附录 1：指令操作码的参考设计

```
  /*
  * @notice opcode枚举用于表示DARC协议的指令。
*/

enum EnumOpcode {

  /**
   * @notice 无效操作
   * ID: 0
   */
  UNDEFINED,

  /**
   * @notice 批量铸造代币操作
   * @param ADDRESS_2DARRAY[0] address[] toAddressArray: 需要铸造新代币的地址数组
   * @param UINT256_2DARRAY[0] uint256[] tokenClassArray: 需要铸造新代币的代币类别索引数组
   * @param UINT256_2DARRAY[1] uint256[] amountArray: 需要铸造的代币数量数组
   *
   * ID: 1
   */
  BATCH_MINT_TOKENS,

  /**
   * @notice 批量创建代币类别操作
   * @param STRING_ARRAY[] string[] nameArray: 需要创建的代币类别名称数组
   * @param UINT256_2DARRAY[0] uint256[] tokenIndexArray: 需要创建的代币类别的代币索引数组
   * @param UINT256_2DARRAY[1] uint256[] votingWeightArray: 需要创建的代币类别的投票权重数组
   * @param UINT256_2DARRAY[2] uint256[] dividendWeightArray: 需要创建的代币类别的分红权重数组
   *
   * ID:2
   */
  BATCH_CREATE_TOKEN_CLASSES,

  /**
   * @notice 批量转移代币操作
   * @param ADDRESS_2DARRAY[0] address[] toAddressArray: 需要转移代币到的地址数组
   * @param UINT256_2DARRAY[0] uint256[] tokenClassArray: 需要转移代币的代币类别索引数组
   * @param UINT256_2DARRAY[1] uint256[] amountArray: 需要转移的代币数量数组
   *
   * ID:3
   */
  BATCH_TRANSFER_TOKENS,

  /**
   * @notice 批量从地址A转移到地址B的代币操作
   * @param ADDRESS_2DARRAY[0] address[] fromAddressArray: 需要从中转移代币的地址数组
   * @param ADDRESS_2DARRAY[1] address[] toAddressArray: 需要转移代币到的地址数组
   * @param UINT256_2DARRAY[0] uint256[] tokenClassArray: 需要转移代币的代币类别索引数组
   * @param UINT256_2DARRAY[1] uint256[] amountArray: 需要转移的代币数量数组
   *
   * ID:4
   */
  BATCH_TRANSFER_TOKENS_FROM_TO,
```

```
/**
 * @notice 批量销毁代币操作
 * @param UINT256_2DARRAY[0] uint256[] tokenClassArray: 需要销毁代币的代币类别索引数组
 * @param UINT256_2DARRAY[1] uint256[] amountArray: 需要销毁的代币数量数组
 *
 * ID:5
 */
BATCH_BURN_TOKENS,

/**
 * @notice 批量从地址A销毁代币操作
 * @param ADDRESS_2DARRAY[0] address[] fromAddressArray: 需要从中销毁代币的地址数组
 * @param UINT256_2DARRAY[0] uint256[] tokenClassArray: 需要销毁代币的代币类别索引数组
 * @param UINT256_2DARRAY[1] uint256[] amountArray: 需要销毁的代币数量数组
 *
 * ID:6
 */
BATCH_BURN_TOKENS_FROM,

/**
 * @notice 批量添加成员操作
 * @param ADDRESS_2DARRAY[0] address[] memberAddressArray: 需要作为成员添加的地址数组
 * @param UINT256_2DARRAY[0] uint256[] memberRoleArray: 需要添加的成员角色数组
 * @param STRING_ARRAY string[] memberNameArray: 需要添加的成员名称数组
 *
 * ID:7
 */
BATCH_ADD_MEMBERSHIP,

/**
 * @notice 批量暂停成员操作
 * @param ADDRESS_2DARRAY[0] address[] memberAddressArray: 需要暂停的成员地址数组
 *
 * ID:8
 */
BATCH_SUSPEND_MEMBERSHIP,

/**
 * @notice 批量恢复成员操作
 * @param ADDRESS_2DARRAY[0] address[] memberAddressArray: 需要恢复的成员地址数组
 *
 * ID:9
 */
BATCH_RESUME_MEMBERSHIP,

/**
 * @notice 批量更改成员角色操作
 * @param ADDRESS_2DARRAY[0] address[] memberAddressArray: 需要更改角色的成员地址数组
 * @param UINT256_2DARRAY[0] uint256[] memberRoleArray: 需要更改的成员角色数组
 *
 * ID:10
 */
BATCH_CHANGE_MEMBER_ROLES,

/**
 * @notice 批量更改成员名称操作
```

```
 * @param ADDRESS_2DARRAY[0] address[] memberAddressArray: 需要更改名称的成员地址数组
 * @param STRING_ARRAY string[] memberNameArray: 需要更改的成员名称数组
 *
 * ID:11
 */
BATCH_CHANGE_MEMBER_NAMES,

/**
 * @notice 批量添加紧急代理操作
 * @param Plugin[] pluginList: 插件数组
 * ID:12
 */
BATCH_ADD_PLUGINS,

/**
 * @notice 批量启用插件操作
 * @param UINT256_ARRAY[0] uint256[] pluginIndexArray: 需要启用的插件索引数组
 * @param BOOL_ARRAY bool[] isBeforeOperationArray: 标志数组，指示插件是否为操作前插件
 * ID:13
 */
BATCH_ENABLE_PLUGINS,

/**
 * @notice 批量禁用插件操作
 * @param UINT256_ARRAY[0] uint256[] pluginIndexArray: 需要禁用的插件索引数组
 * @param BOOL_ARRAY bool[] isBeforeOperationArray: 标志数组，指示插件是否为操作前插件
 * ID:14
 */
BATCH_DISABLE_PLUGINS,

/**
 * @notice 批量添加并启用插件操作
 * @param Plugin[] pluginList: 插件数组
 * ID:15
 */
BATCH_ADD_AND_ENABLE_PLUGINS,

/**
 * @notice 批量设置参数操作
 * @param MachineParameter[] parameterNameArray: 参数名称数组
 * @param UINT256_2DARRAY[0] uint256[] parameterValueArray: 参数值数组
 * ID:16
 */
BATCH_SET_PARAMETERS,

/**
 * @notice 批量添加可提现余额操作
 * @param address[] addressArray: 需要添加可提现余额的地址数组
 * @param uint256[] amountArray: 需要添加的可提现余额数量数组
 * ID:17
 */
BATCH_ADD_WITHDRAWABLE_BALANCES,

/**
 * @notice 批量减少可提现余额操作
 * @param address[] addressArray: 需要减少可提现余额的地址数组
```

```
 * @param uint256[] amountArray: 需要减少的可提现余额数量数组
 * ID:18
 */
BATCH_REDUCE_WITHDRAWABLE_BALANCES,

/**
 * @notice 批量添加投票规则
 * @param VotingRule[] votingRuleList: 投票规则数组
 * ID:19
 */
BATCH_ADD_VOTING_RULES,


/**
 * @notice 批量支付以铸造代币操作
 * @param ADDRESS_2DARRAY[0] address[] addressArray: 需要铸造代币的地址数组
 * @param UINT256_2DARRAY[0] uint256[] tokenClassArray: 需要铸造代币的代币类别索引数组
 * @param UINT256_2DARRAY[1] uint256[] amountArray: 需要铸造的代币数量数组
 * @param UINT256_2DARRAY[2] uint256[] priceArray: 每个代币类别铸造的价格
 * @param UINT256_2DARRAY[3] uint256[1] dividendableFlag: 标志，指示支付是否可分红。1为是（支付购买
 * ID:20
 */
BATCH_PAY_TO_MINT_TOKENS,

/**
 * @notice 支付一些现金以转移代币（可用作商品币）
 * @param ADDRESS_2DARRAY[0] address[] toAddressArray: 需要转移代币到的地址数组
 * @param UINT256_2DARRAY[0] uint256[] tokenClassArray: 需要转移代币的代币类别索引数组
 * @param UINT256_2DARRAY[1] uint256[] amountArray: 需要转移的代币数量数组
 * @param UINT256_2DARRAY[2] uint256[] priceArray: 每个代币类别转移的价格
 * @ param UINT256_2DARRAY[3] uint256[1] dividendableFlag: 标志，指示支付是否可分红。1为是（支付购买
 * ID:21
 */
BATCH_PAY_TO_TRANSFER_TOKENS,

/**
 * @notice 添加一组地址作为紧急代理
 *  （可用作具有新唯一代币类别的产品NFT）
 * @param ADDRESS_2DARRAY[0] address[] 需要添加为紧急代理的地址数组
 * ID:22
 */
ADD_EMERGENCY,

/**
 * @notice 从合约的现金余额中提现现金
 * @param address[] addressArray: 需要提现现金到的地址数组
 * @param uint256[] amountArray: 需要提现的现金数量数组
 * ID:23
 */
WITHDRAW_CASH_TO,

/**
 * @notice 调用紧急代理处理紧急情况
 * @param UINT256_2DARRAY[0] address[] addressArray: 需要调用的紧急代理索引数组
 * ID:24
 */
```

```
CALL_EMERGENCY,


/**
 * @notice 使用给定的abi调用合约
 * @param address contractAddress: 需要调用的合约地址
 * @param bytes abi: 需要调用的函数的abi
 * ID:25
 */
CALL_CONTRACT_ABI,

/**
 * @notice 支付一些现金
 * @param uint256 amount: 需要支付的现金数量
 * @param uint256 paymentType: 需要支付的现金类型，0为以太币/matic/原生代币
 *  1为USDT，2为USDC（目前仅支持0），3为DAI等
 * @param uint256 dividendable: 标志，指示支付是否可分红，
 * 0为否（支付投资），1为是（支付购买）
 * ID:26
 */
PAY_CASH,

/**
 * @notice 计算分红并提供给代币持有者
 *  通过将分红添加到每个代币持有者的可提现余额中
 *
 * ID:27
 */
OFFER_DIVIDENDS,

/**
 * @notice 从可提现分红余额中提取分红
 * @param address[] addressArray: 需要提取分红到的地址数组
 * @param uint256[] amountArray: 需要提取的分红数量数组
 * ID:28
 */
WITHDRAW_DIVIDENDS_TO,

/**
 * @notice 设置所有转移操作的批准通过地址
 * @paran address: 需要设置所有转移操作批准的地址
 * ID:29
 */
SET_APPROVAL_FOR_ALL_OPERATIONS,


/**
 * @notice 批量销毁代币并退款
 * @param UINT256_2D[0] uint256[] tokenClassArray: 需要从中销毁代币的代币类别索引数组
 * @param UINT256_2D[1] uint256[] amountArray: 需要销毁的代币数量数组
 * @param UINT256_2D[2] uint256[] priceArray: 每个代币类别销毁的价格
 * ID:30
 */
BATCH_BURN_TOKENS_AND_REFUND,

/**
```

```
    * @notice 永久地将存储IPFS哈希添加到存储列表中
    * @paran STRING_2DARRAY[0] address: 需要设置所有现金提取操作批准的地址
    * ID:31
    */
  ADD_STORAGE_IPFS_HASH,


  /**
    * 以下两个操作可以在投票等待过程中使用
    */

  /**
    * @notice 为投票等待的程序投票
    * @param bool[] voteArray: 每个程序的投票数组
    * ID:32
    */
  VOTE,

  /**
    * @notice 执行已经投票并获批的程序
    * ID:33
    */
  EXECUTE_PROGRAM,

  /**
    * @notice 紧急模式终止。紧急代理在此操作后不能做任何事情
    * ID:34
    */
  END_EMERGENCY,

  /**
    * @notice 将合约升级到新的合约地址
    * @param ADDRESS_2DARRAY[0][0] 新合约的地址
    * ID:35
    */
  UPGRADE_TO_ADDRESS,

  /**
    * @notice 接受当前DARCs从旧合约地址升级
    * @param ADDRESS_2DARRAY[0][0] 旧合约的地址
    * ID:36
    */
  CONFIRM_UPGRAED_FROM_ADDRESS,

  /**
    * @notice 将合约升级到最新版本
    * ID:37
    */
  UPGRADE_TO_THE_LATEST,

  /**
    * @notice 批量支付转移代币操作
    * ID:38
    */
  op_BATCH_PAY_TO_TRANSFER_TOKENS
}
```

附录 2：程序和操作的参考设计

```
/**
  * 操作的参数或操作数
  */
struct Param {
  uint256[] UINT256_ARRAY;
  address[] ADDRESS_ARRAY;
  string[] STRING_ARRAY;
  bool[] BOOL_ARRAY;
  VotingRule[] VOTING_RULE_ARRAY;
  Plugin[] PLUGIN_ARRAY;
  MachineParameter[] PARAMETER_ARRAY;
  uint256[][] UINT256_2DARRAY;
  address[][] ADDRESS_2DARRAY;
}

/**
  * 要执行的操作，包括操作者地址、操作码和参数
  */
struct Operation {
  address operatorAddress;
  EnumOpcode opcode;
  Param param;
}

/**
  * 要执行的程序，包括程序操作者地址和操作数组
  */
struct Program {
  address programOperatorAddress;

  /**
   * @notice operations: 要执行的操作数组
   */
  Operation[] operations;
}
```

附录 3：插件的参考设计

```
/**
  * 条件节点类型
  */
enum EnumConditionNodeType { UNDEFINED, EXPRESSION, LOGICAL_OPERATOR, BOOLEAN_TRUE, BOOLEAN_FALSE}

/**
  * 逻辑操作符类型
  */
enum EnumLogicalOperatorType {UNDEFINED, AND, OR, NOT }
```

```
enum EnumReturnType {

  /**
   * 默认值。如果没有插件被触发，插件系统将返回UNDEFINED。
   * BEFORE和AFTER操作插件系统都可能返回UNDEFINED。
   */
  UNDEFINED,


  /**
   * 操作被批准，但必须在沙盒中执行以检查操作
   * 在当前机器状态下是否有效。
   * 只有BEFORE操作插件系统可能返回SANDBOX_NEEDED。
   */
  SANDBOX_NEEDED,

  /**
   * 操作被拒绝，在此级别应该被拒绝。
   * BEFORE和AFTER操作插件系统都可能返回NO。
   */
  NO,

  /**
   * 决定待定，应在此级别创建投票项。
   * 只有AFTER操作插件系统可能返回VOTING_NEEDED。
   */
  VOTING_NEEDED,

  /**
   * 操作被批准，应跳过沙盒检查。
   * 只有BEFORE操作插件系统可能返回YES_AND_SKIP_SANDBOX。
   */
  YES_AND_SKIP_SANDBOX,

  /**
   * 操作最终在此级别被批准。
   * 只有AFTER操作插件系统可能返回YES。
   */
  YES
}

/**
  * 条件节点表达式参数
  */
struct NodeParam {
  uint256[] UINT256_ARRAY;
  address[] ADDRESS_ARRAY;
  string[] STRING_ARRAY;
  uint256[][] UINT256_2DARRAY;
  address[][] ADDRESS_2DARRAY;
  string[][] STRING_2DARRAY;
}

/**
  * 条件节点结构
  */
```

```
struct ConditionNode {
  /**
   * 当前条件节点索引
   */
  uint256 id;

  /**
   * 当前条件节点的类型
   */
  EnumConditionNodeType nodeType;

  /**
   * 当前条件节点的逻辑操作符
   */
  EnumLogicalOperatorType logicalOperator;

  /**
   * 当前条件节点的条件表达式
   */
  EnumConditionExpression conditionExpression;

  /**
   * 当前条件节点的子节点列表
   */
  uint256[] childList;

  /**
   * EXPRESSION节点参数数组
   */
  NodeParam param;
}

/**
  * 插件结构
  */
struct Plugin {
  /**
   * 当前条件节点的返回类型
   */
  EnumReturnType returnType;

  /**
   * 限制级别，从0到uint256的最大值
   */
  uint256 level;

  /**
   * 条件二进制表达式树向量
   */
  ConditionNode[] conditionNodes;

  /**
   * 如果返回类型为VOTING_NEEDED，当前插件的投票规则id
   */
  uint256 votingRuleIndex;
```

```
  /**
   * 插件备注
   */
  string note;

  /**
   * 指示插件是否启用的布尔值
   */
  bool bIsEnabled;

  /**
   * 指示插件是否已删除的布尔值
   */
  bool bIsInitialized;

  /**
   * 指示插件是操作前插件还是操作后插件的布尔值
   */
  bool bIsBeforeOperation;

}
```