Valar

Platform Overview

An overview of Valar - a decentralized peer-to-peer staking platform.

Keywords – decentralized, peer-to-peer, staking, delegation, platform

Uroš Hudomalj, Alexander Marinšek

15 January, 2025

Contents

List of Figures

Executive summary

Valar is a decentralized platform for connecting blockchain stakeholders to node runners that offer direct participation in blockchain consensus, i.e. staking. The Platform is based on the peer-to-peer (P2P) direct staking approach, which enables blockchain stakeholders to gain access to blockchain staking rewards while maintaining full custody of their assets. By leveraging smart contracts, the platform enables node runners to be compensated for their services directly by their customers, ensuring transparent collaboration with incentives for all parties. The Valar platform is launched on the Algorand blockchain.

The Valar Platform consists of: Valar Smart Contracts, which manage the collaboration between blockchain stakeholders and node runners; the Valar Daemon, which automates node runners' tasks; and the Valar user interface (UI), which provides a seamless experience for users. This document outlines the Valar Platform's architecture, fee structure, and user journeys to provide an overview of Valar's core functionality. In addition, the document provides an overview of the platform's branding information.

Disclaimer

Chapter 1

Introduction

The invention of the Internet brought people closer to one another by enabling global communication. The invention of blockchain technology goes a step further by enabling people to not only communicate and exchange information but also to reach agreements (i.e. consensus) and to trust these agreements, paradoxically by not having to trust one another. This is possible due to modern cryptography and the principle of decentralization, where as long as the majority of people are honest, the system will be honest. To increase the difficulty for a malicious actor to corrupt the system, blockchain networks strive to get as many independent entities from around the globe to contribute to the network's security. For contributing to the network's security, blockchains provide incentives, commonly called staking rewards.

From a technical perspective, the act of contributing to the network's security is done by validating transactions, i.e. the information that is exchanged on the network, and voting on what one thinks is the truth. The entities that are doing this are commonly referred to as Validators. The process of validating transactions is done by a computer program, which the Validators must operate, i.e. run. The computer that is running such a program, is commonly referred to as a node. This is why Validators are also known as node runners.

To prevent a malicious actor from simply corrupting the system by running the majority of nodes, the Proof of stake (PoS) blockchain networks, such as Algorand, give higher voting power to the entities that have more funds on the network, i.e. larger stake in their blockchain account. The reasoning behind this is that entities that have more funds on the network, have more to lose if they behave maliciously, which other entities will see and act accordingly due to the transparent nature of the blockchain technology.

Not all blockchain stakeholders might have the time, desire, or technical capabilities to operate a node, i.e. be a Validator. However, they can still participate in the network's security by authorizing someone else to represent them on their behalf, i.e. to delegate their stake to a Validator. Entities that delegate their stake to someone else are referred to as Delegator.

1.1 Staking on Algorand

Late 2024 marks the switch to staking rewards for Algorand – accounts that meet the network's requirements and participate in consensus will be eligible for staking rewards. The Algorand Governance vote during Period 10 decided one of the requirements to be to have at least

30k ALGO. This is not to overwhelm the network communication by having millions of nodes that all need to communicate with one another. ALGO owners in possession of an Algorand participation node are able to stake on their own behalf, as well as on behalf of other users. The Algorand community estimates that 3-4 accounts can participate on one participation node, depending on the node's performance. While accounts with less than 30k ALGO are not eligible for staking rewards, they can participate in consensus.

Accounts that own less than 30k ALGO and want to gain access to staking rewards can still do so through stake pooling or liquid staking. Using these, ALGO owners can aggregate their ALGOs at a single blockchain account, which can become eligible for staking rewards due to a balance of over 30k ALGO. However, pooling can decrease network decentralization, since the funds of several persons are gathered at a single, central location – the pool's account, therefore node[1]. While a person with a large amount of ALGO could occupy an entire pool, their funds still have to leave their wallet and enter the custody of the pool or liquid staking provider. This limits the person's freedom in managing their funds and may pose both a security and a legal risk.

Direct participation through the nodes of others avoids the drawbacks of stake pooling and liquid staking, since the stake remains in the owner's custody[2] and is not aggregated at a common account. However, setting up and maintaining the collaboration[3] between a Validator and the owner of the ALGO, i.e., the Delegator, can be challenging and time consuming. To enable secure and transparent collaboration between Validators and Delegators, we have developed Valar - a decentralized peer-to-peer (P2P) staking platform, the Valar Platform in short.

1.2 Valar Peer-to-Peer Staking Platform

The Valar Platform enables Delegators to find Validator to participate in consensus on their behalf, i.e. stake. This gives the Delegators access to staking rewards while they maintain self-custody of the staked funds. The process is outlined in Figure 1.1.

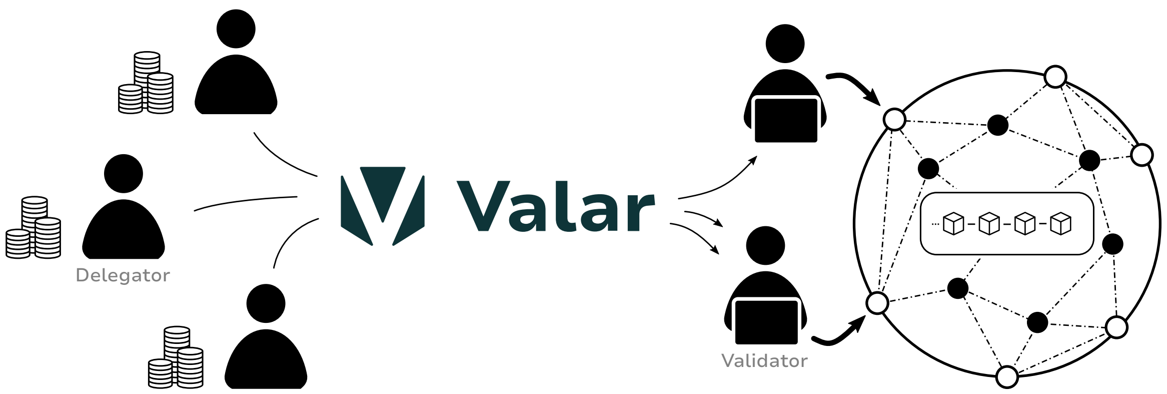

Figure 1.1: The Valar Platform connects Delegators to Validators, enabling P2P delegation and self-custodial staking.

[1]An account can participate in consensus through a single node at a time

[2]Algorand allows self-custodial staking, where the staked funds never have to leave the wallet of the owner of the funds.

[3]A collaboration between a Delegator and Validator refers to the Delegator (the one with the ALGO) delegating consensus participation to the Validator (the one with the node).

Validators can advertise their node running service through the Valar Platform. These advertisements are accessible to anyone, allowing Delegators to browse the offered services and select one of their liking. Moreover, the Valar Platform simplifies the collaboration between Validators and Delegators through the usage of pre-defined smart contract templates, which integrate the collaboration management logic. The Valar Platform is characterized by the following features:

- Peer-to-peer – Delegators connect directly with Validators.
- Delegated – Validators participate in consensus on behalf of Delegators.
- Self-custodial – The staked ALGO never leaves the wallet of the Delegators
- Incentivized – Delegators can receive staking rewards and pay Validators for their service.
- Transparent – The collaboration is established on-chain using immutable smart contracts.
- Decentralized – The platform is available to all.

1.3 Document Structure

The following chapters of this document provide a comprehensive overview of the Valar Platform, focused primarily on the platform's design and users. The contributions of each chapter are summarized as follows:

- Chapter 1: Introduces staking on Algorand and P2P staking.
- Chapter 2: Summarizes of the Valar Platform's architecture and its three components.
- Chapter 3: Outlines the fee structure applied on the Valar Platform.
- Chapter 4: Lists the user roles on the Valar Platform and describes each role.
- Chapter 5: Details the core of the Valar Platform – its smart contracts.
- Chapter 6: Describes each user's journey and the actions they take on the Valar Platform.

In addition, the appendix provides an overview of the the brand definitions.

Chapter 2

Components

The core of the Valar Platform is implemented using immutable smart contracts. Additionally, the Valar Platform consists of the Valar user interface (UI) for easier user interactions with the smart contracts, and the Valar Daemon for automating the validator's tasks. Figure 2.1 outlines the three components, further described in the below subsections.

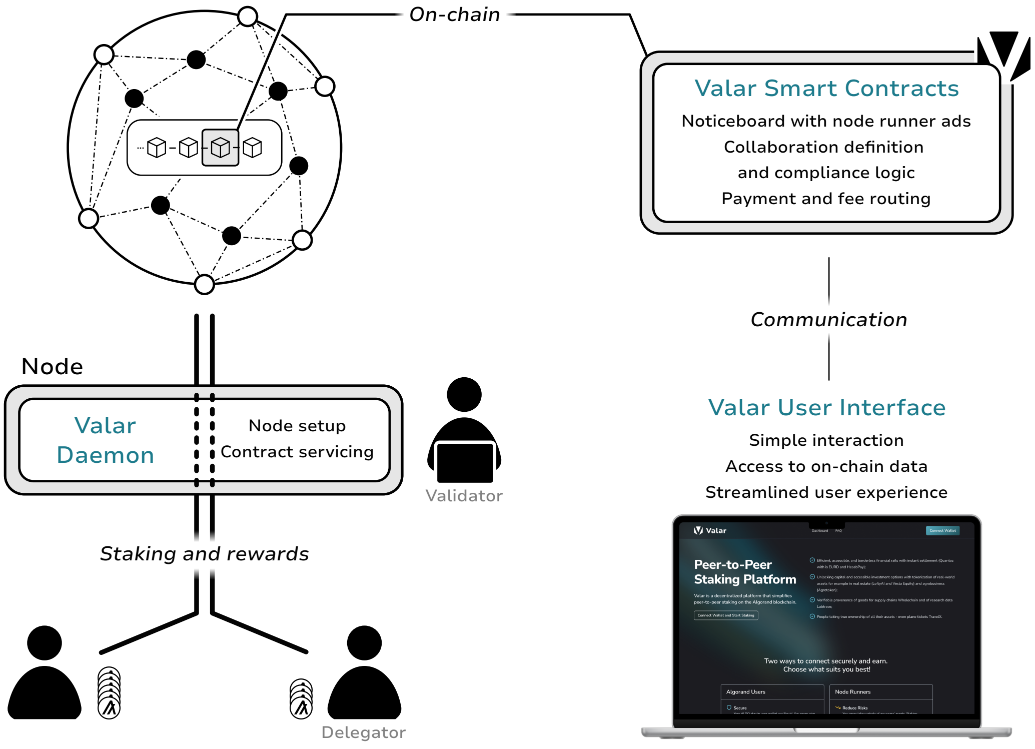

Figure 2.1: An overview of the three components that constitute the Valar Platform.

2.1 Valar Smart Contracts

The Valar Smart Contracts are the core of the Valar Platform, enabling the advertisement of node running services and a streamlined collaboration between Validators and Delegators. These contracts allow Validators to showcase their nodes and collaboration terms while also automating operations such as payment routing, compliance tracking, and contract enforcement. The

Valar Smart Contracts provide a transparent and secure mechanism for managing agreements, minimizing administrative overhead, and ensuring accountability among participants. They save stakeholders' time in establishing and executing a fruitful collaboration.

The Valar Smart Contracts are developed in Python and compiled to TEAL using the Algorand developer tools (i.e. PuyaPy). Implementation details are provided in Chapter 5.

2.2 Valar Daemon

The Valar Daemon is a program that is run by Validators, typically on their advertised nodes, where the staking takes place. The program periodically monitors the Valar Smart Contracts and automates the Validator's tasks. For example, the Valar Daemon services incoming requests of Delegators on behalf of the Validator. This includes setting up the node for Delegators by generating participation keys for the duration of the staking and deleting the participation keys when the collaboration ends. The Valar Daemon also issues reports of breaches of the collaboration's terms and terminates the contract accordingly.

The Valar Daemon is implemented in Python and runs directly as such. While it is designed to assist the Validator, the usage of the Valar Daemon is not strictly necessary, as the Validator could manually monitor the Valar Smart Contracts and execute the required tasks or employ any third-party daemon to automatize their process. Section 6.3.1 provides more information on setting up a node while a guide regarding the Valar Daemon is in development.

2.3 Valar User Interface

The Valar UI is a web application for more convenient access to the Valar Platform. It allows Validators to publish and manage advertisements and the Delegators to start a service contract under their selected advertisement. Both Validators and Delegators can track the progress of their collaboration through the Valar UI. The Delegator can also extend or terminate the collaboration.

The Valar UI is designed using Node.js and deployed to a publicly accessible web address. In addition, the source code can be downloaded from the public Valar repository and run locally, while still accessing the Valar Smart Contracts deployment on the Mainnet, enabling fully independent and decentralized operation.

Chapter 3

Fee Structure

The Valar Platform empowers both Delegators and Validators. The Delegators are able to receive staking rewards directly from the Algorand network into their wallet, without any intermediary, while the Validators can request payment from the Delegators for their service. Additionally, the Valar Platform and its Partners are able to receive a commission for the provided convenience. The described fee and asset flows are outlined in Figure 3.1.

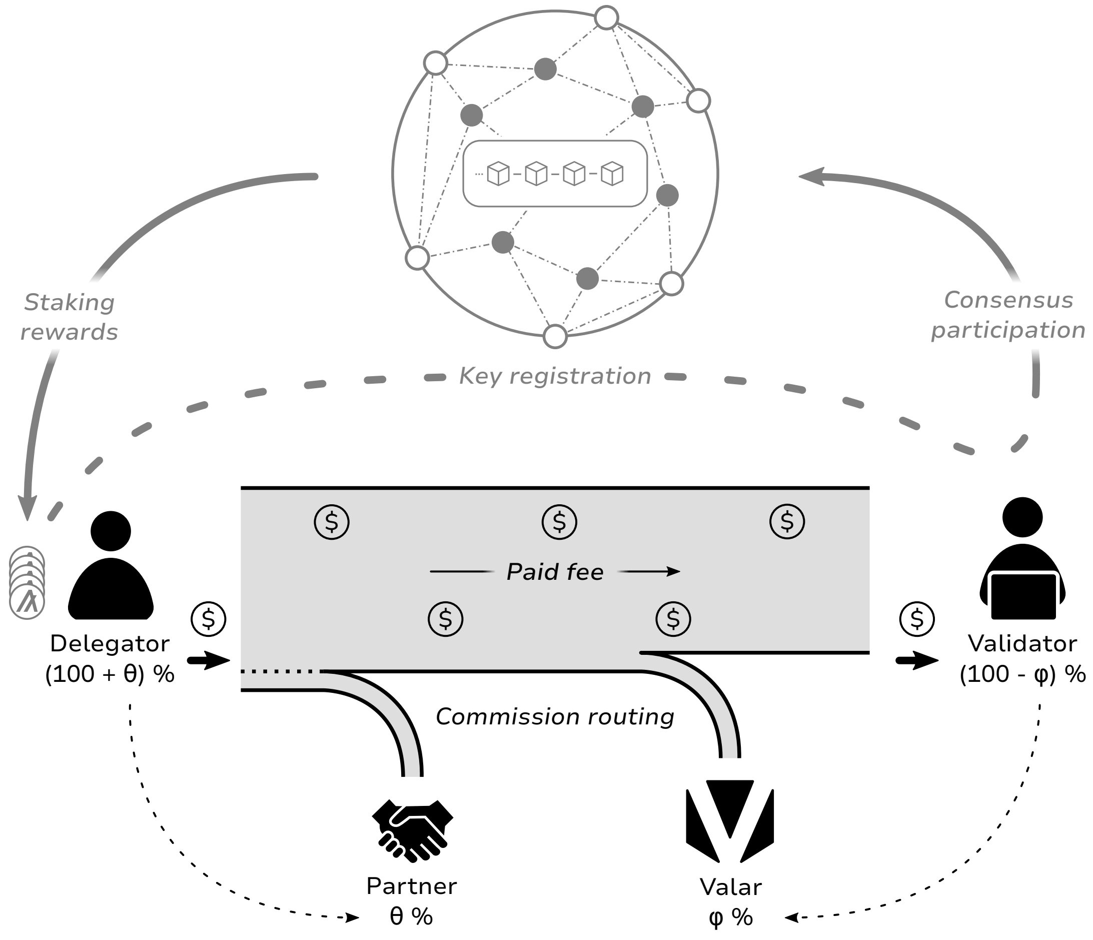

Figure 3.1: Overview of the fees and asset flows. Variables θ and φ denote the 'on-top' commission for the Partner and the subtracted commission for the Valar Platform, respectively. Staking is done using the Delegator's ALGO, while the Delegator carries out payments using a stablecoin.

Normal working conditions, with compliance of all parties, are assumed in the descriptions in the following subsection. Note that only the revenue stream from P2P staking is addressed here.

3.1 Commission Model

A commission (marked by φ in Figure 3.1) for the Valar Platform is automatically subtracted from the fees charged by the Validator. The size of this commission is defined by the Validator and must be within the allowed bounds of the Valar Platform, described in Section 5.1. In addition, a Partner may charge its own commission for referring Delegators to the Valar Platform. The commission for Partners (marked by θ in Figure 3.1) is charged as a percentage on top of the fees paid to the Validator.S Both commissions are automatically routed to the corresponding recipient in the designed Valar Smart Contracts, detailed in Chapter 5.

The Validator fees consist of two parts - a setup fee and an operational fee. These are described in more detail in the following subsections.

3.2 Setup Fee

A Delegator normally selects a node running service of their liking and starts a contract with the corresponding Validator. The Validator then prepares the node so that the Delegator can participate in consensus through it. The necessary setup consists of generating participation keys for the Delegator for the duration of staking (this procedure is further described in Chapter 6).

The Validator charges a setup fee to cover the incurred costs for generating the participation keys. The charged amount is defined by the Validator. The payment is transferred to the Validator once the Validator submits the generated keys to the Valar Platform for the Delegator to use. The payment cannot be reimbursed thereafter. Key generation is a computationally-intensive task, thus the setup fee also acts as a spam prevention mechanism. It is foreseen that Validators will set the setup fee to about 1 USD.

3.3 Operational Fee

Besides charging the setup fee, the Validator additionally charges the operational fee, which corresponds to the duration of the service contract. The amount is paid upfront by the Delegator for the entire duration of the staking. It is locked in the smart contract associated with the specific collaboration. It gets gradually unlocked over time. In case of early termination of the collaboration, any remaining portion of the operational fee is transferred back to the Delegator (this mechanism is further described in Section 5.3.2.8). It is foreseen that the operational fee could range from several USD to a hundred or more USD per month, while It is up to each Validator to determine the size of both the setup fee and the operational fee.

The size of the operational fee is governed by the Validator using two parameters:

- β, the minimal payment amount per unit of time and
- γ, payment factor per maximum staked ALGO[1] per unit of time.

[1]The maximum ALGO stake is considered the actual amount staked when determining the operational fee.

The operational fee payment amount, σ, is determined as follows

$$\sigma = \Delta \max(\beta,\ \lambda\gamma), \tag{3.1}$$

where Δ marks the duration of the staking and λ is the amount of staked ALGO. The term $\Delta\beta$ represents minimal operational fee and $\Delta\lambda\gamma$ is the variable operational fee. Notice that setting $\gamma = 0$ results in a constant price for all stake sizes, while setting $\beta = 0$ imposes no minimal payment requirement. Figure 3.2 shows the resulting piecewise linear function.

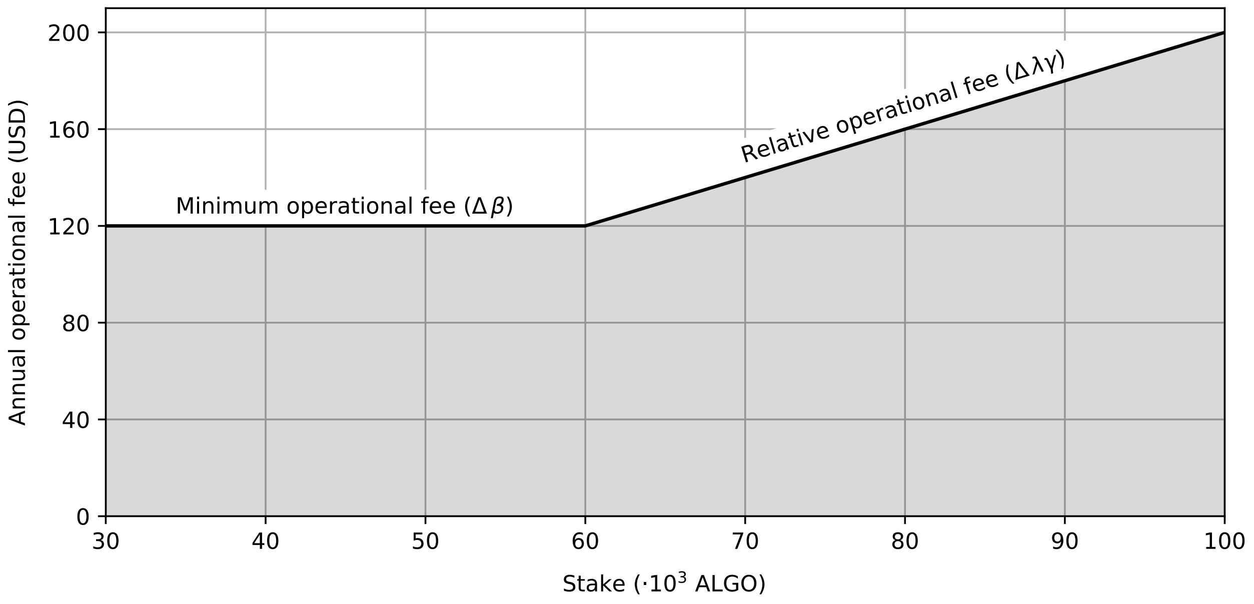

Figure 3.2: An example of determining the annual operational fee for a Validator Ad, dependent on the amount of staked ALGO (λ). The example considers the duration $\Delta = 1$ year, the minimal payment $\beta = 120$ USD/year, and the payment factor $\gamma = 0.002$ USD/ALGO/year.

Chapter 4

User Roles

The main two user roles of the Valar Platform are the Delegator and the Validator, i.e., the owner of the staked ALGO and the owner of the node used for the staking, respectively. The two are further split into two user roles each in order to provide an additional layer of security and modularity. The Delegator is split into the Delegator Beneficiary and Delegator Manager, while the Validator is split into the Validator Owner and Validator Manager. These user roles are described in more detail in the following sections together with the other possible user roles of the Valar Platform. The roles are depicted in Figure 4.1. The interaction between the users and the Valar Platform is detailed in Chapter 6, after introducing the user roles in this chapter and the Valar Smart Contracts in Chapter 5.

4.1 Delegator Beneficiary

The Delegator Beneficiary is the owner of the staked funds that participate in consensus and earns staking rewards. These rewards are directly in the account (i.e., wallet) of the Delegator Beneficiary. The Delegator Beneficiary maintains full and sole custody of the staked funds[1].

The main responsibilities of the Delegator Beneficiary are to maintain the ALGO balance of the account below the contractually agreed maximum while also maintaining any other agreed eligibility requirements throughout the contract duration. In addition, the Delegator Beneficiary has to issue a participation key registration transaction with the parameters that were prepared by the Validator Manager The issued key registration transaction has to include a sufficient fee to make the account eligible for tracking in Algorand consensus performance tracking mechanism. All other aspects of the collaboration management are done by the Delegator Manager. This provides additional security and user convenience as Delegator Beneficiary does not have to interact with any smart contracts or any systems other than the network's native functionality. Moreover, it enables greater modularity for other systems to use Valar Platform.

[1]The staked funds are not locked and may leave the Delegator Beneficiary's wallet at any time, which can influence eligibility the staking rewards. Moreover, if the staked funds are increased, this can result in a breach of the agreed contract terms and a termination of the contract.

Delegator Beneficiary
Stakes and receives staking rewards.

Delegator Manager
Manages P2P delegation on behalf of a Delegator Beneficiary.

Validator Owner
Owns and advertises nodes; receives payments.

Validator Manager
Manages P2P delegation requests and active cooperations.

Platform Creator
Deploys the Valar Smart Contracts.

Platform Manager
Configures the Valar Platform and receives commissions.

Parter
Promotes the Valar Platform and refers users to the platform.

Visitor
Observes information about the Valar Platform.

Figure 4.1: An overview of user roles and their main activities on the Valar Platform.

4.2 Delegator Manager

The Delegator Manager is the one account that is interacting with the Valar Platform, initiating and managing P2P delegation for one or more Delegator Beneficiarys, including paying for the provided service. The payments can be in ALGO or an Algorand standard asset (ASA) (e.g., a stablecoin such as USDC), which the Delegator Manager owns. The only exception is, that the Delegator Manager can not sign participation keys on behalf of a Delegator Beneficiary. This is not possible on the Valar Platform, since, according to the Algorand consensus mechanisms, a key registration transaction is only valid if it is issued by the party that is participating in consensus, i.e., the Delegator Beneficiary.

The separation between Delegator Beneficiary and Delegator Manager is provided in order to enable secure storage of the staked funds in a cold wallet and to allow secure management of foreign funds. For example, an asset manager that manages the staking on behalf of several clients, whose funds remain in their custody, or a person managing the staking for their friends and family.

The same account can assume both the Delegator Beneficiary and Delegator Manager roles simultaneously. For example, if a Delegator stakes funds from the account which also issues the payments for covering node running service costs, then the Delegator Beneficiary and Delegator Manager roles are assigned to the account. Currently, the Valar UI does not yet support separation between the Delegator Manager and the Delegator Beneficiary.

4.3 Validator Owner

A Validator Owner owns one or more Algorand participation nodes which are advertised on the Valar Platform for P2P delegation. Anyone meeting the terms defined by the Validator Owner may participate in the Algorand consensus through the Validator Owner's nodes. The Validator Owner is not necessarily the owner of the hardware infrastructure that is used to run the participation nodes. For example, the Validator Owner can lease cloud infrastructure.

The Validator Owner may generate revenue from their provided service. Any of their revenue is accumulated at the Valar Platform. The Validator Owner has the sole permission to withdraw the accumulated funds. Moreover, the Validator Owner has the sole permission to modify the terms of their services for future Delegators or stop offering the service to new users.

4.4 Validator Manager

A Validator Manager manages the interaction between one or more Algorand participation nodes, owned by a Validator Owner, and the Valar Platform. The Validator Manager can manage the P2P delegation requests and any actions corresponding to it on behalf of the Validator Owner. It is anticipated that the Validator Manager is operated by the tailor-made Valar Daemon, introduced in Section 2.2, which autonomously manages P2P delegation requests and any active collaborations.

The Validator Owner is essentially a cold wallet with elevated privileges, while the Validator Manager is a hot wallet. The Valar Daemon has access to the hot wallet, which it uses to handle a limited set of interactions with the Valar Platform and pay the corresponding transaction fees. The Validator Manager can not withdraw earnings from the Validator Ad on behalf of the Validator Owner. A single account may assume both the Validator Owner and Validator Manager role, while this is not advised since the Validator Manager private key resides on the participation node and therefore poses a larger security risk.

4.5 Platform Creator

The Platform Creator deploys the Valar Platform (the Valar Smart Contracts) to the Algorand blockchain. Hence, the Platform Creator is the first user to interact with the Valar Platform. This is planned to be done only once, by the developer and maintainer of the platform, i.e., Valar Solutions GmbH. After deployment, the platform must first be configured to enable interactions of other users. The majority of Valar Platform's further management is handed over to the Platform Manager. Note that any entity with access to the Valar Smart Contracts code is capable of deploying the Valar Platform. The Valar Platform developer, Valar Solutions GmbH, does not permit such use of Valar Smart Contracts.

4.6 Platform Manager

The Platform Manager is in charge of configuring the Valar Platform and allowing other users to start interacting with it. That is, the Platform Manager can set and later update the platform's terms & conditions (T&C) and operational parameters, such as setting the accepted payment assets. Furthermore, while the Platform Creator deploys the Valar Smart Contracts to the Algorand blockchain, both the Platform Manager and the Platform Creator can decommission the Valar Platform.

The platform commissions, described in Chapter 3, are accumulated at the Valar Platform. The Platform Manager has the sole premission to withdraw the accumulated commission funds on demand. The same account can assume both the Platform Creator and Platform Manager role.

4.7 Partner

A Partner is someone that is recognized by the Platform Manager for promoting the Valar Platform, e.g. by supporting Valar's official branding and/or enabling easy access to users to the Valar Platform. This can be for example, a wallet provider, an non-fungible token (NFT) project, or an individual, that display's Valar's official banner with an integrated link to the Valar Platform. Based on the agreement with Platform Manager, the Partner can receive a commission for the provided convenience, described in Chapter 3.

4.8 Visitor

A Visitor is anyone that retrieves information from the Valar Platform. For example, someone that uses the Valar UI to observe information regarding the Valar Platform, the state of Valar Smart Contracts, or other resources on the Valar UI. A Visitor may also view the Valar Smart Contracts information directly on-chain or using any other means other than the Valar UI.

Chapter 5

Smart Contracts

The Valar Smart Contracts form the core of the Valar Platform. These smart contracts define the terms of the collaboration between Delegators and Validators, store and route funds, and include the needed functionalities for managing the collaboration. The Valar Smart Contracts consist of three contract types – Noticeboard, Validator Ad, and Delegator Contract– which are outlined in the following sections. The reader is referred to the Valar Platform's public code repository for details on the implementation of the Valar Smart Contracts. A simplified description of the technical implementation is also provided here, from a top-down view.

Text marked in this format describes the technical implementation details in simple terms. Code references are marked as `this_is_code`. Bytes in code are written as `b"string to be encoded as UTF-8"`.

All interactions of the Valar Platform are done through the Noticeboard, which routes the calls further to Validator Ad and Delegator Contract, when needed. Figure 5.1 provides an overview of the routing of funds through the Valar Smart Contracts. Note that Algorand transaction fees and minimum balance requirements (MBRs) are not shown in Figure 5.1. Each smart contract level fulfills a separate part of the functionality. The Noticeboard defines the platform terms within which the Validator Ad and Delegator Contract can operate, including the smart contract templates for their creation. The Noticeboard also takes care of user management, serves as the central user notification platform, and holds any earnings from the platform fees. The Validator Ad manages everything related to a Validator and its service terms, including holding of any of its earnings. The Delegator Contract is a particular instance of a service contract and all its terms that is concluded between the Validator Owner and a Delegator Manager on behalf of a Delegator Beneficiary. It also holds any payment funds that were not yet claimed by the Validator Owner.

When methods are routed through multiple levels of smart contracts, the same names are used on all levels. For example, the method named `contract_create` on Noticeboard is routed to the `contract_create` method of the Validator Ad, which further routes it to the `contract_create` method of the Delegator Contract. Despite the same names, the calls on different levels require different inputs but fulfill (a part of) the same functionality. Moreover, a higher-level call might combine multiple lower-level calls. For example, the contract_create method of the Validator Ad sequentially calls contract_create, contract_setup, and contract_pay methods of Delegator Contract within the same call.

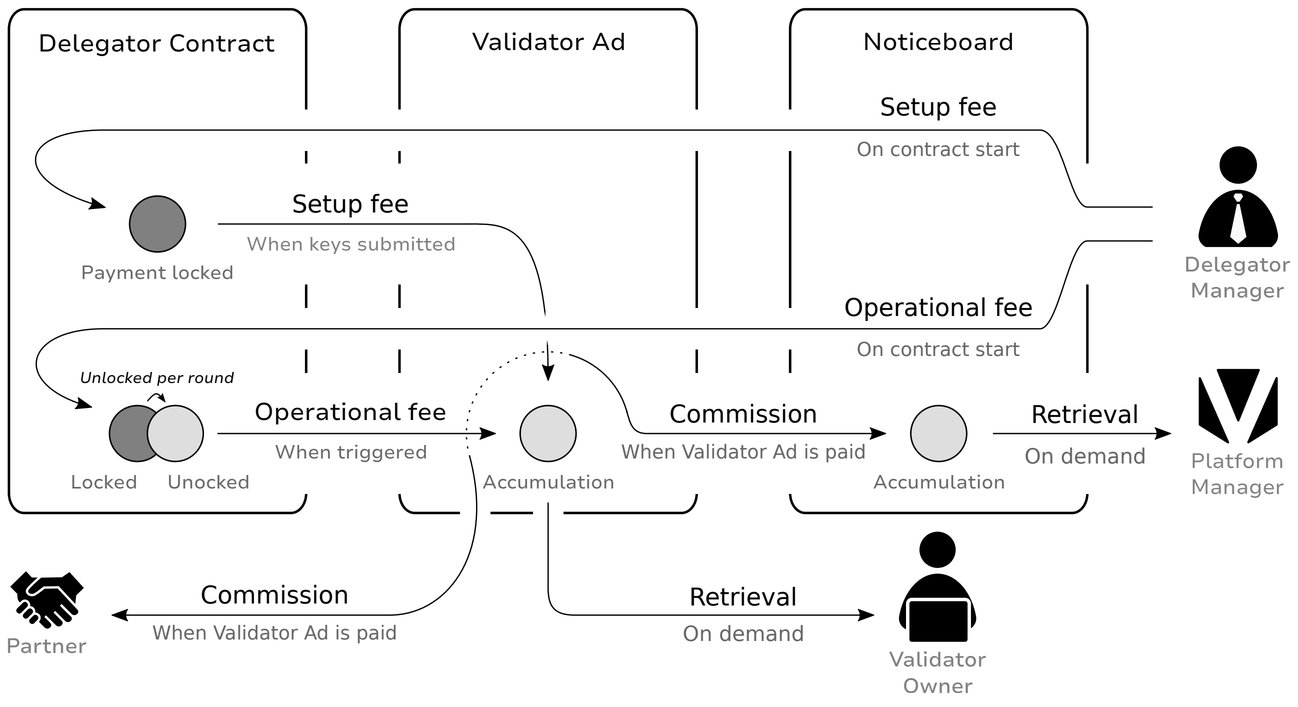

Figure 5.1: Asset flow, accumulation, and locking on the Valar Smart Contracts.

5.1 Noticeboard

The Noticeboard smart contract is the central reference point for the Valar Platform. It defines the terms of operation of the entire platform and stores the addresses of the platform's users and the application IDs of the created smart contracts. It also serves as the user notification platform. An overview of the Noticeboard's states, parameters, and functions is provided in the Noticeboard state machine figure and the the Noticeboard class description. The more high-level descriptions of the Noticeboard's parameters and functionalities, including some technical details, are provided in the following subsections.

5.1.1 Parameters

The Noticeboard's parameters are grouped below according to their purpose.

5.1.1.1 Fees

- Platform commission (commission_min): Minimum commission that Validator Ads have to pay on their earnings to the platform. Optionally, they can define a higher commission. The latter is to simplify certain partnerships.
- Validator user registration fee (val_user_reg): One-time registration fee for a Validator Owner. This is an anti-spam mechanism to not overwhelm the Valar Platform and its users with spam Validator Owners.

- Delegator user registration fee (`del_user_reg`): One-time registration fee for a Delegator Manager. For completeness but not intended to be used.
- Validator creation fee (`val_ad_creation`): One-time Validator Ad creation fee. This is an anti-spam mechanism to not overwhelm the Valar Platform and its users with spam Validator Ads.
- Delegator contract creation fee (`del_contract_creation`): One-time Delegator Contract creation fee. For completeness but not intended to be used.

These parameters are defined in `noticeboard_fees` at Noticeboard.

5.1.1.2 Timing

- Minimum staking duration allowed on the Valar Platform (`rounds_duration_min_min`): Minimum number for the minimum number of rounds the validator can allow the delegation to last.
- Maximum staking duration allowed on the Valar Platform (`rounds_duration_max_max`): Maximum number for the maximum number of rounds the validator can allow the delegation to last.
- Time to notify before expiry (`before_expiry`): How many rounds before contract end can the report about expiry be made.
- Period of notification of contract expiry (`report_period`): How frequently can the report be made.

These parameters are defined in `noticeboard_terms_timing` at Noticeboard.

5.1.1.3 Node Limits

- Maximum stake possible at the platform (`stake_max_max`): Maximum on the maximum ALGO amount that any delegator can stake.
- Minimum stake capacity of validators (`stake_max_min`): Minimum on the maximum ALGO amount that a validator must be able to accept.
- Limit on delegators per node (`cnt_del_max_max`): Maximum on the maximum number of delegators a validator is allowed by the Noticeboard to accept.

These parameters are defined in `noticeboard_terms_ndoe` at Noticeboard.

5.1.1.4 Accepted Payment Currencies

The Noticeboard allows only selected assets to be used by Validator Owner as payment currencies for their service. Currently, the supported assets are: USDC (ASA ID 31566704 on Algorand Mainnet). For each supported asset, the Noticeboard defines whether the asset is currently allowed to be used by Validator Ads or not, and what are the minimum pricing conditions that can be used on the Valar Platform for this asset.

The supported assets are stored in boxes on the Noticeboard. The boxes have name `asset_[asset_id]`. The asset ID equals ASA ID or 0 in case of ALGO. The `asset_id` is represented as uint64. Each asset box entry is structure with four fields:

- `accepted`: whether the asset is currently accepted as payment or not.
- `fee_round_min_min`: minimum flat portion of the operational fee allowed to be charged for the asset.
- `fee_round_var_min`: minimum variable portion of the operational fee allowed to be charged for the asset.
- `fee_setup_min`: minimum setup fee allowed to be charged for the asset.

5.1.1.5 Other

- Platform manager (`pla_manager`): address of the current platform manager.
- Hash (SHA256) of platform Terms and Conditions (`tc_sha256`).
- Applications ID of previous and new (`app_id_old` and `app_id_new`, respectively): This is to track the progression of possible new Noticeboard versions and for easier migration.

5.1.2 Functionality

The core functionalities of the Noticeboard are described below.

5.1.2.1 Deployment

The account that deploys the Valar Smart Contracts becomes the Platform Creator. Each such deployment of the Noticeboard is given an address and an application ID. There should be only one active instance of the Noticeboard contract at any point in time.

Any entity with access to the Valar Smart Contracts code is capable of deploying the Valar Smart Contracts. The Valar Platform developer, Valar Solutions GmbH, does not permit such use of Valar Smart Contracts.

The Platform Creator becomes the account that calls the `noticeboard_deploy` method, creating an instance of the Noticeboard smart contract application. The Noticeboard contract is in `DEPLOYED` state. At this time, the Platform Creator also becomes the Platform Manager.

5.1.2.2 Finalizing deployment

The deployment of the Noticeboard is completed by uploading the Validator Ad and Delegator Contract smart contract templates to the Noticeboard's memory. These templates are used to create the two smart contract types on demand.

After deploying the Noticeboard, the Platform Creator should load the smart contract templates for the Validator Ad and Delegator Contract in to the Noticeboard boxes. These templates are later used by the platform users to create Validator Ad and Delegator Contract smart contracts.

Loading of templates is done by first calling `template_load_init` with the `name` input set to `b"v"`' for Validator Ad and to `b"d"`' for Delegator Contract, respectively, with the `template_size` set to the number of bytes the template program occupies. The `template_load_init` creates the box for the template based on the `template_size`. The templates are then loaded in multiple data chunks by calling `template_load_data` method, with the correct name, data offset and payload chunk. Only the Platform Manager, which equals Platform Creator at this time, can load the templates. It is the responsibility of the Platform Manager to load the correct templates.

5.1.2.3 Configuration

The Platform Creator or Platform Manager can conduct the following configuration steps on the Noticeboard in order to prepare the platform for operation:

- Defining the Platform Manager, who will manage the platform during operation.
- Defining the platform's T&C.
- Defining all the platform's parameters described in Section 5.1.1.
- Defining the platform's payment assets.

Once the configuration is done for the first time, the platform starts to operate.

Once the templates are loaded, the Platform Manager calls `noticeboard_set` method. Inputs to the method define the Platform Manager, hash (SHA256) of the platform Terms and Conditions, platform fees, timing, and node parameters. Calling the `noticeboard_set` method moves the Noticeboard contract from `DEPLOYED` state to `SET` state. Once the Noticeboard contract has moved from the `DEPLOYED` state, it is not possible for the contract templates to change anymore.

The Platform Manager should then define the payment assets that are accepted at the platform. This is a two-step process. The Platform Manager must first opt-in the asset to be accepted (unless the asset is ALGO). This is done by calling `noticeboard_optin_asa` method with the parameters `asa` set to the ASA ID of the payment method. The call must be accompanied by an Algorand payment transaction to the Noticeboard contract address to pay for the increase of its MBR due to the opt-in.

After the Noticeboard has opted into the asset, the Platform Manager must enable the asset as an accepted payment method. This is done by calling the `noticeboard_config_asset` method with the input set to the ASA ID (or 0 in case of ALGO) and a structure that defines whether the asset should be accepted on the platform or not together with its minimum pricing limits. The method call must be accompanied by an Algorand payment transaction to cover the increase of MBR for the box that holds the information about the asset. Note: Defining the platform's accepted payment assets can be done already in the DEPLOYED state.

With these setups, the platform is ready to operate.

5.1.2.4 Updating the Configuration of the Noticeboard

The Platform Manager or Platform Creator can update the configuration of the Noticeboard during the Valar Platform's operation. Any of the before configured parameters can be updated, with the exception that the Noticeboard can no longer be opted out of an asset after it has been opted in.

The Platform Manager or Platform Creator can change any of the operational parameters (Section 5.1.1) of the Noticeboard contract by calling again the `noticeboard_set` method.

The Platform Manager can remove an asset from the accepted methods by calling again `noticeboard_config_asset` with the input set to the ASA ID (or 0 in case of ALGO) and setting the field of the accompanying structure to false. The asset can be accepted again by calling the `noticeboard_config_asset` method. On these calls, the accompanying Algorand payment transaction should equal 0 because the MBR has already been paid for. Note: It is not possible for the Noticeboard to opt out of an ASA once it has opted into one.

5.1.2.5 Commission Accumulation

Funds arising from platform commissions accumulate at the Noticeboard. The transfer of the platform's commission automatically happens every time that funds are transferred from the Delegator Contract to the Validator Ad, as described in Section 5.3.2.

5.1.2.6 Withdrawal of Accumulated Funds

The accumulated funds on the Noticeboard can be withdrawn at any time by the Platform Manager.

The Platform Manager can call the `noticeboard_income` method to withdraw any platform's earnings for a given asset. In case the asset is an ASA, the total ASA balance is considered its income. In the case of ALGO, the income equals the contract's ALGO balance minus its MBR.

5.1.2.7 Consensus Participation

The Noticeboard can participate in consensus using its ALGO balance. The consensus participation is initialized by the Platform Manager.

The Platform Manager can call `noticeboard_key_reg` method to optionally put the Noticeboard contract address online to participate in Algorand consensus with its balance. The call must be accompanied by an Algorand payment transaction. Its amount defines the fee to be paid for the key registration transaction of the Noticeboard.

5.1.2.8 Suspending the Noticeboard

The Noticeboard can be suspended by the Platform Creator or Platform Manager. When suspended, it is temporarily not possible to create new Validator Ads or Delegator Contracts, nor is it possible to update or extend them, respectively. New users cannot register on the platform. All other functionality remains available. The Noticeboard can be unsuspended by the Platform Creator or Platform Manager.

The Platform Manager or Platform Creator can call the `noticeboard_suspend` method. This moves the Noticeboard contract state into `SUSPENDED`.

The Platform Manager or Platform Creator can call the `noticeboard_set` method to move the Noticeboard back into state `SET`.

5.1.2.9 Retiring the Noticeboard

The Noticeboard can be retired by the Platform Creator or Platform Manager after it was suspended. When retired, it is permanently not possible to create new Validator Ads or Delegator Contracts, nor is it possible to update or extend them, respectively. New users cannot register on the platform. All other functionality remains available.

The Platform Manager or Platform Creator can call the `noticeboard_migrate` method. This moves the Noticeboard contract state into `RETIRED`.

5.1.2.10 Registering as Validator Owner or Delegator Manager

A Validator Owner or Delegator Manager has to register first at the Noticeboard before being able to use the Valar Platform. An account can be registered only as either a Validator Owner or a Delegator Manager at a time.

The registration is done by paying a one-time registration fee (in ALGO). The fee is used to cover the Algorand box storage fees for the user. The user can get this fee returned by deregistering from the Noticeboard.

In the case of a Validator Owner, an additional fee is levied as an anti-spam mechanism to not overwhelm the Valar Platform and its users with spam Validators. This fee is currently set to 5 ALGO.

A user can register at the platform by calling the `user_create` method. The method takes as an input a string of 4 bytes that defines the requested user role. Possible roles are:

- `b"val_" for \valown`
- `b"del_" for \delman`

The call must be accompanied by an Algorand payment transaction, to cover the increase in the MBR of the user's created box and to cover any additional user registration fee.

Each user is added upon creation to a double-linked list structure corresponding to the user type. This is to have a consistent way of fetching users to be displayed at the Valar UI. The Noticeboard holds a structure for each user type (`dll_val` and `dll_del`), which includes the address of the first and last user in the double-linked list, as well as the number of users in the list.

5.1.2.11 Deregistering as Validator Owner or Delegator Manager

A Validator Owner or Delegator Manager can deregistered from the Noticeboard once done using the Valar Platform, i.e. when they do not have any active Validator Ad or Delegator Contract. The registration fee that was used for covering the Algorand storage is returned to the user. The user can register again at the platform at any time, with the same or a different role.

A user can deregister from the platform by calling the `user_delete` method. The method succeeds only if the user does not have any smart contracts created anymore on the platform. The method takes no inputs. Upon successful deregistration, the MBR used for covering the user's box storage is returned.

5.1.2.12 Managing Partners

The Platform Manager can declare on the Noticeboard an account as belonging to a Partner of the Valar Platform. This enables the Partner to automatically receive convenience fees for any Delegators referred to the Valar Platform. The Platform Manager sets the convenience fees for the Partner. The fees are charged on top of the Validator fees, separately on setup fee and operational fee.

The Platform Manager can modify the convenience fees charged by the Partner for new users, or remove the Partner from the Valar Platform. It cannot change the convenience fees or the account receiving the fees for active contracts.

Note that the Partner account must be opted into the assets accepted as payment on the Valar Platform. If the Partner account is unable to accept any portion of the convenience fee at any point in time (e.g. because the account is frozen), that portion of the convenience fee will be forfeited and returned to the Delegator Manager. It is up to the Partner to ensure one can accept the convenience fees and keep the account secure in order to obtain any generated convenience fees.

The Platform Manager defines a Partner by calling the `partner_config` method. The method takes as an input the address of the Partner's account, which will be receiving the convenience fees, and the Partner's commissions on the setup fee and operational fee. The commissions are expressed in ppm. Another input to the method call is a flag whether to delete the Partner from the platform or not. The call must be accompanied by an Algorand payment transaction to cover an increase in the platform's MBR.

5.1.2.13 Creation and Management of Validator Ad

A Validator Ad can be created through the Noticeboard by paying a Validator Ad creation fee, which acts as an anti-spam mechanism. The creation fee is determined by the Noticeboard's parameter `val_ad_creation` and is currently set to 5 ALGO. One Validator Owner can have simultaneously up to 110 active Validator Ads. To create more Validator Ads, it is necessary to either delete some or use a different account.

To create a Validator Ad, the Validator must call the `ad_create` method. The method has the following inputs:

- `val_app_idx`: the position of an unoccupied element of the `val_owner`'s list of applications `app_ids`, at which the newly created Validator Ad application ID will be stored.
- `mbr_txn`: an Algorand payment transaction to cover the MBR increase on Noticeboard due to Validator Ad creation and for funding the newly created Validator Ad as well as to cover the Validator Ad creation fee.

The Validator Ad goes into state `CREATED` and is not yet able to service Delegators. It must first be configured. This is done by calling the `ad_terms` method for the first time, which moves the Validator Ad into state `SET`. The method requires as inputs:

- `val_app`: the application ID of the new Validator Ad.
- `val_app_idx`: the position of the Validator Ad application ID in the `val_owner`'s list of applications `app_ids`.
- `tc_sha256`: the SHA256 of the Terms and Conditions that are defined on the platform. This is for Validator Owner to explicitly confirm their acceptance.
- `terms_time`, `terms_price`, `terms_stake`, `terms_reqs`, and `terms_warn`: the Validator Ad's terms (for details see Section 5.2.1). The terms must comply with the Noticeboard parameters (Section 5.1.1.)
- `mbr_delegator_template_box`: the amount of ALGO needed for increase in MBR of Validator Ad due to box creation for Delegator Contract smart contract template.
- `mbr_txn`: an Algorand payment transaction for potential MBR increase of Validator Ad in the case of a new ASA opt-in and the payment for the box creation for the Delegator Contract smart contract template.

The management of any of the created Validator Ad is done through the Noticeboard. The corresponding functionalities of the Validator Ad are described in Section 5.2.2.

5.1.2.14 Creation of a Delegator Contract

The creation of a new Delegator Contract is done through the Noticeboard, where compliance with the platform's terms and the existence of the corresponding users is checked. The creation is finalized through the Validator Ad, as described in Section 5.2.2.5. One Delegator Manager can have simultaneously up to 110 active Delegator Contracts. To create more Delegator Contracts, it is necessary to either delete some or use a different account.

To conclude a Delegator Contract, the Delegator must call the contract_create method on Noticeboard with the inputs rounds_duration, del_beneficiary, and stake_max, set to the requested staking duration, Delegator Beneficiary address, and maximum stake, respectively. These inputs must be accompanied by:

- val_owner: the address of the owner of the Validator Ad that the Delegator selected.
- val_app: the application ID of the selected Validator Ad.
- val_app_idx: the position of the Validator Ad application ID in the val_owner's list of applications app_ids.
- del_app_idx: the position of an unoccupied element of the del_manager's list of applications app_ids.
- tc_sha256: the SHA256 of the Terms and Conditions that are defined on the platform. This is for Delegator Manager to explicitly confirm their acceptance.
- partner_address: the address of the referral partner. In case of no partner, this input should be set to the ZERO_ADDRESS.
- mbr_txn: an Algorand payment transaction, which is forwarded to the selected Validator Ad and used to fund its increase in the MBR, and for funding a newly created Delegator Contract as well as its (potential) opting into an ASA for the payment currency.
- txn: an Algorand payment or ASA transfer transaction which is forwarded to the selected Validator Ad and then further to the created Delegator Contract. The sent amount must equal the sum of the setup_fee and the operational_fee (including any Partner fees), and the payment/transfer asset to the agreed payment currency fee_asset_id.

The contract_create and the accompanied calls are routed to the selected Validator Ad, and consequently to the newly created Delegator Contract. The creation of a new Delegator Contract is blocked if the terms on the Validator Ad, which the Validator Owner accepted, do not match the current terms at the Noticeboard.

Note: the purpose of val_owner, val_app, and val_app_idx is to ensure the user and application exist on the Noticeboard while minimizing the number of opcodes by checking only one index instead of searching through an array.

5.2 Validator Ad

The Validator Ad is a smart contract for advertising the offered node running service, defining the Validator's service terms, and for creating Delegator Contracts. In addition, the Validator Ad stores the references to the created Delegator Contracts and accumulates the Validator Ad's earnings. An overview of the Validator Ad's states, parameters, and functions is provided in the Validator Ad state machine figure and the the Validator Ad class description. The more high-level descriptions of the Validator Ad's parameters and functionalities, including some technical details, are provided below.

5.2.1 Parameters

The Validator Ad's parameters are grouped below according to their purpose.

Note that in addition to the Validator Owner's service terms, the Validator Ad also stores the address of the Validator Owner (`val_owner`) and the ID of the Noticeboard the contract belongs to as well as the information regarding the Validator Owner's earnings, generated through the Validator Ad, and a template for generating Delegator Contracts. The latter two are stored in the so-called box storage. The Validator Ad also stores the self-disclosed information about the Validator (`val_info`). This information is not mandatory and is purely for informational purposes. The information is not verified, thus it must not be trusted lightly. The Valar UI does not currently display information about these parameters.

5.2.1.1 Timing

There are five time-related service terms that are defined in the Validator Ad:

- setup time: the maximum allowed amount of time that it can take the Validator Manager to submit the participation keys to the Valar Platform for the Delegator Beneficiary.
- confirmation time: the maximum allowed amount of time that it can take the Delegator Beneficiary to register the participation keys and the Delegator Manager to confirm this action on the Valar Platform.
- The minimum amount of time that a Delegator Beneficiary may stake through the advertised node.
- The maximal amount of time that a Delegator Beneficiary may stake through the advertised node.
- The time and date by which all its Delegator Contracts must end at the latest.

These parameters are defined in `terms_time` at Validator Ad. All timing parameters are defined as block numbers, i.e. rounds. All times displayed on the Valar UI are only estimations based on the average block time intervals. Any times entered on the Valar UI get converted based on the average block time interval estimation into rounds to be recorded in the smart contracts.

5.2.1.2 Fees

There are five service terms concerning the fees:

- The payment currency, which will be used by the Delegator Manager to pay for the advertised service. The selected currency must be one of the currencies supported on the platform (listed on the Noticeboard).
- The amount that has to be paid for setting up the node by generating participation keys for the Delegator Beneficiary, i.e., the setup fee.
- The minimal operational fee, which determines the minimal payment amount for the operational fee per time unit during staking. Section 3.3 elaborates on how the final operational

fee is determined.

- The variable operational fee, which represents the operational fee per ALGO staked and per time unit.
- The commission percentage that will be paid to the Valar Platform. The Valar Platform defines a minimal commission on the Noticeboard, while the Validator Owner may freely agree to pay a higher commission.

> These parameters are defined in `terms_price` at Validator Ad.
>
> The setup fee (`.fee_setup`) is expressed in base units of `.fee_asset_id` asset. The minimal operational fee (`.fee_round_min`) is expressed in milli base units of the payment asset (`.fee_asset_id`) per block round. The variable operational fee (`.fee_round_var`) is expressed in nano base units of the payment asset (`.fee_asset_id`) per block round and per 1 ALGO (of maximum stake).
>
> The commission (`.commission`) is represented in ppm. The maximum is 10^6.
>
> The units were selected to enable high calculation precision with different stake amounts and staking durations.

5.2.1.3 Stake

There are two service terms concerning the Delegator Contract's stake:

- Maximum allowed stake that any Delegator can stake at the node. This amount defines the maximum of the operational fee per time period unless it is smaller than the minimal operational fee.
- Gratis stake (in ppm) above the maximum stake requested by the Delegator.

The gratis stake acts as a free buffer zone, added to the requested maximum stake. It enables Delegators to hold more ALGO in their account than the initially requested maximum amount. It is envisioned that Delegators will select as the requested maximum stake the amount of ALGO they hold in their account at the time of concluding the Delegator Contract. With the gratis stake, it is possible to prevent term breaches in such cases since a user can receive ALGO without their consent or intent as sending Algorand is completely permissionless.

The agreed maximum stake for a contract is limited to the maximum allowed stake even if the sum of the gratis stake and the requested maximum stake would exceed the maximum allowed stake.

> These parameters are defined in `terms_stake` at Validator Ad.

5.2.1.4 Gating

The Validator can limit its service to Delegators that hold minimum amounts of specific assets. This can be used by Validator Owner e.g. to perform KYC on its Delegators or to make their nodes exclusive to certain NFT holders. The Validator Owner can define two different assets that the Delegator Beneficiary must hold at all times of the staking duration.

These parameters are defined in `terms_reqs` at Validator Ad.

5.2.1.5 Warnings

There are two service terms concerning the warning mechanism:

- The maximum number of warnings given to the Delegator for breaching the agreed stake or gating token balance before the contract is terminated.
- The minimum amount of time that has to pass between consecutive checks on the amount of stake and gating tokens, i.e. the maximum time for the Delegator to correct a mistake in their stake or gating token balance before another warning is given for the same mistake.

These parameters are defined in `terms_warn` at Validator Ad.

5.2.1.6 Operational configuration

The Validator Owner must specify the following operational configuration in addition to the service terms:

- The Validator Manager address for servicing of Delegator Contracts.
- The maximum number of Delegators that the Validator is willing to have at any point in time. This value is limited by the one defined at the Noticeboard.
- An indication whether the Validator Ad is currently accepting new Delegator Beneficiarys or not.

5.2.2 Functionality

The core functionalities of the Validator Ad are described below.

5.2.2.1 Configuring and Updating of Validator Ad

A Validator Ad can be configured by the Validator Owner. During this action, the Validator Ad's terms and operational configuration described in Section 5.2.1 are set. The self-disclosed information about the Validator can also be set.

All the defined configuration parameters in the Validator Ad can be changed at any point in time by the Validator Owner. The new service terms will apply for future Delegator Contracts. The new operational configuration and self-disclosure information take effect immediately. Note that if the new number of maximum accepted users is smaller than the previous one, this will not remove any active Delegator Contracts. Any active Delegator Contract must be fulfilled by the Validator Owner.

Configuration of the Validator Owner terms is done by calling the ad_terms method. The setting of the terms succeeds only if the call explicitly includes as input the SHA256 of the Terms and Conditions of the platform. The set terms must comply with the Noticeboard parameters Section 5.1.1.

When not calling the ad_terms method for the first time (i.e. for changing existing terms), the input mbr_delegator_template_box must be set to zero (as the box for Delegator Contract smart contract template was already created), and the accompanying ALGO payment transaction mbr_txn set to a zero amount, unless the terms define a new ASA as an accepted payment method. In this case, the amount must cover the MBR increase for the new ASA optin.

Configuration of the operational parameters is done by calling the ad_config method, which defines (new) Validator Manager (val_manager), the (new) maximum number of Delegator Beneficiarys the Validator Owner is willing to accept (cnt_del_max), and whether the Validator Owner currently allows accepting new Delegator Contracts or not (live).

Depending on if the Validator Owner decided to currently accept new Delegator Contracts or not, the Validator Ad goes to state NOT_READY or NOT_LIVE, respectively. Note that for Validator Ad to be accepting new Delegator Beneficiary, the Validator Manager must subsequently confirm its readiness by calling the ad_ready method with the readiness flag (ready) set. This moves the Validator Ad into state READY. This is the only state from where new Delegator Contracts can be created.

Configuration of the self-disclosure information is done by calling the ad_self_disclose method.

5.2.2.2 Accumulation of Validator Owner Earnings

The earnings generated by the node running service are transferred from the Delegator Contract to the Validator Ad, as described in Section 5.3.2. These funds are represented as part of the Validator Ad's account balance and are not locked. The amount of all-time earnings and the generated fees for an asset are tracked at the Validator Ad (in box asa_[ASA_ID].

5.2.2.3 Withdrawal of Earnings

The funds that are accumulated on the Validator Ad can be withdrawn at any time by the Validator Owner.

Withdrawing of earnings is done by calling the ad_income method for a particular asset (i.e. ASA ID or 0 for ALGO) defined in the input (asset_id). The whole available account balance is withdrawn from the Validator Ad to the Validator Owner. In case of ASA, the available balance equals the whole balance, while in case of ALGO it is reduced by the account's MBR.

5.2.2.4 Delete Validator Ad

A Validator Owner can delete its Validator Ad if there are no Delegator Contracts, described in Section 5.3, associated with it. Because the possibility to delete the Validator Ad is not in the sole hands of the Validator Owner, this option is currently not available via the Valar UI. Before deleting the Validator Ad, all earnings must be withdrawn from the Validator Ad, which is done by closing out the Validator Ad account from all the assets that the Validator Ad accepted as payment method at any point in time.

If there are no Delegator Contracts on the Validator Ad, the method `ad_asa_close` must first be called for all ASAs that the Validator Ad is opted into. The call closes out the ASA balance to the Validator Owner. Only then can the Validator Ad be deleted by calling the `ad_delete` method.

5.2.2.5 Creation of a Delegator Contract

While the Delegator Contract creation is started at the Noticeboard, as described in Section 5.1.2.14, the action is routed to the corresponding Validator Ad for processing.

On the Validator Ad, it is checked if the defined `rounds_duration` and `stake_max` inputs meet the Validator Ad's terms. Moreover, any `stake_gratis` is applied to determine the actual `stake_max_given`. The operational fee is determined based on the requested `stake_max` and minimum and variable parts of the operational fee, as per Section 3.3.

Any partner fees are also calculated at this point based on the given `partner_address` and its `partner_commissions`, which have been applied based on the information stored at the Noticeboard for the given Partner. The partner fees are similarly divided between `fee_setup_partner` and `fee_round_partner`, and are recorded in the Delegator Contract.

A new Delegator Contract gets created there based on the Delegator Contract template stored in Validator Ad's box `BOX_DELEGATOR_CONTRACT_TEMPLATE_KEY=b"d"`. The created Delegator Contract gets funded with account MBR and for one asset opt-in to prepare it for storing the payment funds (even if the payment asset is ALGO). If the agreed terms include payment in a non-ALGO currency, the Delegator Contract will get opted-in.

The created Delegator Contract contract gets configured with the agreed Validator Ad terms, including the adjusted `stake_max_given`, the stake-adjusted fees, and the applied `partner_address` with its fees.

The payment of the sum of the setup fee and operational fee is re-routed to the Delegator Contract, where it gets checked for amount and payment currency. The creation of the contract, defining its terms, and the payment of fees, moves the Delegator Contract into the `READY` state. The ID of the created Delegator Contract gets stored in the list of contracts managed by the Validator Ad (on the Validator Ad) and in the list of contracts managed by the `del_manager` (on the Noticeboard).

5.3 Delegator Contract

The Delegator Contract is a smart contract that describes the terms of the service contract between a Delegator and Validator, such as the duration of staking and the maximum amount of funds that can be staked. Moreover, it defines all service costs as well as the timing, notification, and eligibility requirements. In addition, the Delegator Contract holds the payments issued by the Delegator Manager before these are forwarded to the Validator Owner, and the participation key parameters that are submitted by the Validator Manager to be used by Delegator Beneficiary. An overview of the Delegator Contract's states, parameters, and functions is provided in the Delegator Contract state machine figure and the the Delegator Contract class description. The more high-level descriptions of the Delegator Contract's parameters and functionalities, including some technical details, are provided below.

5.3.1 Parameters

The Delegator Contract's parameters are grouped below according to their purpose.

5.3.1.1 Fees

- Fee asset ID (`delegation_terms_general.fee_asset_id`): ID of the asset used as a means of payment. Equals 0 in case of ALGO.
- Setup fee (`delegation_terms_general.fee_setup`): Agreed fee for setting up the node, i.e. generating and submitting the participation keys.
- Operational fee (`fee_operational`): Agreed fee for the whole duration of staking, calculated based on the agreed contract terms.
- Platform commission (`delegation_terms_general.commission`): Relative platform commission on any Validator Owner earnings. It gets deducted from the earnings and is paid to the Valar Platform.
- Partner address (`delegation_terms_general.partner_address`): Address of the partner that collects the partner convenience fees if applicable. In case of no partner, the account is `ZERO_ADDRESS`.
- Partner setup fee (`delegation_terms_general.fee_setup_partner`): Agreed commission charged by the partner for convenience on the node setup.
- Partner operational fee (`fee_operational_partner`): Agreed commission charged by the partner for convenience during node operation.

The platform commission (`.commission`) is represented in ppm. The maximum is 10^6.

The operational fees, i.e. `fee_operational` and `fee_operational_partner`, are stored also on a per-block basis as `.fee_round` and `.fee_round_partner`, respectively. This is for calculating the portion of the operational fee that can be claimed by the Validator Owner at a time instance and/or to determine the unspent portion of the operational fee in case of early contract end.

The values of all fees are expressed in base units of `.fee_asset_id` asset, except the operational fees per round, which are expressed in milli base units of `.fee_asset_id` asset per block round to enable high calculation precision.

5.3.1.2 Timing

- Start (`round_start`): Agreed start round of the contract, i.e., the time of its creation.
- Planned end (`round_end`): Agreed end round of the contract.
- Actual end (`round_ended`): Actual round at which the contract ended. Can be smaller than `round_end` in case of early contract end.
- Time for setup (`delegation_terms_general.rounds_setup`): Maximum allowed amount of time that it can take the Validator Manager to submit the participation keys to the Valar Platform for the Delegator Beneficiary.
- Time to confirm the setup (`delegation_terms_general.rounds_confirm`): Maximum allowed amount of time that it can take the Delegator Beneficiary to register the participation keys and the Delegator Manager to confirm this action on the Valar Platform.
- Timing of last claim of operational fee (`round_claim_last`): The round number of the last time the operational fee was claimed.

All timing parameters are defined as block numbers, i.e. rounds. All times displayed on the Valar UI are only estimations based on the average block time intervals. Any times entered on the Valar UI get converted based on the average block time interval estimation into rounds to be recorded in the smart contracts.

The Validator Manager must submit the participation keys before the round `round_start` + `rounds_setup`. The Delegator Beneficiary must register the participation keys and the Delegator Manager confirm this by `round_start` + `rounds_setup` + `rounds_confirm`.

5.3.1.3 Parameters of the participation keys

- Vote key dilution (`vote_key_dilution`): Vote key dilution parameter of the participation keys.
- Vote key (`vote_pk`): The vote public key parameter.
- Selection key (`selection_pk`: The selection public key parameter.
- State proof key (`state_proof_pk`): The state-proof public key parameter.

All participation key parameters are stored as raw bytes, except for the dilution parameter which is an integer. The `vote_first` and `vote_last` parameters of the participation keys must equal `round_start` and `round_end`, respectively. The participation keys must be generated for the Delegator Beneficiary address.

5.3.1.4 Limits and Gating

- Max stake (`delegation_terms_balance.stake_max`): The maximal amount of stake that the Delegator Beneficiary may hold at any point during the contract's duration.
- Gating assets (`delegation_terms_balance.gating_asa_list`): IDs of gating assets and the minimal balance of each asset that the Delegator Beneficiary must hold throughout the contract's duration.

Max stake is represented in microALGO.

There can be at most two gating assets. They are entered as a list of tuples of two elements, whereby the first element is the asset ID (i.e. ASA ID) and the second one the minimum balance the Delegator Beneficiary must hold. If there are no gating assets, the asset IDs should equal zero.

5.3.1.5 Warnings

- Number of allowed limit breaches (`delegation_terms_balance.cnt_breach_del_max`): Maximum allowed number of delegator's breach events of its balance requirements before the contract ends. Having more ALGO than the maximal allowed stake or less than the required amount of a gating asset both count as a breach.
- Rounds between breaches (`delegation_terms_balance.rounds_breach`): Minimum number of rounds between two stake- or ASA-limit breach events to consider them as separate, i.e., to increment `cnt_breach_del` counter.
- Breach counter (`cnt_breach_del`): Counter regarding the number of times the stake or gating asset limit has been breached.
- Timing of latest limit breach (`round_breach_last`): The round number of the latest breach event.
- Timing of latest expiry notice (`round_expiry_soon_last`): The round number of the latest reporting of the fact that the contract will expire soon.

All timing parameters are defined as block numbers, i.e. rounds. All times displayed on the Valar UI are only estimations based on the average block time intervals. Any times entered on the Valar UI get converted based on the average block time interval estimation into rounds to be recorded in the smart contracts.

5.3.2 Functionality

The core functionalities of the Delegator Contract are described below.

5.3.2.1 Creation of a Delegator Contract

While the Delegator Contract creation is started at the Noticeboard (Section 5.1.2.14) and pre-processed on Validator Ad (Section 5.2.2.5), the actual creation and storing of the agreed service terms happens at the Delegator Contract. The Delegator Contract also confirms the transfer of the setup fee and operational fee (including any Partner fees) by the Delegator Manager. This full upfront payment for the service gets stored at the Delegator Contract.

The creation and configuring of the Delegator Contract happens with subsequent calls to `contract_create`, `contract_setup`, and `contract_pay` methods.

The `contract_create` call creates the contract and defines its basic terms like the application IDs of the Noticeboard and Validator Ad it belongs to, the Delegator Beneficiary and Delegator Manager addresses. The contract is in `CREATED` state.

The `contract_setup` call defines the agreed service contract terms (i.e. the parameters from Section 5.3.1) and prepares for the acceptance of payment fees by opting into the payment asset (if this is not ALGO). The terms are provided as method inputs, i.e. `delegation_terms_general` and `delegation_terms_balance`. As part of the inputs, the SHA256 of the platform's Terms and Conditions is given to be recorded in the contract together with the agreed staking duration (`rounds_duration`). The call moves the contract to state `SET`.

The `contract_pay` call ensures the agreed setup fee and operational fee, including any partner commissions, are paid. To ensure this, the call must be accompanied with a correct payment or ASA transfer transaction. The call also checks that the Delegator Beneficiary fulfills the eligibility requirements of the agreed terms. The contract moves to state `READY`, where it is waiting for submission of keys by Validator Manager.

5.3.2.2 Submitting Parameters of Participation Keys

Staking on Algorand requires valid participation keys signed by the Delegator Beneficiary (i.e. the account that is staking) that can be accessed by the participation node through which the staking takes place.

The transaction for registering the participation keys includes the following parameters:

- The account that will participate in consensus.
- First round of validity of the keys (i.e. staking).
- Last round of validity of the keys (i.e. staking).
- The vote key.
- The selection key.
- The state proof key.

The Validator Manager submits these parameters of the generated participation keys to the Delegator Contract for the Delegator Beneficiary to see them. The submission must be done within the agreed setup time. Upon successful submission, the Validator Owner earns the setup fee. The fee, minus the agreed platform commission, goes to the Validator Ad, while the commission goes to the Noticeboard. Any partner commission charged for the setup fee gets transfered to the Partner if it can accept it.

The submission of the keys is done by calling the keys_submit method at the Noticeboard and providing the key registration parameters. The call is re-routed to the Validator Ad. There, it is checked that the call to the Noticeboard was made by the Validator Manager. The call is further re-routed to the Delegator Contract, where it is checked that the vote_first, vote_last, and the address parameters of the keys are correct. Moreover, it is checked that the call has been done before the round round_start + rounds_setup. The setup fee minus the agreed platform commission is sent from the Delegator Contract to the Validator Ad, while the commission amount is sent to the Noticeboard. Any partner commission on the setup fee (i.e. fee_setup_partner) is transfered to the Partner. With a successful keys_submit call, the Delegator Contract moves to SUBMITTED state.

Note that if any of the involved parties cannot accept the earned fees form this action (e.g. if their accounts are frozen or closed-out), the call will still succeed. The party that is not able to accept the earned fees, forfeits them. The Delegator Manager is able to reclaim any forfeited fees when the Delegator Contract ends.

If the payment of the fees cannot be made from the Delegator Contract (e.g. because the Delegator Contract account has been frozen or payment clawed back such that the current amount is insufficient to cover the agreed fees), the call will fail. The method breach_pay should be called in this case (see Section 5.3.2.10).

5.3.2.3 Confirming the Prepared Setup

The Delegator Beneficiary can see in the Delegator Contract the parameters of the participation keys that were prepared for them by the Validator. To start staking, the Delegator Beneficiary must issue a participation key registration transaction with these parameters. The transaction must ensure that the Delegator Beneficiary is opted into the Algorand consensus performance tracking mechanism. After issuing the key registration transaction, the Delegator Manager must confirm on the Valar Platform that the key registration transaction has been issued by the Delegator Beneficiary. The service contract for staking is then live. The confirmation must be done within the agreed confirmation time.

At the current version of the Valar UI, the issuing of the participation key registration and the confirmation at the Valar Platform is done simultaneously in the same group transaction.

After the participation keys have been submitted to the Delegator Contract, the Delegator Beneficiary must issue a key registration transaction with these parameters and with a sufficient fee to opt in the account into the network's staking rewards, i.e. the Algorand consensus performance tracking mechanism. The latter is to be able to track on-chain the performance of the node runner. The del_manager must confirm that the keys have been registered by calling the keys_confirm method on Noticeboard. The call is re-routed to the Validator Ad and then further to Delegator Contract, where it is checked if the confirmation has been done in the agreed time, i.e. before round_start + rounds_setup + rounds_confirm, and if the Delegator Beneficiary is opted into the tracking mechanism. The Delegator Contract moves to state LIVE.

The Delegator Beneficiary starts participating in consensus at the latest 320 rounds after the confirmation.

5.3.2.4 Not Submitting Parameters of Participation Keys in Time

The Validator Manager has a limited amount of time to generate the participation keys and submit them to the Delegator Contract. This must be done at the latest in the agreed setup time after the contract creation. If the keys are not submitted in time, the service contract can get terminated. The fees (i.e. setup fee and operational fee, including any partner fees) are returned to the Delegator Manager in full.

The checking and triggering the corresponding return of funds to the Delegator Manager can be initiated by anyone, including third parties. The Valar team plans to conduct the checking on behalf the platform's users for a better user experience.

The refund is initiated by anyone calling the `keys_not_submitted` method on Noticeboard. As with the other calls, it must be ensured that the contract parties are part of the platform. The call is re-routed via Validator Ad to the Delegator Contract, where it is checked if the setup time has expired, i.e. if the current round is larger than `round_start` + `rounds_setup`. The Delegator Contract moves to state `ENDED_NOT_SUBMITTED` and returns all the fees to `del_manager` if possible. If it is not possible to return the fees (e.g. if `del_manager` opted out of the payment currency or closed out their account), the fees remain in the Delegator Contract, where they can later be reclaimed by the Delegator Manager. The ended Delegator Contract is removed from the list of contracts managed by the Validator Ad, giving space for new Delegator Contracts.

5.3.2.5 Not Confirming the Keys in Time

The Delegator Manager has a limited time window for registering the participation keys after they have been submitted to the Delegator Contract by Validator Manager. If this does not happen within the agreed time window (i.e. confirmation time), the service contract can get terminated. The operational fee, including any partner fee for the operation, are returned to the Delegator Manager in full. The Validator Owner is released from any obligations agreed under the service agreement.

The refund is initiated by anyone calling the `keys_not_confirmed` method on Noticeboard. As with the other calls, it must be ensured that the contract parties are part of the platform. The call is re-routed via Validator Ad to the Delegator Contract, where it is checked if the agreed time has expired, i.e. if the current round is larger than `round_start` + `rounds_setup` + `rounds_confirm`.

The Delegator Contract moves to state `ENDED_NOT_CONFIRMED`. It returns `fee_operational` and `fee_operational_partner` to `del_manager` if possible. If it is not possible to return the fees (e.g. if `del_manager` opted out of the payment currency or closed out their account), the fees remains in the Delegator Contract, where they can later be reclaimed by the Delegator Manager. The ended Delegator Contract is removed from the list of contracts managed by the Validator Ad, giving space for new Delegator Contracts.

5.3.2.6 Claiming Operational Fee

After the contract is live, the Validator Owner continuously earns a portion of operational fee corresponding to the portion of the service duration already provided. The already earned portion must first be explicitly claimed before it can be paid out to the Validator Owner. At the time of claiming, the earned portion of the fee, minus the agreed platform commission, goes to the Validator Ad, while the commission goes to the Noticeboard. The corresponding portion of the partner commission charged for the convenience during operation gets transfered to the Partner if it can accept it.

The claiming of the portion of the operational fee corresponding to the portion of the service duration already provided is done by calling the `contract_claim` method on Noticeboard. This can be initiated by anyone. The call is re-routed to the Validator Ad and further to the Delegator Contract. There, it is checked when was the fee last claimed (`round_claim_last`) and the portion of the fee earned since then calculated. The same is done for any Partner fee. The earned fee minus the agreed platform commission is sent from the Delegator Contract to the Validator Ad, while the commission amount is sent to the Noticeboard. Any earned partner commission is transfered to the Partner.

Note that if any of the involved parties cannot accept the earned fees form this action (e.g. if their accounts are frozen or closed-out), the call will still succeed. The party that is not able to accept the earned fees, forfeits them. The Delegator Manager is able to reclaim any forfeited fees when the Delegator Contract ends.

If the payment of the fees cannot be made from the Delegator Contract (e.g. because the Delegator Contract account has been frozen or payment clawed back such that the current amount is insufficient to cover the agreed fees), the call will fail. The method `breach_pay` should be called in this case (see Section 5.3.2.10).

5.3.2.7 Contract Expiry

After the agreed staking duration expires, the contract automatically ends. The Validator Owner earns the operational fee. The fee, minus the agreed platform commission, goes to the Validator Ad, while the commission goes to the Noticeboard. The Delegator Beneficiary participation keys have expired by then. Any partner commission charged for the convenience during operation gets transfered to the Partner if it can accept it.

Note: Because Algorand consensus uses a 320 round trailing balance for accounts, the Validator Owner should continue to stake for 320 more rounds after participation keys have expired.

The contract end can be processed by anyone calling the `contract_expired` method on Noticeboard. The call is re-routed to the Validator Ad and further to the Delegator Contract. There, it is checked that service contract has expired, i.e. that the current round is larger than `round_end`. Any unclaimed portion of the operational fee minus the agreed platform commission is sent from the Delegator Contract to the Validator Ad, while the commission amount is sent to the Noticeboard. Any partner commission on the unclaimed portion of the operational fee (i.e. `fee_operational_partner`) is transfered to the Partner. With a successful contract_expired call, the Delegator Contract moves to ENDED_EXPIRED state. The ended Delegator Contract is removed from the list of contracts managed by the Validator Ad, giving space for new Delegator Contracts.

Note that if any of the involved parties cannot accept the earned fees form this action (e.g. if their accounts are frozen or closed-out), the call will still succeed. The party that is not able to accept the earned fees, forfeits them. The Delegator Manager is able to reclaim any forfeited fees when the Delegator Contract ends.

If the payment of the fees cannot be made from the Delegator Contract (e.g. because the Delegator Contract account has been frozen or payment clawed back such that the current amount is insufficient to cover the agreed fees), the call will fail. The method breach_pay should be called in this case (see Section 5.3.2.10).

5.3.2.8 Withdrawing from the Contract

The Delegator Manager can decide to withdraw from an active contract at any point in time after confirming the participation keys. At the time the Delegator Manager withdraws form the contract, they get returned a portion of the operational fee corresponding to the percentage of the remaining Delegator Contract's duration. There are no penalties for withdrawing from the contract. The Validator Owner earned only a percentage of the operational fee corresponding to the elapsed time. The earned fee, minus the agreed platform commission, goes to the Validator Ad, while the commission goes to the Noticeboard. The corresponding portion of the partner commission charged for the convenience during operation gets transfered to the Partner if it can accept it.

The Delegator Beneficiary should issue a deregisteration transaction before the Delegator Manager withdraws from the contract. Otherwise, the Delegator Beneficiary might continue to use non-existing keys because the contract ended and thus the obligations of the Validator Owner.

The Delegator Beneficiary cannot directly end the contract on its own. If Delegator Beneficiary wants to stop staking with the Validator Owner, the Delegator Beneficiary simply issues another consensus participation (de)registration transaction. This can be doen at any time, even without the Delegator Manager withdrawing from the contract. In this case, the contract remains valid. The keys can even be reused. The Delegator Beneficiary should inform the Delegator Manager to get back any unspent fees.

Note that it is not possible to withdraw from the contract during the setup time. In this case, the Delegator Manager should just not confirm the prepared setup.

The Delegator Manager withdraws from the service contract by calling contract_withdraw method on Noticeboard. The call is re-routed to the Validator Ad and further to the Delegator Contract. Any unclaimed portion of the operational fee minus the agreed platform commission is sent from the Delegator Contract to the Validator Ad, while the commission amount is sent to the Noticeboard. Any partner commission on the unclaimed portion of the operational fee (i.e. fee_operational_partner) is transfered to the Partner. The Delegator Contract moves to ENDED_WITHDREW state. The ended Delegator Contract is removed from the list of contracts managed by the Validator Ad, giving space for new

Delegator Contracts.

Note that if any of the involved parties cannot accept the earned fees form this action (e.g. if their accounts are frozen or closed-out), the call will still succeed. The party that is not able to accept the earned fees, forfeits them. The Delegator Manager is able to reclaim any forfeited fees when the Delegator Contract ends.

If the payment of the fees cannot be made from the Delegator Contract (e.g. because the Delegator Contract account has been frozen or payment clawed back such that the current amount is insufficient to cover the agreed fees), the call will fail. The method `breach_pay` should be called in this case (see Section 5.3.2.10).

5.3.2.9 Extending the Contract

The Delegator Manager can decide to extend the Delegator Contract with the current Validator Owner. The extension process consists of withdrawing from the existing contract and concluding a new Delegator Contract with the same Validator Owner. A different duration and/or maximum stake can be defined by the Delegator Manager. If the Validator Ad terms of use have changed, the new terms will apply for the new contract. The Valar UI provides the extension functionality as an atomic operation.

5.3.2.10 Paid Fee is Frozen or Clawed

All payments made by the Delegator Manager to the node runner are held in the Delegator Contract until the conditions are met to be released to the Validator Ad for the payment of the Validator Owner's service, the Partner fees, or returned to the Delegator Manager. The payment currencies used might have the option to be frozen or clawed back by the currency issuer. This would prevent the agreed distribution of the fees. In such a case, the contract ends. The responsibility for this mistake is assigned to the Delegator Manager as it is unable to pay for the service.

Any remaining fees are left with the Delegator Contract and can be reclaimed by Delegator Manager if they are unfrozen.

The contract ends in such a case by anyone calling the `breach_pay` method on Noticeboard. The call is re-routed to the Validator Ad and further to the Delegator Contract. The Delegator Contract moves to `ENDED_CANNOT_PAY` state. The ended Delegator Contract is removed from the list of contracts managed by the Validator Ad, giving space for new Delegator Contracts.

5.3.2.11 Delegator Beneficiary not Respecting the Agreed Balance Limits

When concluding the service contract, the Delegator Manager has agreed on a limit of the maximum amount of ALGO that the Delegator Beneficiary can stake with the offered node running service. If the Delegator Beneficiary has at any point in time more ALGO in the account than the agreed limit, the Validator Owner has the right to end the contract.

The Validator Owner can allow in the Validator Ad's terms that the stake limit can be exceeded a few times before the contract is ended, giving a warning to the Delegator Manager each time. The contract will end only when the Delegator Beneficiary has received the specified number of warnings. If the node runner allows for multiple warnings, a minimum defined amount of time must pass between two warnings.

Some node runners might offer their services only to Delegator Beneficiary who fulfill certain requirements like holing a specific NFT or a minimum amount of a particular token. When concluding a service contract, the Delegator Beneficiary agrees to continuously fulfill these requirements. The same counter of maximum amount of warnings and the amount of time between the warnings applies as for the ALGO stake limit.

When the terms have been breached and the contract ends, any remaining portion of the operational fee is returned to the Delegator Manager, while the Validator Owner gets the portion of the operational fee that was spent for the operating time before the breach of terms minus the platform commission. The same applies for any Partner fee.

The contract ends in such a case by anyone calling the `breach_limits` method on Noticeboard. The call is re-routed to the Validator Ad and further to the Delegator Contract, where it is checked if sufficient amount of time (`delegation_terms_balance.rounds_breach`) has passed since last breach. If maximum agreed number of breaches has been reached (`delegation_terms_balance.cnt_breach_del_max`), the Delegator Contract moves to `ENDED_LIMITS` state. Any unclaimed portion of the operational fee minus the agreed platform commission is sent from the Delegator Contract to the Validator Ad, while the commission amount is sent to the Noticeboard. Any partner commission on the unclaimed portion of the operational fee (i.e. `fee_operational_partner`) is transfered to the Partner. The ended Delegator Contract is removed from the list of contracts managed by the Validator Ad, giving space for new Delegator Contracts.

Note that if any of the involved parties cannot accept the earned fees form this action (e.g. if their accounts are frozen or closed-out), the call will still succeed. The party that is not able to accept the earned fees, forfeits them. The Delegator Manager is able to reclaim any forfeited fees when the Delegator Contract ends.

If the payment of the fees cannot be made from the Delegator Contract (e.g. because the Delegator Contract account has been frozen or payment clawed back such that the current amount is insufficient to cover the agreed fees), the call will fail. The method `breach_pay` should be called in this case (see Section 5.3.2.10).

5.3.2.12 Participation Node Under-Performing

The Validator Owner commits to staking on the user's behalf for the agreed account on the Algorand network. The Algorand network is continuously monitoring the performance of the accounts that stake and have opted into the staking rewards. If the network detects that an account is not participating in the network as expected, it will automatically suspend the account from staking, making it ineligible for staking rewards. The suspended account in question is the Delegator Beneficiary on the Valar Platform. This can be initiated by anyone - either the Delegator Manager themselves or by a third party. The Valar team plans to provide this checking service to the platform's users for a more seamless user experience.

When under-performance is reported, any remaining portion of the operational fee is returned to Delegator Manager. The Validator Owner gets the portion of the operational fee that was already spent for the operating time before the report minus the platform commission. The same applies for any Partner fee.

Note that the reporting of under-performance can be done only before the contract has expired.

Algorand network uses a conservative approach to determining if an account is not meeting the expected performance. This is because of the probabilistic nature of its pure PoS (PoS) mechanism. Delegator Managers might want stricter performance requirements. This can be achieved with the help of off-chain monitoring. However, in such a case, the Delegators must either perform the off-chain monitoring themselves or trust a third party to do it because the network is not reaching consensus on more granular performance monitoring. An example of a performance monitoring services is https://alerts.allo.info/. The Delegator Manager can decided e.g. to withdraw from the contract based on off-chain monitoring indication of under-performance.

Due to these reasons, it is recommended that Delegators select reputable and trustworthy Validators for their staking.

The contract ends in such a case by anyone calling the `breach_suspended` method on Noticeboard. The call is re-routed to the Validator Ad and further to the Delegator Contract, where it is checked if Delegator Beneficiary is not eligible anymore for staking rewards. This indicates that either the account was suspended or closed out. Both cases warrant ending of the contract. Any unclaimed portion of the operational fee minus the agreed platform commission is sent from the Delegator Contract to the Validator Ad, while the commission amount is sent to the Noticeboard. Any partner commission on the unclaimed portion of the operational fee (i.e. `fee_operational_partner`) is transfered to the Partner. The Delegator Contract moves to `ENDED_SUSPENDED` state. The ended Delegator Contract is removed from the list of contracts managed by the Validator Ad, giving space for new Delegator Contracts.

Note that if any of the involved parties cannot accept the earned fees form this action (e.g. if their accounts are frozen or closed-out), the call will still succeed. The party that is not able to accept the earned fees, forfeits them. The Delegator Manager is able to reclaim any forfeited fees when the Delegator Contract ends.

If the payment of the fees cannot be made from the Delegator Contract (e.g. because the Delegator Contract account has been frozen or payment clawed back such that the current amount is insufficient to cover the agreed fees), the call will fail. The method `breach_pay` should be called in this case (see Section 5.3.2.10).

5.3.2.13 Reporting Expiry

The Delegator Beneficiary stakes only as long as it has valid participation keys. When they expire, the Delegator Beneficiary stops staking. To remind Delegators of this, a notification about expiry of the service contract can be sent to its Delegator Manager. To prevent spamming Delegator Manager, the notification can be sent via the Valar Platform only at a minimum defined period and only a certain time in advance before expiry (parameters `before_expiry` and `report_period`at Noticeboard, respectively).

Report of expiry is by anyone calling the `contract_report_expiry_soon` method on Noticeboard. The call is re-routed to the Validator Ad and further to the Delegator Contract. There, it is checked if the contract is close enough to its and if enough time has passed since the last call of the `contract_report_expiry_soon` method.

5.3.2.14 Notifications

For all interactions that do not have to be made by the Delegator Manager, the Valar Platform will try to send a message to the Delegator Manager to notify them about relevant changes. The message is a zero ALGO payment transaction issued by the Noticeboard. In case the Delegator Manager cannot accept the message (e.g. if the account is closed out), no message will be sent.

Chapter 6

User Journeys

All users, except for Visitors, interact with the Valar Platform using their Algorand account, i.e., their address for identification and the corresponding private key for approving transactions. Connecting one's wallet and the approving transactions is not required for Visitors, who are merely browsing and observing the information on the blockchain. The Valar UI allows Visitors to browse without the need for a wallet. To interact with the Valar Platform, the users can connect their wallet. For reference, Figure 6.1 recaps the Valar Platform components and users.

The following subsections describe the foreseen user interactions during the usage of the Valar Platform, including the corresponding asset flows. It is assumed that all users are represented by a blockchain account, except for Visitors. The latter do not require an account to view information without interacting with the platform. Note that the transaction fees for executing smart contract calls are not explicitly mentioned. These fees typically range between one and ten microALGO. These fees can change depending on network congestion. Figure 6.2 offers an alternative view of user interactions, where these are displayed on top of the timelines of the Valar Smart Contracts.

6.1 Platform Creator

This section describes the user journey for the Platform Creator. Apart from the existence of an Algorand account with minimal funding, there are no further requirements for this role. The Platform Creator cannot be changed. The Platform Creator interacts with the Valar Smart Contracts e.g. through the command line or any other tool.

6.1.1 Deploying and pre-Configuring the Noticeboard

The account that deploys the Valar Smart Contracts becomes the Platform Creator[1]. Any entity with access to the Valar Smart Contracts code is capable of deploying the Valar Smart Contracts. The Valar Platform developer, Valar Solutions GmbH, does not permit such use of Valar Smart Contracts.

[1]Each Noticeboard deployment gets an address and an application ID. The Valar UI is configured to operate with the Noticeboard deployed by the Valar team. Users should pay attention to which Noticeboard instance they are interacting with.

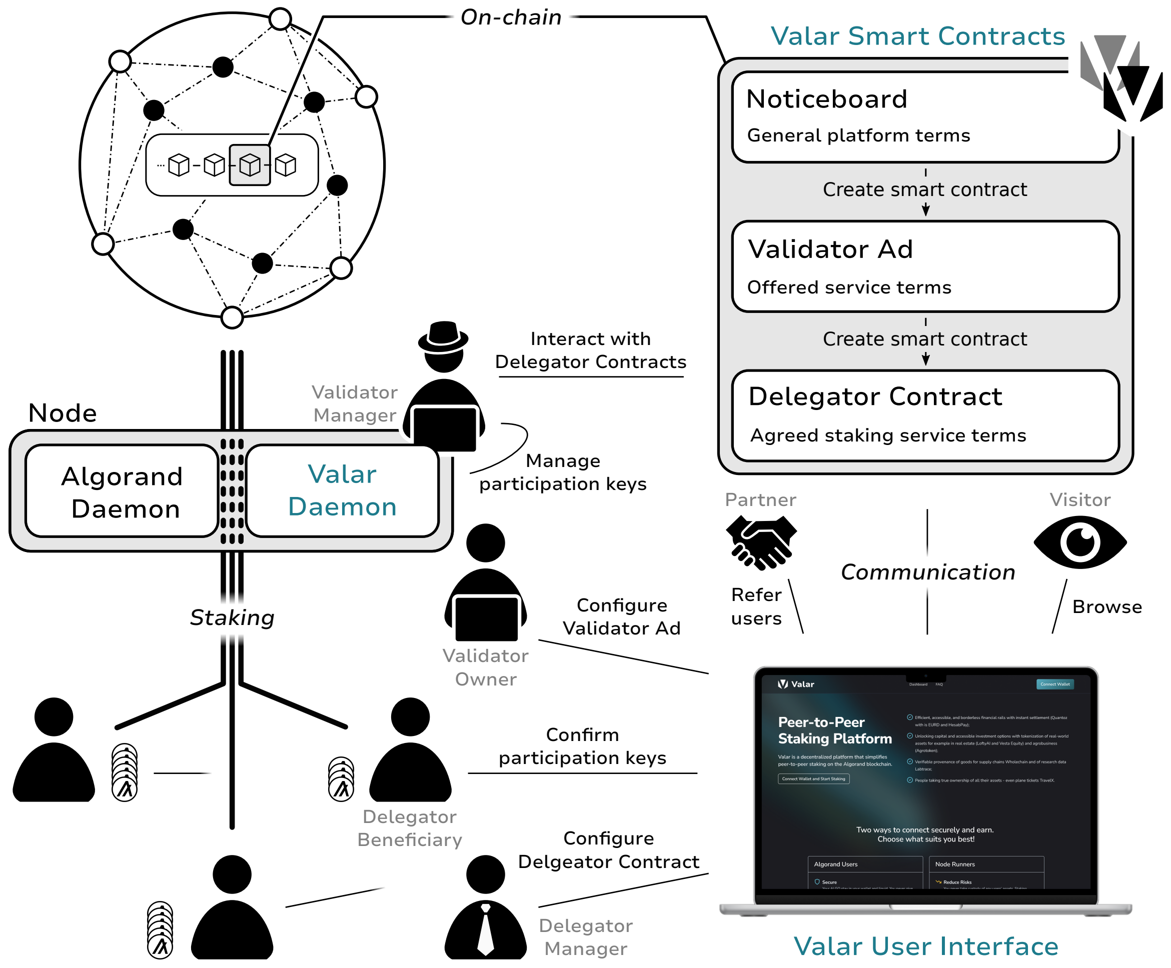

Figure 6.1: Overview of the Valar Platform's architecture, including additional details about its components, users, and user interactions.

After deployment, the Platform Creator conducts of the Noticeboard, also called setting of the Noticeboard. On this step, the Platform Creator uploads the Validator Ad and Delegator Contract templates to the Noticeboard's memory. These templates are later used to create the corresponding contracts. In addition, the Platform Creator appoints the Platform Manager, that will manage the platform during operation. The Platform Manager can be assigned by the Platform Creator without requiring the target account to accept the assignment.

6.2 Platform Manager

This section describes the user journey for the Platform Manager. Apart from the existence of an Algorand account with minimal funding and opting into the assets to withdraw platform's earned fees, there are no further requirements for this role. A new Platform Manager can be assigned by the existing Platform Manager or Platform Creator without requiring the new Platform Manager to accept the assignment. The Platform Manager interacts with the Valar Smart Contracts e.g. through the command line or any other tool.

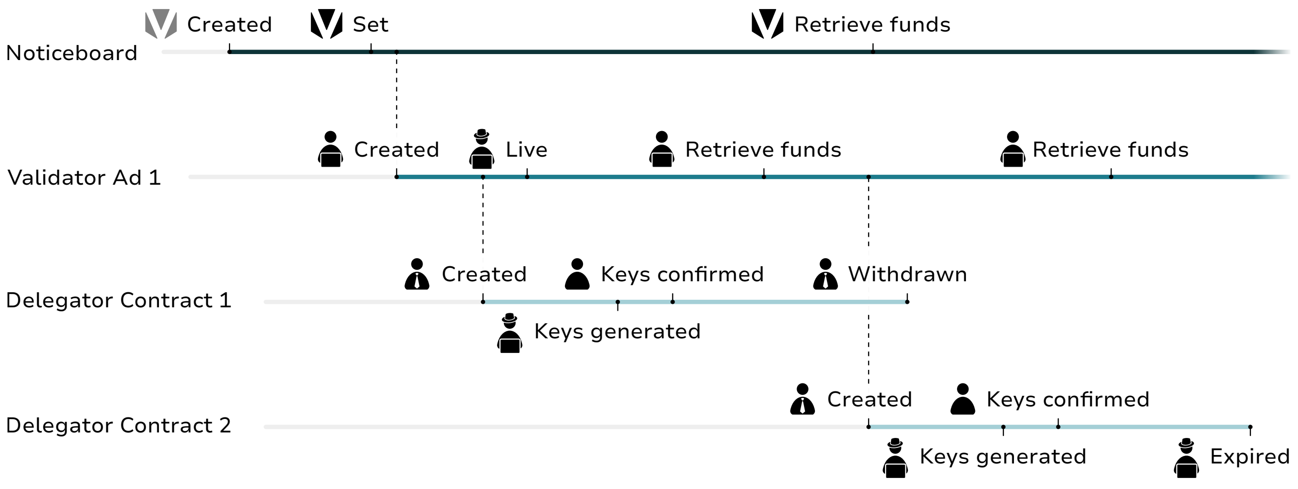

Figure 6.2: Example of the timeline of four smart contracts and corresponding user interaction.

6.2.1 Configuring and publicly launching the Noticeboard

Upon being appointed, the Platform Manager conducts a series of configuration steps on the Noticeboard before Validator Owners can start creating their Validator Ads. These configuration steps include setting the platform's T&C, commission and payment fees limits, accepted payment assets and opting in to these assets, listing platform partners, and other parameters described in Section 5.1.

6.2.2 Withdrawing Funds from the Noticeboard

Funds arising from platform commissions accumulate at the Noticeboard, i.e., in the Noticeboard's address. For example, the commission from payments issued by the Delegator Managers to the Validator Owners. These commissions accumulate at the Noticeboard and can be withdrawn by the Noticeboard manager as outlined in Figure 5.1. The Platform Manager is allowed to withdraw the accumulated funds on demand from the Noticeboard.

6.2.3 Updating the Configuration of the Noticeboard

The Platform Manager can reconfigure the Noticeboard during the Valar Platform's operation. Any of the configuration parameters can be amended, with the exception that the Noticeboard can no longer be opted out of an asset after it has been opted in. All existing Validator Ads and Delegator Contracts continue to operate. The only exception is that after the Noticeboard updates the platform's T&C, the Validator Ads require an update to be able to continue creation of new Delegator Contracts, described in Section 6.3.7.

6.3 Validator Owner

This section describes the user journey for the Validator Owner. In addition to the existence of an Algorand account with minimal funding, the Validator Owner has to register at the Valar Platform by paying a one-time registration fee (in ALGO). The fee acts as an anti-spam mechanism and is currently set to 5 ALGO. The Validator Owner is foreseen to interact with the Valar

Smart Contracts through the Valar UI. The following subsections describe the corresponding user journey.

6.3.1 Setting up the Participation Node

The Validator Owner must run a participation node in order to offer the node running service[2]. The node software can be installed by following the official documentation or community guides such as the Aust One Click Node (simple installation for any operating system), avm-win-node (for Windows), or a solution for Raspberry Pi 4.

6.3.2 Creating and Configuring a Validator Ad

A Validator Owner starts by visiting the Valar UI's and navigating to the page for Validator Ad creation. There, they configure the Validator Ad by setting the supported staking duration limits, the requested fees, the maximum staked amount limits, any eligibility requirements for staking, and other parameters listed in the Section 5.2.1. Note that the Validator Owner is requested to enter the monthly minimal operational fee and variable operational fee. The Valar Platform assumes that one month corresponds to 30 days. The values entered at the Valar UI are converted into per-round values based on the estimated average block time intervals to be stored in the smart contracts. Therefore, there can be a discrepancy in the actual monthly fees charged to the Delegator Manager.

To proceed with Validator Ad creation, the Validator Owner pays the Validator Ad creation fee. This is determined by the Noticeboard and is currently set to 5 ALGO. The fee serves as an anti-spam mechanisms. On the Valar UI, the user registration and the corresponding payment of the of the user registration fee is done at the same time as the creation of the first ad.

The Validator Ad configuration is successful if the entered configuration parameter values comply with the terms of the Noticeboard. When created, a Validator Ad allows the creation of new Delegator Contracts as soon as the Validator Owner enables accepting of new users and the Validator Manager (e.g. Valar Daemon) signals that it is ready to service them, as described in Section 6.4.1. The Validator Owner can decide to disable the creation of Delegator Contracts when configuring the Validator Ad, whereby the Validator Ad will not accept P2P delegation requests.

The Validator Owner can create multiple Validator Ads. It is recommended to use one Validator Ad for one node.

6.3.3 Configuring and Running the Valar Daemon

The Validator Owner should download the Valar Daemon from Valar's public repository and enter the configuration parameters. The configuration consists of Validator Manager's mnemonic and the Validator Ad associated with the node[3]. Additionally, the configuration allows setting

[2]A Validator Owner may create a Validator Ad and accept Delegators without operating a node. However, this will result in termination of Delegator Contracts since the Algorand's consensus monitoring mechanism will notice incorrect performance of Delegator's accounts.

[3]The Valar Daemon supports servicing of multiple Validator Ads on the same node. If this is desired, they can all be entered as part of Valar Daemon's configuration.

the Valar Daemon servicing response rate, as well as other parameters listed in Valar's public repository. The configuration is saved in a text file, which the Valar Daemon reads at the start of its runtime.

Running the Valar Daemon requires the installation of the defined version of Python and the dependency modules. The Valar Daemon (i.e. Validator Manager) enables automatic servicing of P2P delegation requests by handling all Delegator Contracts of a Validator Ad. A more detailed explanation of setting up the node and Valar Daemon is in development.

6.3.4 Updating the Configuration of a Validator Ad

The Validator Owner can update the Validator Ad configuration at any time through the Valar UI. The updated configuration will apply for future Delegator Contracts, while the terms and operation of existing Delegator Contracts will not be affected. The Validator Ad configuration update is only successful if the entered configuration parameters comply with the terms of the Noticeboard. Note that reducing the Validator Ad's maximum number of Delegators will not remove any active Delegator Contracts.

The Validator Owner can decide during reconfiguration to start accepting new P2P delegation requests if previously disabled. If accepting delegation was enabled, the Validator Owner can disable it, in which case existing Delegator Contracts are not affected.

6.3.5 Withdrawing Funds

The Validator Owner can at any time withdraw the earnings that have been accumulated at a Validator Ad. This is done individually for each Validator Ad through the Valar UI. Earnings need to be withdrawn individually for each payment asset that was accepted at any point in time by the Validator Ad.

6.3.6 Deleting a Validator Ad

The Validator Owner can delete the Validator Ad if there are no Delegator Contracts tied to it – all Delegator Contracts must have ended and must have been deleted by this point as per Section 6.6.5. This option is currently not available via the Valar UI.

6.3.7 Reconfiguring a Validator Ad following a Noticeboard Update

All Validator Ads are unable to create new Delegator Contracts following an update of Noticeboard's T&C. The Validator Owner can re-activate a Validator Ad by accepting the new terms during re-configuration of the ad, as described in section 6.3.4. This re-enables creation of new Delegator Contracts.

6.4 Validator Manager

The Validator Manager is most commonly controlled by the Valar Daemon on the participation node for automatic request servicing.

6.4.1 Setting the Validator Ad to Ready

The Validator Manager (e.g. Valar Daemon) must signal that it is ready for creation of new Delegator Contracts after the Validator Owner has enabled accepting of new delegation requests (see Section 6.3.2). The Validator Manager should periodically check the state of the Validator Ad that it is servicing and update the Validator Ad's state to `READY` if possible.

The checking period of the Valar Daemon is set to three seconds by default and can be changed by in its configuration.

6.4.2 Generating and Submitting Participation Keys

The Validator Manager (e.g. Valar Daemon) should monitor the state of all Delegator Contracts of the Validator Ad that it is servicing. When a new Delegator Contract is created by a Delegator Manager (see Section 6.6.1), the Validator Manager should start generating the corresponding participation keys on the participation node.

Once the participation keys are generated, the Validator Manager submits the required parameters (see Section 5.3.1) to the Delegator Contract. The waiting period starts for the Delegator Beneficiary to confirm the generated participation keys (see Section 6.5.1).

6.4.3 Checking and Reporting Contract Breaches

The Validator Manager (e.g. Valar Daemon) should monitor the compliance of Delegator Beneficiarys with the corresponding Delegator Contracts. Namely, the amount of staked funds, the presence of the agreed ASAs for eligibility, and the status of the payment (that no freeze or clawback has been issued). The Delegator Contract is ended according to the procedures described in Section 5.3 if a noncompliance with the terms is detected.

6.5 Delegator Beneficiary

The Delegator Beneficiary does not have to directly interact with the Valar Platform. The only required action by the Delegator Beneficiary is to issue the correct participation key registration transaction.

The Valar UI currently does not support separation of Delegator Beneficiary and Delegator Manager roles.

6.5.1 Confirming participation keys

The Delegator Beneficiary has to confirm the participation keys that have been generated by the Validator Manager (e.g. Valar Daemon) in order to stake and be eligible for Algorand staking rewards.

In the current version of Valar UI, where the Delegator Manager and Delegator Beneficiary roles are not separated, this is done at the same time when the Delegator Manager confirms that the keys have been signed. It is foreseen that the separation will be supported, whereby Delegator Beneficiary will be able to connect their wallet to the Valar UI (or use another solution) to sign keys before the Delegator Manager confirms them.

6.5.2 Breaching the Terms regarding Stake Amount or Eligibility

The maximal amount of stake that the Delegator Beneficiary account may hold during staking is defined in the Delegator Contract's terms. Breaching these limits can result in a warning and ultimately the Delegator Contract's early ending (see Section 5.3.2.11).

Similarly, the Delegator Contract may require the Delegator Beneficiary to maintain certain assets in their possession to be eligible to use their service. If the Delegator Beneficiary no longer has the required assets in the agreed amounts, this can result in a warning and ultimately the Delegator Contract's early ending (see Section 5.3.2.11).

6.6 Delegator Manager

The Delegator Manager is foreseen to interact with the Valar Smart Contracts through the Valar UI. The following subsections describe the corresponding user journey.

Note that the Valar UI currently does not support separation of Delegator Beneficiary and Delegator Manager roles.

6.6.1 Creating and Configuring a Delegator Contract

A Delegator Manager starts by visiting the Valar UI and navigating to the page that lists the available Validator Ads. There, the Delegator Manager is asked to enter the desired staking duration, payment currency, and maximum stake amount. Upon entering the three parameters, the platform will recommend a few Validator Ads that support the indicated parameters. The recommendation depends on the three entered parameters and the internal logic of the Valar UI. Alternatively, the Delegator Manager can browse the full list of Validator Ads and select one where it is eligible to stake.

Once a Validator Ad is selected, the Delegator Manager is presented with the contract details and a summary of the associated payment. The Delegator Manager can at this stage confirm the presented information and pay the associated setup fee and operational fee. Both of these are paid in their entirety upfront, together with any additional partner convenience fees. Before the Delegator Manager concludes their first Delegator Contract, they are also requested to pay an additional user registration fee to cover Algorand storage fees. The contract is created, awaiting the Validator Manager to generate and submit the participation keys (see Section 6.4.2). The Delegator Beneficiary must then confirm these keys before staking begins (see Section 6.5.1).

6.6.2 Ending a Delegator Contract by Withdrawing

The Delegator Manager can withdraw from the Delegator Contract at any time. The setup fee can not be refunded if the Validator Manager submitted the keys. The remaining (locked) operational fee is returned to the Delegator Manager upon withdrawal.

6.6.3 Breaching the Terms regarding Payment Assets

The Delegator Manager asset used for paying the setup fee and operational fee may be frozen or clawbacked for whatever reason by the asset issuer. If this happens and is reported, for example, by the Validator Manager (e.g. Valar Daemon), the Delegator Contract is ended early. Any remaining (locked) operational fee can be reclaimed by the Delegator Manager upon Delegator Contract end.

6.6.4 Extending a Delegator Contract

The Delegator Manager can request Delegator Contract extension through the Valar UI at any time. This automatically withdraws the current Delegator Contract and initiates a new one. Hence, any remaining operational fee is returned to the Delegator Manager, while the setup fee has to be paid again for generation of new participation keys. For extension, the Delegator Manager must define again the staking duration and the maximum stake. If the Validator Ad terms have changed since the initial contract, the new terms apply.

Note that the Delegator's place on the Validator Ad is not locked during the extension of the contract, so it can be occupied by another Delegator in this period. This is why the Valar UI does all the parts of contract extension atomically.

6.6.5 Deleting an Ended Delegator Contract

Deleting an ended Delegator Contract requires an additional manual action by the Delegator Manager. This action is present to enable claiming of any accumulated assets on the Delegator Contract, such as any portion of failed payments.

6.7 Partner

The Partner does not need to interact with the Valar Smart Contracts. Configuring of the Partner is done by Platform Manager. The Partner must only opt into and remain eligible to accept any asset that is supported as means of payment on the Noticeboard because any Partner commission is issued in the same asset as used for the payment by the Delegator Manager. The Partner automatically receives the commission fees directly to their account balance during fee-distribution transactions (see Section 5.3.2).

6.8 Visitor

A Visitor can browse the information provided by the Valar Platform without the need to connect a wallet. They can do this through the Valar UI, a blockchain explorer, or other means.

Appendix A

Brand Definitions

A.1 Mission

Valar's goal is to increase the security and robustness of the Algorand network in order to protect user funds and empower applications that rely on the network. Examples of applications running on Algorand include:

- Efficient, accessible, and borderless financial rails with instant settlement (Quantoz with is EURD and HesabPay).
- Unlocking capital and accessible investment options with tokenization of real-world assets for example in real estate (LoftyAI and Vesta Equity) and agrobusiness (Agrotoken).
- Verifiable provenance of goods for supply chains Wholechain and of research data Labtrace.
- People taking true ownership of all their assets - even plane tickets TravelX.

To fulfill its goals, the Valar platform tries to increase the number of nodes and the amount of online stake on the Algorand network. It does this by enabling direct delegation of users' security budgets to the node runners using a peer-to-peer staking approach. The platform provides a safe system with mutual benefits to participate in the network's protection. The system is based on a simple generation of (smart) contracts between an Algorand user and a node runner for the node running service. The contracts have embedded automatic, permissionless, fully transparent, and verifiable checks to ensure both parties fulfill the mutually agreed contract terms.

A.2 Descriptions

Keywords

- Protect (protection)
- Empower (empowerment)
- Decentralize (decentralization)
- Connect (connection)
- Stake (staking)

- Simplify (simple)
- Secure (security)
- Keep/take/give custody (custody)
- Peer-to-peer (direct)

Taglines

- Peer-to-peer staking, simplified (made easy).
- Secure. Decentralized. Transparent.
- Connecting Algorand users to node runners.

Boilerplate Valar is a decentralized platform for connecting Algorand users (blockchain stakeholders) to node runners, enabling direct staking of their ALGO (assets). By increasing the number of nodes and the amount of online stake, Valar protects on-chain assets and empowers solutions leveraging Algorand technology. Join a worldwide network of node runners now!

Elevator Pitch - Broad Valar is progressing humanity by building the infrastructure for the most robust and secure computer, accessible to anyone, anywhere, anytime. Valar connects people and incentivizes them to join this global infrastructure.

Elevator Pitch - Technical Valar connects Algorand users (blockchain stakeholders) to node runners, enabling access to staking rewards, while allowing full self-custody of one's own assets. If you would like to protect and grow your digital assets or monetize your infrastructure, then come and join a worldwide network of Algorand users and node runners!

Boilerplate Valar is a decentralized platform for connecting Algorand users (blockchain stakeholders) to node runners, enabling direct staking of their ALGO (assets). By increasing the number of nodes and the amount of online stake, Valar protects on-chain assets and empowers solutions leveraging Algorand technology. Join a worldwide network of node runners now!

Brief Description for Algorand Users Setting up a node or finding a node runner to stake your ALGO can be complex and time-consuming. With Valar you can connect with node runners and directly stake your ALGO with them without your ALGO ever leaving your wallet. By staking, you contribute to further decentralization of the Algorand network, thus protecting all on-chain assets and empowering solutions that are leveraging Algorand technology. The best part? The Algorand network rewards you for this with staking rewards! With Valar's direct peer-to-peer staking, you can spend your ALGO at any time, it is completely liquid without any lockups or slashing, and with you having full custody at all times. You pay no commission on the received staking rewards. Your staking costs are fixed in advance. You can pay them in stablecoins. Thanks to Valar, you can also get automatic performance guarantees.

Brief Description for Node Runners Finding and connecting with Algorand users can be a complex and time-consuming task. With Valar, you can easily let others utilize your nodes by delegating their stake to you, while paying you for the service. Valar fully automates the management of this process, and allows you to simply configure your requirements for accepting new Delegators as well as determine your pricing model, which can be denominated in stablecoins. The delegation process is done in a secure, transparent, and decentralized way as it happens fully on-chain. During the delegation process, you never take custody of user assets. The additional online stake further decentralizes the Algorand network, thus protecting on-chain assets and empowering solutions that are leveraging Algorand technology.

A.3 Color Pallette

Below are two views of the same color pallette with the main colors highlighted.

🟦	**Core primary** (`#0E3438`)	⬛	**Dark darkest** (`#1C1C21`)
🟦	Core primary light (`#145C69`)	⬛	Dark less dark (`#25252C`)
🟦	Core primary lightest (`#1D7B8C`)	⬛	Dark least dark (`#33333D`)
🟦	Core secondary dark (`#4D7177`)	⬜	Light least light (`#CBD5E1`)
🟦	**Core secondary** (`#6EA2AB`)	⬜	Light less light (`#F1F5F9`)
🟦	Core secondary light (`#A4CFD6`)	⬜	**Light lightest** (`#FFFFFF`)
🟦	Core tertiary darkest (`#3FAEC9`)	🟥	Error (`#DC2626`)
🟦	Core tertiary dark (`#69CBE3`)	🟧	Warning (`#EA580C`)
🟦	**Core tertiary** (`#A2EDFF`)	🟩	Success (`#16A34A`)
🟧	Gold (`#CF7F06`)		

Figure A.1: Color pallette including annotation colors. The main colors are marked in bold.

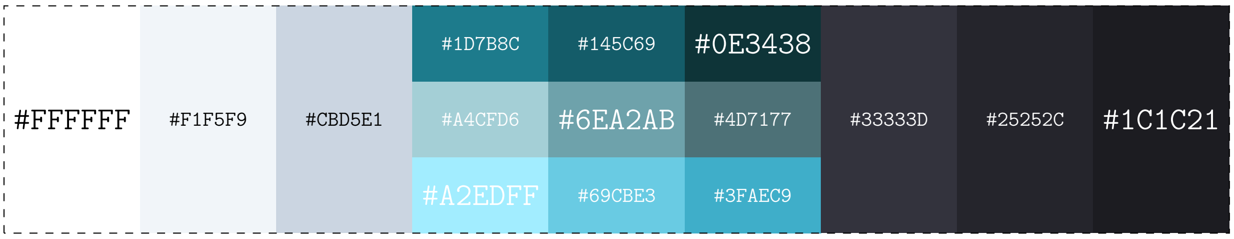

Figure A.2: Color pallette without annotation colors. The main colors are larger in size.

A.4 Logo and Distinction with Valar Solutions

Note that the Valar Platform primarily employs a white logo with a filled shield icon on a dark teal background, while the company Valar Solutions uses a dark teal logo with an empty shield icon (edges only) shield icon on a white background. The Valar Platform (Figure A.3) and the Valar Solutions (Figure A.4) company logo are shown below.

Figure A.3: The Valar Platform logo.

Figure A.4: The Valar Solutions logo.

Glossary

account Also known as an address, is a public identifier on the blockchain. It fulfills a similar role in blockchain as a bank account or phone number do in traditional systems. The access rights to an accounts are stored in a wallet, while the accounts balance is tracked on the blockchain.

address See "account".

ALGO The token of the Algorand blockchain network. It is used to secure the network by staking, to pay for the fees for its use, and in the network's governance.

Algorand A Pure Proof of Stake (PPoS) layer 1 blockchain.

blockchain explorer A tool to view and explore all the information on a blockchain including, transactions, blocks, addresses, and network metrics – e.g., `https://allo.info`.

blockchain account See "account".

cold wallet A wallet that stores the keys for an account with high security concerns. For example, an account that is used for long-term storage of a large amount of funds.

confirmation time The maximum amount of time that a Delegator on the Valar Platform has to confirm the setup that the Validator prepared for them. If the delegator does not confirm the setup in time, the contract ends. The setup fee is kept by the Validator.

consensus The process by which nodes in a distributed network agree on the validity of transactions and the state of the ledger.

consensus participation See "consensus".

Delegator See "Delegator Beneficiary".

Delegator Contract A Valar smart contract that defines the cooperation terms for delegated staking.

Delegator Manager Someone who manages Delegator Contracts for Delegator Beneficiarys.

Delegator Beneficiary A person or an entity that stakes their ALGO not by themselves.

hot wallet A wallet that stores the keys for an account with low security concerns. For example, an account with a small amount of funds that is used for automating recurring transactions.

liquid staking The act of sending one's funds to a third party's account for staking and receiving a token in exchange for the duration of the staking – the token can typically be traded or exchanged for the initial stake plus part of the staking rewards.

Mainnet A public Algorand network, which is under convention used for transactions of the Algorand cryptocurrency.

minimal operational fee The minimal fee per time unit that a Validator Owner accepts in exchange for allowing delegated staking.

node A computer that is running a specific computer program without interruption.

Node.js A runtime environment for the JavaScript programming language, commonly used in web development.

Noticeboard The central Valar smart contract, used for defining general terms, creating new contracts, and managing existing contracts.

operational fee A fee paid by the delegator to the node runner for the operation of the node. It is proportionate to the agreed duration of staking and the maximum stake.

participation keys A set of cryptographic private keys that are used solely for voting on and confirming the blocks of the blockchain network, i.e. for staking. These keys cannot be used to move any assets.

participation node A node that is participating in consensus.

Partner A person, project, company or other entity that is partnered with the Valar Platform and refers users to it.

Platform Manager The entity that configures and maintains the Noticeboard smart contract.

Platform Creator The entity that deploys the Noticeboard smart contract.

private key A cryptographic code to securely sign messages or authorize transactions – synonymous to a spending key.

Python A programming language used for developing the Valar smart contracts and daemon.

setup time the maximum amount of time the delegator must wait for the Validator to prepare the node for them, after having paid the setup fee. If the Validator does not prepare the node in time, the delegator can automatically refund the paid setup fee, ending the contract.

setup fee A flat fee paid by the delegator to the node runner for preparation of the node for staking. The setup fee serves as an anti spam mechanism because the node preparation is a computationally intensive task.

spending key Cryptographic private keys used to sign, i.e. authorize and approve, all transactions of a user on a blockchain.

stablecoin A crypto asset typically pegged to a fiat currency (like USD).

stake pooling The act of accumulating funds from multiple accounts in a single account and staking the accumulated funds.

staking The process of securing the blockchain network by validating transactions and producing new blocks on the blockchain network – synonymous to consensus participation.

staking rewards Incentives that are give out for participating in consensus (staking).

TEAL Transaction Execution Approval Language, an assembly-like language that is processed by the Algorand Virtual Machine.

token An asset with a unique identifier on the blockchain.

USDC A stablecoin pegged to USD.

Valar Platform for peer-to-peer staking on Algorand.

Valar Daemon A computer program for Validators that want their nodes to automatically service the cooperation requests of Delegators made via the Valar Platform.

Valar Smart Contracts The designed smart contracts that make up the core of Valar Platform.

Valar Platform See "Valar".

Validator Also called a node runner, is a person or an entity that is validating transactions and producing new blocks on the blockchain network.

Validator Manager The Validator's account associated with its hot wallet that interacts with delegator contracts.

Validator Owner The Validator's account associated with its cold wallet that configures the Validator Ad and manages the validator's funds.

Validator Ad The validator advertisement smart contract.

variable operational fee The part of the operational fee, which is proportional to the amount of staked ALGO and the duration of the staking.

Visitor Anyone that is browsing the Valar smart contracts or other information on the Valar user interface.

wallet A digital wallet that holds the spending keys of one or more blockchain accounts.

ASA Algorand standard asset.

MBR minimum balance requirement.

NFT non-fungible token.

P2P peer-to-peer.

PoS proof of stake.

PPoS pure PoS.

T&C terms & conditions.

UI user interface.