# Specification White Paper

[Alex Batlin](file:///C:\wiki\display\~alex.batlin)

# Abstract

One of Enterprise Ethereum Alliance's (EEA) primary goals is to develop a holistically cohesive Enterprise Ethereum Specification (EES) that compliments the Ethereum Foundation (EF) baseline offering i.e. geth codebase, EIPs and papers; to bridge the requirements gap between public blockchain and the needs of enterprises. To account for EEA's diversity, EES is not expected to be a single coarse-grained specification document, but rather a corpus of fine grained, code backed and working group (WG) defined Enterprise Ethereum Proposals (EEPs), which can be assembled into multiple configurations to meet different requirements, as defined by relevant domain experts. Each EEP specifies how its competing factor's offering levels e.g. confidentiality, are different to the EF's baseline. Software vendors are free to implement some or all of the EEPs, depending on which end user configurations they wish to target. This paper intends to offer advice and guidance to EEA's WGs as they start to work on EEPs.

# Introduction

EEA is a member driven organisation, which means it's up to the members to drive EEPs. There is no one better than the context domain expert to create a set of requirements, and no one better than an Ethereum client developer to develop an EEP and backing code that meets those requirements. In a cross-border payments example, a Banking industry WG's legal, regulatory and business domain experts may define its requirements and collaborate with the Quorum client WG, to write a code-backed specifications (EEPs).

EEA is an agile organisation, which means that members should not wait for someone to create some high level specifications or frameworks, rather they should get on with the work of defining their EEPs within WGs as soon as they identify a need. Since WG's charters are approved by the EEA Board, with assistance from the Governance Working Group (GWG), charter overlaps should be spotted and resolved at charter review stage. Furthermore, EEPs that do not modify EF's client codebase can be self-approved by EEA's WGs. Those that do, must be approved by the Technical Steering Committee (TSC), working on behalf of the EEA Board.

It's likely that some EEPs can be reviewed and approved by TSC members during regular meetings, yet others may require follow up work such as general advice, improvement proposals, clarification requests, ghost writing, reasoned rejections, merge arbitration, re-factoring supervision and proposal deprecation. The Standards Working Group (SWG) is there to carry out such works when requested by the TSC. It will do so by collaborating with the WGs that are submitting the EEP for approval. Whilst WGs are not required to seek approval for codebase changing EEPs, they may still submit the EEP to TSC for comments and advice.

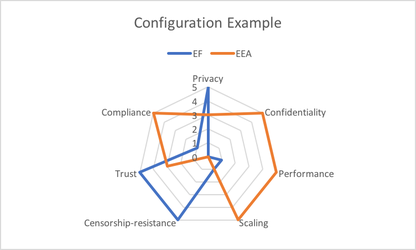
SWG will not create sub-working groups to define all possible factor specific requirements up-front. Instead, it will consider each EEP on it's own merits against already approved EEPs; merging, re-factoring and deprecating as needed on a per-proposal basis. A sub-working group will only be formed as and when a need arises to focus on a set of specific factors. It will be disbanded as soon as it's goals are met.

SWG is expected to publish white papers, such as this one, reference material and policies, which once approved by the TSC, should guide and streamline WGs proposal efforts. For example, since there is ambiguity as to what actually constitutes Ethereum Foundation's canonical corpus of specifications, the SWG should maintain it's own baseline reference list. The SWG should also be striving to achieve EEA's long term EF re-integration aspiration, by submitting approved EEPs as EIPs, assuming they are vision-aligned.

# Process

This is an outline of the proposed high level process:

* WG identifies a need use-case e.g. documentary collection trade finance process requires near-real-time cross-border fiat payments.
* WG identifies relevant domain subject matter experts (SMEs) e.g. business, legal, regulatory, finance, accounting, technical, operations.
* SMEs break down the need use-case into a set of requirements.
* SMEs map requirements to competing factors e.g. EEA's codebase may compete with EF's on confidentiality, performance, scalability, censorship.
* SMEs define competing factors offering levels required by their need e.g. confidentiality and performance may need to be high as compared to EF's.
* SMEs calculate their required client codebase configuration e.g. {'confidentiality': 5, 'privacy': 3, ...}, see radar diagram below.
* SMEs consult EEA's client code configuration database to see if any vendor or open source project already supports their client codebase needs.
* If there is a supporting codebase, good news, the WG can focus on defining and self-approving use-case specific standards e.g. configuration options and smart contract interfaces.
* Otherwise, the WG not only has to define use-case specific standards, but also client codebase changing EEPs, either on it's own, if it has the relevant technical skills, or in collaboration with one or more client WGs, SWG and TSC.



# Domains

It is rare for a single individual to know all of the requirements for a specific use case. Take banking for example. Front office folks will know very well their needs when concerning OTC trades, but they are un-likely to know all the requirements that are placed on them by law and regulation. Nor are they aware of the middle and back office operations frictions. And operations folks do know abut technical frictions, issues and risks. So, in order to define a complete set of requirements, you need to take a holistic view by consulting SMEs from all relevant to use-case domains, so that you can create define cohesive specifications. While the domain list below is anything by exhaustive, it should be a good starting point for EEP authors to build on:

|  |  |
| --- | --- |
| **Domain** | **Description** |
| Legal | What do you need to do in order not to go to jail or be sued. |
| Regulatory | What do you need to do in order not to loose your operating license. |
| Finance | If you can't make it for itself, it ain't going to work. |
| Accounting | Make it easy to account for your activities or face high costs of doing so or worse loose your license. |
| Audit | If it can't be audited and reported, it can't be approved by regulators. |
| Tax | Death and taxes, so make it's easy to pay them or they will be the death of you. |
| Operations | Make sure you can operate the solution in cost efficient manner. |
| Business | Get to know you business functional and non-functional requirements, they drive your revenues. |
| Technology | Makes sure your requirements are technically implementable. |
| Risk | Consider all of the domains from what-can-fail point of view, and propose mitigation requirements. |

# Factors

EF's geth client codebase can be described by a set of features. For example, it supports very high level of privacy through the use of pseudonymity, after all if no one knows who you are, then you can do whatever you want, and then simply change your identity. So whilst everyone can observe what you do, since you can change your face at any time, just like the Faceless Men in Game of Thrones, you actually get a lot of privacy. Current banking systems are the opposite, they know exactly who you are and what you do, but they offer high level of confidentiality i.e. they will not tell others what they know about you, unlike public Ethereum, where everyone knows what is happening.

Features can thus be split into factors e.g. privacy and confidentiality, and their offering levels e.g. high, medium, low or non-existent. This helps to compare similar offerings e.g. EF's geth codebase versus EEA's configuration X codebase, by treating factors as competing offerings, and comparing them by the offering levels. See Configurations sections for a template table. Below is the authoritative set of factors and EF's baseline offering levels. As new factors are discovered, the table below must be updated.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Factors / Levels** | **EF** | **Description** | **0** | **1** | **2** | **3** | **4** | **5** |
| Privacy | 5 | Does anyone know who are you and what you do? | No |  | | | | Yes |
| Confidentiality | 0 | How easy it is to share some data with a sub-set of participants and make sure others can't access that data. | No |  | | | | Yes |
| Performance | 1 | How quickly can a transaction be settled e.g. 10 mins minimum for Bitcoin, 12 seconds minimum for Ethereum. |  | 10 mins | tbc | tbc | tbc | 1 sec |
| Scaling | 1 | How many transactions can be settled within a specific time frame e.g. 1 second. |  | 15 tx/s | tbc | tbc | tbc | 100k tx/s |
| Censorship-resistance | 5 | Can a central authority prevent miners confirming transactions, or unilaterally increase money supply? |  |  |  |  |  |  |
| Trust | 5 | How likely is it that someone can subvert 51% of the network i.e. number and independence of nodes matters. |  |  |  |  |  |  |
| Compliance | 1 | This is a tricky one, but largely can the system support existing regulations. Needs more work. |  |  |  |  |  |  |
| Finality | 3 | Can transactions be reversed, and if so how long for and what are the chances of doing so. |  |  |  |  |  |  |

# Configurations

A client code configuration specifies competing factor's offering levels. Software vendors can quickly advertise to their users which configurations their client software supports. Users can quickly decide which client to use by mapping their requirements to competing factor's offering levels, aggregating the results into a configuration, and searching for vendors that support that configuration. Below is a template configuration table that can be used by both software vendors and end users to match needs to clients.

|  |  |  |
| --- | --- | --- |
| **Competing Factors / Offering Levels** | **EF** | **EEA Configuration X** |
| Privacy | 5 |  |
| Confidentiality | 0 |  |
| Performance | 1 |  |
| Scaling | 1 |  |
| Censorship-resistance | 5 |  |
| Trust | 5 |  |
| Compliance | 1 |  |

# Example

Let's use cross-border fiat payments in a documentary collection trade finance process as an example. Below are the kind of outputs that an EEP author should consider producing:

## Need Use-Case

Importers need to pay exporters for purchased goods. They two do not necessarily trust each other i.e. exporter does not trust the importer to give them money once they hand over the rights to their goods, and visa-versa. In such cases, they two counter-parties may choose to buy a documentary collection service from trusted third parties, such as banks.

A documentary collection is a trade transaction in which the exporter hands over the task of collecting payment for goods supplied to his or her bank, which sends the shipping documents to the importer's bank together with payment instructions. In other words, banks facilitate delivery-versus-payment atomic transactions i.e. money and goods will only be exchanged together or not at all.

Today's cross-border payments can be slow to settle, and expensive to execute. When they are slow to execute, this causes liquidity issues for the payer, and introduced settlement risk. One of the reasons for delays is reliance on correspondent banks. For example, the importer banks with Bank A in country X, whilst the exporter banks with Bank B in country Y. The two banks do not have a direct relationship via nostro-vostro accounts to settlement payments. However, there is a Bank C that has branches in both X and Y countries. Both countries have central banks with RTGS systems that all licensed commercial banks have direct or indirect access to. So, Bank A can send a payment instruction to Bank C, settling it via Central Bank X, Bank C will then perform an internal book transfer between its X and Y branches, and then send a payment instruction from itself to Bank B, settling it via Central Bank Y. Most banks still clear and settle transactions in batches, so if each hop is cleared and then settled in 24 hours batches, it may take up to 72 hours to complete. Further, since each payment instructions is independent of the originating transactions, no single bank can tell the customer where the payments is at if it gets stuck, they can only report once the final confirmation makes its way back along the chain. SWIFT gpi promises to improve tracking and reporting, but not necessarily the sequential batch processing time delays.

Today's documentary collection process is highly manual, slow and expensive. Human operators often receive posted or faxed paper documents such as bills of lading, a legal document between the shipper of goods and the carrier detailing the type, quantity and destination of the goods being carried. They inspect that everything is ok, and transfer the ownership of goods by re-assigning the bill of lading to the exporter, only once the payment is received by the importer. If any leg fails, they reverse both i.e. cancel ownership transfer and payments. They may or may not hold both the title and money in escrow for a short period of time to allow for transaction reversal.

What is needed is much faster and cheaper system, to both reduce liquidity impact, reduce settlement risk and reduce transaction costs. Automation is one key driver to achieving these needs. By converting bills of lading from paper or electronic unstructured documents to structured data documents, humans can be replaced by code to perform verification checks. If an asset ledger existed that could allow code, not humans, to escrow assets and only transfer them if swap conditions are met, then the escrow and atomic swap processes can also be automated. However, to achieve that, not only the bill of lading assets must be managed on such a system, but also fiat money. Once such a system is available, it should allow for automated straight-through-processing with settlement finality, all within several seconds, and at a low cost. Not much to ask for!

## Requirements

Table below details the requirements for the need use case along with competing factor offering levels.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Domain** | **Requirements** | **Performance** | **Scaling** | **Confidentiality** | **Privacy** | **Censorship** | **Trust** |
| Legal |  |  |  | Yes | Yes | 3 | 3 |
| Regulatory | REQ3 - Payments transaction must only be seen by counter-parties   * requires client codebase changing EEP2 | n/a | n/a | Yes | Yes |  |  |
| Finance |  |  |  |  |  |  |  |
| Accounting |  |  |  |  |  |  |  |
| Audit |  |  |  |  |  |  |  |
| Tax |  |  |  |  |  |  |  |
| Operations | REQ1 - Support 15 tx/s | 1 | n/a | n/a |  |  |  |
|  | REQ2 - Settlement to take no longer than 10 seconds | n/a | 1 | n/a |  |  |  |
|  | REQ4 - Use ISO20022 payments messages standards   * requires smart contract EEP1 but no codebase changes | n/a | n/a | n/a |  |  |  |
| Business |  | 500 tx/s |  |  |  |  |  |
| Technology |  |  |  |  |  |  |  |
| Risk |  |  |  |  |  |  |  |
| **Required** |  | 1 | 1 | 5 |  |  |  |
| **EF** |  | 1 | 1 | 1 |  |  |  |

## Proposals

|  |  |  |  |
| --- | --- | --- | --- |
| **EEP** | **Headline** | **WGs** | **TSC** |
| 001 | ISO20022 compatible payments smart contract interface | Banking | No |
| 002 | Support private payment transactions | Banking, Quorum, Standards | Yes |

# Terms

* Enterprise Ethereum Alliance (EEA) - an industry-supported, non-profit established to build, promote, and broadly support Ethereum-based technology best practices, open standards, and open source reference architectures
* Enterprise Ethereum Specification (EES) -
* Ethereum Foundation (EF) -
* Enterprise Ethereum Proposal (EEP) -
* Governance Working Group -
* geth - abbreviation for "Go Ethereum"; also the command-line interface
* codebase -
* Ethereum Improvement Proposal (EIP) - document describing standards and specifications of Ethereum; can have a status of draft, accepted, final, or deferred.
* public blockchain - a decentralized and distributed blockchain without membership restrictions
* enterprise -
* working group (WG) -
* domain -
* configuration -
* Standards Working Group (SWG) -
* Software Vendor (ISV) -
* Technical Steering Commmitee (TSC) -
* refactoring - process of changing pieces of code to improve nonfunctional characteristics (complexity, readibility) with the intention of reducing complexity around maintenance
* deprecation -
* baseline -
* subject matter expert (SME) -