Identity (XID)

"Bridging the legacy traditional banking platform with a public blockchain"

Marshall Hayner Glenn Mariën

Introduction to Identity Chain (XID)

Vitalik Buterin's development of Ethereum in 2013 has often been hailed as a phenomenal achievement in smart contracts and digital currency. Built to be a decentralized cloud computer that would offer a layer on top of the internet to enable DACs and DAOs, also known as decentralized autonomous corporations/organizations, Ethereum instead has proven itself to be useful as: 1) a platform to issue new assets and tokens (many of which used Ethereum as a launch pad to access crypto markets), 2) a store of value, and 3) a medium of exchange between alternative cryptocurrency and cryptocommodity markets. We are now seeing the beginning of what is called the Decentralized Finance movement or "DeFi," which, in its nascent form, has run into several major obstacles to mass adoption and development. The cost of transactions have risen exponentially and unpredictably with the markets, thus limiting the potential for micro transactions or embedding identity from large to small corporations. One solution proposed by Origin, is the usage of meta-transactions and relays to process those transactions[1]. The question then becomes, when and how do those transactions eventually get processed? Another proposal is EIP-1559[2], which lowers fees by preventing gamification from miners by burning the actual fee and only accepting the 'tip', a nominal amount selected by the user. Unfortunately, this further exacerbates the situation of rising costs of fees and thus kicks the proverbial can down the road.

The next problem we must ask ourselves becomes 'who controls the central repository of code?' 'Who sets the standards?' With the introduction of Facebook Libra[3] and JP Morgan Chase Quorum[4] Interbank Information Network[5], it has become increasingly evident that blockchain must have a diversified, multi-party Blockchain Engineering Task Force (BCETF), similar to the Internet Engineering Task Force (IETF), to help establish inter-blockchain standards for the web. Similar to the IETF, the BCETF has no board of directors, is not a corporation, and does not have any dues. The focus of the BCETF is to establish standards for the integration of traditional legacy payment systems into public blockchain infrastructure.

---

[1] https://medium.com/originprotocol/driving-user-adoption-with-meta-transactions-3539aa6c5ae3
[2] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md
[3] https://libra.org/en-US/wp-content/uploads/sites/23/2019/06/LibraWhitePaper_en_US.pdf
[4] https://github.com/jpmorganchase/quorum
[5] https://www.jpmorgan.com/global/treasury-services/IIN

Identity (XID) acts as the sandbox for experimentation by the BCETF for decentralized sovereign identity, payment flows, and transaction monitoring by extending the existing Ethereum codebase and creating a bridge from legacy payment protocols (ACH, SEPA, SWIFT) and services to modern protocols and services (blockchain). Identity (XID) aims to remove the concept of sharing private keys (BIN and bank account numbers) directly with merchants, instead creating a permission-less layer that exists on top of traditional card networks and banks. XID enables the fintechs, banks, and regulators of the world to experiment with the concept of monetary systems existing on public blockchain ledgers, before moving into production. In the case of governments experimenting with digital national currency, XID acts as a sandbox before setting sovereign digital currencies into motion as national currencies existing on a blockchain (nation-state sponsored cryptocurrencies).

History

The history of cryptocurrency has been well documented, from the birth of Bitcoin to the introduction of Ethereum in 2014. We have witnessed the birth of stable coins, smart contracts, securities on the blockchain, and decentralized finance (DeFi). Metal launched a platform, called Metal Pay, that proved a working demonstration of a Proof-of-Processed-Payment minting mechanism for a crypto-commodity called MTL. Brave launched a web publishing platform, called Basic Attention Token, that proved a working demonstration of Proof-of-Attention minting mechanism for a crypto-commodity called BAT. We have seen the creation of unique digital collectables like crypto-kitties and decentralized ownership of digital parcels of land with Decentraland (MANA). As the decentralized future of the web progresses, it becomes increasingly obvious that we must establish standards between blockchains, and simultaneously must establish a global sandbox that does not enforce austrian economics or any unsustainable market mechanics that would prevent further development by large and small corporations and groups. This platform would grant XID and other crypto-commodities and cryptocurrencies to academics, researchers, governments, banks, payment processors, and financial technology providers, in order to set a foundation for experimentation. The purpose of extending grants is to allow for these institutions to embed identity contracts and pay for gas costs, with the intention of keeping transaction and contract fees extremely low while allowing everyone globally universal access to identity.

Regulatory

The Blockchain Engineering Task Force (BCETF) seeks to establish clear legal definition from the regulatory agencies of the world as to which digital assets are regulated by what agencies and how. Furthermore, defining these assets into three distinct categories: crypto securities, crypto commodities, and crypto currencies and seeks the most efficient possible ways to directly implement the automation of the creation of said assets directly from the core commands of XID protocol.

Identity

Identity builds on the concepts of identity key management originally proposed by Fabian Vogelsteller with ERC 725[6] and ERC 735[7], with the intent to merge these proposals directly into the core chain functionality. We propose several interactions beyond the original and arguably most widely used function of Ethereum, which is the ERC 20 standard. Ethereum Request for Comment was adopted from the Internet Engineering Task Force standards for “Request for Comment” (RFC)[8], which was originally used to create a formal document. Identity makes use of the ERC formats 20, 721, 725, 735, and 1404 as core functions of the blockchain. Additionally, XID permits the banks of the world to experiment with a live production environment with other decentralized applications actively on it, without the consequence of serious gas costs or restrictive network operating fees. This is achieved through a hybrid consensus proof-of-work (PoW), proof-of-stake (PoS), and merged-mine method that allows several fall-back mechanisms preventing malicious hard-forks.

Governance

Governance is modeled after the Maker (MKR) concept on Ethereum and dedicated towards leveraging the wisdom of the crowd[9]. In order to vote on risk management, business logic and technical development decisions, one can cast a vote through locking XID into the governance contract.

Items that will be voted on will consist of (a) XRC XID Requests for Comment (b) Mining rewards (c) Staking reward (d) Grants to be offered in order to establish further users, organizations and developers on the platform.

Mining is split into a hybrid Proof-of-Work, Proof-of-Stake, and merged-mined model against Litecoin in which all three comprise of 75% of the block reward and the remaining 25% to consist against the Treasury administered and managed by the Blockchain Engineering Task Force (BCETF).

KYC Flow
XID can accept hashes for encrypted documents, that can be passed independently of the blockchain and verified through an immutable ledger. The process would look something like this:

---

[6] https://github.com/ethereum/EIPs/issues/725
[7] https://github.com/ethereum/EIPs/issues/735
[8] https://en.wikipedia.org/wiki/Request_for_Comments
[9] https://makerdao.com/en/whitepaper/#mkr-token-governance

KYC Verify

Verifier is defined as the entity verifying the KYC information
Application is defined as the stand-alone application provided by the payment processor or financial institution
Sender is defined as the consumer or business sending KYC/KYB documents for verification

1. Sender submits KYC documents to Application using the specified formatting
2. The documents are then separated and signed with the users private key
3. Documents are then compressed and encrypted according to the Verifiers public key
4. Application then pushes the transaction with the flags: -KYCrequest
5. Application sends the compressed file over third party networks labeled as the transaction hash
6. Verifier receives the compressed file labeled as the transaction hash
7. Verifier receives the XID transaction for KYC verification
8. The transaction and corresponding compressed file are matched by transaction ID
9. KYC verification is processed by the Verifier
10. Upon success or failure of verification the Verifier sends a transaction to the Sender with the flags: -KYCverify -KYCStatus:Success -RiskScore:XXXXX
11. The Sender has successfully become verified by the Verifier with a time stampPayment Request

Pending Payment

Receiver is defined as the entity requesting the payment
Application is defined as the stand-alone application provided by the payment processor or financial institution
Sender is defined as the entity sending the payment

1. Receiver forms a payment request with their desired currency and amount
2. Application broadcasts the request transaction with flags: -PaymentRequest -Currency -Amount
3. Sender public key receives a payment request transaction
4. Application catches the transaction and displays it in the form of a payment request with any public data available from the Receiver public key such as `RiskScore`

Identity Request

Receiver is defined as the entity requesting identity
Application is defined as the stand-alone application provided by the payment processor or financial institution
Verifier is defined as the entity that has independently verified the identifying documents
Sender is defined as the entity sending the identifying documents

1. Receiver forms a payment request with their desired currency and amount
2. Application broadcasts the request transaction with flags: -PaymentRequest -Currency:XXX -Amount:XXXX
3. Sender public key receives a payment request transaction
4. Application monitors for the transaction and displays it in the form of a payment request with any public data available from the Receiver public key such as `RiskScore`
5. Sender approves or denies the request

Pending payment

Receiver is defined as the entity receiving the pending payment
Application is defined as the stand-alone application provided by the payment processor or financial institution
Sender is defined as the entity sending the pending payment
Transmitter is defined as the entity that has independently processed and verified the pending payment transaction

1. Sender forms a pending payment transaction with their desired currency and amount to the Transmitter
2. Application broadcasts the pending payment transaction with flags: -PendingPayment:Request -Currency:XXX -Amount:XX -Receiver:XXX to the Transmitter
3. Transmitter approves or denies the transaction and sends the output as a transaction with flags: -PendingPayment:Approved -Currency:XXX -Amount:XX to the Receiver
4. Application monitors for the transaction and displays it in the form of a pending payment transaction with any public data available from the Sender public key such as `RiskScore`

Bank Linking

Receiver is defined as the entity receiving the pending payment
Application is defined as the stand-alone application provided by the payment processor or financial institution
Sender is defined as the entity sending the pending payment
Transmitter is defined as the entity that has independently processed and verified the pending payment transaction

1. Sender forms a pending payment transaction with their desired currency and amount to the Transmitter
2. Application broadcasts the pending payment transaction with flags: -PendingPayment:Request -Currency:XXX -Amount:XX -Receiver:XXX to the Transmitter
3. Transmitter approves or denies the transaction and sends the output as a transaction with flags: -PendingPayment:Approved -Currency:XXX -Amount:XX to the Receiver
4. Application monitors for the transaction and displays it in the form of a pending payment transaction with any public data available from the Sender public key such as `RiskScore`

Self sovereign identity

While there is no solid consensus of the term “self sovereign identity,” many believe it means individuals and businesses being in full control of their personal/sensitive information[10]. The idea behind this is not new, but it is becoming more apparent in an era where data breaches involving sensitive personally identifiable information are becoming weekly headlines.

In a world fully entangled in the digital transformation, loss of control over one’s identity information is becoming a rampant problem. When a person’s information is stored and scattered across multiple institutions, some of which may not be as protective of your sensitive information as you are, data breaches and identity fraud proliferate freely.

XID aims to solve this by providing a platform to register one’s identity and the accompanying sensitive information in a fully decentralized, yet controlled, manner. The owner of this identity has the ability to choose when and what information she wishes to share. An inversion of control is at the center of this idea, effectively giving back control of the information to the entity (i.e., the individual) it belongs to. The owner maintains the private key in confidence, while permitting institutions who wish to authenticate the owner to check with trusted institutions via public keys. The verification from the trusted institution will confirm the owner’s identity without revealing the owner’s associated private key.

[10] https://sovrin.org/faq/what-is-self-sovereign-identity/

Identity

An identity can be established by either an individual or a business. Xeth (the XID clone of geth) will include natively built in commands to handle the deployment of the identity contract.

The specification of this contract builds upon the ERC725v2 spec designed by Fabian Vogelsteller and acts as a proxy between the identity holder and the claims made against it.

Specification

Identity Proxy	
address public owner;	Returns owner of the identity
function changeOwner(address _owner);	Changes owner of the identity, can only be called by the current owner → Triggers OwnerChanged event
getData(bytes32 _key) external view returns (bytes _value);	Returns data of the identity stored at a specific key
function setData(bytes32 _key, bytes calldata _value) external;	Stores data at a specific key, can only be used by the owner of the identity → Triggers DataChanged event
function execute(uint256 _operationType, address _to, uint256 _value, bytes _data) external;	Executes an operation on behalf of the identity, can only be called by the owner. The operationType should be one of the following: - 0 for call type transactions - 1 for create type transactions

Claims

An identity can have one or more claims made against it. These claims can be self-attested, or in most cases, attested by a third party. Claims attested by third parties will always have to be approved by the claim holder. The idea builds upon the ERC-735 standard established by Fabian Vogelsteller.

Valid examples of claims are:

- Claim holder claiming he owns a certain Twitter handle, this is an example of self-attestation;
- Twitter claiming an identity owns a certain Twitter handle, this is an example of an attestation by a third party, which in this case is much more powerful;
- A financial institution claiming a certain identity holds a valid government issued ID.

Specification

Claim	
topic	Topic of this claim, examples: ● Proof of residence ● Biometric data ● Social security number ● … The topic essentially describes what this claim is about.
schema	Defines how the claim should be verified, examples: ● Contract call where ***data*** will be the call data ● ECDSA ● ...
issuer	The identity behind issuing this claim which has to be the same key that was used to build the signature.
signature	Signature generated by the issuer using the claim's data.
data	Data about this claim, to be used in conjunction with the scheme. Can be a hash

	of the data behind this claim or the actual data itself.
uri	The location of this claim, examples: ● IPFS hashes ● HTTP links ● ...

Virtual Asset Service Providers

Virtual asset service providers, or **VASPs,** are entities that deal with the transmission of virtual assets; usually these are financial institutions transmitting and storing funds on behalf of their customers. VASPs play a vital role in today's modern age and will be one of the key actors of the Identity network[11].

Registered VASPs on the Identity network will be able to perform the following actions:

- Request payments from other VASPs on behalf of their customers, similar to an ACH or SEPA debit
- Send payments to other VASPs on behalf of their customers, similar to an ACH or SEPA credit
- Share transaction related data to comply with the FATF travel rule using a standardized format
- Share customer information related data to comply with KYC requirements

Identity

Since Identity builds upon the public-key infrastructure provided by Ethereum, each registered VASP will have a 160-bit public key that is linked to their smart contract, usually prefixed with “0x” for readability, which will act as the main identifier of the VASP.

Identifier Code

The main identifier is a rather long 160-bit hexadecimal code which can be cumbersome and inefficient to use, so we will use a routing code similar to how BIC/SWIFT codes identify financial institutions. This code, which we’ll be referring to as “VIC” (VASP identifier code), is assembled using the last 32 bits of the public key and allows for a more usable form of identifying VASPs.

160-bit VASP Identity
0x8dEB12D8774d22a53FdD1Be3314e9384851139BD

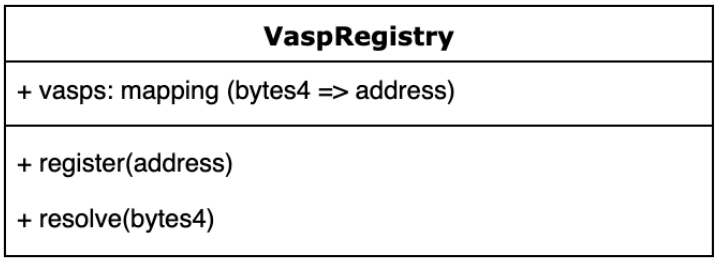
VASP Identifier Code (last 32 bits)

[11] https://www.openvasp.org/wp-content/uploads/2019/11/OpenVasp_Whitepaper.pdf

851139BD

Identifier Code Registry

As previously described, the last 32 bits of the main identifier will be used as the VASP identifier code. This does however create a potential attack vector where someone could try and generate a public key where the last 32 bits equal an existing VASP's identifier code. To mitigate this risk, we will be implementing a VASP Identifier Code Registry where VASPs register and effectively claim their identifier code. A unique identifier code can only be registered once. This code registry will be a decentralized registry in the form of an Ethereum smart contract using the following interface:



The register() function consumes the address parameter and adds the address to the VASPs list using the last 32 bit as the mapping key. The resolve() function consumes the 32 bit key and returns the matching address.

Account Numbers

Having already addressed generating "routing numbers" that allow routing funds/requests to the proper VASP using the VASP Identifier Code, the next step is defining a method of identifying which customer a request relates to. Traditionally this is accomplished with an account number that allows the VASPs to tie the request to a customer they manage. We will use a similar format, called the Virtual Assets Account Number or VAAN. This account number is generated using the existing VASP Identifier Code (**VIC**) and a 56-bit Customer Identifier (**CI**):

VAAN = VIC + CI + CheckSum8 Mod 256(VIC + CI)

Furthermore, a VAAN is assigned permissions:

- Debit: VAAN can send funds
- Credit: VAAN can receive funds

An example of a how a VAAN can be generated is displayed below:

VASP Identifier Code (last 32 bits)	Customer Identifier (56-bit)
8511 39BD	**AB09 3E9C 01DA 94**
Concatenated string	
8511 39BD AB09 3E9C 01DA 94	
Checksum8 Mod 256 of concatenated string	
89	
VAAN	
8511 39BD AB09 3E9C 01DA 9489	

The customer identifier can be chosen at random by the VASP. Typically a customer has a single account number, but it all depends on which level of privacy the customer wants to achieve. The VASP could go as granular as generating a unique VAAN per transfer in order to achieve a higher level of privacy.

Communication between VASPs

VASPs need to be able to communicate with each other over a secure and authenticated channel. Ethereum comes with a built-in messaging protocol called Whisper, which establishes a secure communication channel in a P2P environment and offers a high level of privacy as messages are encrypted using the public key of the recipient.

The general principle of Whisper consists of a network of nodes broadcasting encrypted envelopes to one another. Every participating node on the network will receive every single envelope transmitted by other nodes. However, since these envelopes are encrypted using the public key of the recipient, the recipient is the only node capable of decrypting the payload.

Decrypting every single message can be quite resource intensive, so a topic filter is used to make sure the VASP's Whisper node only decrypts messages sent to its topic. The topic we will be using is the VASP Identifier Code, the 32-bit short identifier explained above.

Travel Rule Compliance

Operating as a VASP comes with many obligations set forth by regulators. One is the **Travel Rule** defined by the Financial Action Task Force (FATF) or **Bank Secrecy Act** (BSA) defined by FinCEN. Identity will provide a platform to efficiently exchange information between **trusted** VASPs, so they can comply with the Travel Rule enforced by the FATF and FinCEN.

The key characteristics behind this are:

A. A standardized data format regarding originator and beneficiary information;
B. A protocol that defines the data exchange between VASPs.

Features

Requesting payments

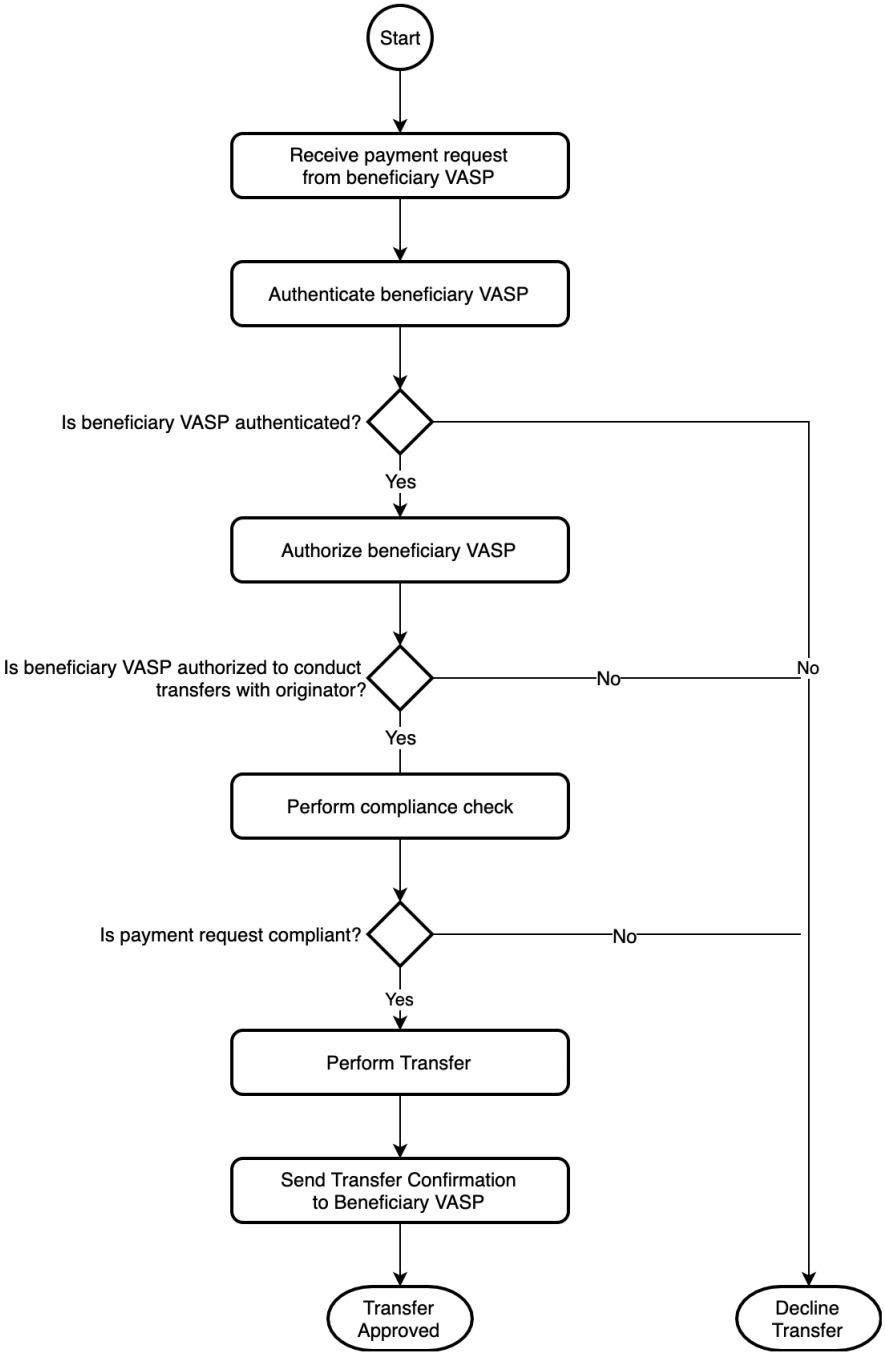
A beneficiary VASP can request funds on behalf of another customer from another originating VASP. In traditional payments systems this could be compared to an ACH or SEPA debit request. A similar functionality will be provided within Identity, with the added benefit of being able to facilitate asset agnostic payment requests.

A beneficiary VASP can initiate a payment request using the VAAN provided by the originating VASP, similar to how merchants can collect payments using the account and routing number on the ACH payments network.

Imagine Acme, Inc. wants to collect a bill payment from Alice Doe. This would mean the beneficiary VASP used by Acme, Inc. would simply need the VAAN belonging to Alice Doe.

Alice Doe using the ACH network	
Routing number	**Account number**
011103093	128438194

Alice Doe using the Identity network	
VASP Identifier Code	**Customer Identifier**
8511 39BD	AB09 3E9C 01DA 94
VAAN	
8511 39BD AB09 3E9C 01DA 9489	



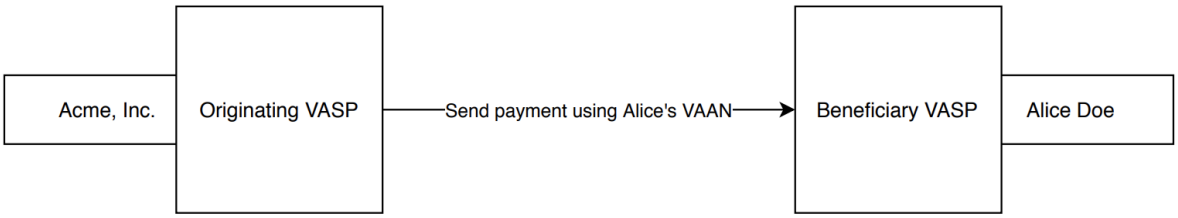
The first step of fulfilling a payment request is the beneficiary VASP sending the standardized payment request to the originator VASP. This data format will also contain KYC information specific to the beneficiary recipient of this payment request. The originating VASP will then authenticate the request using the signature of the incoming request and the VaspRegistry. Assuming the authentication passes, the originating VASP will authorize the request. It is up to the originating VASP to build their own policies surrounding the authorization; an originating VASP may maintain a blacklist of other VASPs it will decline by default, etc.

A compliance check will follow the successful authorization of the payment request by the originating VASP. This involves making a risk assessment based on the KYC information provided by the beneficiary VASP. When the risk assessment checks out, the transfer will be facilitated, and a transfer confirmation will be sent to the beneficiary VASP in a standardized format.

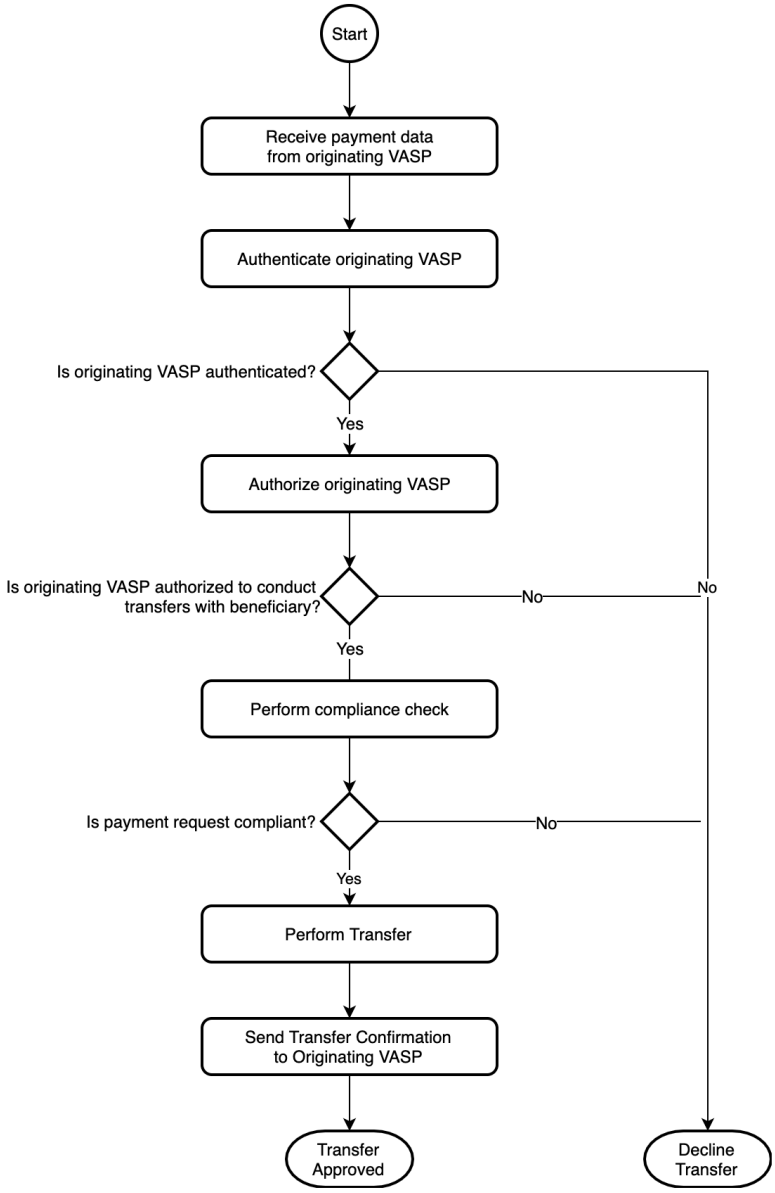
An additional step that has been omitted, since it depends on each specific use-case, would be the originator approving or declining the payment request. This is not usually done in traditional payment methods, but might be desirable in some cases. This does however impact settlement times in a negative way, since the originating customer would need to have additional time to review the transfer.

Sending payments

Sending payments involves the originating VASP pushing funds into the beneficiary VASP by using the VAAN of the beneficiary. It works very similar to how the payment request flow works, however in this case the initiating party is reversed. Following our previous example, assumeAcme, Inc. wants to send a refund to Alice Doe. Acme, Inc. would use Alice's VAAN to push funds into Alice's virtual asset account.



The process flow can be described as seen below:

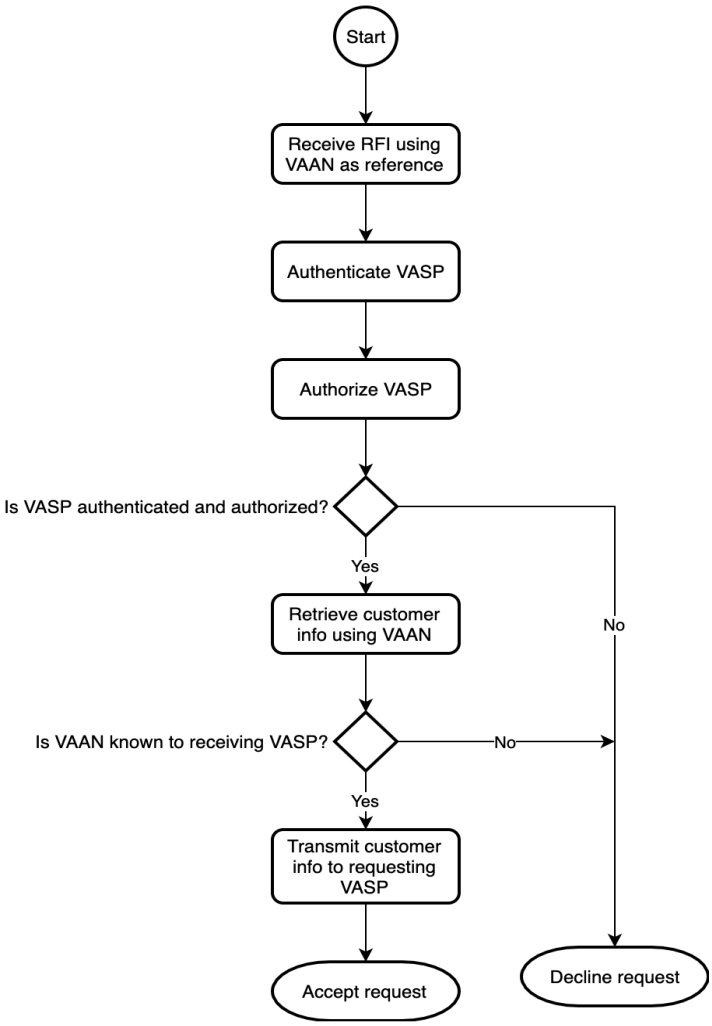


Request for Identity

In some cases a participating VASP may want to fetch the identity behind a certain VAAN. This can be done using a Request for Identity request, or RFI in short. A VASP can initiate this request any time but the VASP receiving the request may decline the request at their own discretion.

Valid use cases for a request for identity:

- During a fund transfer between beneficiary and originator
- As part of an AML investigation
- As part of a KYC process



Flagging a payment

Any identity can issue a payment flag with several codes, which can act as a digital watermark preventing a payment from being further propagated by an institution. The payment flags act as a way for individuals and institutions to communicate where a payment is potentially problematic in the areas of: theft, money-laundering, terrorism, or any other form of criminal activity.

Regardless XID network remains agnostic and will process payments, however nodes may decide not to process transactions that are tagged with a payment flag. Additionally, VASPS may choose not to process or honor transactions that originate from an address that have a payment flag attached.

Payment flags consist of (a) stolen funds (b) currently under investigation (c) money-laundering (d) terrorism or OFAC listed (d) please inquire.

Link external funding sources

The initial goal of XID is to collaborate with existing payment networks and operate alongside them. To accomplish this we will provide a mechanism for linking external funding methods. Whenever a funding method is successfully linked, a matching VAAN will be associated to it so funds can be pushed and pulled from the source.

A common use case is to link external bank accounts such as ACH/SEPA so the beneficiary VASP can pull funds into an account managed by it. Another common use case is for VASPs initiating the request to link a funding method to disperse micro-deposits to make sure the requesting customer has access to the funding method. This is not necessary if the identity of the requesting user is known to the receiving VASP and is able to verify the signature of the request. Otherwise, the customer may be required to verify they are the legitimate owner of the funding method by verifying two small deposits made to their account. This may not be necessary in every use case, but should be determined based on a risk score.

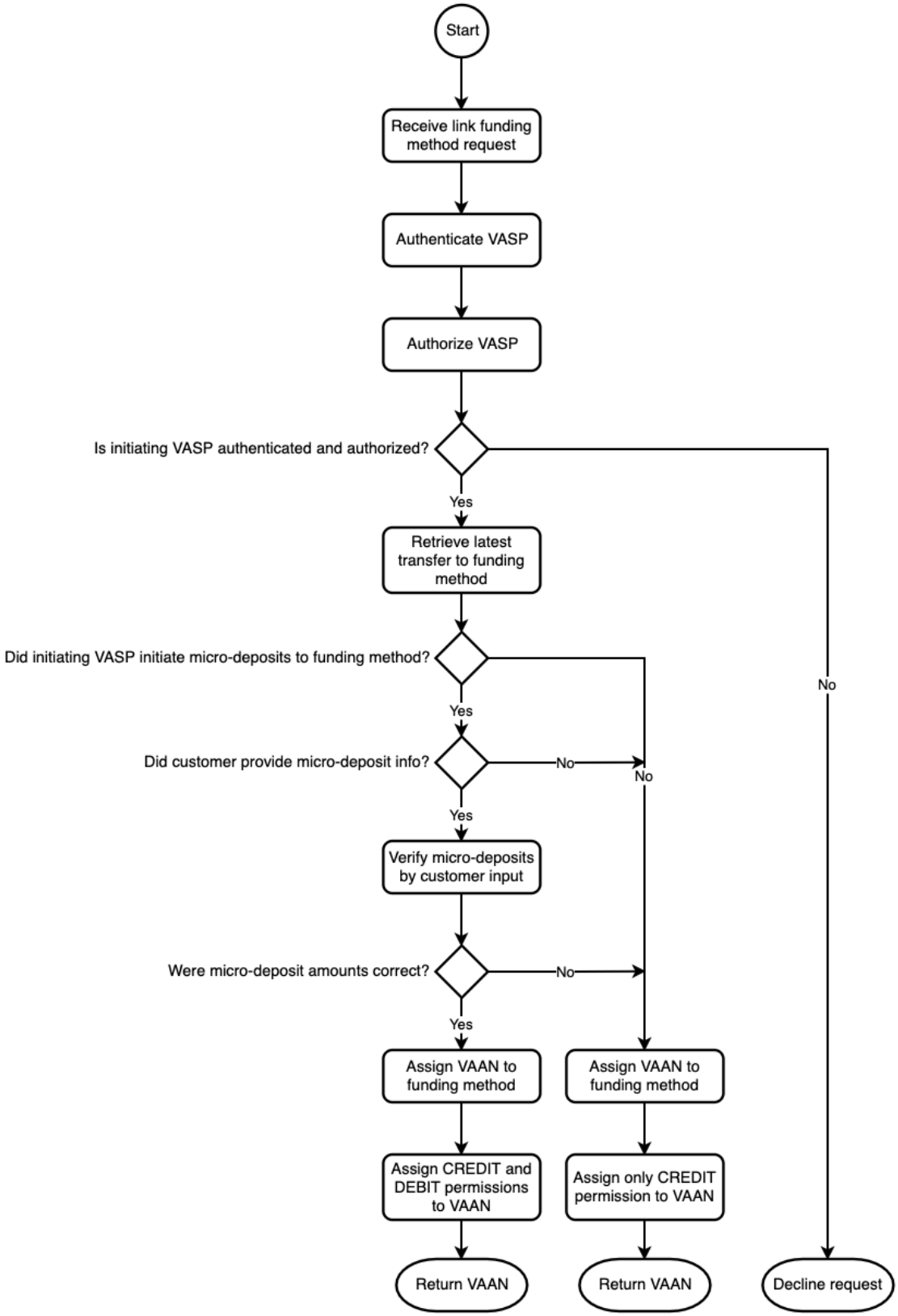
Let’s continue with the example where Alice is trying to link her bank account into the Metal Pay application. Metal Pay and Alice’s bank are both registered VASPs on the network. Metal Pay would request Alice’s bank to link her bank account using the account and routing number provided. When Alice’s bank authorizes the request it will issue a VAAN specific for Metal Pay and Alice’s bank account. This way if Alice claims it was a fraudulent request, Alice’s bank can just mark the VAAN as deactivated and effectively revoke the authorization.

Alice's bank account	
Routing number	**Account number**
011103093	128438194
Alice's bank issues new VAAN specific to Metal Pay	
8511 39BD AB09 3E9C 01DA 9489	

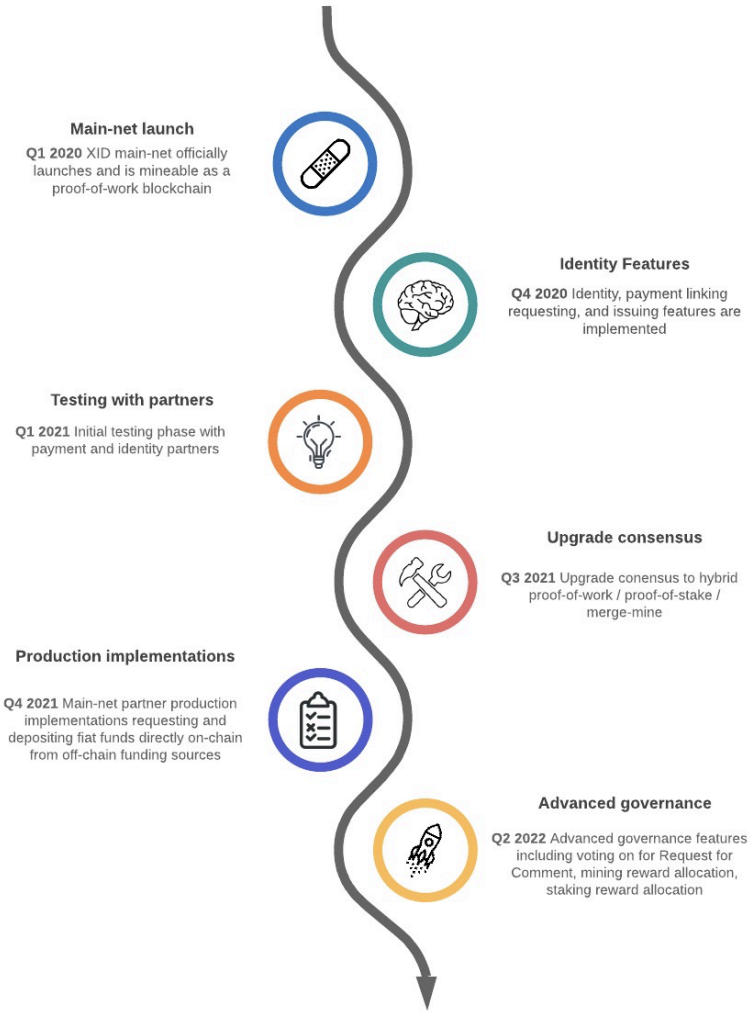
All payments and payment request to and from this VAAN will be handled by Alice's bank over the ACH network while still utilizing the same communication channels to update participating VASPs.

Initial funding method types supported by Identity:

- ACH
- SEPA
- EFT
- CARD



XID Initial Development Timeline



XID Token

Allocation