Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: May 11, 2022

M. Hargreaves
Quant Network
T. Hardjono
MIT
R. Belchior
Technico Lisboa

November 7, 2021

Open Digital Asset Protocol
draft-hargreaves-odap-03

Abstract

This memo describes the Open Digital Asset Protocol (ODAP). ODAP is an asset transfer protocol that operates between two gateway devices. The protocol includes a description of virtual or digital assets held on distributed ledgers in an open and interoperable format, a session negotiation part and message passing flows between gateways connecting disparate distributed ledger technologies (DLTs).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There is a lack of interoperability between individual blockchains, but also a general difficulty building open DLT networks. Extant networks are custom built and relatively closed, usually limited to networks of a single DLT type.

This memo proposes at DLT-agnostic protocol in order to allow the creation of business applications that use and modify multiple DLTs, through a single programmatic interface.

The target DLTs can be of any type, operated by different owners and managed using different DLT interoperability and management platforms that implement ODAP interfaces.

These platforms may act as gateways or relays for the application to interact with the hosted DLTs. They are referred to herein as DLT Gateways.

When correctly implemented and deployed, the protocol should provide the basis for solutions involving asset migration between two DLT systems, as well as use-cases when one side is non-DLT system (e.g. legacy system).

This memo focuses on the core aspects of the ODAP protocol, with the understanding that profiles of this core protocol would need to be utilized for implementations that target specific use-cases and employing different technologies to implement their DLT network.

The mechanisms used by client applications to interact with a sender gateway is out of scope for this specification and will be specified elsewhere.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

3. Terminology

The following are some terminology used in the current document. Further terminology can be found in [Arch].

Client application: This is the application employed by a user to interact with a gateway node.

Gateway: The node of the DLT system functionally capable of acting as a gateway in an asset transfer.

Sender gateway: The gateway that initiates a unidirectional asset transfer.

Transfer client: An alternate name for the sender gateway (protocol level).

Recipient gateway: The gateway that is the recipient side of a unidirectional asset transfer.

Transfer server: An alternate name for the recipient gateway (protocol level).

DLT resources: The various interior protocols, data structures and cryptographic constructs that are a core part of a DLT system.

Off-DLT resources: The various resources that are outside a DLT system, and are not part of the operations of the DLT system.

Role: As in the classic client-server roles. In the gateway-to-gateway interaction, one gateway will take the role of the client while the other takes the role of the server, depending on the type of interaction flow.

Claim: An assertion made by an Entity [JWT].

Claim Type: Syntax used for representing a Claim Value [JWT].

DLT Claim: An assertion made by a Gateway regarding the status or condition of resources (e.g. asset, public keys, etc.) accessible to that gateway within its DLT system.

4. The Open Digital Asset Protocol

4.1. Overview

The Open Digital Asset Protocol (ODAP) is a gateway-to-gateway protocol used by a sender gateway with a recipient gateway to perform a unidirectional transfer of a virtual asset [Arch].

The protocol defines a number of API endpoints, resources and identifier definitions, and message flows corresponding to the asset transfer between the two gateways.

```
                +----------+                +----------+
                |  Client  |                | Off-DLT  |
                | (Applic) |                | Resource |
                +----------+                +----------+
                     |                      |API Type-3|
                     |                      +----------+
                     |                           ^
                     V                           |
                +----------+                     |
                |API Type-1|                     |
     +------+   +----------+----+        +----+----------+   +------+
     |      |   |          |    |        |    |          |   |      |
     | DLT  |   | Gateway  |API |        |API | Gateway  |   | DLT  |
     |  L1  |---|    G1    |Type|<------>|Type|    G2    |---|  L2  |
     |      |   |          | 2  |        | 2  |          |   |      |
     +------+   +----------+----+        +----+----------+   +------+
```

Figure 1

4.2. ODAP Model

Following the gateway interoperability architecture [Arch], the model for ODAP is shown in Figure 1.

The Client (application) interacts with its local gateway (G1) over an interface (API Type-1) in order to provide instructions directly

or indirectly to the gateway with regards to actions related resources located in the local DLT system (L1) and resources located in remote DLT system (L2).

Gateways interact with each other over a gateway interface (API Type-2). A given gateway may be required to access resources that are not located in DLT system L1 or DLT system L2. Access to these types of resources are performed over an Off-DLT interface (API Type-3).

4.3. Types of APIs

The following are the types of APIs in ODAP:

o Gateway APIs for client (API Type-1): This the REST APIs that permit a Client (application) to interact with a local gateway, and issue instructions for actions pertaining to resources accessible to the gateway in the local DLT system.

o Gateway APIs for peer gateways (API Type-2): This is the REST APIs employed by two (2) peer gateways in performing unidirectional asset transfers.

o APIs for validation of off-DLT resources (API Type-3): This is the REST APIs made available by a resource server (resource owner) at which a gateway can access Off-DLT resources.

The use of these APIs is dependent on the mode of access and the type of flow in question.

4.4. Types of Flows

The ODAP protocol defines the following three (3) flows:

o Transfer Initiation flow: This flow deals with the asset profile verification, asset ownership evidence verification and identities verification.

o Lock-Evidence flow: This flow deals with the conveyance of evidence regarding the lock (escrow) status of an asset by one gateway, and the verification of the evidence by the other gateway.

o Commitment Establishment flow: This flow deals with the asset transfer and commitment establishment between two gateways on behalf of their DLT systems.

These flow will be discussed below.

4.5. Resources and Identifiers

- (a) Resource addressing for DLTs, using the URL syntax.
- (b) 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.
- (c) Protocol message family for negotiating authentication, authorisation, and parameters for confidential channel establishment.
- (d) Resource discovery mechanism for developers and applications to discover DLT-based resources hosted at a DLT gateway. The gateway response is subject to the level of access granted to that developer or application.

5. ODAP Message Format, identifiers and Descriptors

5.1. Overview

This section describes (i) the phases of the ODAP protocol; (ii) the format of ODAP messages; (iii) the format for resource descriptors; (iv) a method for gateways to implement access controls; (iv) protocol for negotiating security capabilities; (v) discovery and accessing resources and provisions for backward compatibility with existing systems.

5.2. ODAP Message Format

ODAP messages are exchanged between applications (clients) and DLT gateways (servers). They consist of protocol negotiation and functional messages.

Messages are JSON format, with protocol specific mandatory fields, support for arbitrary authentication and authorization schemes and support for a free format field for plaintext or encrypted payloads directed at the DLT gateway or an underlying DLT.

JSON format message, mandatory fields are shown below:

- Version: ODAP protocol Version (major, minor).
- Session ID: unique identifier (UUIDv2) representing a session
- Sequence Number: monotonically increasing counter that uniquely represents a message from a session.

- o ODAP Phase: The current ODAP phase.
- o Resource URL: Location of Resource to be accessed.
- o Developer URN: Assertion of developer / application identity.
- o Action/Response: GET/POST and arguments (or Response Code)
- o Credential Profile: Specify type of auth (e.g. SAML, OAuth, X.509)
- o Credential Block: Credential token, certificate, string
- o Payload Profile: Asset Profile provenance and capabilities
- o Application Profile: Vendor or Application specific profile
- o Payload: Payload for POST, responses, and native DLT txns. The payload is specific to the current ODAP phase.
- o Payload Hash: hash of the current message payload.
- o Message signature: Gateway EDCSA signature over the message

Other relevant attributes may exists that need to be captured for logging purposes. See [ODAP-2PC].

5.3. Digital Asset Resource Descriptors

Resources are identified by URL [RFC 1738] as described below:

- o The type is new: application/odapres
- o The access protocol is ODAP.

Data included in the URL includes the folowing:

5.3.1. Organization Identifier

This Legal Entity Identifier (LEI) or other identifier linking resource ownership to real world entity. Any scheme for identifying DLT Gateway owners may be implemented (e.g. LEI directory, closed user group membership, SWIFT BIC, etc.).

The developer or application MAY validate the identity with the issuing authority. The identifier is not a trusted identity, but MAY be relied on where trust has been established between the two parties (e.g. in a closed user group).

The mechanisms to determine organizations identifiers is out of scope for the current specification.

5.3.2. DLT Gateway / Endpoint ID

FQDN of the ODAP compliant DLT gateway. Required to establish IP connectivity. This MUST resolve to a valid IP address.

5.3.3. DLT Identifier

Specify to gateway behind which the target DLTs operates. This field is local to the DLT gateway and is used to direct ODAP interactions to the correct underlying DLT.

For example: "Hyperledger1", "Bitcoin, "EU-supply-chain".

5.3.4. Resource

Specifies a resource held on the underlying DLT. This field must be meaningful to the DLT in question but is otherwise an arbitrary string. The underlying object it points to may be a DLT address, block, transaction ID, alias, etc. or a future object type not yet defined.

5.3.5. Examples

```
odapres://quant/api.gateway1.com/ripple

odapres://quant/api.gateway1.com/bitcoin/xxxxxADDRESSxxxxx
```

5.4. Digital Asset Resource Client Descriptors

Resources are identified by URN as described below:

o The type is new: application/odapclient

The URN format does not imply availability or access protocol.

Data included in the URN includes the folowing:

5.4.1. Organization Identifier

Legal Entity Identifier (LEI) or other identifier linking resource ownership to real world entity. Any scheme for identifying DLT Gateway owners may be implemented (e.g. LEI directory, closed user group membership, BIC, etc.).

The DLT Gateway MAY validate the identity with the issuing authority. The identifier is not a trusted identity, but MAY be relied on where trust has been established between the two parties (e.g. in a closed user group).

5.4.2. DLT Gateway / Endpoint ID

Multi-DLT applications can operate in a mode whereby the application connects to its local DLT gateway, which then forwards application traffic to local DLTs and to remote DLTs via other ODAP gateways.

Where this is the case, this field identifies the "home" gateway for this application. This may be required to carry out Gateway to Gateway handshaking and protocol negotiation, or for the server to look up use case specific data relating to the client.

5.4.3. Organizational Unit

The organization unit within the organization that the client (application or developer) belongs to. This assertion should be backed up with authentication via the negotiated protocol.

The purpose of this field is to allow DLT gateways to maintain access control mapping between applications and resources that are independent of the authentication and authorization schemes used, supporting future changes and supporting counterparties that operate different schemes.

5.4.4. Name

A locally unique (within the OU) identifier, which can identify the application, project or individual developer responsible for this client connection. This is the most granular unit of access control, and DLT Gateways should ensure appropriate identifiers are used for the needs of the application or use case.

5.4.5. Examples

```
odapclient:quant/api.overledger.quant.com/research/luke.riley
```

5.5. Gateway Level Access Control

Gateways can enforce access rules based on standard naming conventions using novel or existing mechanisms such as AuthZ protocols using the resource identifiers above, for example:

```
odapclient://hsbc/api.overledger.hsbc.com/lending/eric.devloper
```

can READ/WRITE

odapres://quant/api.gateway1.com/bitcoin

AND

odapres://quant/api.gateway1.com/ripple

These rules would allow a client so identified to access resources directly, for example:

odapres://quant/api.gateway1.com/bitcoin/xxxxxADDRESSxxxxx

This example could be an client subscribing to or writing to an address associated with a smart contract as part of its functionality.

This method allows resource owners to easily grant access to individuals, groups and organizations. Individual gateway implementations may implement access controls, including subsetting and supersetting or applications or resources according to their own requirements.

5.6. Negotiation of Security Protocols and Parameters

5.6.1. TLS Established

TLS 1.2 or higher MUST be implemented to protect gateway communications. TLS 1.3 or higher SHOULD be implemented where both gateways support TLS 1.3 or higher.

5.6.2. Client offers supported credential schemes

Capability negotiation prior to data exchange, follows a scheme similar to the Session Description Protocol [RFC 5939]. Initially the client (application) sends a JSON block containing acceptable credential schemes, e.g. OAuth2.0, SAML in the "Credential Scheme" field of the ODAP message.

5.6.3. Server selects supported credential scheme

The server (DLT Gateway) selects one acceptable credential scheme from the offered schemes, returning the selection in the "Credential Scheme" field of the ODAP message.

If no acceptable credential scheme was offered, an HTPP 511 "Network Authentication Required" error is returned in the Action/Response field of the ODAP message.

5.6.4. Client asserts of proves identity

The details of the assertion / verification step are specific to the chosen credential scheme and are out of scope of this document.

5.6.5. Sequence numbers initialized

Sequence numbers are used to allow the server to correctly order operations from the client, some of which may be asynchronous, synchronous, idempotent with duplicate requests handled in different ways according to the use case.

The initial sequence number is proposed by the client (Application) after the finalization of credential verification. The server (DLT gateway) MUST respond with the same sequence number to indicate acceptance.

The client (application) increments the sequence number with each new request. Sequence numbers can be reused for retries in the event of a gateway timeout.

5.6.6. Messages can now be exchanged

Handshaking is complete at this point, and the client (application) can send ODAP messages to perform actions of DLT resources, which MAY reference the ODAP Payload field.

5.7. Asset Profile Identification

The client and server must mutually agree as to the asset type or profile that is the subject to the current transfer from the client and server. The client must provide the server with the asset-identification number, or the server may provide the client with the asset-identification numbers for the digital asset supported by the server.

Formal specification of asset identification is out of scope of this document. Global numbering of digital asset types or profiles is expected to be performed by a legally recognized entity.

5.8. Application Profile Negotiation

Where an application relies on specific extensions for operation, these can be represented in an Application Profile.

For example, a payments application tracks payments through the use of a cloud based API and will only interact with Gateways that log messages to that API, a resource profile can be established:

Application Name: TRACKER

X-Tracker_URL: https://api.tracker.com/updates

X-Tracking-Policy: Always

As Gateways implement this functionality, they support the TRACKER application profile, and the application is able to expand its reach by periodically polling for the availability of the profile.

This is an intentionally generalized extension mechanism for application or vendor specific functionality.

5.9. Discovery of Digital Asset Interior Resources

Applications located outside the DLT network MUST be able to discover which resources they are authorized to access to the level of individual DLTs. They MAY be able to discover lower level resources.

Resource discovery is handled by the DLT gateway. For instance using a GET request against the gateway URL with no DLT or resource identifier could return a list of URLs available to the requester at DLT level. This list is subject to the access controls above.

DLT Gateways MAY allow applications to discover resources they do not have access to. This should be indicated in the free text field, and gateways SHOULD implement a process for applications to request access.

Formal specification of supported resource discovery methods is out of scope of this document.

5.10. Response Codes

The DLT Gateway MUST respond with return codes indicating the failure or success of each operation. For DLTs with slow consensus mechanism, the Gateway may return codes indicating the operation has been submitted. The application may carry out further operation in future to determine the ultimate status of the operation.

For Non-native transactions, the Gateway is responsible for translating the request into the appropriate native format and ensuring correct signing takes place.

5.11. Backward Compatibility

It is also possible to send a fully formatted native message to the underlying DLT in the Payload field using the NATIVE_TXN operation, directed to a resource URL. This allows existing DLT native code to be ported to ODAP infrastructures with minimal change.

6. Transfer Initiation Flow (Phase 1)

This section describes ODAP initialization phase, where a sender gateway (transfer client) interacts with a recipient gateway (transfer server), proposing a session.

The session may be triggered directly or indirectly by a client application behind the sender gateway. The method employed by an application to invoke the sender gateway is out of scope for this specification.

For this phase, several artifacts related to the asset transfer must be validated: asset profile, asset ownership evidence, identities, and logging-related operations (log profile, access control profile [ODAP-2PC]).

In this phase, gateways implement the Transfer Initiation Flows endpoint.

The flow follows a request-response model. The sender gateway (transfer client) makes a request (POST) to the Transfer Initiation endpoint at the recipient gateway (transfer server).

Gateways MUST support the use of the HTTP GET and POST methods defined in RFC 2616 [RFC2616] for the endpoint.

The transfer clients MAY use the HTTP GET or POST methods to send messages in this phase to the transfer server. If using the HTTP GET method, the request parameters maybe serialized using URI Query String Serialization.

The transfer client and server may be required to sign certain messages in order to provide standalone proof (for non-repudiation) independent of the secure channel between the client and server. This proof maybe required for audit verifications post-event.

Flows occur over TLS secure session that must be established between the client and server.

6.1. Initialization Request Message

This message is sent from the sender gateway to the recipient gateway This message may be the result of a client application invoking its local gateway to request an asset transfer event

Depending on the transfer proposal, multiple rounds of communications between the client application and its sender gateway may occur. Similarly, multiple rounds of communication between gateways may occur.

The parameters of this message consists of the following:

- Version: ODAP protocol Version (major, minor).
- Developer URN: Assertion of developer / application identity.
- Credential Profile: Specify type of auth (e.g. SAML, OAuth, X.509)
- Payload Profile: Asset Profile provenance and capabilities
- Application Profile: Vendor or Application specific profile
- logging_profile REQUIRED: contains the profile regarding the logging procedure. Default is local store.
- 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.
- Initialization Request Message signature REQUIRED: Gateway EDCSA signature over the message
- Source_gateway_pubkey REQUIRED: the public key of the gateway initiating a transfer
- Source_gateway_dlt_system REQUIRED: the ID of the source DLT
- Recipient_gateway_pubkey REQUIRED: the public key of the gateway involved in a transfer
- Recipient_gateway_dlt_system REQUIRED: the ID of the recipient gatewayinvolved in a transfer
- Escrow type: faucet, timelock, hashlock, hashtimelock, multi-claim PC, destroy/burn (escrowed cross-claim).

- Expiry time: when will the escrow or lock expire

- Multiple claims allowed: true/false

- Multiple cancels allowed: true/false

- 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.

- Originator address: along with the source gateway DLT, allows for the correct identification of the entity in the origin DLT transmitting the asset.

- Beneficiary address: along with the recipient gateway DLT, allows for the correct identification of the entity in the destination DLT receiving the asset.

- Subsequent calls: details possible escrow actions

- 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.

The sender gateway makes the following HTTP request using TLS (with extra line breaks for display purposes only).

Example: TBD.

6.2. Initialization Request Message Response (ACK)

After receiving an Initialization Request Message from the sender gateway (transfer client), the recipient gateway (transfer server) must validate the profiles stated in the request message.

This validation could be performed automatically (using a defined set of rules), or by requiring approval by an application operated by the beneficiary.

If one of the profiles is erroneous or rejected, the recipient gateway constructs an Initialization Denied Message, stating what was rejected, and proposing an alternative (if applicable).

Otherwise, if approved, the recipient gateway constructs a Initialization Request Message Response.

The purpose of this acknowledgement message is for the recipient gateway to indicate agreement to proceed with the proposed operations, under the proposed profiles.

This message is sent from the recipient gateway to the sender gateway in response to a received Initialization Request from the sender gateway.

The message must be signed by the recipient gateway.

The parameters of this message consists of the following:

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.

Example: TBD.

7. Lock-Evidence Verification Flow (Phase 2)

This section describes the conveyance of claims regarding to the status of assets or resources from a sender gateway to a recipient gateway.

In this phase, gateways implement the Lock-Evidence Agreement endpoint.

In the following, the sender gateway takes the role of the transfer-client while the recipient gateway takes the role of the transfer-server.

The flow follows a request-response model. The client makes a request (POST) to the Lock-Evidence Agreement endpoint at the server.

Gateways MUST support the use of the HTTP GET and POST methods defined in RFC 2616 [RFC2616] for the endpoint.

Clients MAY use the HTTP GET or POST methods to send messages in this phase to the server. If using the HTTP GET method, the request parameters maybe serialized using URI Query String Serialization.

The client and server may be required to sign certain messages in order to provide standalone proof (for non-repudiation) independent of the secure channel between the client and server. This proof maybe required for audit verifications post-event.

(NOTE: Flows occur over TLS. Nonces are not shown).

7.1. Transfer Commence Message

This message is sent from the transfer-client (sender gateway) to the Transfer Request Endpoint at the transfer-server (recipient gateway). It signals to the server that the client is ready to start the transfer of the digital asset.

The message must contain claims related to the information from the previous flow (Phase 1). It must be signed by the client (sender gateway).

The parameters of this message consists of the following:

o message_type REQUIRED. MUST be the value urn:ietf:odap:msgtype:transfer-commence-msg

o originator_pubkey REQUIRED. This is the public key of the asset owner (originator) in the origin DLT system.

o beneficiary_pubkey REQUIRED. This is the public key of the beneficiary in the destination DLT system.

o sender_dlt_system REQUIRED. This is the identifier of the origin DLT system behind the client.

o recipient_dlt_system REQUIRED. This is the identifier of the destination DLT system behind the server.

o client_identity_pubkey REQUIRED. The client who sent this message.

o server_identity_pubkey REQUIRED. The server for whom this message is intended.

- o hash_asset_profile REQUIRED. This is the hash of the asset profile previously agreed upon in Phase 1.

- o asset_unit OPTIONAL. If applicable this is the unit amount of the asset being transferred, previously agreed upon.

- o hash_prev_message REQUIRED. The hash of the last message in Phase 1.

- o client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only the client.

- o client_signature REQUIRED. The digital signature of the client.

For example, the client makes the following HTTP request using TLS (with extra line breaks for display purposes only):

```
 POST /token HTTP/1.1
   Host: server.example.com
   Authorization: Basic awHCaGRSa3F0MzpnWDFmQmF0M2ZG
   Content-Type: application/x-www-form-urlencoded

       {
       "message_type": "urn:ietf:odap:msgtype:transfer-commence-msg",
       "originator_pubkey":"zGy89097hkbfgkjvVbNH",
       "beneficiary_pubkey": "mBGHJjjuijh67yghb",
       "sender_dlt_system": "originDLTsystem",
       "recipient_dlt_system":"recipientDLTsystem",
       "client_identity_pubkey":"fgH654tgeryuryuy",
       "server_identity_pubkey":"dFgdfgdfgt43tetr535teyrfge4t54334",
       "hash_asset_profile":"nbvcwertyhgfdsertyhgf2h3v4bd3v21",
       "asset_unit": "ghytredcfvbhfr",
       "hash_prev_message":"DRvfrb654vgreDerverv654nhRbvder4",
       "client_transfer_number":"ji9876543ewdfgh",
       "client_signature":"fdw34567uyhgfer45"
       }
```

Figure 2

7.2. Transfer Commence Response Message (Ack)

The purpose of this message is for the transfer server to indicate agreement to proceed with the asset transfer.

This message is sent from the transfer server (recipient gateway) to the transfer client (sender gateway) in response to a Transfer Commence Request from the client.

The message must be signed by the server.

The parameters of this message consists of the following:

- message_type REQUIRED urn:ietf:odap:msgtype:transfer-commenceack-msg

- client_identity_pubkey REQUIRED. The client for whom this message is intended.

- server_identity_pubkey REQUIRED. The server who sent this message.

- hash_commence_request REQUIRED. The hash of previous message.

- server_transfer_number OPTIONAL. This is the transfer identification number chosen by the server. This number is meaningful only to the server.

- server_signature REQUIRED. The digital signature of the server.

An example of a success response could be as follows:

```
    HTTP/1.1 200 OK
    Content-Type: application/json;charset=UTF-8
    Cache-Control: no-store
    Pragma: no-cache

    {
      "message_type":"urn:ietf:odap:msgtype:transfer-commenceack-msg",
      "client_identity_pubkey":"fgH654tgeryuryuy",
      "server_identity_pubkey":"dFgdfgdfgt43tetr535teyrfge4t54334",
      "hash_commence_request":"DRvfrb654vgreDerverv654nhRbvder4",
      "server_transfer_number":"ji9876543ewdfgh",
      "server_signature":"aaw34567uyhgfer66"
    }
```

Figure 3

7.3. Lock Evidence Message

The purpose of this message is for the transfer client (sender gateway) to deliver the relevant asset-lock (or escrow) evidence to the server (recipient gateway).

The format of the evidence is dependent on the DLT fronted by the client and is outside the scope of this specification.

The message must be signed by the client.

This message is used client (sender gateway) to convey lock/escrow evidence to the server (recipient gateway).

This message is sent from the client to the Evidence Validation Endpoint at the server. The server must validate the lock evidence claims in this message.

The message must be signed by the client (sender gateway).

The parameters of this message consists of the following:

- message_type REQUIRED urn:ietf:odap:msgtype:lock-evidence-req-msg

- client_identity_pubkey REQUIRED. The client who sent this message.

- server_identity_pubkey REQUIRED. The server for whom this message is intended.

- lock_evidence_claim REQUIRED. The lock or escrow evidence (on the ledger L1 fronted by the client G1).

- lock_claim_format OPTIONAL. The format of the evidence.

- lock_evidence_expiration REQUIRED. The duration of time of lock on ledger L1 (after which the lock is released).

- hash_commence_ack_request REQUIRED. The hash of previous message.

- client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only to the client.

- client_signature REQUIRED. The digital signature of the client.

7.4. Lock Evidence Response Message (Ack)

The purpose of this message is for the transfer server (recipient gateway) to indicate acceptance of the asset-escrow (or lock) evidence delivered by the client (sender gateway) in the previous message.

The message must be signed by the server.

The parameters of this message consists of the following:

- o message_type REQUIRED urn:ietf:odap:msgtype:lock-evidence-ack-msg
- o client_identity_pubkey REQUIRED. The client for whom this message is intended.
- o server_identity_pubkey REQUIRED. The server who sent this message.
- o hash_lockevidence_request REQUIRED. The hash of previous message.
- o server_transfer_number OPTIONAL. This is the transfer identification number chosen by the server. This number is meaningful only to the server.
- o server_signature REQUIRED. The digital signature of the server.

8. Commitment Establishment Flow (Phase 3)

This section describes the transfer commitment agreement between the sender gateway (transfer client) to a recipient gateway (transfer server).

This phase must be completed within the asset-lock duration time specificied in the previous lock_evidence_expiration parameter (message 7.3).

In this phase gateways implement the Transfer Commitment endpoint.

In the following, the sender gateway takes the role of the client while the recipient gateway takes the role of the server.

The flow follows a request-response model. The client makes a request (POST) to the Transfer Commitment endpoint at the server.

Gateways MUST support the use of the HTTP GET and POST methods defined in RFC 2616 [RFC2616] for the endpoint.

Clients MAY use the HTTP GET or POST methods to send messages in this phase to the server. If using the HTTP GET method, the request parameters maybe serialized using URI Query String Serialization.

The client and server may be required to sign certain messages in order to provide standalone proof (for non-repudiation) independent of the secure channel between the client and server. This proof maybe required for audit verifications post-event.

(NOTE: Flows occur over TLS. Nonces are not shown).

8.1. Commit Preparation Message

The purpose of this message is for the client to indicate its readiness to begin the commitment of the transfer.

The message must be signed by the client.

The parameters of this message consists of the following:

o message_type REQUIRED. It MUST be the value urn:ietf:odap:msgtype:commit-prepare-msg

o client_identity_pubkey REQUIRED. The client who sent this message.

o server_identity_pubkey REQUIRED. The server for whom this message is intended.

o hash_lockevidence_ack REQUIRED. The hash of previous message.

o client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only the client.

o client_signature REQUIRED. The digital signature of the client.

8.2. Commit Preparation Response

The purpose of this message is for the server to indicate to the client its readiness to proceed with the commitment finalization step.

The message must be signed by the server.

The parameters of this message consists of the following:

- o message_type REQUIRED. It MUST be the value urn:ietf:odap:msgtype:commit-prepare-ack-msg

- o client_identity_pubkey REQUIRED. The client for whom this message is intended.

- o server_identity_pubkey REQUIRED. The server who sent this message.

- o hash_commitprep REQUIRED. The hash of previous commit preparation message.

- o server_transfer_number OPTIONAL. This is the transfer identification number chosen by the server. This number is meaningful only the server.

- o server_signature REQUIRED. The digital signature of the server.

8.3. Commit Final Message

The purpose of this message is for the client to indicate to the server that the client (sender gateway) has completed local extinguishment of the asset on its DLT (L1), and that now on its part the server (recipient gateway) must re-generated the asset on its DLT (L2).

The message must contain claims related to the extinguishment of the asset by the client. It must be signed by the client.

The parameters of this message consists of the following:

- o message_type REQUIRED. It MUST be the value urn:ietf:odap:msgtype:commit-final-msg

- o client_identity_pubkey REQUIRED. The client who sent this message.

- o server_identity_pubkey REQUIRED. The server for whom this message is intended.

- o commit_final_claim REQUIRED. This is one or more claims signed by the client that the asset in question has been extinguished by the client in its local DLT.

- o commit_final_claim_format OPTIONAL. This is the format of the claim provided by the client in this message.

- o hash_commitprepare_ack REQUIRED. The hash of previous message.

- client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only the client.

- client_signature REQUIRED. The digital signature of the client.

8.4. Commit Final Response Message

The purpose of this message is for the server to indicate to the client that it has completed the asset re-generation at its DLTS (L2).

The message must contain claims related to the re-generated of the asset by the server. It must be signed by the server.

The parameters of this message consists of the following:

- message_type REQUIRED. It MUST be the value urn:ietf:odap:msgtype:commit-final-ack-msg

- client_identity_pubkey REQUIRED. The client for whom this message is intended..

- server_identity_pubkey REQUIRED. The server who sent this message.

- commit_acknowledgement_claim REQUIRED. This is one or more claims signed by the server that the asset in question has been regenerated by the server in its local DLT.

- commit_acknowledgement_claim_format OPTIONAL. This is the format of the claim provided by the server in this message.

- hash_commitfinal REQUIRED. The hash of previous commit final message.

- server_transfer_number OPTIONAL. This is the transfer identification number chosen by the server. This number is meaningful only the server.

- server_signature REQUIRED. The digital signature of the server.

8.5. Transfer Complete Message

The purpose of this message is for the client to indicate to the server that the asset transer has been completed and that no further messages are to be expected from the client in regards to this transfer instance.

The message closes the first message of Phase 2 (Transfer Commence Message). It must be signed by the client.

The parameters of this message consists of the following:

- o message_type REQUIRED. It MUST be the value urn:ietf:odap:msgtype:commit-transfer-complete-msg

- o client_identity_pubkey REQUIRED. The client who sent this message.

- o server_identity_pubkey REQUIRED. The server for whom this message is intended.

- o hash_commit_final_ack REQUIRED. The hash of previous message.

- o hash_transfer_commence REQUIRED. The hash of the Transfer Commence message at the start of Phase 2 (see section 7.1).

- o client_transfer_number OPTIONAL. This is the transfer identification number chosen by the client. This number is meaningful only the client.

- o client_signature REQUIRED. The digital signature of the client..

9. Security Consideration

(TBD)

10. IANA Consideration

(TBD)

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <https://www.rfc-editor.org/info/rfc2234>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

11.2. Informative References

[Arch] Hardjono, T., Hargreaves, M., and N. Smith, "An Interoperability Architecture for Blockchain Gateways. draft-hardjono-blockchain-interop-arch-02", April 2021, <https://datatracker.ietf.org/doc/draft-hardjono-blockchain-interop-arch/>.

[HS2019] Hardjono, T. and N. Smith, "Decentralized Trusted Computing Base for Blockchain Infrastructure Security, Frontiers Journal, Sepcial Issue on Blockchain Technology, Vol. 2, No. 24", December 2019, <https://doi.org/10.3389/fbloc.2019.00024>.

[NIST] Yaga, D., Mell, P., Roby, N., and K. Scarfone, "NIST Blockchain Technology Overview (NISTR-8202)", October 2018, <https://doi.org/10.6028/NIST.IR.8202>.

[ODAP-2PC]
Belchior, R., Correia, M., and T. Hardjono, "Gateway Crash Recovery Mechanism. draft-belchior-gateway-recovery-01", March 2021, <https://datatracker.ietf.org/doc/draft-belchior-gateway-recovery/>.

[RFC5939] Andreasen, F., "Session Description Protocol (SDP) Capability Negotiation", RFC 5939, DOI 10.17487/RFC5939, September 2010, <https://www.rfc-editor.org/info/rfc5939>.

Authors' Addresses

Martin Hargreaves
Quant Network

Email: martin.hargreaves@quant.network

Thomas Hardjono
MIT

Email: hardjono@mit.edu

Rafael Belchior
Technico Lisboa

Email: rafael.belchior@tecnico.ulisboa.pt