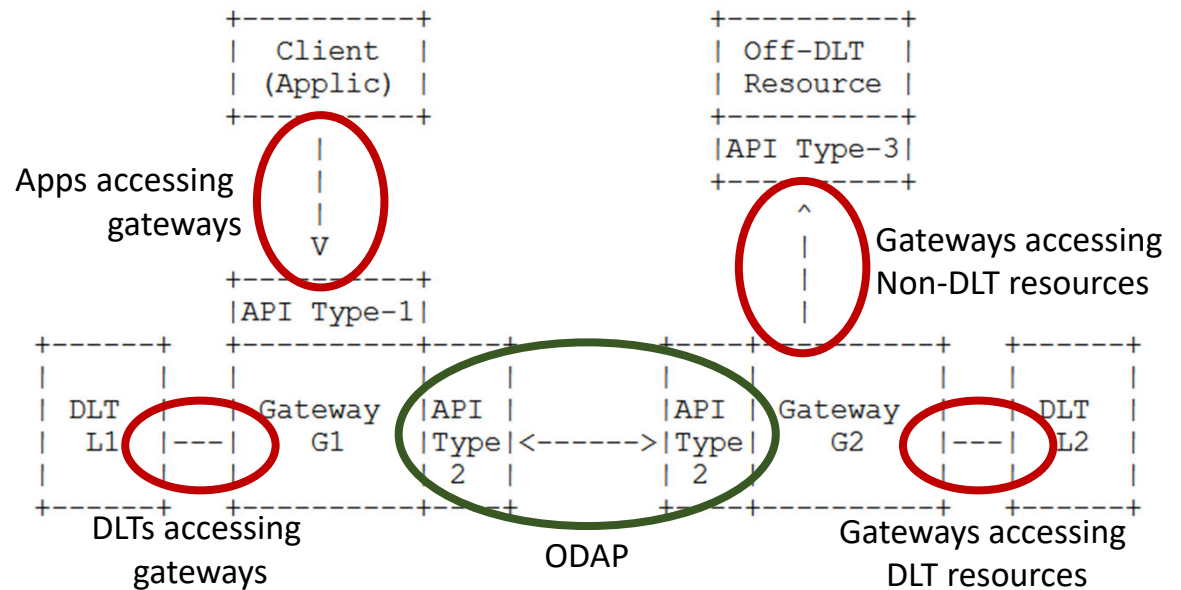
Open Digital Asset Protocol

- A protocol for the exchange of digital assets between distributed ledger networks (DLNs)
- Interoperability between DLNs is low
- Multiple proprietary schemes are proposed, often embedding particular distributed ledger technologies
- We propose a border gateway approach and protocol insulating applications from the underlying DLT / DLNs
- This approach allows standards based transfer of assets between DLNs, typically tokenised securities, debt, currency, data assets and cryptocurrencies

ODAP: Scope

Gateway to Gateway communications

Primarily to atomically transfer data objects between dissimilar DLTs



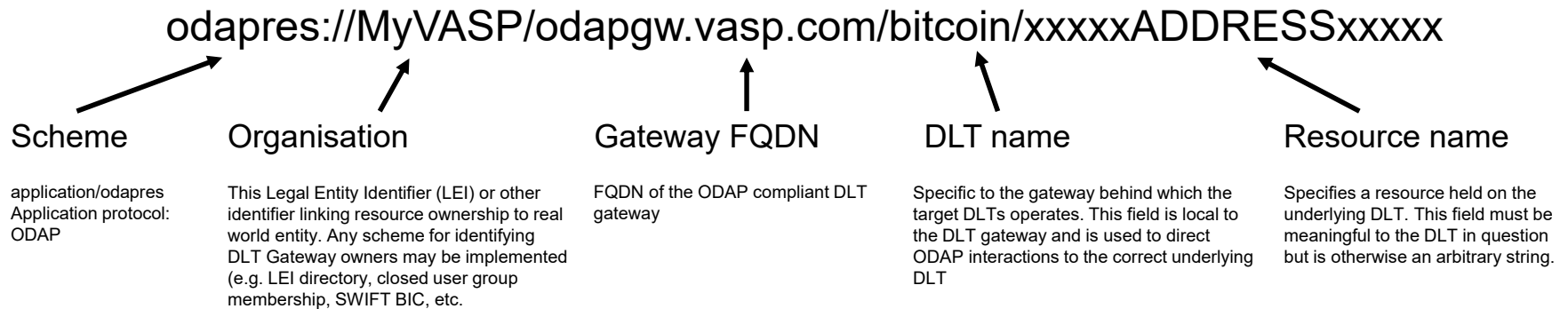
ODAP Draft 3 Coverage

- The phases of the ODAP protocol
- The format of ODAP messages
- The format for resource descriptors
- A method for gateways to implement access controls
- Protocol for negotiating security capabilities
- Discovery and accessing resources and provisions for backward compatibility with existing systems.

ODAP maps DLT resources to RFC 1738 (URL)

Resource addressing for DLTs, using the URL syntax, abstracting a diverse range of underlying schemes.

Client identification based on the URN format. These are for identifying clients (developers and applications) who access these resources, and which in some use-cases require access authorization.



ODAP Flows

Negotiation of Security Protocols and Parameters

- TLS Established
- Client offers supported credential schemes
- Server selects supported credential scheme
- Client asserts of proves identity
- Sequence numbers initialized
- Messages can now be exchanged

Transfer Initiation Flow (Phase 1)

- Initialization Request Message
- Initialization Request Message Response (ACK)

Lock-Evidence Verification Flow (Phase 2)

- Transfer Commence Message
- Transfer Commence Response Message (Ack)
- Lock Evidence Message
- Lock Evidence Response Message (Ack)

Commitment Establishment Flow (Phase 3)

- Commit Preparation Message
- Commit Preparation Response
- Commit Final Message
- Commit Final Response Message

ODAP Phases (asset transfer flows)

Phase 1

- o Secure channel establishment between G1 and G2
- o Mutual device attestations
- o Validation of the gateway ownership
- o Validation of VASP status
- o Identification and validation of asset profile
- o Exchange of Travel Rule information and validation
- o Negotiation of asset transfer protocol parameters

Phase 2

- o Commencement
- o Acknowledgement
- o G1 lock/escrow asset
- o G2 logs incoming asset
- o Lock Evidence
- o Evidence receipt

Phase 3

- o Commit-prepare
- o Ack-prepare
- o Lock-final
- o Commit-final
- o Asset-create
- o Ack-final
- o Location-record
- o Transfer complete

ODAP Defines Message Formats

Transfer Initiation Flow Example

Request

- o **Version**: ODAP protocol Version (major, minor).
- o **Developer URN**: Assertion of developer / application identity.
- o **Credential Profile**: Specify type of auth (e.g. SAML, OAuth, X.509)
- o **Payload Profile**: Asset Profile provenance and capabilities
- o **Application Profile**: Vendor or Application specific profile
- o **logging_profile** REQUIRED: contains the profile regarding the logging procedure. Default is local store.
- o **Access_control_profile** REQUIRED: the profile regarding the confidentiality of the log entries being stored. Default is only the gateway that created the logs can access them.
- o **Initialization Request Message signature** REQUIRED: Gateway EDCSA signature over the message
- o **Source_gateway_pubkey** REQUIRED: the public key of the gateway initiating a transfer
- o **Source_gateway_dlt_system** REQUIRED: the ID of the source DLT
- o **Recipient_gateway_pubkey** REQUIRED: the public key of the gateway involved in a transfer
- o **Recipient_gateway_dlt_system** REQUIRED: the ID of the recipient gateway involved in a transfer
- o **Escrow type**: faucet, timelock, hashlock, hashtimelock, multi-claim PC, destroy/burn (escrowed cross-claim).
- o **Expiry time**: when will the escrow or lock expire
- o **Multiple claims allowed**: true/false
- o **Multiple cancels allowed**: true/false
- o **Permissions**: list of identities (addresses or X.509 certificates) that can perform operations on the escrow or lock on the asset in the origin DLT network.
- o **Originator address**: along with the source gateway DLT, allows for the correct identification of the entity in the origin DLT transmitting the asset.
- o **Beneficiary address**: along with the recipient gateway DLT, allows for the correct identification of the entity in the destination DLT receiving the asset.
- o **Subsequent calls**: details possible escrow actions
- o **History** (optional): provides an history of the escrow, in case it has previously been initialized. This includes a list of the transactions on that escrow (transaction ID) and which action it performed (ActionCategory), the origin and destination, balance, current status, and ActionLockSpecificParameters.

Response

- o **Session ID**: unique identifier (UUIDv2) representing a session.
- o **Sequence Number**: monotonically increasing counter that uniquely represents a message from a session.
- o **ODAP Phase**: The current ODAP phase.
- o **Hash of the Initialization Request Message** REQUIRED: the hash of the proposal.
- o **Destination**: if the recipient gateway calculates the destination address dynamically.
- o **Timestamp** REQUIRED: timestamp referring to when the Initialization Request Message was received.
- o **Processed Timestamp** REQUIRED: timestamp referring to when the Initialization Response Message was constructed.

Implementations

- Hyperledger Cactus (code exists)
- Quant Overledger Gateway (2022)

Thank You