# OpenVASP: An Open Protocol to Implement FATF's Travel Rule for Virtual Assets

David Riegelnig, Bitcoin Suisse

November 14, 2019

Summary

In June 2019, the Financial Action Task Force (FATF) issued updated guidance on "Virtual Assets and Virtual Asset Service Providers" [1], providing additional clarifications on how its recommendations should be understood in the context of virtual asset transactions. Implementation of the newest recommendations is under way in some countries[2], yet for most FATF members, awareness of the requirement is still in the very early stages.

The FATF has been monitoring and providing guidance on money laundering and terrorism financing prevention since 2014. In 2018, the international body adopted changes to its recommendations to define "virtual assets" and introduced the term "virtual asset service provider" (VASP). Yet, the implementation of some of the most recent guidance is a challenge for the still nascent crypto-financial industry. Recommendation 16, often referred to as "Travel Rule", presents particular challenges, requiring industry participants to agree on common standards. It requires any VASP to obtain, hold, and transmit originator and beneficiary information when conducting virtual asset transactions with obliged entities as defined by the FATF (other VASPs, banks and financial intermediaries).

This white paper outlines an open protocol among VASPs for the mutual exchange of originator and beneficiary information. It does so in a fully decentralized manner, leveraging cryptographically secure peer-to-peer communication and capabilities of the Ethereum blockchain for authentication. The protocol works with any blockchain or distributed ledger technology (DLT) used for the underlying virtual asset transfer. It puts privacy of transferred data at the center of its design.

The protocol facilitates robust compliance, purely based on a set of principles for VASPs around the world, regardless of jurisdiction or technology used and without membership or registration with a centralized third-party. It is meant as a contribution to the ongoing dialogue of members of the crypto-financial community and regulators around the globe.

[1] http://www.fatf-gafi.org/publications/fatfrecommendations/documents/public-statement-virtual-assets.html

[2] E.g. in Switzerland: Anti-Money Laundering Ordinance (AMLO-FINMA), issued by Swiss Financial Market Supervisory Authority (FINMA), in particular Article 10; FINMA Guidance 02/2019 Payments on the blockchain

## Contents

# 1 Overview

## Design Principles (Chapter 2)

An outline of the seven design principles guiding the protocol's design.

## Messaging Layer (Chapter 3)

Requirements are defined for the underlying asynchronous communication layer so that the protocol can possibly be implemented on top of different existing messaging systems. The Whisper protocol is introduced as an exemplary peer-to-peer messaging layer.

## Addressing (Chapter 4)

Making sure the protocol works with any blockchain or distributed ledger system used for the underlying virtual asset transfer, including upcoming Layer 2 solutions, a generic addressing and routing system is required. We introduce the Virtual Assets Account Number (VAAN), which borrows elements from proven payment routing systems (such as the IBAN), without compromising on the decentralized approach.

## Authentication (Chapter 5)

Robust end-to-end encryption requires strong mutual authentication between VASPs in the first place. The protocol leverages the Ethereum blockchain as a public key infrastructure, avoiding the known weaknesses of central certificate authorities. Mutual authentication between VASPs is at the center of the protocol's trust model, complemented by attestations from trusted third parties like self-regulatory organizations and trade associations.

## Protocol Flow (Chapter 6)

The protocol defines a simple flow of steps to facilitate a structured communication between two VASPs transferring virtual assets on behalf of their respective customers.

## Protocol Messages (Chapter 7)

A set of structured messages is specified, which borrows proven elements from modern payment standards (i.e. ISO 20022), tailored for the specific needs arising for VASPs.

# 2 Design Principles

In designing this proposal for an open protocol to facilitate compliance with FATF's travel rule for virtual assets, we were guided by the following seven principles.

## 2.1 Travel Rule Compliance

Establish a shared communication protocol for VASPs to exchange virtual asset transfer information as specified in the FATF requirements.

This principle includes:

a) a common transfer data standard for required originator and beneficiary information;

b) a suitable set of rules to facilitate the data exchange between VASPs.

## 2.2 Decentralized Approach

Pursue a decentralized approach that enables any two VASPs to use the protocol without consent or even knowledge of any third party.

This principle includes that the protocol:

a) does not require a VASP to obtain any form of membership or registration with any third party;

b) does not require the usage of a central component at any time;

c) assumes that each VASP has a non-delegable obligation to carefully select other VASPs it wants to work with, following a risk-based approach.

## 2.3 Technology Agnostic

Make sure the protocol works with any blockchain or distributed ledger technology (DLT) used for the underlying virtual asset transfer.

This principle includes that the protocol:

a) requires no changes to the underlying blockchain / DLT;

b) does not assume specific characteristics of the underlying blockchain / DLT (e.g. the existence of a unique identifier or comment field in transactions).

## 2.4 Privacy by Design

Make sure the protocol puts privacy of transferred data at the center of its design.

This principle includes that the protocol:

a) requires robust authentication of the VASPs involved;

b) requires robust end-to-end encryption between VASPs;

c) applies perfect forward secrecy (protecting data transferred in the past against future compromises of the private keys);

d) allows for two VASPs to transfer data without the knowledge of any third party.

## 2.5 Broad Applicability

Ensure that the protocol facilitates all applicable usage scenarios where VASPs need to exchange transfer data.

This principle includes that the protocol:

a) supports VASPs exchanging data as part of a one-off virtual asset transfer;

b) supports VASPs exchanging data for a high number of routine transactions;

c) supports situations where the beneficiary VASP is not known to the sending VASP or where there is uncertainty whether a target address is controlled by a VASP;

d) supports situations where virtual asset transfers between VASPs are initiated or facilitated by smart contracts.

## 2.6 Extensibility

Ensure that the protocol allows for custom extensions.

This principle includes that the protocol:

a) allows for VASPs to add custom data;

b) while doing so, has rules that prevent any weakening of the common core.

## 2.7 Efficient to Use

Ensure minimal cost for VASPs to deploy and maintain the protocol's implementation.

This principle includes that the protocol:

a) supports straight-through processing capabilities;

b) supports implementations where a single server instance can process virtual asset transfers in all blockchains / DLTs used by the VASP.

# 3 Messaging Layer

## 3.1 Generic requirements

On the underlying communication layer, the protocol requires to send and receive asynchronous messages in simple datagram format.

[ Header, Data ]

| | |
|---|---|
| Header: | Addressing information & reference for receiver how to decrypt the data (4 bytes) |
| Data: | Encrypted message content |

The proposed architecture does not assume a secure connection between sender and receiver and is therefore not requiring the messaging layer to handle authentication. Instead, the protocol defines a communication handshake based on structured messages, where the session key used for subsequent encryption is generated.

Addressing is based on the VASP's unique identity (for initial handshake) and the session (for subsequent messages). Again, no specific requirements are imposed on the messaging layer, allowing for different approaches how routing is implemented.

Details on addressing, session handling and message structure are described in further chapters.

## 3.2 Range of possible underlying messaging protocols

Given these minimal requirements, a wide range of underlying messaging protocols could be used, which is helpful in supporting a wide range of usage scenarios.

To illustrate this, one could utilize standard unencrypted email to exchange protocol messages, simply using the subject field for header information and the email body for data. Doing so is not very efficient and would be vulnerable to traffic analysis, albeit only regarding the identity of the sender and the receiver, but it would still work. Naturally, any other direct connection between sender and receiver being able to exchange messages specified above can be used.

However, in line with its decentralized approach, the protocol is designed to work with a peer-to-peer communication layer as well. In such a system, each node passes on every message it receives to some (gossiping) or all (flooding) of its neighbors. Routing works because the receiving node can decide based on the message header, whether he is the addressee and then try to decrypt the message data with his private key or a previously agreed shared key.

Ideally, a set of initially supported messaging layers will be part of the final protocol specification, following broader discussion with industry participants. For this paper, Whisper is taken as an exemplary peer-to-peer protocol.

## 3.2 Ethereum Whisper as an example

The Whisper messaging system has been developed as part of the Ethereum technology stack but can be used separately from the blockchain. It was designed to provide resilience and strong privacy in a peer-to-peer environment and targets message delivery below five seconds on a global scale.

Given the design goals for the OpenVASP protocol, Whisper seems a good fit with its focus on secure, untraceable communication between peers. It benefits from using Ethereum's proven network layer and is available for all to use. Adoption will be facilitated for VASPs already running Ethereum nodes.

The reader is recommended to consult the online documentation about the Whisper protocol (https://geth.ethereum.org/docs/whisper/whisper-overview). Hence, the following sections provide just a brief overview about Whisper and its key functionality, partially citing from the mentioned source.

### 3.2.1 Functionality

#### *3.2.1.1 Resilience and Privacy*

Whisper has been designed as a peer-to-peer messaging system with focus on resilience and privacy. As a matter of principle, a Whisper message is sent to every Whisper node and every message is encrypted by default. The one who can decrypt the message, is the intended recipient. That way, Whisper can ideally provide absolute anonymity for the recipient. However, nodes can choose their level of anonymity by leaking limited information for better performance if so desired.

As each node passes on all incoming messages, neighboring nodes learn nothing about the origin of a message. In addition, a node can send random messages to maintain a constant level of “noise”. Therefore, Whisper protects the sender as well, even in case of a powerful adversary attempting to do targeted traffic analysis.

#### *3.2.1.2 Dark Routing*

Decrypting every incoming message would put a too heavy strain on computational resources. For this reason, Whisper messages include a 4-byte “topic”, which is a probabilistic hint for the recipient to watch out for and to try to decrypt. The receiver can subscribe to several topics along with the respective symmetric and asymmetric key.

The occurrence of collisions, meaning that a receiver attempts to decrypt a message with a matching topic to find out it cannot be decrypted as it was for another receiver, are deliberate to ensure plausible deniability.

#### *3.2.1.3 Latency*

It has been designed for message delivery below five seconds.

#### *3.2.1.4 Time-to-live (TTL)*

When sending a Whisper message, the sender decides on a time-to-live (specified in seconds) with a maximum time span of two days. Messages are kept in every Whisper node until their TTL expires. When a new node connects, it will receive all messages whose TTL has not yet expired. The Whisper network is therefore storing a message for a specific time, which is useful in situations when a node has a short downtime.

#### *3.2.1.5 Protection from denial-of-service attacks*

To protect the Whisper network from denial-of-service attacks, sending a message requires a proof-of-work to be accepted by any forwarding node. The required proof-of-work is proportional to both message size and TTL.

#### *3.2.1.6 Encryption*

Every Whisper message is encrypted either symmetrically or asymmetrically. Messages can be decrypted by anyone who possesses the corresponding key.

Asymmetric encryption uses the standard Elliptic Curve Integrated Encryption Scheme with SECP-256k1 public key. Symmetric encryption uses AES GCM algorithm with random 96-bit nonce.

#### *3.2.1.7 Underlying transport layer*

Whisper is implemented on top of the RLPx, a TCP-based transport protocol also used for communication among Ethereum nodes.

### 3.2.2 Whisper envelopes and messages

Envelopes are the packets sent and received by Whisper nodes. They contain the encrypted message and metadata for decryption. Once decoded, they have the following format:

[ Version, Expiry, TTL, Topic, AESNonce, Data, EnvNonce ]

| | |
|---|---|
| Expiry time: | 4 bytes (UNIX time in seconds). |
| TTL: | 4 bytes (time-to-live in seconds). |
| Topic: | 4 bytes of arbitrary data. |
| AESNonce: | 12 bytes of random data (only present in case of symmetric encryption). |
| Data: | byte array of arbitrary size (contains encrypted message). |
| EnvNonce: | 8 bytes of arbitrary data (used for PoW calculation). |

Upon receipt of a message, if the node detects a known topic, it tries to decrypt the message with the corresponding symmetric and asymmetric key. In case of failure, the node assumes that topic collision occurs, e.g. the message was encrypted with another key. If decryption is successful, the message is revealed with the following structure:

| | |
|---|---|
| Flags: | 1 byte |
| Padding: | byte array of arbitrary size, optional |
| Message: | byte array of arbitrary size (actual message content) |
| Signature: | 65 bytes, optional, Ethereum ECDSA signature (Keccak-256 hash of unencrypted data) |

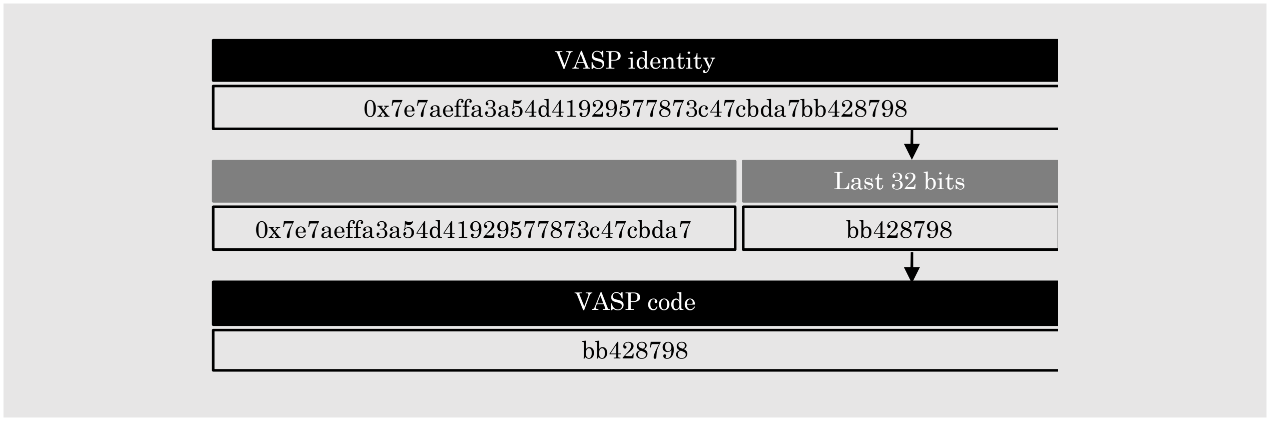
Those unable to decrypt the message data are also unable to access the signature.

# 4 Addressing

## 4.1 VASP identity & VASP code

The protocol uses the Ethereum blockchain as a decentralized public-key infrastructure (see next chapter). Each participating VASP must deploy a standardized smart contract, which represents its identity on the blockchain, similar to how a traditional public key certificate would function.

The Ethereum address of the standardized smart contract deployed by a VASP is defined to be the VASP identity and the last 32 bits are called VASP code. Both values are numbers encoded as hexadecimals, which are easy to process and human-readable. However, the “0x” radix, typically used to indicate the hexadecimal format, is omitted for the VASP code.


VASP identity
0x7e7aeffa3a54d41929577873c47cbda7bb428798
Last 32 bits
0x7e7aeffa3a54d41929577873c47cbda7
bb428798
VASP code
bb428798

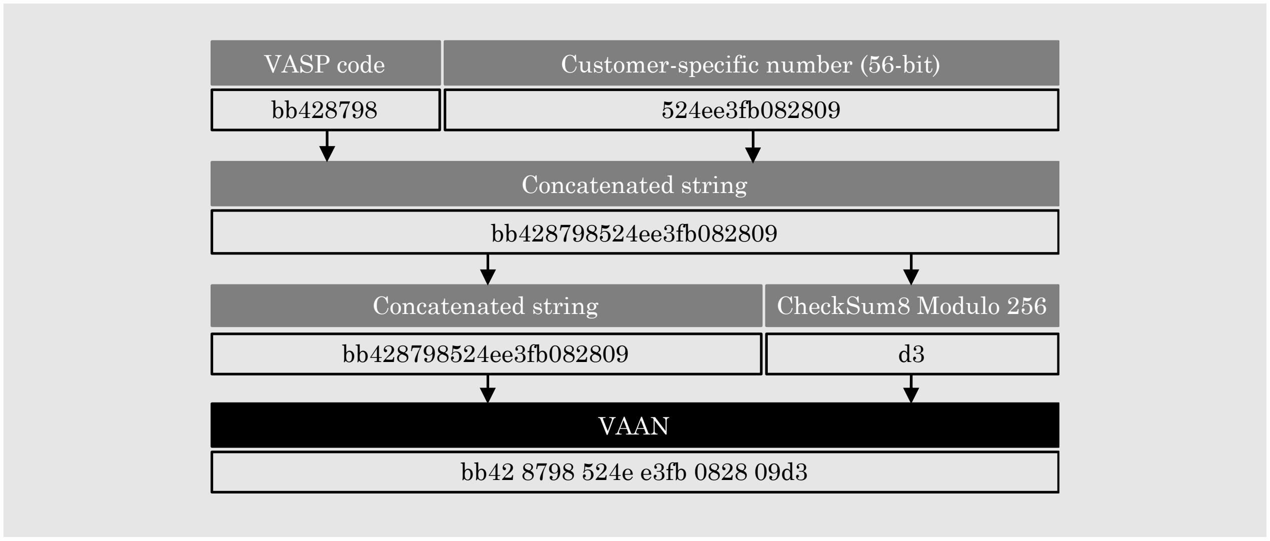
## 4.2 Virtual Assets Account Number (VAAN)

At the outset, the beneficiary wants to receive virtual assets on a wallet hosted by a VASP and therefore provides the originator with routing information on where to send them. The originator then instructs his VASP to transfer the virtual assets based on the routing information.

In traditional payment systems, bank account numbers in combination with bank identifiers (e.g. BIC/SWIFT, IBAN) are used as routing information.

We suggest a similar, but decentralized approach in form of the Virtual Assets Account Number (VAAN), which is a 24-character hex code including a 2-character checksum. The leading eight characters correspond to the VASP code, while the remaining characters are customer specific.

The VAAN should not contain spaces when transmitted electronically. When printed, it can be expressed in groups of four characters separated by a single space for readability.



It is at the discretion of each VASP to define the details of how VAANs are assigned and used by their customers. The proposed length of 14 hex characters (56 bits) for the client-specific number of the VAAN gives each VASP a sufficiently large address space of $7.2 \times 10^{16}$ identifiers, allowing plenty of room including vanity numbers.

A VASP could assign a single, fixed VAAN to each customer by default, similar to how a traditional bank account number is used. Another VASP might provide multiple VAANs on request or limit their period of validity. Privacy-minded customers might even prefer to use a VAAN only once.

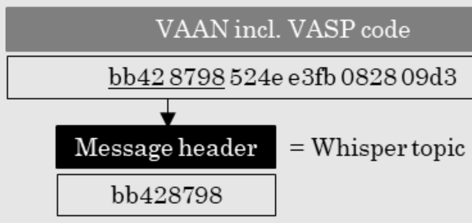
While VASPs are completely free in assigning VAANs to their customers within the boundaries of the format, they must ensure that each VAAN is uniquely assigned to exactly one of their customers. By doing so, the regulatory requirement for unique identification is ensured.

The proposed format allows the VAAN to be exchanged easily (e.g. via email) between beneficiary and originator and facilitates VASP-to-VASP transfers of virtual assets beyond the purpose of FATF compliance.

## 4.3 VAAN-based message routing

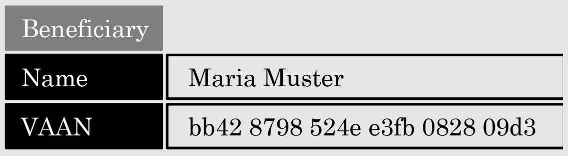
The VASP code, being the first 8 characters of the VAAN, is used as the header to send messages to the respective VASP.

In case of Whisper, the header (VASP code) can directly be used as topic to route message envelopes to the receiver (see previous chapter).



## 4.4 Transfer instructions to be given by a customer

From a customers' perspective, only the name and the VAAN have to be provided to receive virtual assets to his/her wallet hosted by a VASP.



| Beneficiary | |
|---|---|
| Name | Maria Muster |
| VAAN | bb42 8798 524e e3fb 0828 09d3 |

# 5. Authentication

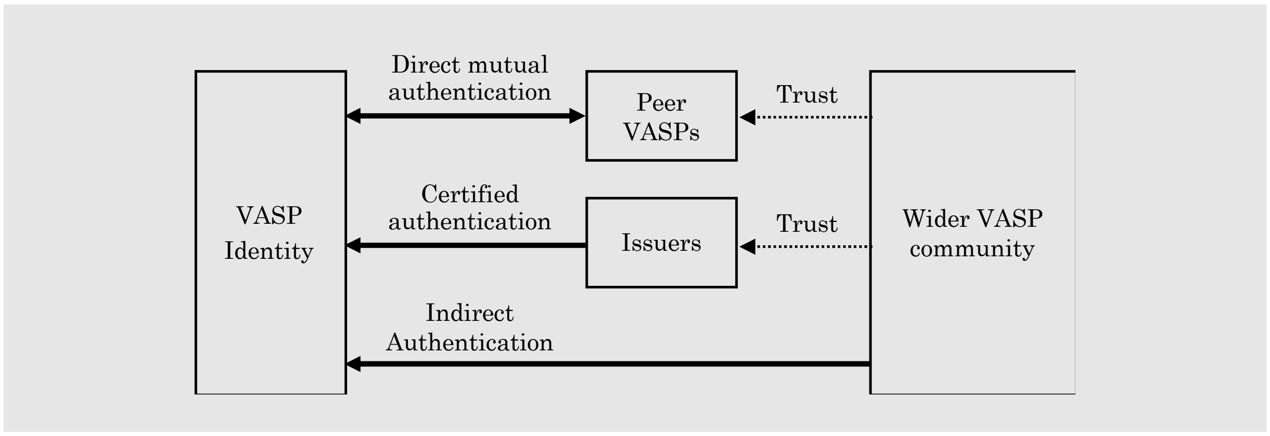
A standardized Ethereum smart contract (called VASP contract) is used to represent the identity of a VASP. Each VASP deploys its own contract and can also withdraw it at any time. As described in the previous chapter (section 4.1), the Ethereum address of the contract is defined to be the VASP identity.

## 5.1 Direct and indirect authentication among VASPs

Direct mutual authentication between counterparty VASPs forms the basis of the protocol's trust model. Whenever two VASPs establish their business relationship, their respective identities can be directly authenticated, meaning first-hand evidence is available that the other VASP's identity is genuine. For significant VASP-to-VASP relationships with virtual asset transfers occurring regularly (e.g. broker/exchange, bank/custodian) authenticating each other's identity will happen as part of the usual onboarding process between financial intermediaries. While there is no requirement to make this step public, most VASPs will find visible recognition from reputable counterparties to be a sign of good standing.

Certified authentication by trusted third parties is the protocol's second approach to ensure robust authentication. Recognized self-regulatory organizations and trade associations are particularly suitable to issue identity claims for their respective members. Claims by such issuers can directly include an attestation about the VASP's license or registration status obtained from competent authorities.

Identity claims made by either peer VASPs or trusted issuers are signed and recorded in the VASP contract for everyone's reference. Claims can also be revoked by the issuing party at any time.



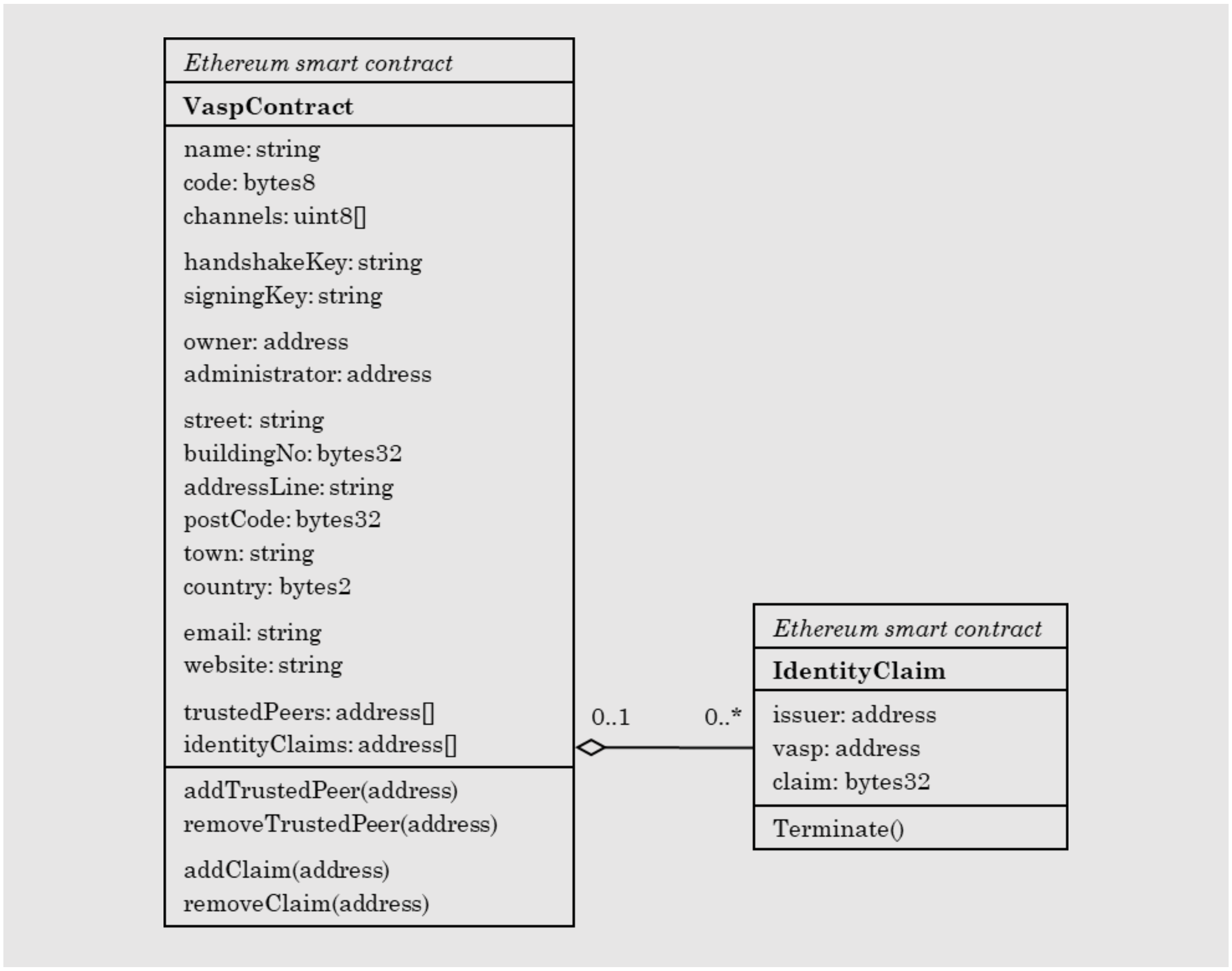
The ability to retrieve proven identity claims about a VASP on the blockchain at any time, enables the wider community of VASPs to partially rely on indirect trust relationships. While risk-based decision making is required, indirect authentication will be often feasible for one-time transfers of moderate value.

## 5.2 VASP contract

A high-level specification of the VASP contract is outlined in this section, focusing on the most important aspects required for the functioning of the protocol.

As mentioned earlier, each VASP deploys its own contract instance and takes the contracts' deployment address as identity identifier and the last 32 bits as the VASP code. The contract includes relevant information about the VASP.

For better security and to facilitate separation of duties, different roles should be implemented (e.g. owner, administrator). Addresses assigned to these roles can again point to multi-sig smart contracts for better security. However, the VASP contract itself should be as simple as possible and delegate access control and governance to calling contracts and systems.



### 5.2.1 VASP contract attributes (selection)

| | |
|---|---|
| name | Legal name of the VASP |
| code | Last 32 bit of VASP contract address, used as abbreviation for VASP identity |
| channels | Communication channels the VASP is accepting for messages (e.g. Whisper, Email, etc.) |
| handshakeKey | Asymmetric public key used to securely establish sessions |
| signingKey | Asymmetric public key used to verify message signatures |
| owner | Address assigned as the owner for the VASP contract |
| administrator | Address assigned to change smart contract attributes |
| *postal address* | Attributes for the VASP's postal address, in line with section 7.10 |
| *email, website* | Contact information |
| trustedPeers | Trusted peer VASPs |
| identityClaims | Identity claims made by trusted issuers |

### 5.2.2 VASP contract methods (selection)

| | |
|---|---|
| addTrustedPeer(address) | Adds peer VASP to the set of trusted peers. Must be called by the administrator. |
| removeTrustedPeer(address) | Removes peer VASP from the set of trusted peers. Must be called by the administrator. |
| addClaim(address) | Adds address of an identity claim contract, issued by a trusted third party, to the set of accepted claims. Must be called by the administrator. |
| removeClaim(address) | Removes address of an identity claim contract from the set of accepted claims. Must be called by the administrator. |

## 5.3 Claim contract

While mutual trust between VASPs can easily be recorded by reciprocal entries in their VASP contracts, identity claims made by third party issuers require a different setup. We propose the usage of a simple claim contract as part of the OpenVASP protocol.

Deployed by the issuer, the claim contract allows to specify the content and potential qualifications of the claim made about the VASP. Since the contract remains under control of the issuer, it can be revoked (terminated) at any time.

VASPs accept a claim made about them by adding the respective claim contract address. If for whatever reason necessary, claims can be repudiated by again removing the reference.

## 5.4 Registering the VASP code in the Ethereum Name Service (ENS)

The Ethereum Name Service (ENS) allows to use short, human-readable names as pointers to otherwise long Ethereum addresses. Its logic and governance are completely decentralized, operating solely as a set of smart contracts. The reader can find more information under https://ens.domains/.

As described in chapter 4, the VASP code is an abbreviation of the VASP contract address (VASP identity) and serves as prefix in the proposed VAAN format. Alternatively, the full contract address could have been used to uniquely bind a VAAN to the VASP identity on the blockchain. Using the shorter VASP code is obviously more convenient for everyday handling. However, it introduces the risk that an imposter could try to set up a smart contract with the same abbreviation as an existing VASP. Other VASPs may then be deceived into interacting with the wrong counterparty.

In order to mitigate this risk, VASPs must register their VASP code as an ENS domain and point it to their VASP contract. Taking the example from section 4.1, the VASP would register bb428798.eth and point it to the address 0x7e7aeffa3a54d41929577873c47cbda7bb428798.

The probability for a new VASP contract address to be the same as an existing one, is relatively low with one in $4.29 \times 10^9$. Nevertheless, if a collision was to happen, it would immediately be detected as the corresponding ENS domain would no longer be available for registration. In such as case, a simple redeployment of the contract generates a new contract address.

Besides providing security against imposters and accidental collisions, using the VASP code as ENS domain brings convenience as any VASP's identity can be quickly retrieved by the 8-character code.

## 5.5 Communication channels offered

As described in chapter 3, different communication channels might be used to exchange protocol messages, and each VASP could decide which cannels to offer. Available options can be retrieved from the *channels* attribute of the VASP contract.

## 5.6 Summary: Steps to authenticate a VASP

Whenever a counterparty VASP must be authenticated for the first time, the following steps are required to verify that a valid VASP identity is available.

1) Check if the 8-character VASP code, followed by the .eth domain suffix, resolves to the VASP contract address.
2) Check if the standardized VASP contract was deployed and used as VASP identity.
3) Check whether key attributes are properly filled out in the VASP contract.
4) Check which of the VASPs listed as trusted peers in the VASP contract have reciprocally listed the counterparty VASP as a trusted peer.
5) Review available identity claims issued by trusted third parties and verify their authenticity (e.g. by checking whether the known third party public address was used to set up the identity claim contract).

Steps 1) to 3) are formal requirements and must be concluded with a positive result.

Information obtained in steps 4) and 5) require an assessment whether claims made by trusted peers and third parties is deemed to be sufficient for indirect authentication. If this is not the case, the VASP must be directly authenticated by obtaining first-hand evidence that the identity is genuine, e.g. by confirming the VASP identity via personal contact.

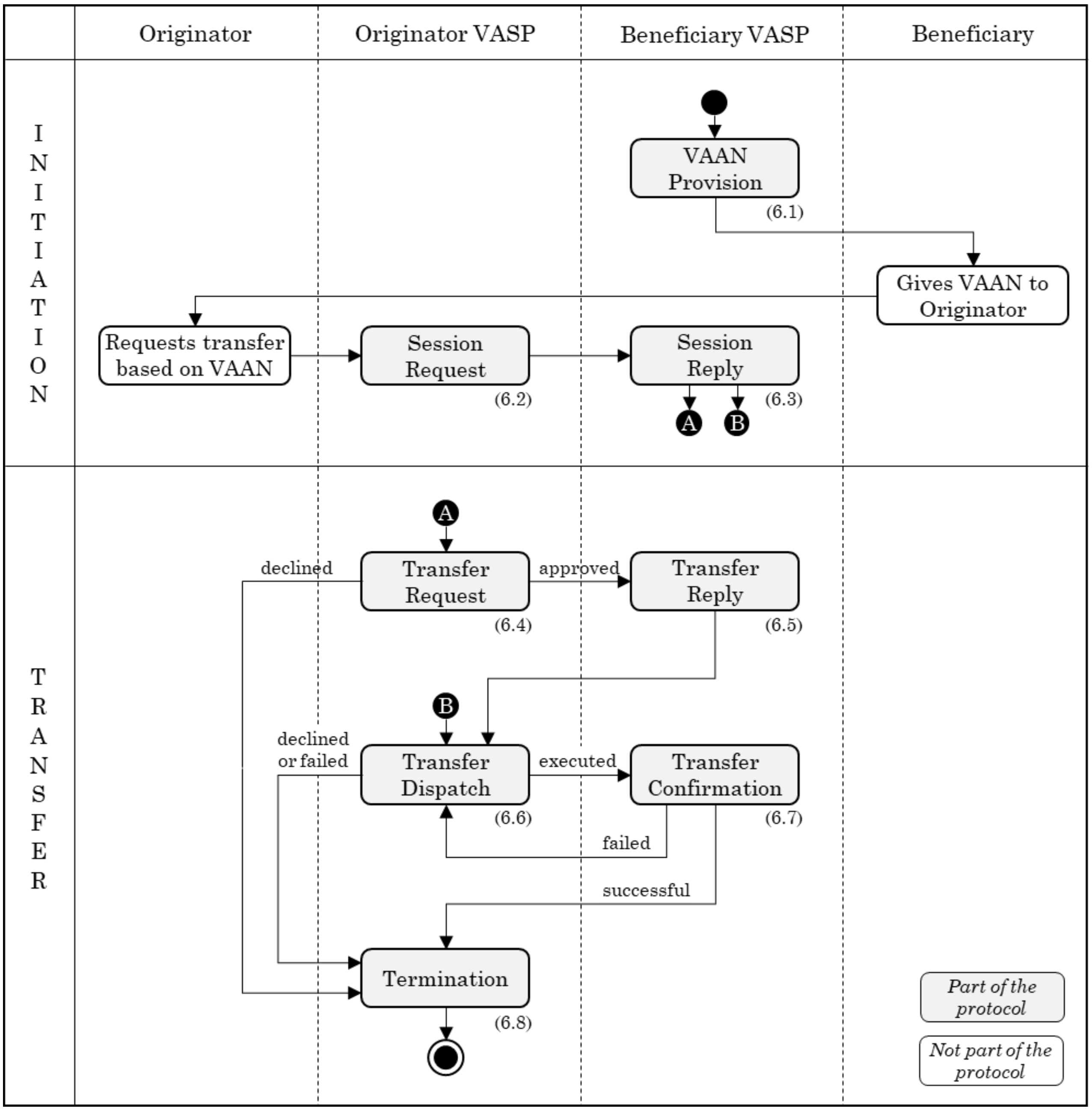
The above mentioned steps can largely be automated with exception of the final decision whether indirect authentication is enough to proceed.

# 6 Protocol Flow

The protocol provides a set of requests and responses to facilitate a structured communication between the originator VASP and the beneficiary VASP transferring virtual assets on behalf of their respective customers.

At the outset, the beneficiary wants to receive virtual assets on a wallet hosted by the beneficiary VASP and therefore provides the originator with routing information on where to send them. The originator then instructs the originator VASP to transfer the virtual assets.

In the initiation phase, the involved VASPs establish communication and mutually authenticate and authorize each other. In transfer phase, originator/beneficiary information is exchanged, the transfer gets mutually approved and execution on the blockchain is notified and confirmed.



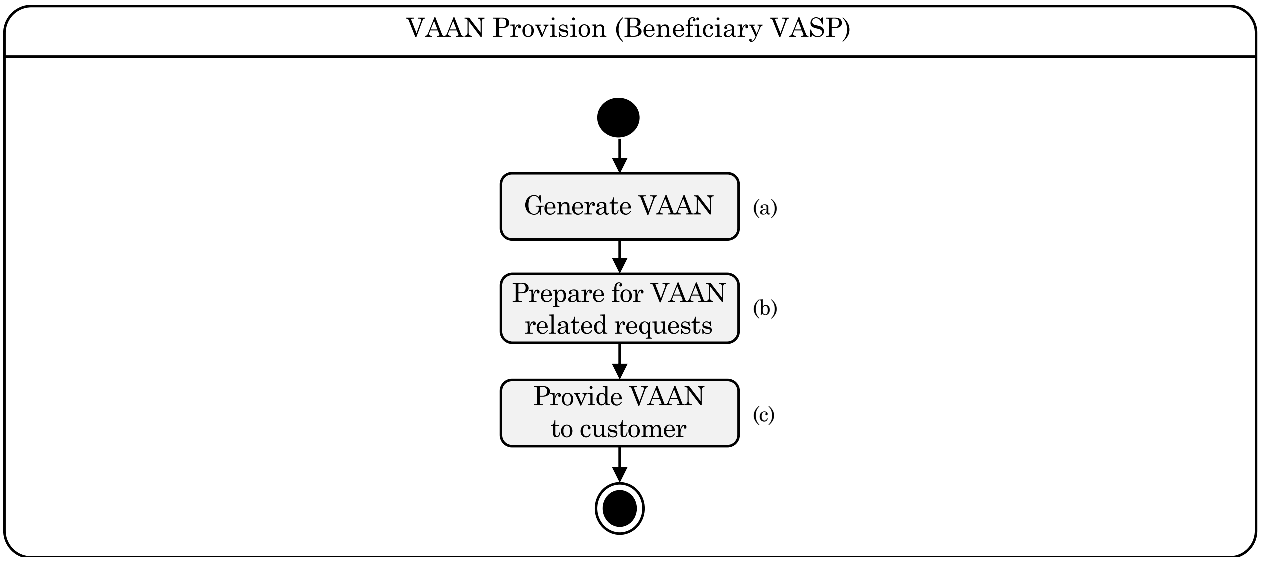
For the transfer phase, the protocol supports two different policies how a beneficiary VASP accepts virtual assets on behalf of their customers:

A) **Approval required before blockchain transaction**
Given the compliance obligations, the beneficiary VASP will generally want to approve a virtual asset transfer based on originator and beneficiary information before the actual blockchain transaction is executed. In addition, the beneficiary VASP can specify the destination address on the blockchain to be used for this transfer.

B) **Approval NOT required before blockchain transaction**
For efficiency reasons the VASPS may have agreed that an approval before the actual blockchain transaction is not required for certain transfers (e.g. transfers between two specific customers). As no message is exchanged before the blockchain transaction, either always the same destination address is used, or the originator VASP knows the destination address from a different source (e.g. using deterministic addresses based on a shared master public key). Such a practice can make sense when the VASPs regularly process numerous transactions for the same originator and beneficiary, e.g. as part of an automated setup.

The following sections provide more details about each of the protocol steps.

## 6.1 VAAN Provision


VAAN Provision (Beneficiary VASP)
Generate VAAN
(a)
Prepare for VAAN related requests
(b)
Provide VAAN to customer
(c)

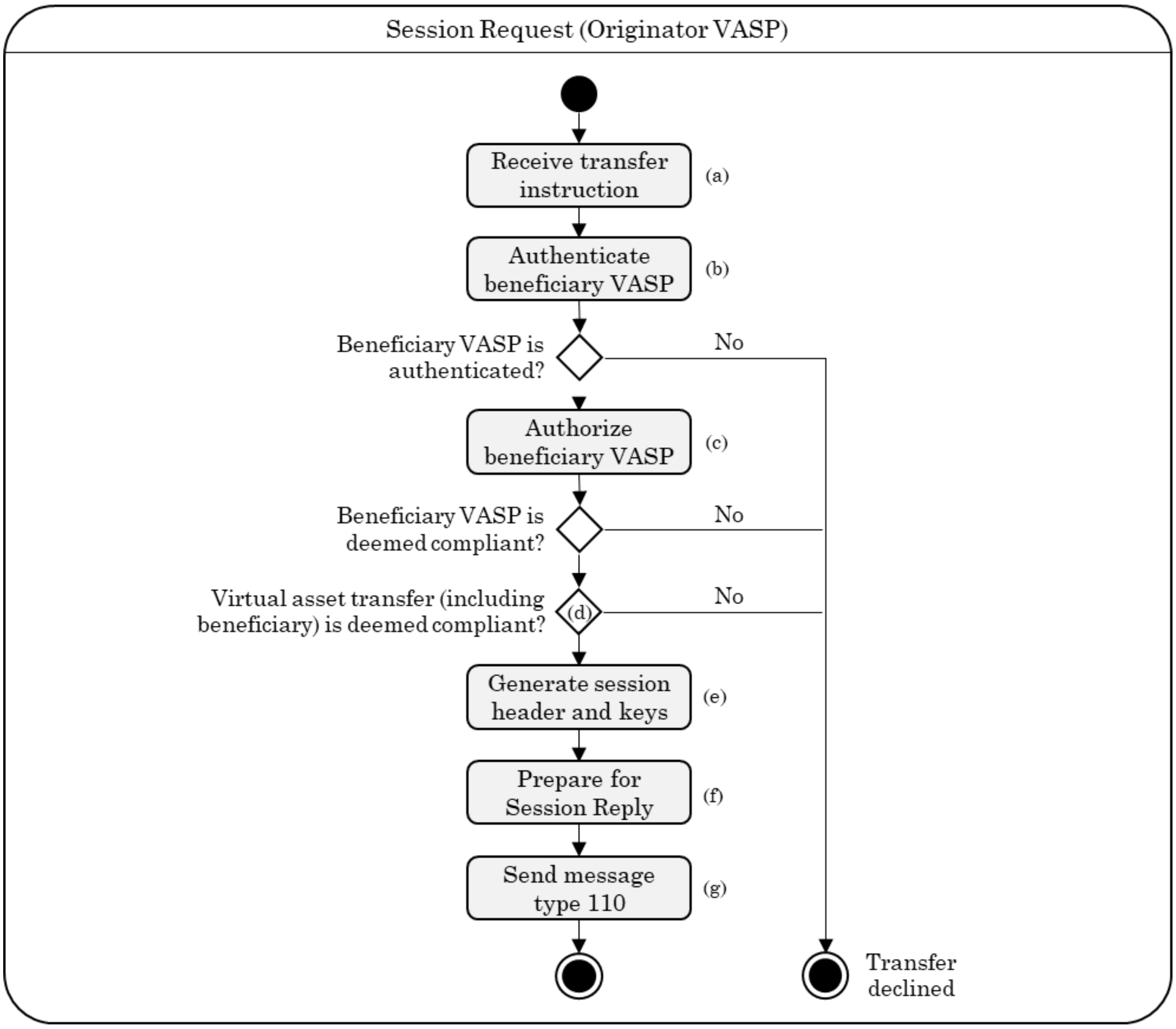
### 6.1.1 Processing steps by the beneficiary VASP

a) Generate a customer specific VAAN based on the steps described in section 4.2.

b) Ensure that incoming messages containing the VASP code as header can be received and processed (e.g. that the Whisper node is set up) and prepare internal systems and procedures to receive and process protocol messages related to the VAAN generated in the previous step.

c) Provide the VAAN to the customer who can use it as routing information to receive virtual assets to the hosted wallet.

## 6.2 Session Request


Session Request (Originator VASP)
Receive transfer instruction
(a)
Authenticate beneficiary VASP
(b)
Beneficiary VASP is authenticated?
No
Authorize beneficiary VASP
(c)
Beneficiary VASP is deemed compliant?
No
Virtual asset transfer (including beneficiary) is deemed compliant?
(d)
No
Generate session header and keys
(e)
Prepare for Session Reply
(f)
Send message type 110
(g)
Transfer declined

### 6.2.1 Processing steps by the originator VASP

a) Receive instruction from customer to transfer virtual asset to a wallet hosted with another VASP, specified by name and VAAN (see section 4.2) of the beneficiary.

b) Identify and authenticate beneficiary VASP specified in the VAAN. See section 6.9 for authentication steps.

c) Once successfully authenticated, authorize the beneficiary VASP (see section 6.10).

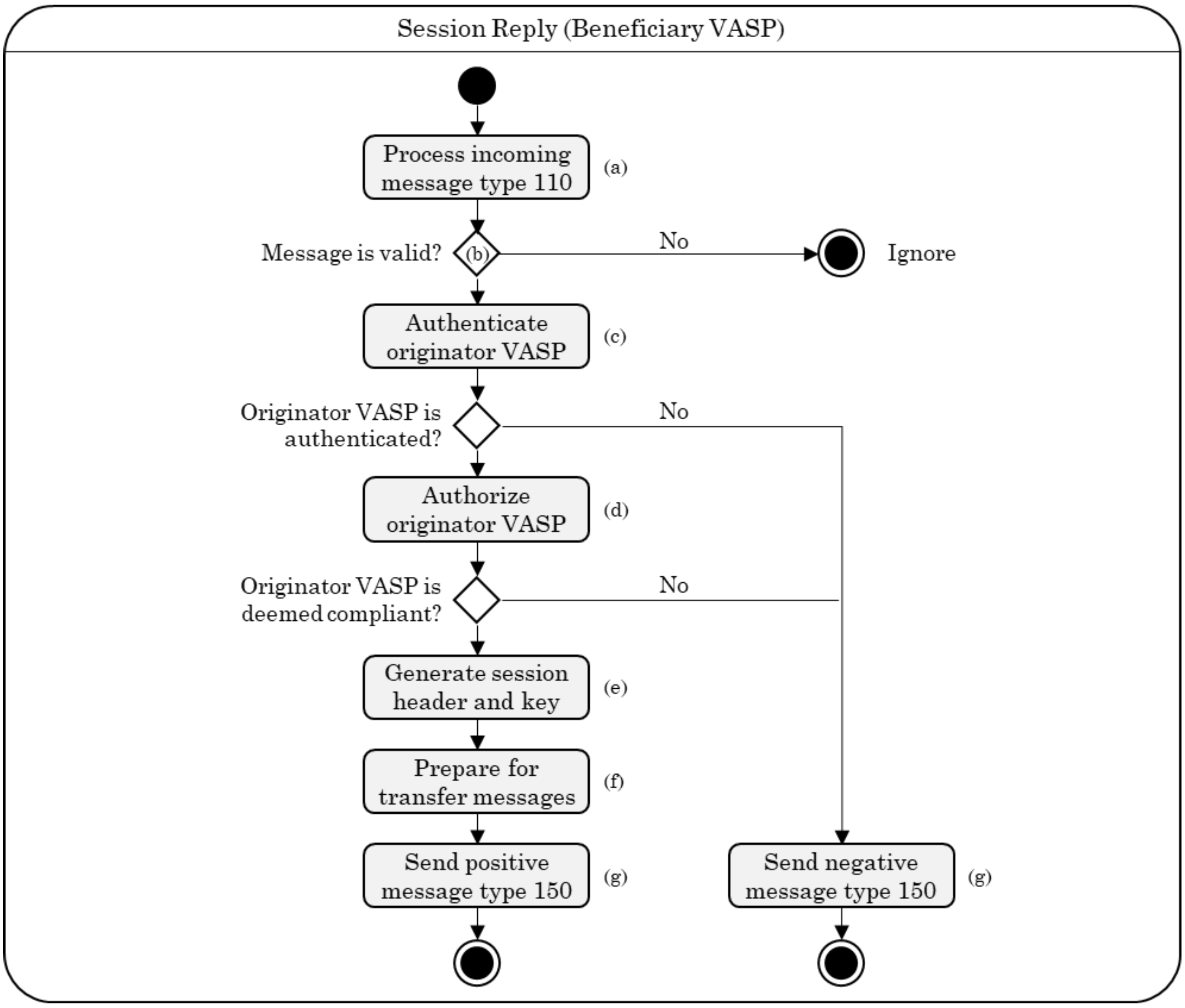
d) Perform a risk-based decision whether a virtual asset transfer can be executed, based on all relevant information including information about the originator, the beneficiary, the beneficiary VASP, the virtual asset type and the amount of the requested transfer.

e) Generate a random header (topic A) to be used in this session. Generate private and public keys based on the elliptic curve Diffie–Hellman (ECDH) key-exchange protocol to use for ephemeral encryption in transfer phase. Derive shared key based on the ECDH public key from the beneficiary VASP (handshakeKey in VASP contract) and own ECDH public key generated in the previous step.

f) Prepare systems to receive the VARC Reply message from the beneficiary VASP (e.g. in the Whisper node, subscribe to messages with topic A and being encrypted with the shared key).

g) Send Session Request message (type 110, see section 7.3) to the beneficiary VASP.

## 6.3 Session Reply


Session Reply (Beneficiary VASP)
Process incoming message type 110
(a)
Message is valid?
(b)
No
Ignore
Authenticate originator VASP
(c)
Originator VASP is authenticated?
No
Authorize originator VASP
(d)
Originator VASP is deemed compliant?
No
Generate session header and key
(e)
Prepare for transfer messages
(f)
Send positive message type 150
(g)
Send negative message type 150
(g)

### 6.3.1 Processing steps by the beneficiary VASP

a) Receive Session Request message (type 110), e.g. from the Whisper node.

b) Verify if incoming message contains mandatory data in valid format and is correctly signed.

c) Authenticate the originator VASP. See section 6.9 for authentication steps.

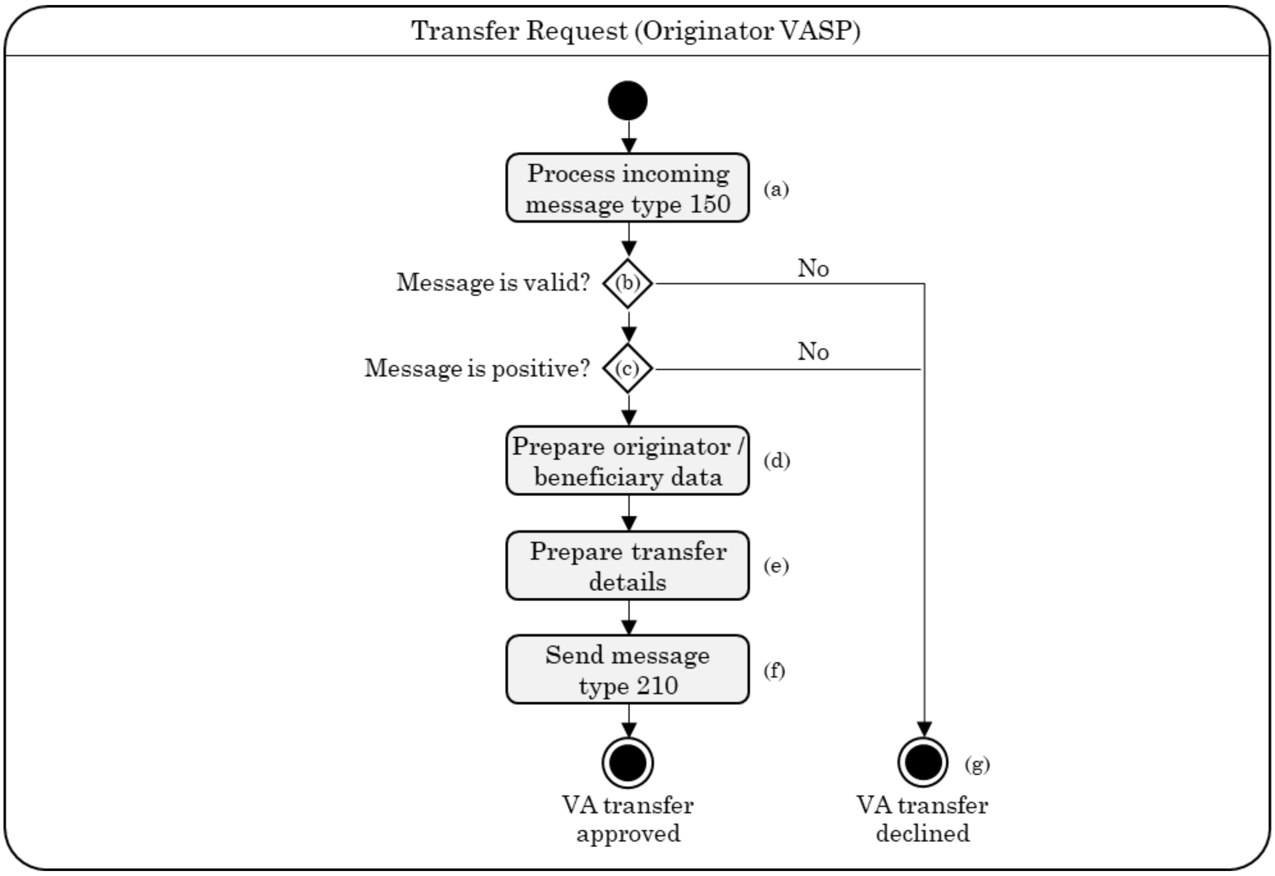
d) Once successfully authenticated, authorize the beneficiary VASP (see section 6.10).

e) Generate a random header (topic B) to be used in this session. Derive shared key based on the ECDH public key from the originator VASP (received in message) and own ECDH public key.

f) Prepare systems to receive subsequent messages from the originator VASP (e.g. in the Whisper node, subscribe to messages with topic B and being encrypted with the shared key).

g) Send Session Reply message (type 150, see section 7.4) to the originator VASP. In case of failed authentication (step c) or failed authorization (step d), a negative message is sent by providing the response code specifying the reason for failure.

## 6.4 Transfer Request



### 6.4.1 Processing steps by the originator VASP

a) Receive Session Reply message (type 150), e.g. from the Whisper node.

b) Verify if incoming message contains mandatory data in valid format and is correctly signed.

c) Check whether received message has a positive response code.

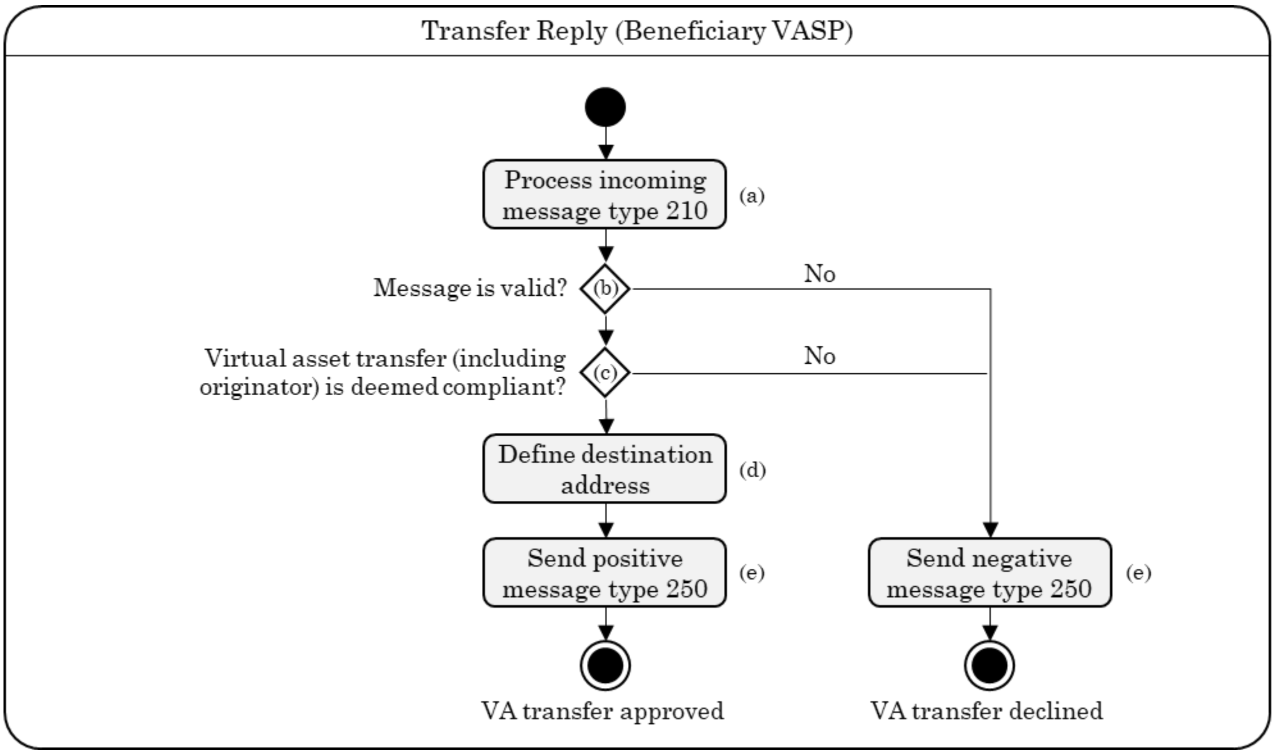
d) Prepare originator and beneficiary data (see sections 7.11 and 7.12 for details).

e) Prepare transfer session details (virtual asset type, transfer type and amount).

f) Send Transfer Request message (type 210, see section 7.5) to the beneficiary VASP.

g) In case of an invalid or negative incoming message (steps b or c), the session is terminated following the steps in section 6.6.

## 6.5 Transfer Reply


Transfer Reply (Beneficiary VASP)
Process incoming message type 210
(a)
Message is valid?
(b)
No
Virtual asset transfer (including originator) is deemed compliant?
(c)
No
Define destination address
(d)
Send positive message type 250
(e)
Send negative message type 250
(e)
VA transfer approved
VA transfer declined

### 6.5.1 Processing steps by the beneficiary VASP

a) Receive Transfer Request message (type 210), e.g. from the Whisper node.

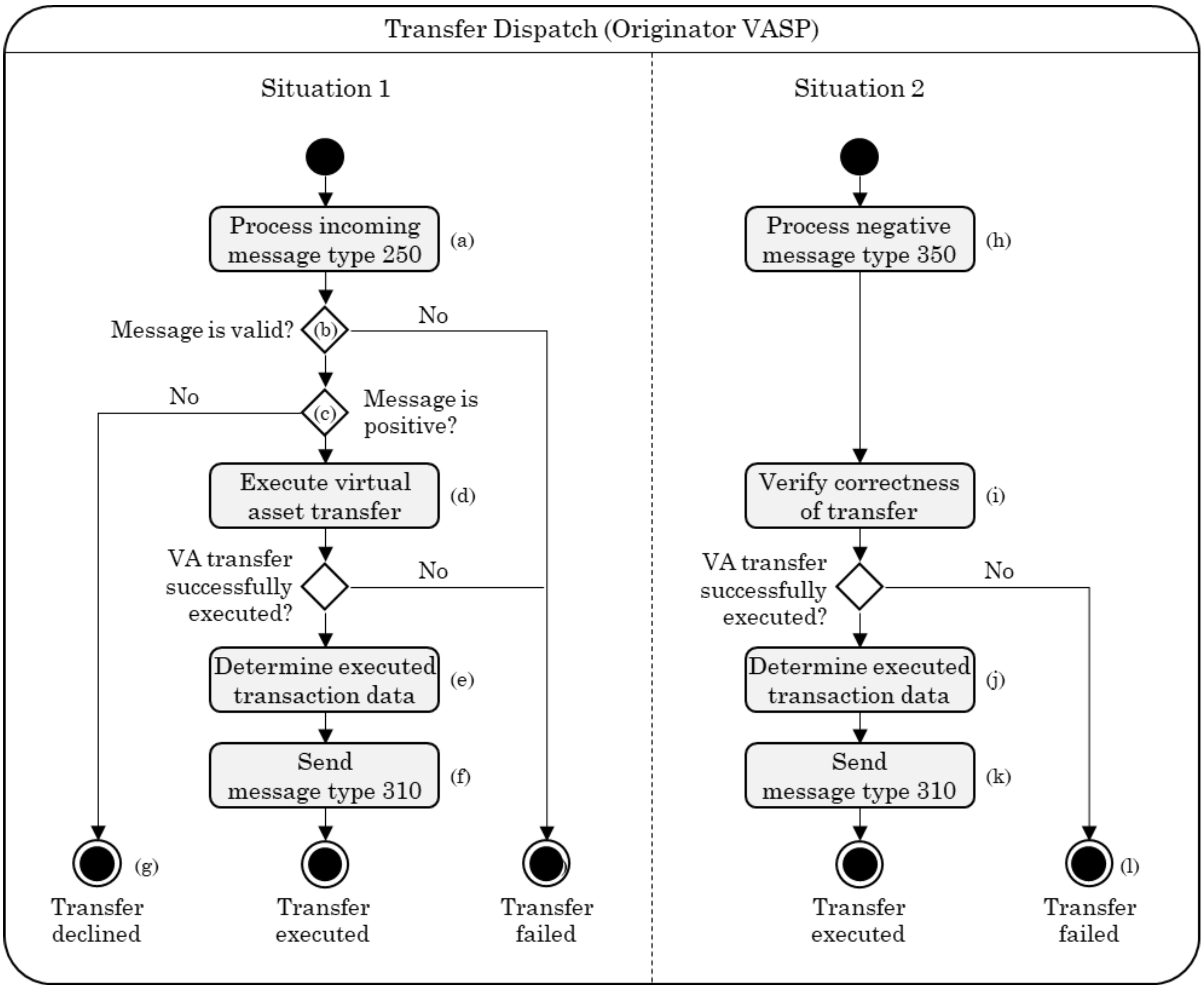
b) Verify if incoming message contains mandatory data in valid format and is correctly signed.

c) Perform a risk-based decision whether the requested virtual asset transfer can be executed, based on all relevant information including information about the originator, the beneficiary, the beneficiary VASP, the virtual asset type and the amount of the requested transfer.

d) Define destination address to be used for the virtual asset transfer.

e) Send Transfer Replay message (type 250, see section 7.6) to the originator VASP. In case of an invalid incoming message (step b) or if the transfer is deemed to be non-compliant, a negative message is sent by providing the response code specifying the reason for failure.

## 6.6 Transfer Dispatch


Transfer Dispatch (Originator VASP)
Situation 1
Process incoming message type 250
(a)
Message is valid?
(b)
No
No
(c)
Message is positive?
Execute virtual asset transfer
(d)
VA transfer successfully executed?
No
Determine executed transaction data
(e)
Send message type 310
(f)
(g)
Transfer declined
Transfer executed
Transfer failed
Situation 2
Process negative message type 350
(h)
Verify correctness of transfer
(i)
VA transfer successfully executed?
No
Determine executed transaction data
(j)
Send message type 310
(k)
(l)
Transfer executed
Transfer failed

### 6.6.1 Processing steps by the originator VASP (situation 1)

a) Receive Transfer Reply message (type 250), e.g. from the Whisper node.

b) Verify if incoming message contains mandatory data in valid format and is correctly signed.

c) Check whether received message has a positive response code.

d) Execute the virtual asset transfer by posting the transaction to the blockchain or distributed ledger technology (DLT) system. Check whether the transaction has been successfully executed, considering the specifics of the technology used for the transfer (e.g. awaiting a number of confirmations). Repeat if necessary.

e) Determine transaction information (identifier, date and time of execution and sending address, if applicable).

f) Send Transfer Dispatch message (type 310, see section 7.7) to the beneficiary VASP.

g) In case of an invalid or negative incoming message (steps b or c) or if the transfer could not be executed (e.g. for technical reasons), a negative message is sent by providing the response code specifying the reason for failure.

### 6.6.2 Processing steps by the originator VASP (situation 2)

h) Receive negative Transfer Confirmation message (type 350), e.g. from the Whisper node.

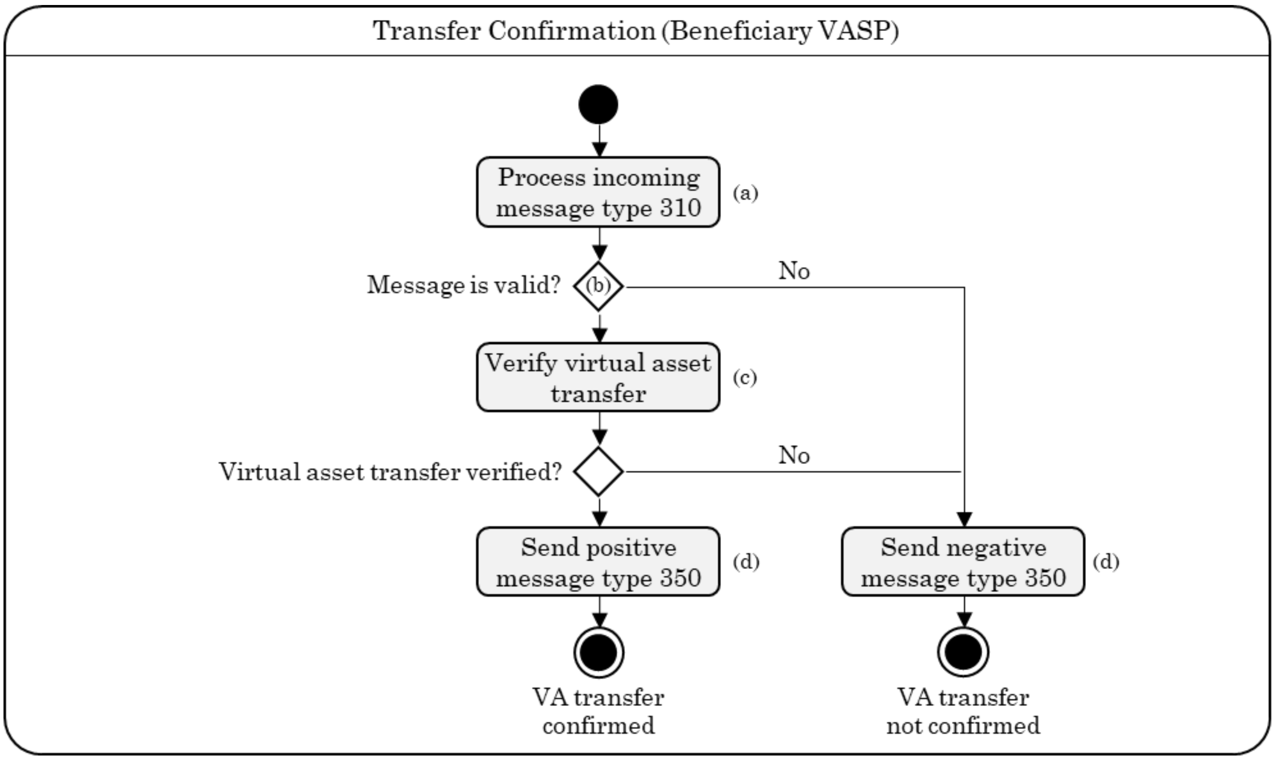
i) Verify whether transaction has been correctly executed and analyze the reason for negative confirmation from the beneficiary VASP. Repeat execution of transaction if necessary.

j) Determine transaction information (identifier, date and time of execution and sending address, if applicable).

k) Send new Transfer Dispatch message (type 310, see section 7.7) to the beneficiary VASP, specifying the result of the verification.

l) If the transfer could (still) not be executed (e.g. for technical reasons), a negative message is sent with a response code specifying the reason for failure.

## 6.7 Transfer Confirmation


Transfer Confirmation (Beneficiary VASP)
Process incoming message type 310
(a)
Message is valid?
(b)
No
Verify virtual asset transfer
(c)
Virtual asset transfer verified?
No
Send positive message type 350
(d)
Send negative message type 350
(d)
VA transfer confirmed
VA transfer not confirmed

### 6.7.1 Processing steps by the beneficiary VASP

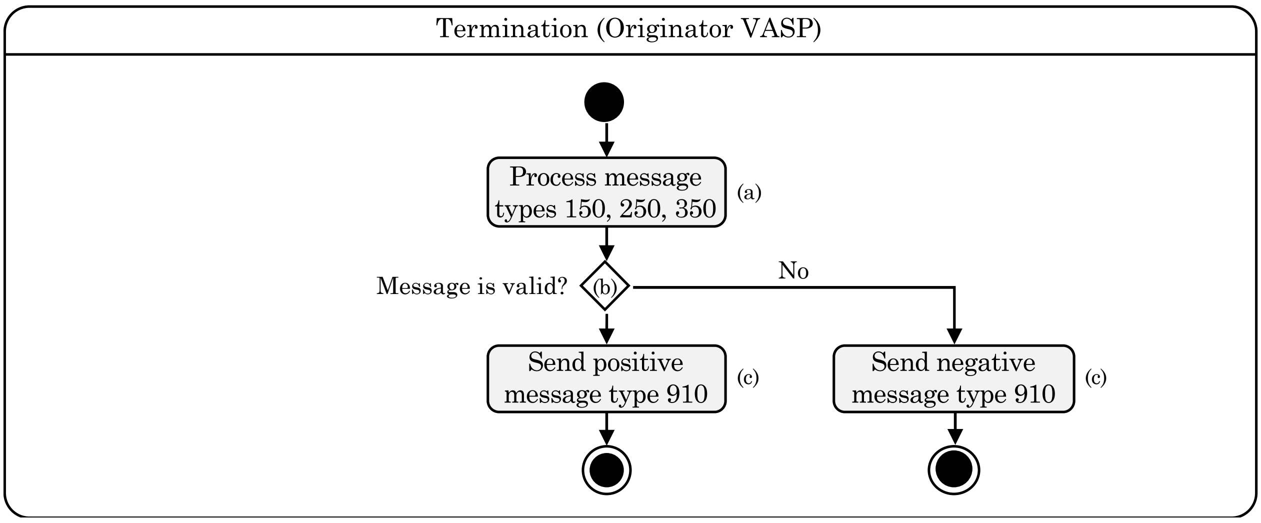
a) Receive Transfer Dispatch message (type 310), e.g. from the Whisper node.

b) Verify if incoming message contains mandatory data in valid format and is correctly signed.

c) Check whether the transaction has been successfully executed on the blockchain or distributed ledger technology (DLT) system, considering the specifics of the technology used for the transfer (e.g. awaiting a number of confirmations). Verify whether received transaction information is accurate.

d) Send Transfer Confirmation message (type 350, see section 7.8) to the originator VASP. In case of an invalid incoming message (step b) or if the transaction has not been successfully executed or is not matching the transaction information received in message 310 (step c), a negative message is sent by providing the response code specifying the reason for failure.

## 6.8 Termination



Termination (Originator VASP)
Process message types 150, 250, 350
(a)
Message is valid?
(b)
No
Send positive message type 910
(c)
Send negative message type 910
(c)

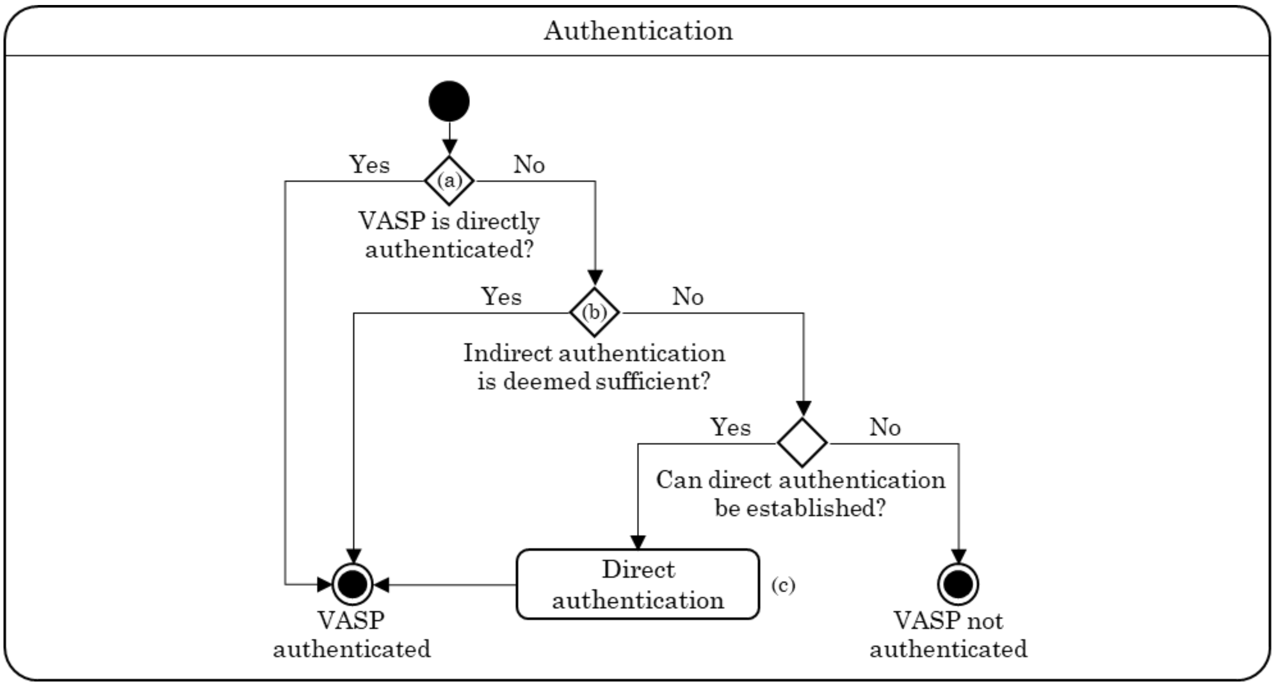
### 6.8.1 Processing steps by the originator VASP

a) Receive Session Reply (type 150), Transfer Reply (type 250) or Transfer Confirmation (type 350) message, e.g. from the Whisper node.

b) Verify if incoming message contains mandatory data in valid format and is correctly signed.

c) Send Termination message (type 910, see section 7.8) to the originator VASP. In case of an invalid incoming message (step b) or if the transaction has not been successfully executed or is not matching the transaction information received in message 310 (step c), a negative message is sent by providing the response code specifying the reason for failure.

## 6.9 Authentication



Authentication
Yes
(a)
No
VASP is directly authenticated?
Yes
(b)
No
Indirect authentication is deemed sufficient?
Yes
No
Can direct authentication be established?
Direct authentication
(c)
VASP authenticated
VASP not authenticated

### 6.2.1 Processing steps

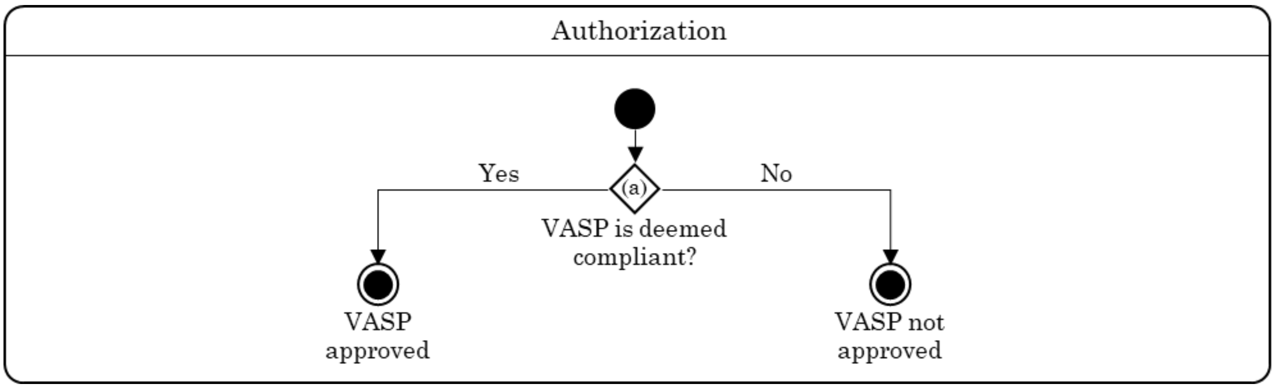
a) Ideally, the receiving VASP can be directly authenticated, which means first-hand evidence that the identity is genuine (e.g. obtaining the VASP identity via personal contact).

b) Indirect authentication can be achieved if both sending and receiving VASP share a web of trust involving other VASPs or by checking the counterparty's identity claims issued by thrusted third parties.

c) If indirect authentication is deemed not to be enough, establishing direct authentication should be considered.

See chapter 5 for more details about authentication.

## 6.10 Authorization



### 6.3.1 Processing steps

a) The VASP must evaluate whether the counterparty VASP is acceptable or not in order to comply with applicable laws and regulations. Particularly the rules regarding anti-money laundering, sanctions and the combating the financing of terrorism must be applied when evaluating the counterparty VASP. Decision is taken following a risk-based approach and differs from VASP to VASP.

Important: the decision whether a virtual asset transfer can be executed is not only based on the counterparty VASP, but primarily by analyzing originator and beneficiary information. This step is solely about the assessment on the counterparty VASP involved in the transfer once identified and authenticated.

# 7. Protocol Messages

## 7.1 Message Types

The protocol includes seven structured message types as summarized in the table below. Protocol messages are specified to use the JSON format, which is human-readable as well as common for data processing.

Each message contains relevant header information and is signed with the sender's private key. Certain content is deliberately repeated in the different messages to provide complete information about the virtual asset transfer in its respective state when looking at a specific message. In this way, messages can be directly filed and maintained in compliance with applicable regulations.

This set of message types is meant to form the basis for a more extended VASP-to-VASP protocol with additional message types to be included based on community requirements. Therefore, session initiation using message types 110 and 150 is designed to be a generic handshake for any message types to follow. Similarly, message type 910 is solely closing the session in the end.

One of the most obvious additions would be message types for bulk transfers of virtual assets, which might be particularly relevant for larger VASPs. However, it seems reasonable to first gain experience on single transfers first and expand the protocol based on the learnings of the user base.

Details for the seven message types can be found in sections 7.3 to 7.9 of this chapter.

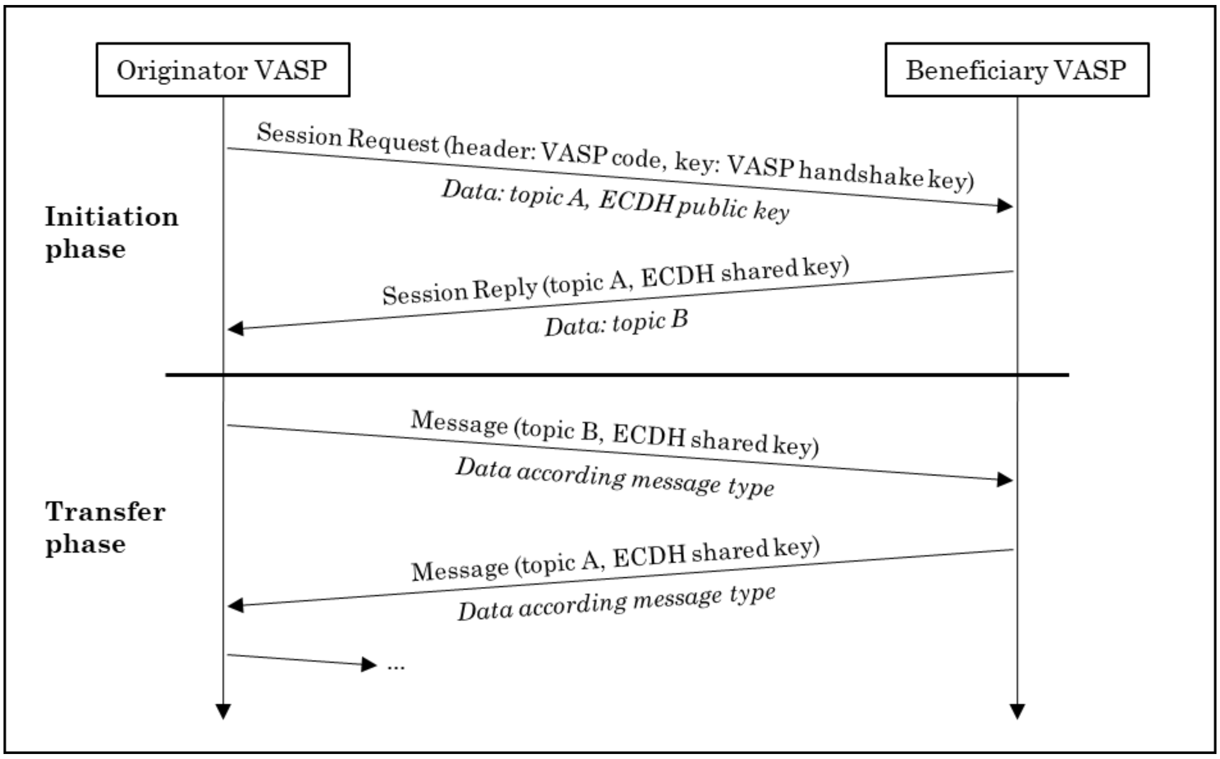
| | 110 | 150 | 210 | 250 | 310 | 350 | 910 |
|---|---|---|---|---|---|---|---|
| **Message Type** | **Session Request** | **Session Reply** | **Transfer Request** | **Transfer Reply** | **Transfer Dispatch** | **Transfer Confirmation** | **Termination** |
| Actor | Originator VASP | Beneficiary VASP | Originator VASP | Beneficiary VASP | Originator VASP | Beneficiary VASP | Originator VASP |
| **Message Envelope** | | | | | | | |
| Topic | Ben. VASP code | Topic A | Topic B | Topic A | Topic B | Topic A | Topic B |
| Encryption Key | Ben. VASP handshake key | ECDH shared key | ECDH shared key | ECDH shared key | ECDH shared key | ECDH shared key | ECDH shared key |
| Encryption Type | Asymmetric | Symmetric | Symmetric | Symmetric | Symmetric | Symmetric | Symmetric |
| **Message Content** | | | | | | | |
| Message | Message id | Message id | Message id | Message id | Message id | Message id | Message id |
| | Session id | Session id | Session id | Session id | Session id | Session id | Session id |
| | Message code | Message code | Message code | Message code | Message code | Message code | Message code |
| Handshake | Topic A | | | | | | |
| | ECDH public key | | | | | | |
| | | Topic B | | | | | |
| Originator | | | Originator information | Originator information | Originator information | Originator information | |
| Beneficiary | | | Beneficiary information | Beneficiary information | Beneficiary information | Beneficiary information | |
| Transfer | | | Virtual asset type | Virtual asset type | Virtual asset type | Virtual asset type | |
| | | | Transfer type | Transfer type | Transfer type | Transfer type | |
| | | | Amount | Amount | Amount | Amount | |
| | | | | Destination address | Destination address | Destination address | |
| Transaction | | | | | Transaction id | Transaction id | |
| | | | | | DateTime | DateTime | |
| | | | | | Sending address | Sending address | |
| Comment | Comment | Comment | Comment | Comment | Comment | Comment | Comment |
| VASP | VASP Information | VASP Information | VASP Information | VASP Information | VASP Information | VASP Information | VASP Information |
| Signature | Signature | Signature | Signature | Signature | Signature | Signature | Signature |
| **Message Flow** | | | | | | | |
| Preceding Messages | - | 110 | 150 | 210 | 150, 250 | 310 | 150, 250, 350 |
| See section: | 7.3 | 7.4 | 7.5 | 7.6 | 7.7 | 7.8 | 7.9 |

## 7.2 Session Model

All messages exchanged for a virtual asset transfer form a session. Sessions are initiated by message type 110 and acknowledged by message type 150.

The initiation phase includes the exchange of public keys to generate a shared symmetric key following the elliptic curve Diffie–Hellman (ECDH) protocol. In addition, both VASPs exchange random headers to be used for message routing in transfer phase (topic A and topic B).

Following initiation phase, originator and beneficiary information can be exchanged ensuring forward secrecy, as the long-term keys are used for authentication purposes only.


Originator VASP
Beneficiary VASP
Initiation phase
Session Request (header: VASP code, key: VASP handshake key)
Data: topic A, ECDH public key
Session Reply (topic A, ECDH shared key)
Data: topic B
Transfer phase
Message (topic B, ECDH shared key)
Data according message type
Message (topic A, ECDH shared key)
Data according message type
...

### 7.2.1 Encryption system and parameter selection

Cryptographic algorithms used and all its parameter (e.g. key lengths) must be thoroughly challenged and carefully selected as part of the final protocol specification. The following suggestions are meant as a starting point for discussion:

- **Session request message**:
  Asymmetric encryption SECP-256k1 as per Whisper standard, using the handshakeKey published in the VASP contract (see section 5.2).
- **All other messages:**
  Symmetric encryption with the AES GCM algorithm as per Whisper standard, 256-bit key with random 96-bit nonce.

  Shared key generated via the X25519 key exchange protocol with a fixed key length of 256 bit. Salting as well as an appropriate key derivation function must be defined, considering that one of the two public ECDH keys is published in the VASP contract and therefore reused.

## 7.3 Session Request

### 7.3.1 Message purpose

Initiates a session between two VASPs. First part of a two-way handshake between VASPs to finally generate a shared symmetric session key based on the Diffie–Hellman key-exchange protocol

### 7.3.2 Message structure

| | |
|---|---|
| Name | Session Request |
| Number | 110 |
| Actor | Originator VASP |

| Envelope | Parameter Value | Type |
|---|---|---|
| Topic | Beneficiary VASP code | Hex(32-bit) |
| Encryption Key | Beneficiary VASP handshake key | Hex(256-bit) |
| Encryption Type | Asymmetric | SECP-256k1 |

| Level 1 | Level 2 | Name | Type | Mult. | Comment |
|---|---|---|---|---|---|
| Message | | msg | | 1..1 | |
| | Message type | type | String | 1..1 | Fixed value: '110' |
| | Message identifier | msgid | Hex(128-bit) | 1..1 | Randomly set |
| | Session identifier | session | Hex(128-bit) | 1..1 | Randomly set and used for entire session |
| | Message code | code | String | 1..1 | Currently not used, fixed value '1' |
| Handshake | | handshake | | 1..1 | |
| | Topic A | topica | Hex(32-bit) | 1..1 | Randomly set |
| | ECDH public key | ecdhpk | Hex(256-bit) | 1..1 | According to specified domain parameters |
| Comment | | comment | String | 0..1 | |
| VASP | *see 7.10 for elements* | vasp | | 1..1 | VASP information incl. public signing key |
| Signature | | sig | String | 1..1 | Message signed with actor's private signing key |

## 7.4 Session Reply

### 7.4.1 Message purpose

Response (positive or negative) to a previous request for initiating a session between two VASPs. Second part of a two-way handshake.

### 7.4.2 Message structure

| | |
|---|---|
| Name | Session Reply |
| Number | 150 |
| Actor | Beneficiary VASP |

| Envelope | Parameter Value | Type |
|---|---|---|
| Topic | Topic A | Hex(32-bit) |
| Encryption Key | ECDH shared key | Hex(256-bit) |
| Encryption Type | Symmetric | AES GCM |

| Level 1 | Level 2 | Name | Type | Mult. | Comment |
|---|---|---|---|---|---|
| Message | | msg | | 1..1 | |
| | Message type | type | String | 1..1 | Fixed value: '150' |
| | Message identifier | msgid | Hex(128-bit) | 1..1 | Randomly set |
| | Session identifier | session | Hex(128-bit) | 1..1 | As set in message 110 |
| | Message code | code | String | 1..1 | See 7.4.3 |
| Handshake | | handshake | | 1..1 | |
| | Topic B | topicb | Hex(32-bit) | 1..1 | Randomly set |
| Comment | | comment | String | 0..1 | |
| VASP | *see 7.10 for elements* | vasp | | 1..1 | VASP information incl. public signing key |
| Signature | | sig | String | 1..1 | Message signed with actor's private signing key |

### 7.4.3 Possible message codes

1 = Session accepted

2 = Session declined; request not valid

3 = Session declined; originator VASP could not be authenticated

4 = Session declined; originator VASP declined

5 = Session declined; temporary disruption of service

## 7.5 Transfer Request

### 7.5.1 Message purpose

Seeking approval from the beneficiary VASP for a virtual asset transfer by specifying transfer details including originator and beneficiary information.

### 7.5.2 Message structure

| | |
|---|---|
| Name | Transfer Request |
| Number | 210 |
| Actor | Originator VASP |

| Envelope | Parameter Value | Type |
|---|---|---|
| Topic | Topic B | Hex(32-bit) |
| Encryption Key | ECDH shared key | Hex(256-bit) |
| Encryption Type | Symmetric | AES GCM |

| Level 1 | Level 2 | Name | Type | Mult. | Comment |
|---|---|---|---|---|---|
| Message | | msg | | 1..1 | |
| | Message type | type | String | 1..1 | Fixed value: ‘210’ |
| | Message identifier | msgid | Hex(128-bit) | 1..1 | Randomly set |
| | Session identifier | session | Hex(128-bit) | 1..1 | As set in message 110 |
| | Message code | code | String | 1..1 | Currently not used, fixed value ‘1’ |
| Originator | *see 7.11 for elements* | originator | | 1..1 | Originator information |
| Beneficiary | *see 7.12 for elements* | beneficiary | | 1..1 | Beneficiary information |
| Transfer | | transfer | | 1..1 | |
| | Virtual asset type | va | String | 1..1 | See 7.5.3 |
| | Transfer type | ttype | Number | 1..1 | See 7.5.4 |
| | Amount | amount | Decimal | 1..1 | 18 digits |
| Comment | | comment | String | 0..1 | |
| VASP | *see 7.10 for elements* | vasp | | 1..1 | VASP information incl. public signing key |
| Signature | | sig | String | 1..1 | Message signed with actor’s private signing key |

### 7.5.3 Possible transfer types

Examples: BTC = Bitcoin, ETH = Ethereum.

To be defined as part of community governance.

### 7.5.4 Possible transfer types

1 = Blockchain transaction

(no further codes defined at the moment, but used to specify alternative transfer types, e.g. Layer 2 mechanisms)

## 7.6 Transfer Reply

### 7.6.1 Message purpose

Response (positive or negative) to an originator VASP having sought approval for a virtual asset transfer by specifying transfer details including originator and beneficiary information.

### 7.6.2 Message structure

| | |
|---|---|
| Name | Transfer Reply |
| Number | 250 |
| Actor | Beneficiary VASP |

| Envelope | Parameter Value | Type |
|---|---|---|
| Topic | Topic A | Hex(32-bit) |
| Encryption Key | ECDH shared key | Hex(256-bit) |
| Encryption Type | Symmetric | AES GCM |

| Level 1 | Level 2 | Name | Type | Mult. | Comment |
|---|---|---|---|---|---|
| Message | | msg | | 1..1 | |
| | Message type | type | String | 1..1 | Fixed value: ‘250’ |
| | Message identifier | msgid | Hex(128-bit) | 1..1 | Randomly set |
| | Session identifier | session | Hex(128-bit) | 1..1 | As set in message 110 |
| | Message code | code | String | 1..1 | See 7.6.3 |
| Originator | *see 7.11 for elements* | originator | | 1..1 | Originator information |
| Beneficiary | *see 7.12 for elements* | beneficiary | | 1..1 | Beneficiary information |
| Transfer | | transfer | | 1..1 | |
| | Virtual asset type | va | String | 1..1 | See 7.5.3 |
| | Transfer type | ttype | Number | 1..1 | See 7.5.4 |
| | Amount | amount | Decimal | 1..1 | 18 digits |
| | Destination address | destination | String | 0..1 | |
| Comment | | comment | String | 0..1 | |
| VASP | *see 7.10 for elements* | vasp | | 1..1 | VASP information incl. public signing key |
| Signature | | sig | String | 1..1 | Message signed with actor’s private signing key |

### 7.6.3 Possible message codes

1 = Transfer accepted

2 = Transfer declined; request not valid

3 = Transfer declined; no such beneficiary

4 = Transfer declined; virtual asset not supported

5 = Transfer declined; transfer not authorized

6 = Transfer declined; temporary disruption of service

## 7.7 Transfer Dispatch

### 7.7.1 Message purpose

Notifies the beneficiary VASP that a virtual asset transaction has been committed to the blockchain.

### 7.7.2 Message structure

| | |
|---|---|
| Name | Transfer Dispatch |
| Number | 310 |
| Actor | Originator VASP |

| Envelope | Parameter Value | Type |
|---|---|---|
| Topic | Topic B | Hex(32-bit) |
| Encryption Key | ECDH shared key | Hex(256-bit) |
| Encryption Type | Symmetric | AES GCM |

| Level 1 | Level 2 | Name | Type | Mult. | Comment |
|---|---|---|---|---|---|
| Message | | msg | | 1..1 | |
| | Message type | type | String | 1..1 | Fixed value: '310' |
| | Message identifier | msgid | Hex(128-bit) | 1..1 | Randomly set |
| | Session identifier | session | Hex(128-bit) | 1..1 | As set in message 110 |
| | Message code | code | String | 1..1 | Currently not used, fixed value '1' |
| Originator | *see 7.11 for elements* | originator | | 1..1 | Originator information |
| Beneficiary | *see 7.12 for elements* | beneficiary | | 1..1 | Beneficiary information |
| Transfer | | transfer | | 1..1 | |
| | Virtual asset type | va | String | 1..1 | See 7.5.3 |
| | Transfer type | ttype | Number | 1..1 | See 7.5.4 |
| | Amount | amount | Decimal | 1..1 | 18 digits |
| | Destination address | destination | String | 0..1 | |
| Transaction | | tx | | 1..1 | |
| | Transaction identifier | txid | String | 0..1 | Format specific to virtual asset / transfer type |
| | Transaction datetime | datetime | String | 1..1 | ISO 8601 (YYYY-MM-DDThh:mm:ssZ) |
| | Sending address | sendingadr | String | 0..1 | Blockchain-specific format, sending address used for transaction (non-UTXO systems) |
| Comment | | comment | String | 0..1 | |
| VASP | *see 7.10 for elements* | vasp | | 1..1 | VASP information incl. public signing key |
| Signature | | sig | String | 1..1 | Message signed with actor's private signing key |

## 7.8 Transfer Confirmation

### 7.8.1 Message purpose

Positive or negative acknowledgement to the originator VASP about the receipt of virtual assets transferred via a blockchain transaction.

### 7.8.2 Message structure

| | |
|---|---|
| Name | Transfer Confirmation |
| Number | 350 |
| Actor | Beneficiary VASP |

| Envelope | Parameter Value | Type |
|---|---|---|
| Topic | Topic A | Hex(32-bit) |
| Encryption Key | ECDH shared key | Hex(256-bit) |
| Encryption Type | Symmetric | AES GCM |

| Level 1 | Level 2 | Name | Type | Mult. | Comment |
|---|---|---|---|---|---|
| Message | | msg | | 1..1 | |
| | Message type | type | String | 1..1 | Fixed value: '350' |
| | Message identifier | msgid | Hex(128-bit) | 1..1 | Randomly set |
| | Session identifier | session | Hex(128-bit) | 1..1 | As set in message 110 |
| | Message code | code | String | 1..1 | See 7.8.3 |
| Originator | *see 7.11 for elements* | originator | | 1..1 | Originator information |
| Beneficiary | *see 7.12 for elements* | beneficiary | | 1..1 | Beneficiary information |
| Transfer | | transfer | | 1..1 | |
| | Virtual asset type | va | String | 1..1 | See 7.5.3 |
| | Transfer type | ttype | Number | 1..1 | See 7.5.4 |
| | Amount | amount | Decimal | 1..1 | 18 digits |
| | Destination address | destination | String | 0..1 | |
| Transaction | | tx | | 1..1 | |
| | Transaction identifier | txid | String | 0..1 | Format specific to virtual asset / transfer type |
| | Transaction datetime | datetime | String | 1..1 | ISO 8601 (YYYY-MM-DDThh:mm:ssZ) |
| | Sending address | sendingadr | String | 0..1 | Blockchain-specific format, sending address used for transaction (non-UTXO systems) |
| Comment | | comment | String | 0..1 | |
| VASP | *see 7.10 for elements* | vasp | | 1..1 | VASP information incl. public signing key |
| Signature | | sig | String | 1..1 | Message signed with actor's private signing key |

### 7.8.3 Possible message codes

1 = Transfer confirmed

2 = Transfer not confirmed; dispatch not valid

3 = Transfer not confirmed; assets not received

4 = Transfer not confirmed; wrong amount

5 = Transfer not confirmed; wrong asset

6 = Transfer not confirmed; transaction data mismatch

## 7.9 Termination

### 7.9.1 Message purpose

Terminates a session between two VASPs.

### 7.9.2 Message structure

| | |
|---|---|
| Name | Termination |
| Number | 910 |
| Actor | Originator VASP |

| Envelope | Parameter Value | Type |
|---|---|---|
| Topic | Topic B | Hex(32-bit) |
| Encryption Key | ECDH shared key | Hex(256-bit) |
| Encryption Type | Symmetric | AES GCM |

| Level 1 | Level 2 | Name | Type | Mult. | Comment |
|---|---|---|---|---|---|
| Message | | msg | | 1..1 | |
| | Message type | type | String | 1..1 | Fixed value: '910' |
| | Message identifier | msgid | Hex(128-bit) | 1..1 | Randomly set |
| | Session identifier | session | Hex(128-bit) | 1..1 | As set in message 110 |
| | Message code | code | String | 1..1 | See 7.9.3 |
| Comment | | comment | String | 0..1 | |
| VASP | *see 7.10 for elements* | vasp | | 1..1 | VASP information incl. public signing key |
| Signature | | sig | String | 1..1 | Message signed with actor's private signing key |

### 7.9.3 Possible message codes

1 = Session closed; transfer occurred

2 = Session closed; transfer declined by beneficiary VASP

3 = Session closed; transfer canceled by originator VASP

## 7.10 VASP Information

Mandatory and optional elements for transmitting information about the VASPs involved in the virtual asset transfer.

| Level 2 | Level 3 | Name | Type | Mult. | Rules | Comment |
|---|---|---|---|---|---|---|
| Name | | name | String | 1..1 | | Legal name |
| VASP identity | | id | String | 1..1 | | Ethereum address of VASP contract |
| VASP public key | | pk | String | 1..1 | | Ethereum public key, see sec. 5.2 |
| Postal address | | address | | 1..1 | | |
| | Street name | street | String | 0..1 | 1 | |
| | Building number | number | String | 0..1 | 1 | |
| | Address line | adrline | String | 0..1 | 1 | Alternative to street/number |
| | Post code | postcode | String | 1..1 | | |
| | Town name | town | String | 1..1 | | |
| | Country | country | String | 1..1 | | ISO 3166-1 alpha-2 code |
| Date / place of birth | | birth | | 0..1 | 2 | |
| | Birth date | birthdate | Date | 1..1 | | ISO 8601 (yyyymmdd) |
| | City of birth | birthcity | String | 1..1 | | |
| | Country of birth | birthcountry | String | 1..1 | | ISO 3166-1 alpha-2 code |
| Natural person ID | | nat | | 0..* | 2 | |
| | Identification type | natid_type | Integer | 1..1 | | 1 = Passport number<br>2 = National identity number<br>3 = Social security number<br>4 = Tax identification number<br>5 = Alien registration number<br>6 = Driver's license number<br>7 = Other |
| | Identifier | natid | String | 1..1 | | |
| | Issuing country | natid_country | String | 0..1 | 5 | ISO 3166-1 alpha-2 code |
| | Non-state issuer | natid_issuer | String | 0..1 | 5 | |
| Juridical person ID | | jur | | 0..* | 3 | |
| | Identification type | jurid_type | Integer | 1..1 | | 1 = Country identification number<br>2 = Tax identification number<br>3 = Certificate of incorporation no.<br>4 = Legal Entity Identifier (LEI)<br>5 = Bank party identification<br>6 = Other |
| | Identifier | jurid | String | 1..1 | | |
| | Issuing country | jurid_country | String | 0..1 | 6 | ISO 3166-1 alpha-2 code |
| | Non-state issuer | jurid_issuer | String | 0..1 | 6 | |
| BIC | | bic | String | 0..1 | 4 | ISO 9362 Bank Identifier Code |

### 7.10.1 Rules

1) Either [street] and [number] must both be present or [adrline]. All three positions can be present at the same time.
2) [birth] or [nat] is allowed, if neither [jur] nor [bic] is present.
3) [jur] is allowed, if neither [birth] nor [nat] nor [bic] is present.
4) [bic] is allowed, if neither [birth] nor [nat] nor [jur] is present.
5) If [nat] is present, either [natid_country] or [natid_issuer] or both must be present.
6) If [jur] is present, either [jurid_country] or [jurid_issuer] or both must be present.

## 7.11 Originator Information

Mandatory and optional elements for transmitting originator information to be included in message types 210, 250, 310 and 350.

| Level 2 | Level 3 | Name | Type | Mult. | Rules | Comment |
|---|---|---|---|---|---|---|
| Name | | name | String | 1..1 | | |
| VAAN | | vaan | Hex(96-bit) | 1..1 | | Assigned by VASP, see section 4.2 |
| Postal address | | address | | 0..1 | | |
| | Street name | street | String | 0..1 | 1 | |
| | Building number | number | String | 0..1 | 1 | |
| | Address line | adrline | String | 0..1 | 1 | Alternative to street/number |
| | Post code | postcode | String | 1..1 | | |
| | Town name | town | String | 1..1 | | |
| | Country | country | String | 1..1 | | ISO 3166-1 alpha-2 code |
| Date / place of birth | | birth | | 0..1 | 2 | |
| | Birth date | birthdate | Date | 1..1 | | ISO 8601 (yyyymmdd) |
| | City of birth | birthcity | String | 1..1 | | |
| | Country of birth | birthcountry | String | 1..1 | | ISO 3166-1 alpha-2 code |
| Natural person ID | | nat | | 0..* | 2 | |
| | Identification type | natid_type | Integer | 1..1 | | 1 = Passport number<br>2 = National identity number<br>3 = Social security number<br>4 = Tax identification number<br>5 = Alien registration number<br>6 = Driver's license number<br>7 = Other |
| | Identifier | natid | String | 1..1 | | |
| | Issuing country | natid_country | String | 0..1 | 5 | ISO 3166-1 alpha-2 code |
| | Non-state issuer | natid_issuer | String | 0..1 | 5 | |
| Juridical person ID | | jur | | 0..* | 3 | |
| | Identification type | jurid_type | Integer | 1..1 | | 1 = Country identification number<br>2 = Tax identification number<br>3 = Certificate of incorporation no.<br>4 = Legal Entity Identifier (LEI)<br>5 = Bank party identification<br>6 = Other |
| | Identifier | jurid | String | 1..1 | | |
| | Issuing country | jurid_country | String | 0..1 | 6 | ISO 3166-1 alpha-2 code |
| | Non-state issuer | jurid_issuer | String | 0..1 | 6 | |
| BIC | | bic | String | 0..1 | 4 | ISO 9362 Bank Identifier Code |

### 7.11.1 Rules

1) Either [street] and [number] must both be present or [adrline]. All three positions can be present at the same time.

2) [birth] or [nat] is allowed, if neither [jur] nor [bic] is present.

3) [jur] is allowed, if neither [birth] nor [nat] nor [bic] is present.

4) [bic] is allowed, if neither [birth] nor [nat] nor [jur] is present.

5) If [nat] is present, either [natid_country] or [natid_issuer] or both must be present.

6) If [jur] is present, either [jurid_country] or [jurid_issuer] or both must be present.

## 7.12 Beneficiary Information

Mandatory and optional elements for transmitting originator data to be included in message types 210, 250, 310 and 350.

| Level 2 | Level 3 | Name | Type | Mult. | Rules | Comment |
|---|---|---|---|---|---|---|
| Name | | name | String | 1..1 | | |
| VAAN | | vaan | Hex(96-bit) | 1..1 | | Assigned by VASP, see section 4.2 |