

Tokens

Introduction

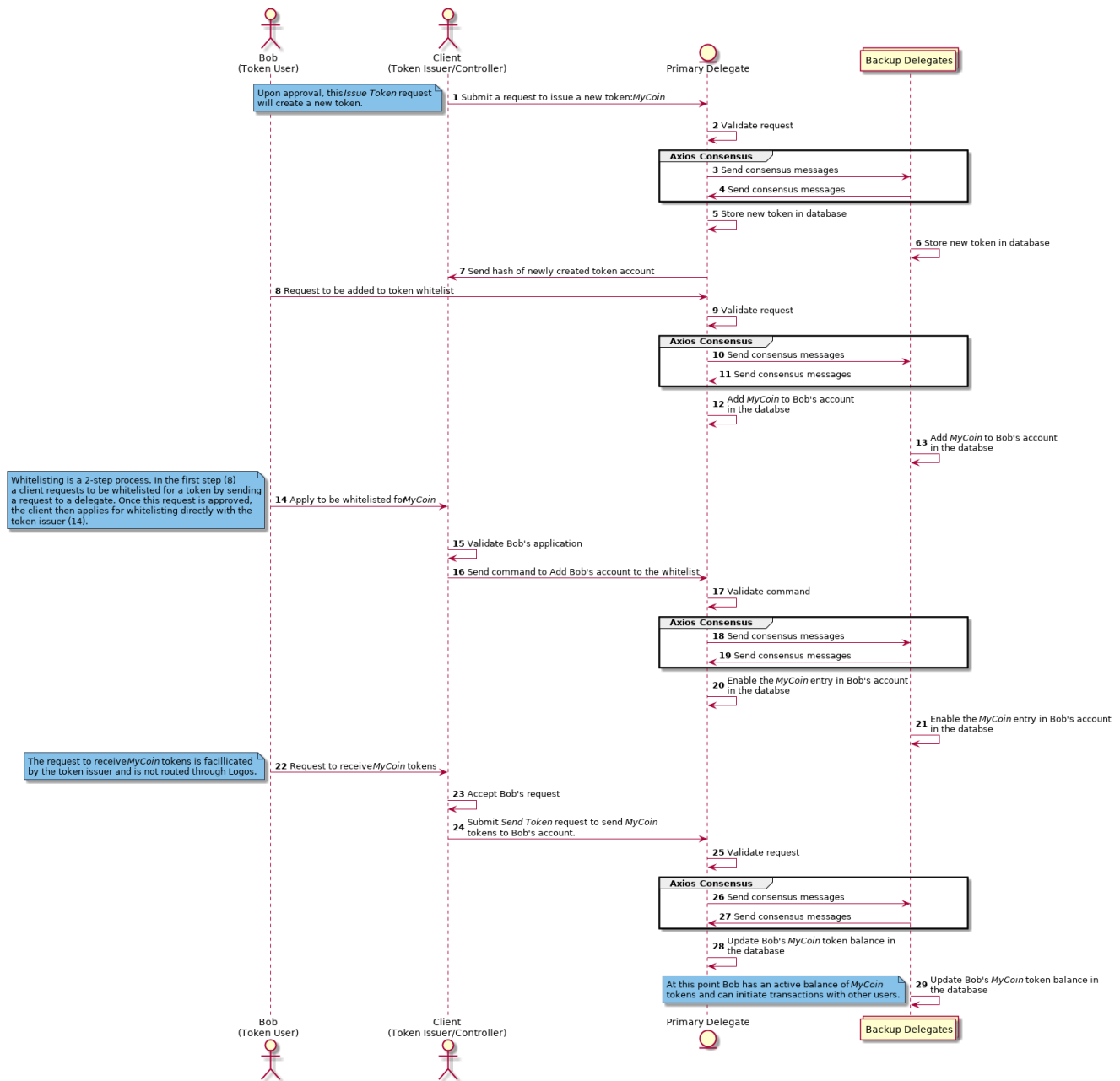
The Logos token platform is an extension of the Logos Network in that it supports the issuance, management and daily operations of tokens on top of the Logos Network architecture. All token operations are initiated by requests that are submitted to delegates and then approved via Axios Consensus. As the token platform is an extension of the existing Logos Network architecture, this section assumes that the reader has a basic understanding of that underlying system, and some details may be omitted.

Token accounts are extensions of conventional Logos accounts in that they have a balance that can be increased or decreased by making transactions on the network, resulting in a historical chain of operations. Additionally, token accounts contain settings that dictate which operations the token does or does not support. These settings also have mutability options that dictate whether or not the settings can be changed. For example, if the total supply for a particular token is set as immutable, then additional tokens cannot be issued. Furthermore, operations performed on a token account are chained together (as with transactions between regular Logos user accounts) with each operation referring to the hash of the request that preceded it. The following sections describe the Logos token platform in greater detail.

Token Issuance

Issue and Request Tokens





Freezelist

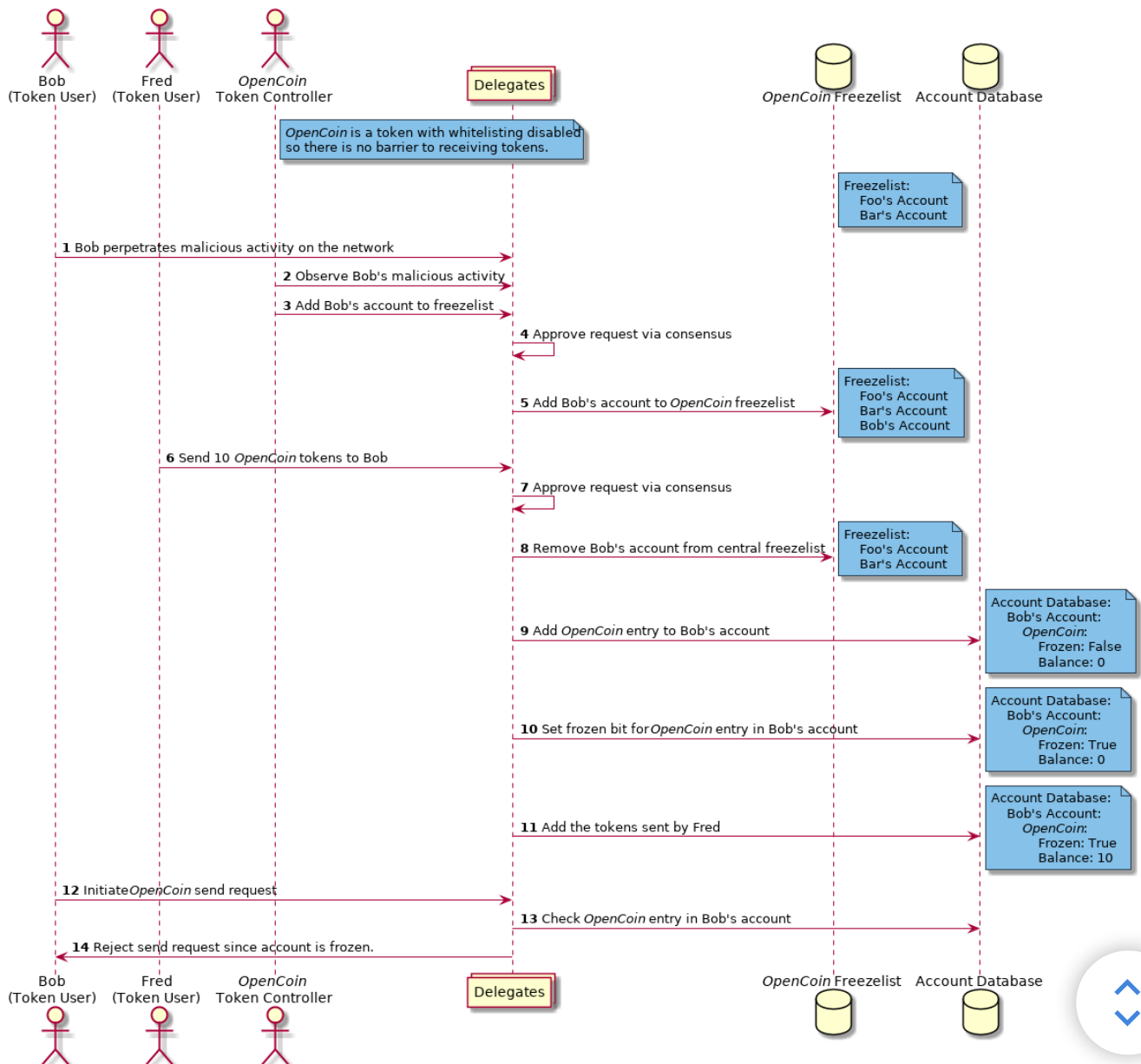
Freezing an account prevents it from initiating send transactions and is enforced per token account. Requests to send tokens from frozen accounts will be rejected. Regarding the implementation, the freezelist was originally to be represented as a central list containing the address of each frozen account, but this approach has the following drawbacks:

- This can result in database bloat if many accounts are frozen.
- Nodes may have to scan through many entries to confirm whether a particular account is or isn't frozen, impacting performance.



One solution is to simply use a single bit in the token entry stored within an individual account to indicate whether this account is frozen. However, a token account controller may want to freeze accounts that have not yet been tethered to the token (an account is tethered by being whitelisted by a controller or by receiving tokens). To allow this we would have to enable a controller to add a token entry to an untethered account in order to set the appropriate bit, however, this can be exploited by a malicious controller who spams the network by freezing every account, permanently increasing the size of each account. To mitigate this, a central freezelist will be used for untethered accounts. Since the users' accounts themselves will be used to implement the freezelist, the central freezelist doesn't need to be checked for transactions between accounts with preexisting token entries. The only time the freezelist needs to be checked is when an account is being tethered and an entry for this token is being created in the individual account. When this occurs, the freezelist will be checked, and if the account in question is listed therein, its address will be removed from the freezelist and the frozen bit will be set within the account entry. In this way, the freezelist will be pruned automatically.

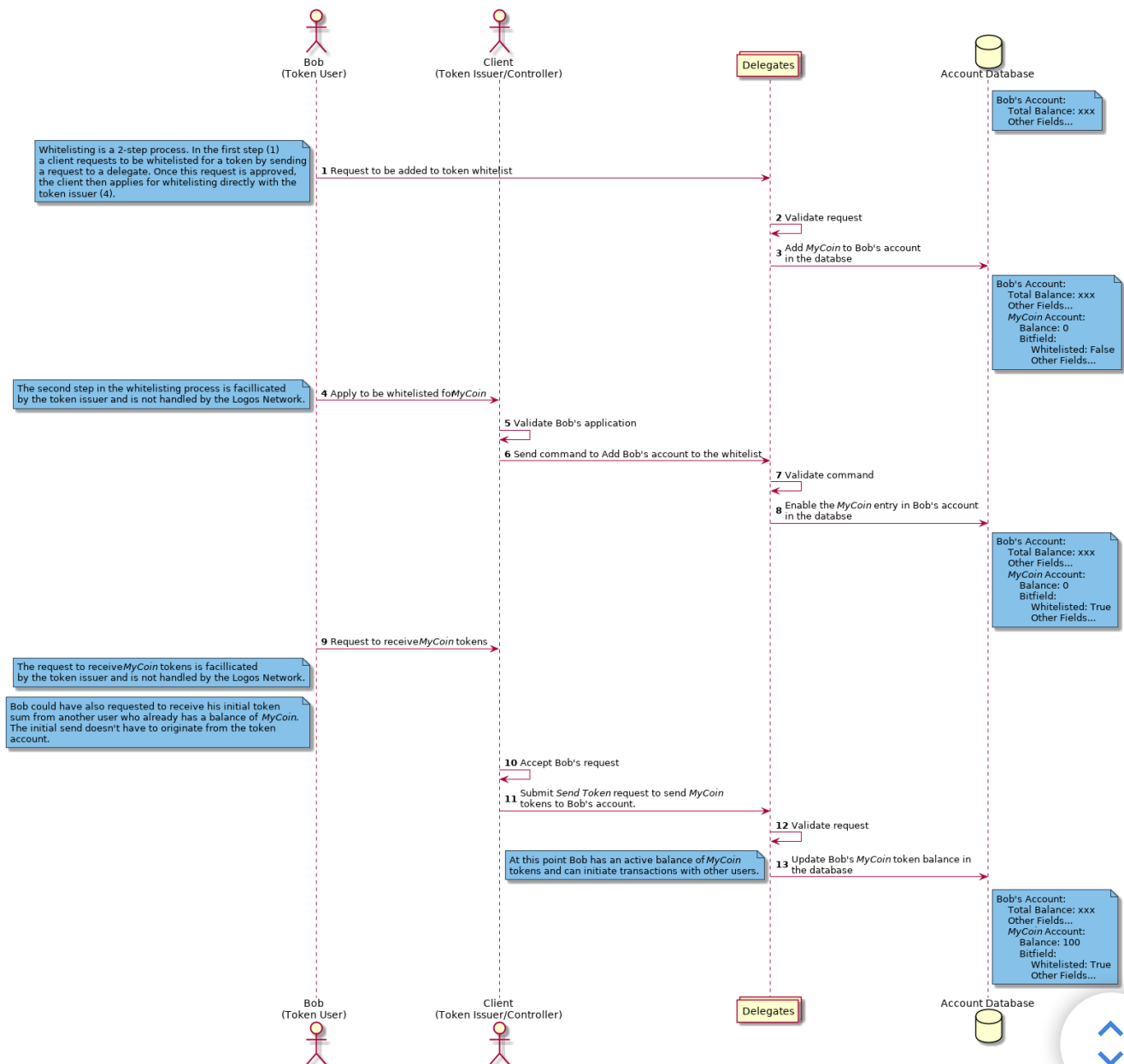
Freezing an Untied Account



Whitelist

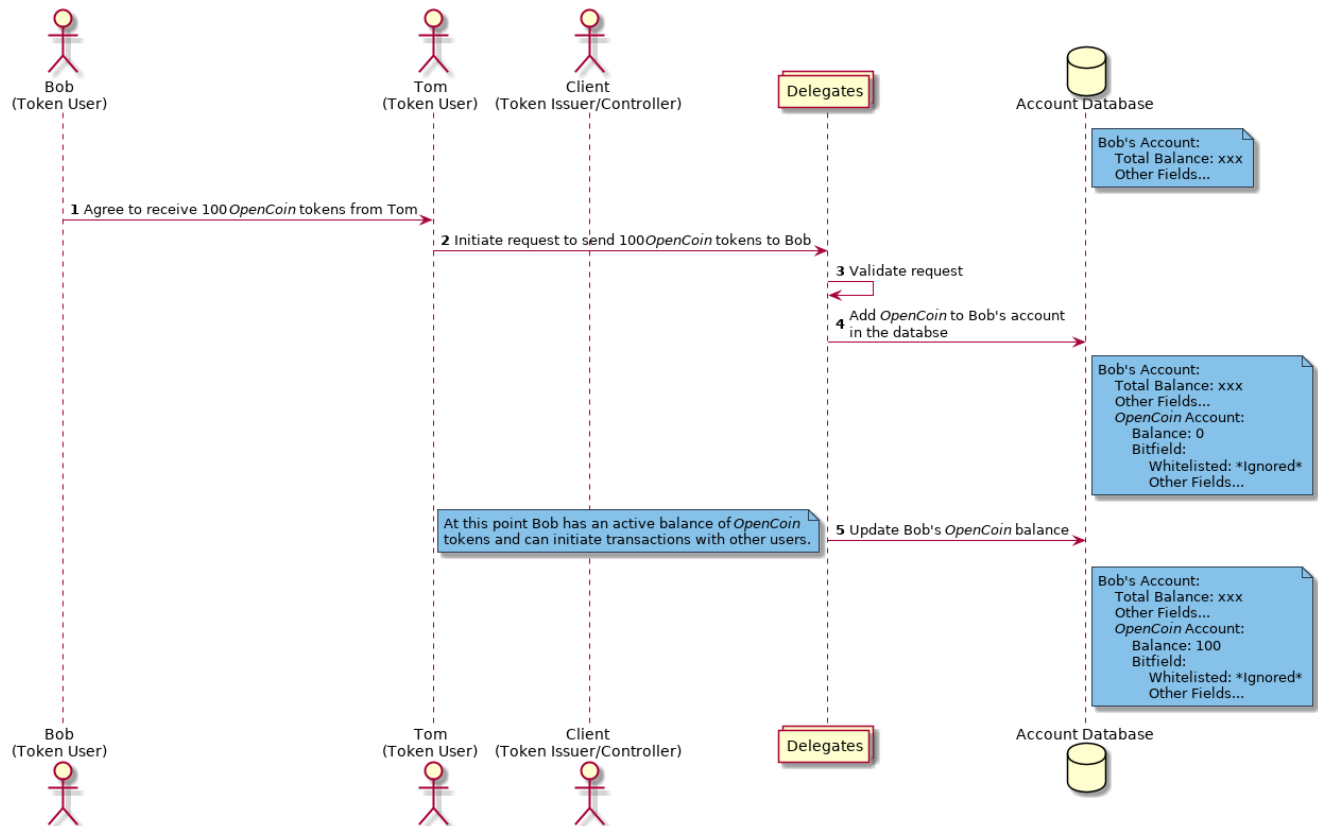
Tokens that use the whitelisting feature require that before receiving any amount of the token, the users first complete a two-step application process, after which their account address will 'whitelisted' and able to receive tokens. To save space, rather than maintain a centralized list to store the addresses of accounts that are whitelisted for a particular token, we leverage the token entry added to individual accounts to indicate that this account has been whitelisted by a controller for the token.

Whitelisting Process



Whitelisting Disabled

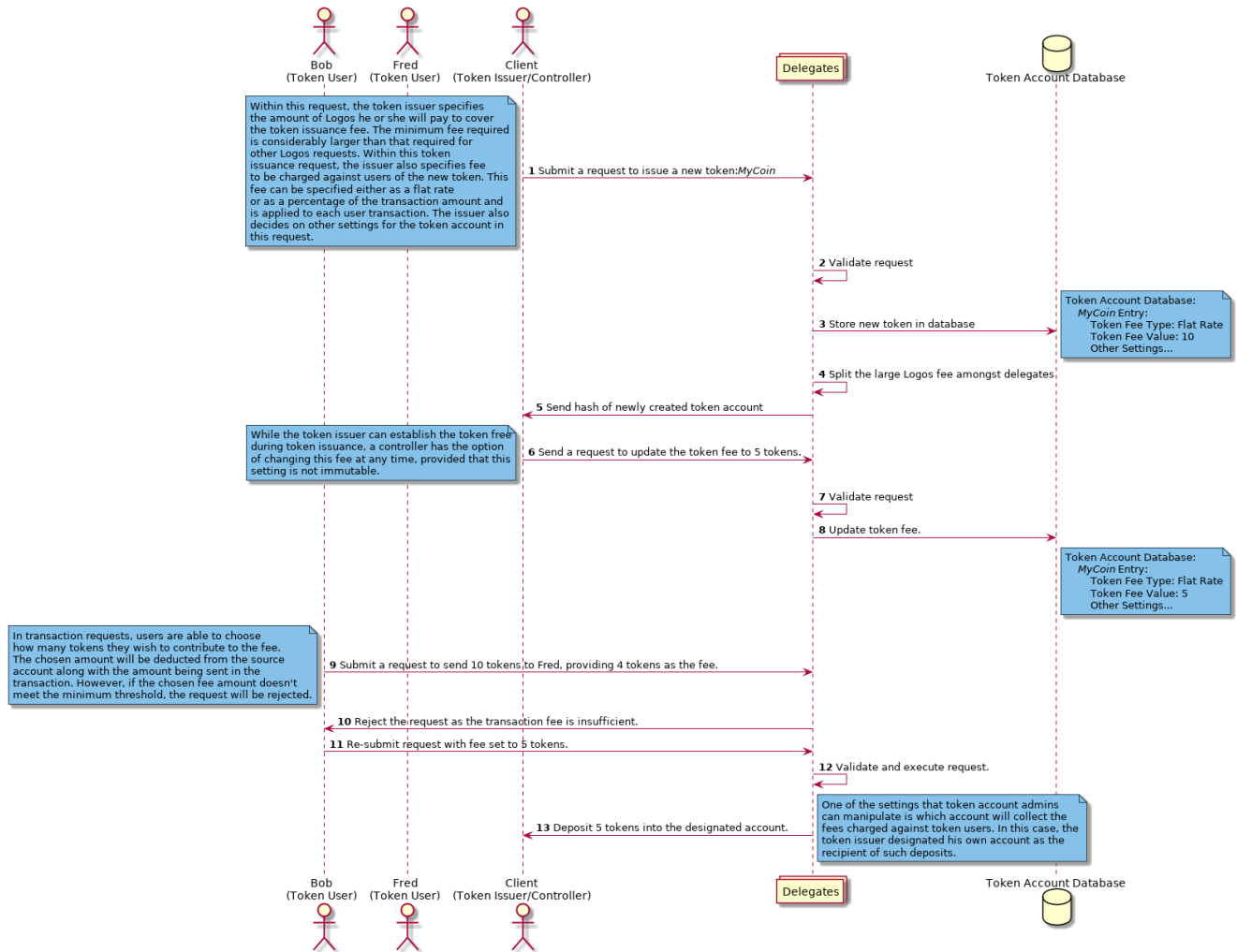
Tokens that do not use the whitelisting feature provide no barrier to entry, and any account can receive tokens at will.



Fees

Issuing a new token on the Logos Network requires the issuer to pay a considerable fee (higher than the minimum fee for native send transactions) in Logos. The exact amount required is a configuration parameter determined by the system and as with all Logos fees, the sum will be split among the delegates. Once a token has been issued, all requests involving it are subject to the standard Logos Network fee which is paid in Logos, plus an additional fee designated by the token's administrators (Token Issuer/Controllers), which is paid in tokens. The token fee for such requests is deducted from the source account's token balance, and deposited in an account determined by the token admins.





Token Settings

Each token created on the Logos Network has a list of settings that describe the operations that can be performed on it. The values for these settings are determined by the token issuer when the token is first created, and can be subsequently modified by token controllers with sufficient privileges.

Setting	Description
Allow Additional Tokens	This determines whether the initial supply of tokens can ever be increased.
Allow Additional Tokens Mutable	*
Allow Revoke	This determines whether we can revoke tokens from user account
Allow Revoke Mutable	*

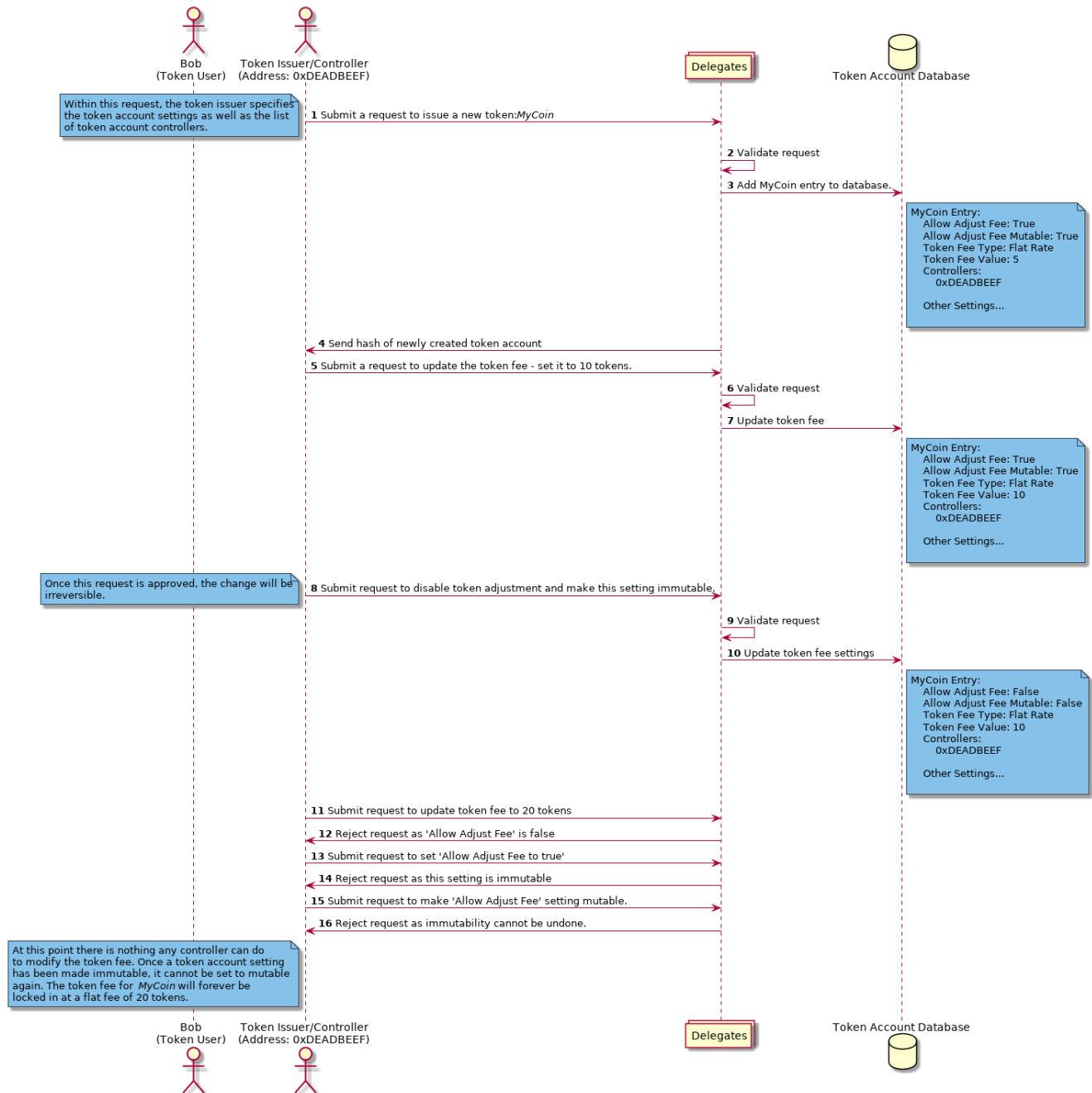
Setting	Description
Allow Freeze	This determines whether we can freeze user accounts.
Allow Freeze Mutable	*
Allow Adjust Fee	This determines whether we can change the transaction fee rate.
Allow Adjust Fee Mutable	*
Need Whitelist	This determines whether accounts need to be whitelisted before receiving tokens.
Need Whitelist Mutable	*

* Mutability setting descriptions are omitted as these settings just determine whether the corresponding setting can be modified. Once a setting is marked as immutable (the mutability setting is set to false), it cannot be changed.

Mutability

A request to modify a token account parameter must be allowed by the corresponding token account setting. Immutable settings cannot be changed at all.





Controllers

A controller is an entity that has administrative privileges for modifying the settings governing a particular token. Token controllers can be appointed by the token issuer when a token is first created, or by a current controller once the token is already live. In order to modify a token account setting, a controller must submit (to a delegate) a request that expresses the desired change. Once this request is approved via Axios Consensus, the change will take effect on the Logos Network. The changes that a controller can make to a token account are dictated by its set of privileges.



Controller Privileges

Privilege	Description
Allow Additional Tokens	Whether the controller can modify the corresponding token account setting.
Allow Additional Tokens Mutable	"
Allow Revoke	"
Allow Revoke Mutable	"
Allow Freeze	"
Allow Freeze Mutable	"
Allow Adjust Fee	"
Allow Adjust Fee Mutable	"
Need Whitelist	"
Need Whitelist Mutable	"
Promote Controller	Whether the controller can: <ul style="list-style-type: none"> • Promote another account, making it a controller. • Assign privileges to new or existing controllers.
Issue Additional Token	Whether the controller can issue additional tokens.
Revoke	Whether the controller can revoke tokens from an account.
Freeze	Whether the controller can freeze an account.
Adjust Fee	Whether the controller can change the transaction fee rate.
Whitelist	Whether the controller can whitelist an account.
Burn	Whether the controller can burn tokens from the token account balance.
Withdraw Balance	Whether the controller can withdraw tokens from the token account balance.
Withdraw Fee	Whether the controller can withdraw tokens from the token account transaction fee pool.



Databases

The Logos Network token platform leverages several databases for storing various information concerning tokens created on the platform:

- **Token Account DB:** This database will be used to store token accounts. Token accounts stored in the database contain their list of settings with current values, as well as the list of token controllers and their respective privileges.
- **Freezelist DB:** The freezelist database will be used to store accounts that have been frozen for a particular token, forming a centralized list. Each token account gets its own freezelist. Note that the freezelist stored here is partial, as tethered accounts will be marked frozen by using a bitfield stored within the user's account.
- **Account DB:** The account database is used by the base Logos Network and is also leveraged by the token platform, which stores token entries inside of user accounts. Each token entry contains that user's balance of tokens among other useful information.

[→ Next to Echo](#)

