1 Principles of DARC Protocol

In designing DARC to be a regulated, programmable, customizable, profitable, and sustainable business entity, we adhere to the following principles:

1.1 Plugin as a Law

For real-world companies, compliance with a multitude of laws and regulations is essential. This compliance encompasses not only a company's By-laws but also its internal policies, agreements between the company and its employees, contracts with customers and suppliers, and agreements among shareholders. These agreements and rules define the operational procedures and boundaries of the company, forming the foundation for its healthy operation and growth. They not only determine the company's structure but also provide detailed descriptions of how the company operates and how profits are distributed.

As a purely virtual corporate entity existing solely on an EVM-compatible blockchain, within the DARC protocol, plugins serve as a core and foundational mechanism, representing a variety of by-laws, contracts, legal agreements, documents, and more. In each DARC protocol, there exists a set of plugins that form the fundamental laws governing the DARC. All operations and activities conducted within the DARC must rigorously adhere to and comply with all restrictions and conditions outlined by these plugins.

For the DARC protocol, plugins need to adhere to the following principles:

1.1.1 Design of Plugins

Each plugin consists of two crucial components: "condition" and "decision". The "condition" represents the triggering criteria that activate the plugin, while the "decision" specifies the actions to be taken when the condition is met.

The "condition" of each plugin is programmable and configurable, allowing for flexibility. When an operation is executed or when the DARC's state aligns with the specified condition, the plugin responds by implementing the corresponding "decision".

1.1.2 Levels of Plugins

For each operation, there is an associated "level". Within each DARC protocol, there are multiple plugins, each of which may have the same or different levels. When multiple plugins are triggered simultaneously for a single operation, the DARC protocol selects the plugin with the highest level and uses that plugin's decision as the final decision.

Additionally, all plugins at the same level must have the same decision type. This requirement ensures that when multiple plugins at the same level are triggered simultaneously, the final decision is consistent. This design simplifies the decision-making process and ensures that there is no ambiguity when multiple plugins of the same level are involved.

1.1.3 Immutability of Plugins

Once a plugin is added to the DARC protocol, it cannot be changed or modified. Users have the capability to enable or disable one or multiple plugins through the use of "enable operation" or "disable operation". When users wish to modify the rules represented by a plugin, they must disable the existing plugin that represents those rules and then add and enable a new plugin to replace it. This approach ensures that the integrity of the protocol is maintained while allowing for changes in the rules as needed by the users.

1.1.4 Authority of Plugins

In the DARC protocol, the plugin system holds authoritative control. For each operation, if it is rejected by the plugin system, it cannot proceed further. If the plugin system requires a vote, the operation can only be executed after undergoing the voting process. Only when the plugin system grants full approval can an operation be directly executed. This design ensures that the plugin system has the final say in the execution and validation of operations within the DARC protocol.

In the DARC Protocol, operations related to plugins include enabling plugins, disabling plugins, adding plugins, and adding and enabling a plugin. When users execute these actions associated with plugins, they are required to adhere to the regulations and constraints set by the existing plugins, just like any other operations. These operations targeted at plugins can only be executed after receiving approval from the current set of plugins.

In summary, any operations involving modifications to plugins must also receive approval from the current set of plugins. This ensures fairness and compliance with the rules and regulations established by the existing plugins within the DARC Protocol.

1.2 Program and Operations

For each DARC program, it consists of a series of operations, where each operation includes an opcode, a set of parameters, and an operator address. When an operator requests the DARC to execute a program, several scenarios may arise:

1. If one or more operations within this program receive a rejection decision from the plugin system, the entire program will be denied execution. In such a scenario, even if some of the operations meet the requirements, the program as a whole will be unable to proceed.

2. If all operations within this program do not receive rejected decisions from the plugin system, but one or more operations in this program receive a "voting needed" decision, a voting process is initiated. In this scenario, all voting items will be consolidated into a single voting process, and the DARC enters a voting state. This allows all token holders to participate in the voting. If the program is approved through the voting process, it can proceed with execution. However, if it is rejected through the voting, the project will be rejected.

3. If each operation within this program receives an "approved" decision from the plugin system, in this scenario, all operations within the program can be executed sequentially and directly.

1.3 Multi-level Token System

The DARC protocol features a multi-level token system, with each level of tokens having independent voting weights and dividend weights, where the minimum values for these weights can be set to 0. It's important to note that the voting weight and dividend weights for each level of tokens are immutable and cannot be altered. Users have the capability to initialize a new level and perform a range of operations, including minting, burning, transferring, and more for all tokens.

By assigning voting weights and dividend weights to tokens at each level, imposing limitations on token quantities, and incorporating additional plugins to restrict and design various token-related operations, multi-level tokens can serve a multitude of purposes, including common stock, bonds, board votes, A/B shares, preferred stock, common commodities, Non-Fungible Tokens (NFTs), and more.

1.4 Dividends

A company has two methods of spending money: one is through direct cash payments, typically used for purchasing, salary disbursement, bill payments, and bond redemption, among other purposes. These payments generally involve one-time or multiple transactions of fixed amounts. The other method is through dividend payments, where the company allocates a specific amount or a certain percentage of funds and distributes them to all shareholders based on dividend weights.

In the DARC protocol, the dividend mechanism distributes a dividend, which consists of X permyriad of the accumulated income from every N transactions. This dividend is distributed to all token holders based on their dividend weight.

In the DARC protocol, there is a dividend cycle counter. Every time the DARC receives a dividendable payment, this counter automatically increments by 1, and the amount of this payment is added to the dividendable fund pool. When the dividend cycle counter reaches the predefined dividend cycle N within the DARC protocol, users can, subject to approval by the plugin system, execute the "`OFFER_DIVIDENDS`" operation. This operation disburses X permyriad of the funds from the dividendable fund pool, calculates the dividends, and distributes them to the accounts of each token holder. Subsequently, both the dividendable fund pool and the dividend cycle counter are reset to zero.

1.5 Voting

In the DARC protocol, for operations that cannot be directly approved or denied by the plugin system after triggering specific conditions, a voting mechanism can be employed to make the final decision. The voting mechanism can serve various purposes, including:

1. Democratic decision-making where all token holders have one vote each, suitable for major decisions involving a large number of token holders.

2. Quick and simple decision-making for smaller groups such as boards or committees.

3. Approval processes for routine tasks involving multiple managers taking turns to approve daily affairs.

4. Weighted voting processes for different groups, including A/B classes or multiple levels of voting rights.

For each plugin, if the decision is "voting needed", the plugin must be associated with a voting item. When the condition of this plugin is triggered, and the plugin has the highest level among all the plugins with triggered conditions, DARC will select the voting item specified by that plugin as the voting rule. Once all the voting items are collected by DARC, a voting process will be initiated based on these items. All token holders who meet the criteria specified by this series of voting items are eligible to participate in the voting.

After a program undergoes evaluation by the plugin system, each operation within the program may be subject to voting as requested by one or more plugins. Each plugin will point to a specific voting item. When DARC initiates a voting process, it collects all the voting items required for the operations and proceeds to the voting phase. Assuming that the total number of collected voting items is N, each voter must submit a program containing only one voting operation. This operation must include a boolean value array with a length of N, corresponding to the voting results for the N voting items. Each time an operator submits their vote, the voting process calculates the voting weight associated with that operator for each of the N voting items. It tallies the votes separately for each of the N voting items, adding the voting weight to the respective voting result. If a user's token balance is zero for one or more voting items, their voting weight for those specific items is also considered as zero.

1.6 Emergency

In a rule-based corporate virtual machine, DARC operators may encounter various types of errors, including operational errors, incorrect parameters in operations, and various potential non-technical conflicts and disputes. DARC maintains a backdoor, known as "emergency agents", to serve as firefighters in emergency situations.

In a DARC protocol, operators have the ability to designate multiple addresses as emergency agents. In the event of an emergency, and with the permission of the plugin system, operators can call upon one or more of these emergency agents for assistance. Once an emergency agent is summoned, they gain super-administrator privileges within the DARC, granting them the authority to perform any action. This includes adding, enabling, disabling plugins, conducting token operations, and managing cash assets.

Since emergency agents possess the highest level of comprehensive administrative authority, their role can be likened to that of both a court and a firefighter. Therefore, it is essential for all members of the DARC to have complete trust in these emergency agents. Additionally, specific conditions should be established for calling upon emergency agents to prevent them from taking untimely or inappropriate actions, which could lead to damage or losses within the DARC.