1 多级别代币系统

多级别代币系统是 DARC 协议的核心机制，用户可以设计具有不同投票权重和分红权重的不同级别的代币。通过插件作为限制，多级别代币系统可用于代表组织中的不同组成部分或资产。

1.1 普通股

普通股是股份公司机制的中心元素，作为筹集资本和分配所有权的手段。普通股股东通常享有投票权，允许他们在股东会议上对重要公司决策发表意见。此外，普通股股东可能会收到分红，这是公司利润分配给股东的一部分。投票和分红的双重属性使普通股成为股东参与公司治理和财务回报的关键工具。

如果一个代币级别被分配了 1 的投票权重和 1 的分红权重，并且 DARC 中的所有重要事项都必须得到该级别所有代币持有者的批准，那么这个代币级别可以被视为 DARC 中的普通股。

要实例化这样一个代币级别，首先必须在法规脚本中建立它，分配 1 的投票权重和 1 的分红权重。成功执行此脚本后，DARC 协议将初始化一个具有 1 的投票权重和分红权重的 0 级别代币。

```
batch_create_token_classes(
    ["TOKEN_0"],   // 代币符号字符串
    [0],           // 代币级别
    [1],           // 投票权重
    [1]            // 分红权重
);
```

考虑到公司事务的复杂性，创建了两个用于不同目的的投票规则：

投票规则 1：要批准一个操作，必须满足以下条件：超过 50% 的有效投票权力支持该操作，并且这次投票需要是绝对多数模式，意味着它必须包括至少 50% 的总代币投票权力。投票规则 1 的投票时间为 7 天 (604800 秒)。程序获批后，操作者需要在 1 小时 (3600 秒) 内执行程序。

投票规则 2：要批准一个操作，必须满足以下条件：超过 75% 的有效投票权力支持该操作，并且这次投票需要是相对多数模式，意味着它必须包括至少 75% 的有效投票权力。投票规则 2 的投票时间为 1 天 (86400 秒)。程序获批后，操作者需要在 1 小时 (3600 秒) 内执行程序。

以下是投票规则 1 和 2 的初始化法规脚本。当添加一个带有投票规则索引 1 或 2 的新插件时，插件将确保当条件被触发且插件优先级最高时，将根据选定的投票规则启动投票过程。

```
batch_add_voting_rule([
    // 添加规则1
    {
        // 仅允许0级别代币进行投票
        votingTokenClassList: [0],

        // 批准门槛：50%
        approvalThresholdPercentage: 50,

        // 投票持续时间：604800秒
        votingDurationInSeconds: 604800,

        // 执行待定持续时间：3600秒
        executionPendingDurationInSeconds: 3600,

        // 投票规则是否启用：是
        isEnabled: true,

        // 关于投票规则的注释
        note: "这是投票规则索引1",

        // 是否是绝对多数
        bIsAbsoluteMajority: true
    },

    // 添加规则2
```

```
    {
        // 仅允许0级别代币进行投票
        votingTokenClassList: [0],

        // 批准门槛：75%
        approvalThresholdPercentage: 75,

        // 投票持续时间：86400秒
        votingDurationInSeconds: 86400,

        // 执行待定持续时间：3600秒
        executionPendingDurationInSeconds: 3600,

        // 投票规则是否启用：是
        isEnabled: true,

        // 关于投票规则的注释
        note: "这是投票规则索引2",

        // 是否是绝对多数
        bIsAbsoluteMajority: false
    }
]);
```

1.2 A/B 类股

A 类和 B 类股是公司财务的基本工具。A 类提供投票权和分红优势，通常由创始人持有。B 类具有限制的投票权，用于在保留控制权的同时筹集外部资本。这种双重类别结构平衡了治理需求和增长需求，对股东的决策制定和财务利益有所影响。

首先初始化两个代币级别：0 级别具有 1 的投票权重和 1 的分红权重，1 级别具有 100 的投票权重和 1 的分红权重。

```
batch_create_token_classes(
    ["TOKEN_0", "TOKEN_1"],   // 代币符号字符串
    [0, 1],           // 代币级别
    [1, 100],         // 投票权重
    [1, 1]            // 分红权重
);
```

接下来，我们将铸造 10000 个 0 级别代币和 200 个 1 级别代币。这将使 0 级别代币的总投票权力为 10000，1 级别代币为 20000。这样，DARC 的总投票权力为 30000，其中 66.7% 在 1 级别代币。通过这种方式，共同创始人和早期投资者将控制 DARC 的大多数投票权力，允许所有 0 级别和 1 级别代币持有者参与投票过程。

我们假设 addr1 和 addr2 总共持有 10000 个 0 级别代币，每人 5000 个，每个 addr3、addr4、addr5 和 addr5 持有 50 个 1 级别代币。首先在法规脚本中将代币铸造给这些地址。

```
batch_mint_tokens(
  [0, 0, 1, 1, 1, 1 ],
  [5000, 5000, 50, 50, 50, 50],
  [addr1, addr2, addr3, addr4, addr5, addr6]
);
```

接下来添加一个插件以限制 0 级别和 1 级别代币的总供应量分别为 10000 和 20000。

投票规则 1：要批准操作，0 级别和 1 级别代币的总投票权力的 90% 必须投 YES。投票过程需要是绝对多数模式。投票规则 1 的投票时间为 7 天 (604800 秒)。程序获批后，操作者需要在 1 小时 (3600 秒) 内执行程序。

操作前插件规则 1：如果操作是 BATCH_MINT_TOKENS 并且铸造的代币级别是 0 或 1，操作需要在沙箱中执行后再做决定。

操作后插件规则 1：如果操作是 BATCH_MINT_TOKENS 并且铸造的代币级别是 0 或 1，操作需要经过投票规则 3 的批准。

操作前插件规则 2：如果操作是 BATCH_DISABLE_PLUGIN 并且索引在操作前插件 1、操作前插件 2 或操作后插件 1 之前，拒绝操作。此插件具有最高优先级。

在操作前插件 1 和操作后插件 1 中，当通过铸造新的 0 级别或 1 级别代币改变股东结构时，操作需要经过 90% 的总投票权力的批准。这将防止 1 级别股东稀释 0 级别代币的价值，并保护那些主要是零售投资者的 0 级别代币持有者。

操作前插件 2 是对上述其他插件的保护。当任何操作者试图禁用这三个插件中的任何一个时，操作将被拒绝。这将永久锁定该机制并保证插件不能被禁用。

```
batch_add_voting_rules([
    // 添加操作前插件1
    {
        // 允许0级别和1级别代币进行投票
        votingTokenClassList: [0, 1],

        // 批准门槛：90%
        approvalThresholdPercentage: 90,

        // 投票持续时间：604800秒
        votingDurationInSeconds: 604800,

        // 执行待定持续时间：3600秒
        executionPendingDurationInSeconds: 3600,

        // 投票规则是否启用：是
        isEnabled: true,

        // 关于投票规则的注释
        note: "这是投票规则1（索引0）",

        // 是否是绝对多数
        bIsAbsoluteMajority: true
    },
]);
```

然后向 DARC 添加三个插件。

```
batch_add_and_enable_plugins([
    // 添加操作前插件，索引0
    {
        returnType: SANDBOX_NEEDED, // 需要沙箱
        level: 255, // 等级255
        condition:
            operation_equals(BATCH_MINT_TOKENS) &
            (
                mint_token_class_equals(0) | mint_token_class_equals(1)
            ),
        votingRuleIndex: 0,  // 不需要投票规则索引
        note: "操作前插件1",
        bIsBeforeOperation: true  // 插件将作为操作前插件添加
    },

    // 添加操作后插件，索引0
    {
        returnType: VOTING_NEEDED, // 需要投票
```

```
        level: 258, // 等级258
        condition:
            operation_equals(BATCH_MINT_TOKENS) &
            (
                mint_token_class_equals(0) | mint_token_class_equals(1)
            )
        votingRuleIndex: 0,  // 投票规则1，索引 = 0
        note: "操作后插件1",
        bIsBeforeOperation: false  // 插件将作为操作前插件添加
    },

    // 添加操作前插件1
    {
        returnType: NO, // 拒绝
        level: 257, // 等级257
        condition:
            operation_equals(BATCH_DISABLE_PLUGIN) &
            (
                disable_before_op_plugin_index_equals(0) |
                disable_before_op_plugin_index_equals(1) |
                disable_after_op_plugin_index_equals(0)
            )
        votingRuleIndex: 0,  // 不需要投票规则索引
        note: "操作前插件2",
        bIsBeforeOperation: true  // 插件将作为操作前插件添加
    }
]);
```

1.3 优先股

优先股，也称为非投票股，是一种在公司中不享有投票权但提供财务利益如分红优先权、清算优先权和稳定性的所有权份额。它可以是可转换的或可赎回的，并且因其稳定的收入而受到青睐。然而，它不提供对公司决策的发言权，使其成为寻求资本保值并对公司事务有限控制的收入型投资者的选择。在 DARC 协议中，当一个代币被初始化为具有 1 的分红权重和 0 的投票权重时，它可以被归类为非投票股。

1.4 带印花税的股票

带印花税的股票是在购买或出售时需缴纳政府税收的证券，这种税收称为“印花税”。这种税适用于各种金融工具，包括股票和债券，通常由买方或卖方支付，具体取决于当地规定。人们在交易股票时支付印花税是因为它作为政府收入的来源，也可以作为调节金融市场和抑制过度交易的工具。印花税的具体原因和税率因地区而异。

在 DARC 协议中，我们有能力创建一个机制，强制代币持有者在转让不同级别的代币时支付交易费，称为“印花税”。例如，我们可以将 0 级别的代币指定为 DARC 协议中的标准股票，每次代币转移应用固定的 1000 wei 印花税。

首先，我们必须实现一个插件来禁用所有 `BATCH_TRANSFER_TOKENS` 操作，当被转让的代币属于 0 级别时。接下来，我们应该开发一个插件，仅通过 `BATCH_PAY_TO_MINT_TOKENS` 操作限制转让 0 级别的代币，前提是每个代币的交易费大于或等于 1000 wei。此外，用户需要将 `BATCH_PAY_TO_MINT_TOKENS` 操作的 `dividendable` 标志设置为 0（假），确保交易费（印花税）不会被视为 DARC 的分红收入。

```
batch_add_and_enable_plugins([
    // 添加操作前插件1：禁用0级别代币的转让
    {
        returnType: NO, // 拒绝
        level: 255, // 等级255
        condition:
            operation_equals(BATCH_TRANSFER_TOKENS) &
```

```
            transfer_tokens_level_equals(0)
        votingRuleIndex: 0,  // 不需要投票规则索引
        note: "禁止转让0级别代币",
        bIsBeforeOperation: true  // 插件将作为操作前插件添加
    },

    // 添加操作前插件2:
    // 允许支付转让0级别代币，
    // 每个代币的交易费 >= 1000 wei
    {
        returnType: YES_AND_SKIP_SANDBOX, // 允许并跳过沙箱
        level: 256, // 等级256
        condition:
            operation_equals(BATCH_PAY_TO_TRANSFER_TOKENS) &
            (
                pay_to_transfer_tokens_level_equals(0)  &
                transaction_fee_per_token_GE(1000)   &
                pay_to_trasnfer_tokens_dividendable_flag_equals(0)
            )
        votingRuleIndex: 0,  // 不需要投票规则索引
        note: "允许支付转让0级别代币，交易费 > 1000 且分红标志为0",
        bIsBeforeOperation: true  // 插件将作为操作前插件添加
    }
]);
```

1.5 董事会

在 DAOs 中，通常通过使用治理代币进行投票来达成决策。然而，当每个决策都涉及成千上万的代币持有者时，这个过程可能变得缓慢且低效。此外，缺乏对待执行提案智能合约的规定或预定义规则意味着，批准的投票可以潜在地授权 DAO 内进行广泛的行动。

相比之下，在传统的股份公司中，日常运营和事务是由董事会而不是依靠所有股东的决策来监督的。董事会以其专业知识、战略眼光、快速决策能力以及维护公司治理稳定性和一致性而著称。虽然股东投票对于问责制和代表性至关重要，但董事会制度提供了一个结构化和见多识广的决策方法，确保公司的长期利益得到周到的优先考虑和有效管理。

以下是一个示例，展示了 DARC Y 实验如何使用多级别代币系统设计其董事会。存在四种类型的代币，每种都在组织内扮演着独特的特性和角色，如表 **??** 所示。

0 级别（A 类代币）：这些代币总供应量为 400，拥有 1 的投票权重和 1 的分红权重。它们可能代表 DARC Y 协议中的普通利益相关者。

1 级别（B 类代币）：B 类代币的总供应量为 60，拥有 10 的显著投票权重，使其在决策中具有影响力。它们还具有 1 的分红权重，暗示着可能的收入分配份额。

2 级别（独立董事）：只有一个独立董事代币，设计上没有投票权重和分红权重，表明其在治理结构中的独特角色。

3 级别（董事会）：董事会由五个代币组成，具有 1 的投票权重但没有分红权重，表明其在决策中的主要功能。

级别	名称	总供应量	投票权重	分红权重
0	A 类代币	400	1	1
1	B 类代币	60	10	1
2	独立董事	1	0	0
3	董事会	5	1	0

表 1: DARC Y 中的代币分配结构

以下是设计董事会机制的一些指导原则：

原则 1：董事会应由最多 5 名成员组成，所有董事会决策必须得到超过 2/3 成员的批准。

原则 2：如果一个地址持有超过 2/3 的 0 级别代币，并且不持有任何 1 级别、2 级别或 3 级别代币，它可以代表零售投资者的利益，并可能被选为 A 类董事会成员。如果没有任何地址持有超过 2/3

的这些代币，董事会将没有 A 类代表。

原则 3：董事会应始终包括一名独立董事，独立董事有权选择其继任者。不允许将独立董事从董事会中移除。

原则 4：董事会应由最多 3 名 B 类董事会成员组成，允许任何持有超过 5% B 类代币的地址添加。现有董事会成员可以提名 B 类董事会成员，需得到至少 2/3 所有董事会成员的批准。

原则 5：现有董事会成员有权提议移除 B 类董事会成员。这样的提案可以在以下条件下获批：（1）被移除的地址持有的 B 类代币数量小于或等于总供应量的 5%，或（2）该地址不持有超过 2/3 的 0 级别代币总供应量或独立董事代币，并且提案获得超过 2/3 的董事会投票权力的支持。

原则 6：A 类代币董事、B 类代币董事和独立董事应保持独立于代币持有量。这意味着（1）A 类代币董事只能持有 A 类代币和董事会代币，禁止持有 B 类代币或独立董事代币；（2）B 类代币董事仅允许持有 B 类代币和董事会代币，不得持有 A 类代币或独立董事代币；（3）独立董事只允许持有独立董事代币。

接下来，我们将设计以下插件以实现上述定义的规则和流程：

操作前插件规则 1：如果操作者地址持有超过 2/3 的 0 级别代币，并且不持有任何 1 级别、2 级别或 3 级别代币，它可以为自己铸造 1 个 3 级别代币而无需沙箱检查。此插件规则用于提名 A 类代币董事会成员。

操作前插件规则 2：当操作者地址打算将 2 级别代币转让给另一个目标地址，且目标地址不持有任何 0 级别、1 级别或 3 级别代币时，此操作可以在不需要沙箱检查的情况下获批。此插件便于独立董事职位的转让。

操作前插件规则 3：在情况下，操作者地址持有 1 个 2 级别代币和 0 个 3 级别代币并希望为自己铸造一个 3 级别代币，此操作可以在不需要沙箱检查的情况下获批。此插件用于提名独立董事。

操作前插件规则 4：涉及铸造或销毁 2 级别代币的操作需要被拒绝。此插件限制独立董事的数量。

操作前插件规则 5：如果操作者地址打算为目标地址铸造 1 个 3 级别代币，且目标地址持有超过 5% 的 1 级别代币总供应量，并且操作者地址持有 1 个 3 级别代币，此操作必须进行沙箱检查。此插件用于提名 B 类代币董事会成员。

操作前插件规则 6：如果操作者地址打算从目标地址销毁一个 3 级别代币，且目标地址持有的 0 级别代币总供应量小于或等于 2/3，持有的 1 级别代币小于或等于 5%，且拥有 0 个 2 级别代币，此操作可以在不需要沙箱检查的情况下获批。此插件便于移除不合格的董事会成员。

操作前插件规则 7：如果操作者地址打算从目标地址销毁一个 3 级别代币，且目标地址持有的 0 级别代币总供应量小于或等于 2/3，持有的 1 级别代币超过 5%，且拥有 0 个 2 级别代币，并且操作者地址至少持有 1 个 3 级别代币，此操作必须进行沙箱检查。此插件用于移除 B 类代币董事会成员。

操作后插件规则 1：如果操作者地址打算从目标地址销毁一个 3 级别代币，且目标地址持有的 0 级别代币总供应量小于或等于 2/3，持有的 1 级别代币超过 5%，拥有 0 个 2 级别代币，并且操作者地址至少持有 1 个 3 级别代币，此操作必须通过投票规则 1 进行投票。此次投票涉及所有董事会成员，需要绝对多数模式，批准率超过 66%。此插件用于 B 类董事会成员选举。

接下来是法规脚本中实现的插件。

```
const plugin_before_op_1 = {
    returnType: YES_AND_SKIP_SANDBOX,
    level: 255,
    condition: operation_equals(BATCH_MINT_TOKENS)
               & number_of_token_mint_equals(1)
               & level_of_token_mint_equals(3)
               & operator_owns_num_of_token_GE(0,267)
               & operator_owns_num_of_token_equals(1, 0)
               & operator_owns_num_of_token(2, 0)
               & operator_mint_to_itself() ,
    votingRuleIndex: 0,
    note: "操作前插件1",
    bIsBeforeOperation: true
},

const plugin_before_op_2 = {
    returnType: YES_AND_SKIP_SANDBOX,
    level: 255,
    condition: operation_equals(BATCH_TRANSFER_TOKENS)
               & transfer_token_level_equals(2) ,
```

```
    votingRuleIndex: 0,
    note: "操作前插件2",
    bIsBeforeOperation: true
},

const plugin_before_op_3 = {
    returnType: YES_AND_SKIP_SANDBOX,
    level: 255,
    condition: operation_equals(BATCH_MINT_TOKENS)
               & operator_owns_num_of_token_equals(2, 1)
               & number_of_token_mint_equals(1)
               & level_of_token_mint_equals(3) ,
    votingRuleIndex: 0,
    note: "操作前插件3",
    bIsBeforeOperation: true
},

const plugin_before_op_4 = {
    returnType: NO,
    level: 257,
    condition: (operation_equals(BATCH_BURN_TOKENS)
               & level_of_token_burned_equals(2)) |
               (operation_equals(BATCH_MINT_TOKENS)
               & level_of_token_mint_equals(2)) ,

    votingRuleIndex: 0,
    note: "操作前插件4",
    bIsBeforeOperation: true
},

const plugin_before_op_5 = {
    returnType: SANDBOX_NEEDED,
    level: 256,
    condition: operation_equals(BATCH_MINT_TOKENS)
               & level_of_token_mint_equals(3)
               & number_of_token_mint_equals(1)
               & operator_owns_num_of_tokens_equals(3,1)
               & batch_mint_tokens_target_address_ownes_num_of_tokens_greater(1,12),
    votingRuleIndex: 0,
    note: "操作前插件5",
    bIsBeforeOperation: true
},

const plugin_before_op_6 = {
    returnType: YES_AND_SKIP_SANDBOX,
    level: 255,
    condition: operation_equals(BATCH_BURN_TOKENS)
               & level_of_token_burned_equals(3)
               & batch_burn_token_target_address_owns_num_of_tokens_LE(1, 12)
               & batch_burn_token_target_address_owns_num_of_tokens_LE(0, 267)
               & batch_burn_token_target_address_owns_num_of_tokens_equals(2,0),
    votingRuleIndex: 0,
    note: "操作前插件6",
    bIsBeforeOperation: true
},
```

```
const plugin_before_op_7 = {
    returnType: SANDBOX_NEEDED,
    level: 256,
    condition: operation_equals(BATCH_BURN_TOKENS)
               & level_of_token_burned_equals(3)
               & batch_burn_token_target_address_owns_num_of_tokens_GE(1, 12)
               & batch_burn_token_target_address_owns_num_of_tokens_LE(0, 256)
               & batch_burn_token_target_address_owns_num_of_tokens_equals(2,0),
    votingRuleIndex: 0,
    note: "操作前插件7",
    bIsBeforeOperation: true
},

const plugin_after_op_1 = {
    returnType: VOTING_NEEDED,
    level: 253,
    condition: operation_equals(BATCH_BURN_TOKENS)
               & level_of_token_burned_equals(3)
               & batch_burn_token_target_address_owns_num_of_tokens_GE(1,12)
               & batch_burn_token_target_address_owns_num_of_tokens_LE(0, 267)
               & batch_burn_token_target_address_owns_num_of_tokens_equals(2,0)
               & operator_ownes_num_of_tokens_equals(3,1),
    votingRuleIndex: 1,
    note: "操作后插件1",
    bIsBeforeOperation: false
},

batch_add_and_enable_plugins([
    plugin_before_op_1,
    plugin_before_op_2,
    plugin_before_op_3,
    plugin_before_op_4,
    plugin_before_op_5,
    plugin_before_op_6,
    plugin_before_op_7,
    plugin_after_op_1
]);
```

1.6 公司债券

公司债券是企业为筹集资金而发行的债务证券。投资者向公司借出资金，以换取定期利息支付和到期时本金的返还。它们为公司提供了筹集资本的手段，并为投资者提供了可预测的收入来源。

在 DARC 协议的背景下，BATCH_PAY_TO_MINT_TOKENS 和 BATCH_BURN_TOKENS_AND_REFUND 命令便于在特定时间框架内以指定价格购买和赎回债券代币，使任何地址都可进行购买。购买和退款这些债券代币的两条规则如下：

规则 1：在 2020 年 1 月 1 日至 2020 年 2 月 1 日期间，任何地址都可以使用 BATCH_PAY_TO_MINT_TOKENS 以每个代币 10000 wei 的价格购买不超过 100 个债券代币 (2 级别)。债券代币的总供应量不得超过 10000。

规则 2：地址有权使用 BATCH_BURN_TOKENS_AND_REFUND 以每个代币 15000 wei 的价格在 2030 年 1 月 1 日后赎回债券代币。

```
batch_add_and_enable_plugins([
    {
        returnType: YES_AND_SKIP_SANDBOX,
        level: 253,
        condition: operation_equals(BATCH_PAY_TO_MINT_TOKENS)
                   & pay_to_mint_tokens_price_per_token_equals(10000)
```

```
                & timestamp_greater(1577858400) // 2020-01-01-0-0-0
                & timestamp_less(1580536800) // 2020-02-01-0-0-0
                & pay_to_mint_tokens_level_equals(2) // 代币级别
                & number_of_token_pay_to_mint_LE(100) // 铸造数量
                & total_number_to_tokens_LE(2,99900), // 总代币供应量
        votingRuleIndex: 0,
        note: "",
        bIsBeforeOperation: true
    },
    {
        returnType: YES_AND_SKIP_SANDBOX,
        level: 253,
        condition: operation_equals(BATCH_BURN_TOKENS_AND_REFUND)
                & burn_tokens_and_refund_price_per_token_equals(15000)
                & timestamp_GE(1893477600) // 2030-01-01-0-0-0
                & burn_tokens_and_refund_level_equals(2), // 代币级别
        votingRuleIndex: 0,
        note: "",
        bIsBeforeOperation: true
    },
]);
```

1.7 产品代币和非同质化代币

当我们将投票权重和分红权重都设置为 0，并强制要求操作者以特定价格铸造代币时，这些代币可以被视为代表 DARC 实例的付费产品或服务的产品代币。

当我们进一步配置几个级别的代币以允许每笔交易仅支付铸造一个代币，禁止铸造和销毁操作，并禁止在某一级别已存在一个代币的情况下进行额外的支付铸造交易。同时，我们允许这些代币的持有者自由将它们转让给其他地址，此时，这些特定级别的代币可以被视为 NFTs（非同质化代币）[?]。NFTs 可以用作并被视为代表特定物品或内容的所有权或真实性证明的独特数字资产，在 DARC 协议中使用。