

BSN-DDC EOS 合约

详细设计

文档信息

项目名称：BSN-DDC	项目编号：
项目负责人： 谢兆颜	所属部门：
编 制 人：李铭坤	编制时间：2021.10.16
审 核 人：谢兆颜	审核时间：2021.10.16
批 准 人：	批准时间：
版 本 号：	流 水 号：

修改记录

日期	版本	修改说明	修改者
2021.12.07	v1.2	版本初始化	谢兆颜
2021.12.07	v1.3	按照 eos 的模式重新更新合约设计和整体设计说明	谢兆颜
2021.12.08	v1.4	更新接口命名规范，使其符合 eos 标准	鲍宏飞
2021.12.10	v1.5	重新设计表结构;根据表结构修改 API 接口	李铭坤
2021.12.16	v1.5.3	增加业务模块授权表结构；增加业务模块的账户授权、ddc 授权；权限模块增加 添加运营账户 方法；更新整体设计说明	李铭坤
2021.12.16	v1.5.4	在表结构，接口中删除 ddc_name;erc-721 接口重命名;重构业务模块的表结构	李铭坤
2021.12.17	v1.5.5	讨论后的表结构优化，计费 and 权限合约区分 erc-721 和 erc-1155；增加 business_type 字段做业务逻辑区分	李铭坤
2022.01.07	v1.5.6	删除查询方法，部分接口逻辑描述按实现修改，增加 erc-721 和 erc-1155 的主键记录	李铭坤
2022.01.14	v1.6.0	1.将数据表按照智能合约的实现进行了名称修改；整体二级索引的顺序和实现规则，使其和实现逻辑一致 2.修改部分接口的描述，使其和实现一致	李铭坤

		3.mint 和 mintbatch 中增加 memo 字段; burn 和 burnbatch 增加 owner 字段, 并修改实现逻辑	
2022.01.15	v1.6.1	1.删除添加账户(addaccount) 和运营方添加终端用户方法(addconsumer),添加运营方添加账户(operatoradd)和平台方添加账户(manageradd) 方法, 注意: manageradd 暂时不开放调用 2.计费模块对于 asset 类型添加了示例	
2022.01.20	v1.6.2	1.业务模块增加 uri 设置方法(seturi) 2.权限模块增加跨平台授权方法(crossappr) 3.权限模块增加跨平台授权表(permapprs)	
2022.01.24	v1.6.3	1. 添加 名称符号设置 方法	
2022.03.17	v1.6.4	权限模块: 1.增加批量充值(rechargebat) 方法 721 主模块: 1.增加批量生成(mintbatch) 方法 2.增加批量转移(batchtrans) 方法 3.增加批量授权(approvebatch) 方法 4.增加批量销毁(burnbatch) 方法 权限模块: 1.增加权限全局(permglobal) 表 2.增加平台方批量添加账户(managaddbat) 方法 3.增加运营方批量添加账户(operaddbat) 方法 4.增加平台方添加账户开关(setswitcher) 方法	
2022.06.10	v1.6.4.1	对设置收费规则和删除授权的说明做了调整	
2022.08.03	v1.6.5	1.在充值(rechage)接口增加memo 字段 2.在批量充值(rechargebat)接口增加memo 字段	
2022.08.04	v1.6.5.1	1.在充值(rechage)、批量充值(rechargebat)接口 删除 memo 字段 2. 增加留言的充值(recharge2)、批量充值(rechargebat2)接口	
2022.11.17	v1.7.0	增加元交易模块	李佳音

目录

文档信息.....2

修改记录.....2

目录.....4

1 编写目的.....5

2 需求文档.....5

3 整体设计.....6

 3.1 合约整体结构.....6

 3.2 DDC 业务交易时序图.....6

 3.3 计费充值交易时序图.....6

 3.4 账户管理交易时序图.....7

 3.5 整体设计说明.....7

4 合约设计.....8

 4.1 BSN-DDC-计费模块.....8

 4.1.1 功能介绍.....8

 4.1.2 数据结构.....8

 4.1.3 API 定义.....10

 4.2 BSN-DDC-721 业务主模块.....15

 4.2.1 功能介绍.....15

 4.2.2 数据结构.....15

 4.2.3 API 定义.....19

 4.3 BSN-DDC-1155 业务主模块.....28

 4.3.1 功能介绍.....28

 4.3.2 数据结构.....28

 4.3.3 API 定义.....30

 4.4 BSN-DDC-权限模块.....38

 4.4.1 功能介绍.....38

 4.4.2 数据结构.....38

 4.4.3 API 定义.....41

 4.5 BSN-DDC-元交易模块.....48

 4.5.1 功能介绍.....48

 4.5.2 数据结构.....48

 4.5.3 API 定义.....49

删除[1]: 文档信息 2

修改记录 2

目录 4

1 编写目的 5

2 需求文档 5

3 整体设计 5

 3.1 合约整体结构 5

 3.2 DDC 业务交易时序图 5

 3.3 计费充值交易时序图 6

 3.4 账户管理交易时序图 7

 3.5 整体设计说明 7

4 合约设计 7

 4.1 BSN-DDC-计费模块 7

 4.1.1 功能介绍 8

 4.1.2 数据结构 8

 4.1.3 API 定义 9

 4.2 BSN-DDC-721 业务主模块 14

 4.2.1 功能介绍 14

 4.2.2 数据结构 14

 4.2.3 API 定义 17

 4.3 BSN-DDC-1155 业务主模块 27

 4.3.1 功能介绍 27

 4.3.2 数据结构 27

 4.3.3 API 定义 29

 4.4 BSN-DDC-权限模块 36

 4.4.1 功能介绍 36

 4.4.2 数据结构 37

 4.4.3 API 定义 40

1 编写目的

为了让项目组成员以及各开放链盟链框架方对 BSN-DDC 合约的整体设计有一个全面详细的了解，同时为项目的开发、测试、验证、交付等环节提供原始依据以及开发指导，特此整理 BSN-DDC 合约整体设计规范方案说明文档。

2 需求文档

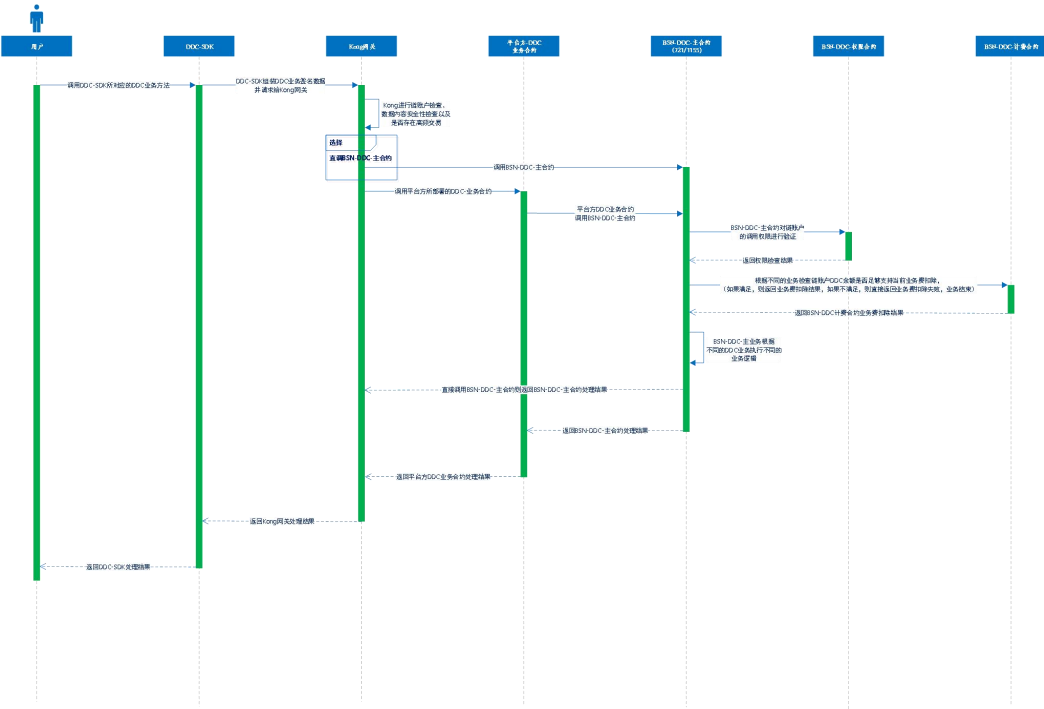
需求文档引用 BSN-DDC_需求说明书 V1.1.docx

3 整体设计

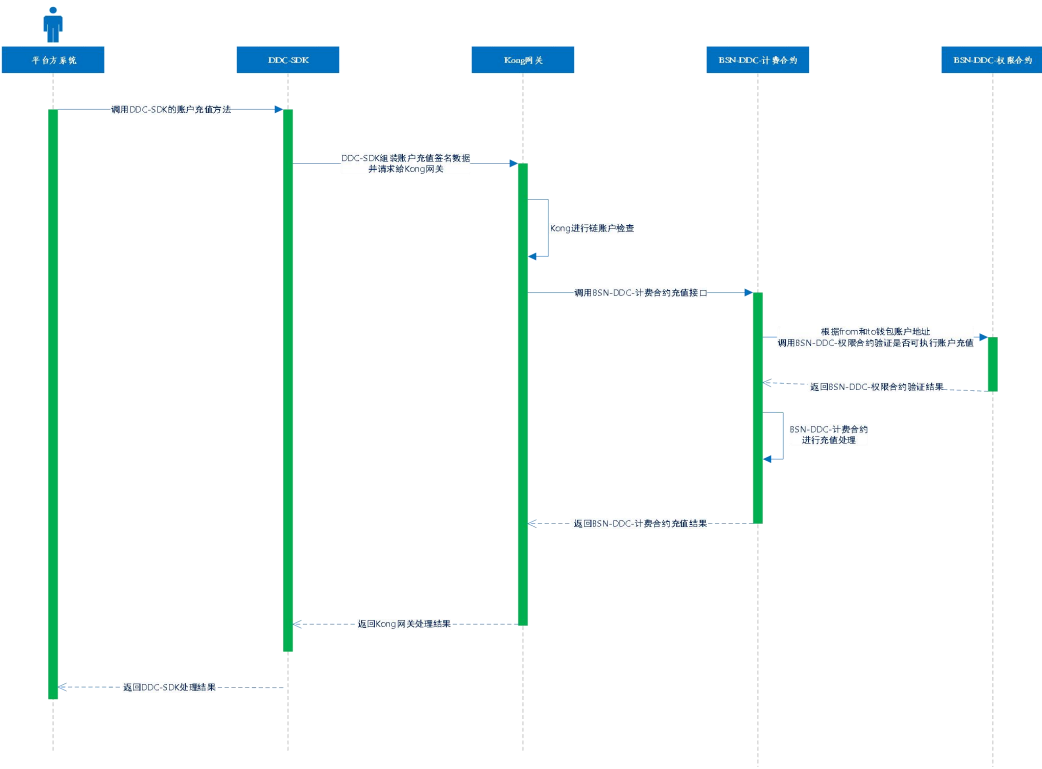
3.1 合约整体结构



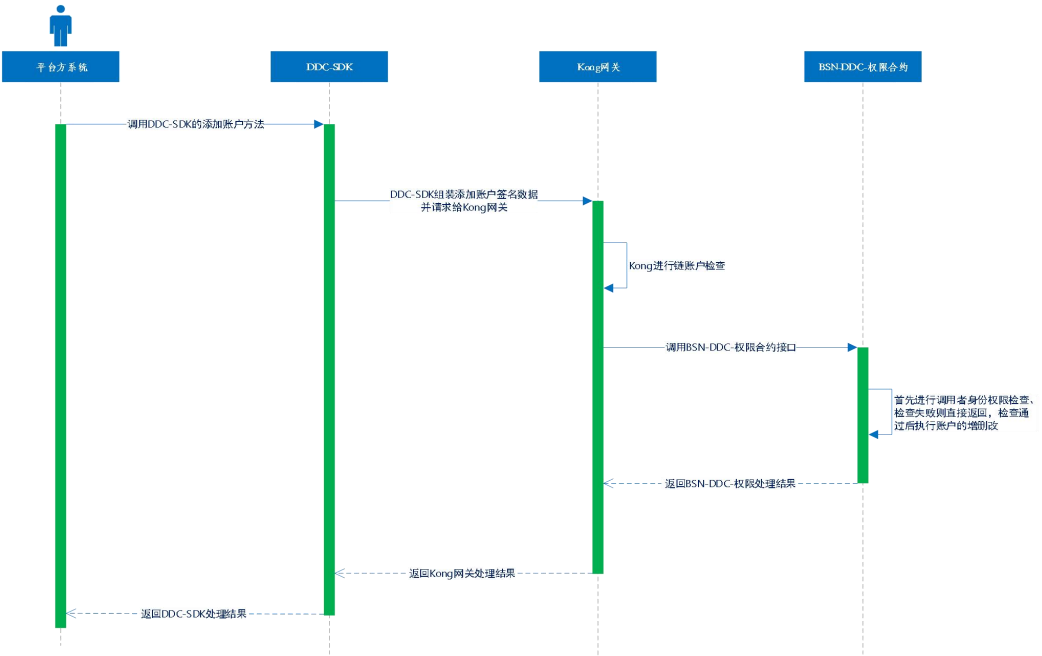
3.2 DDC 业务交易时序图



3.3 计费充值交易时序图



3.4账户管理交易时序图



3.5整体设计说明

BSN-DDC EOS 合约相对于 ETH 合约有较大的灵活性，不需要分层设计以确

保整体的灵活性和安全性。

BSN-DDC EOS 合约可以设计成一个通用合约,其中存在权限,计费,业务模块。
每个平台方都可以调用该合约创建自己的 BSN-DDC。

EOS 合约允许重复升级，但需要尽可能在初始阶段对数据表做出完备设计，
如果需要对数据层结构做修改，需要构建新表，并进行迁移操作。

4 合约设计

4.1BSN-DDC-计费模块

业务费：用于定义 BSN-DDC 业务费所对应的费用计量单位名称。

计费模块用于对参与 DDC 业务中的各方的链上账户进行统一管理，其中
包括计费规则的定义以及各种类型账户按照计费规则调用 DDC 合约所扣除的
DDC 业务费。

4.1.1 功能介绍

计费合约用于对参与 DDC 业务中的各方的链上账户进行统一管理，其中
包括计费规则的定义以及各种类型账户按照计费规则调用 DDC 合约所扣除的
DDC 业务费。

- 各参与方的链上账户类型包含以下：
- 1. 运营方：运营方在计费合约中所对应的账户。
 - 2. 平台方：平台方在计费合约中所对应的账户。
 - 3. 普通用户：普通用户在计费合约中所对应的账户，每个普通用户只能
所属于一个平台方。

4.1.2 数据结构

➤ DDC 计费全局表（feeglobal）

数据表信息：

编号	字段名	字段	类型	备注
1.	主键 id	primary	uint64_t	唯一性主键
2.	费用	total_fee	asset	总体的业务费 (参考格式： 1.0000 FEE)
3.	费用	total_cost	asset	消耗的业务费 (参考格式： 1.0000 FEE)

➤ DDC 计费规则表 (feerules)

数据表信息：

编号	字段名	字段	类型	备注
1.	业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
2.	计费规则集合	func_fee	map<name, asset>	调用 BSN-DDC-业务主合约方法名和对应的业务费
3.	DDC 状态	used	bool	DDC 业务主合约的删除状态

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	business_type	业务类型形成的索引

➤ 用户账户(feeaccounts)

数据表信息：

编号	字段名	字段	类型	备注
1.	账户名	account	name	用户账户名
2.	账户余额	balance	asset	用户账户余额（参考格式： 1.0000 FEE）
3.	账户总额	supply	asset	用户充值总额（参考格式： 1.0000 FEE）

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	account.value	区块链账户名形成的唯一性索引

4.1.3 API 定义

4.1.3.1 充值

运营方、平台方调用该 API 将账户的 DDC 业务费充值给下级用户。

- 输入参数：发送账户名，接收方账户名、充值额度(参考格式: 1.000 FEE);
- 方法命名：recharge;
- 方法举例：recharge(name from, name to, asset value);
- 核心逻辑
 - 检查转出者账户与接收者账户是否相同，相同则终止调用；
 - 检查转出者账户状态是否活跃，不活跃则终止调用；
 - 检查接收者账户状态是否活跃，不活跃则终止调用；
 - 检查转出者账户所对应的账户类型是否为运营方账户(运营方给平台方或终端账户进行充值)或检查转出者账户对应的账户 DID 与接收者账户的上级 DID 是否相同(运营方给平台方进行充值或平台方给终端账户进行充值)或检查转出者账户与接收者账户所对应的账户 DID 相同且接收者账户所对应的账户类型不是终端客户(平台方自己的链账户之间的相互充值), 如果都不满足则终止调用；
 - 判断账户余额或者授权额度是否满足充值，不满足则终止调用；
 - 否则调用数据合约保存充值结果，并触发事件；
- 业务规则

- 运营方可以充值给平台方以及终端用户；
 - 平台方可以充值所属自己平台的用户，不可以充值给其他平台方以及其他平台方所属的用户；
 - 用户不能调用该方法充值给任何账户；
- 业务场景
- 平台方在运营方处充值，运营方调用充值 API 给平台方充值；
 - 用户在某平台方充值，平台方调用充值 API 给该用户充值；

4.1.3.2 批量充值

- 运营方、平台方调用该 API 将账户的 DDC 业务费批量充值给下级用户。
- 输入参数：发送账户名，接收方账户名列表、充值额度列表；
- 方法命名：rechargebat；
- 方法举例：rechargebat(name from, std::vector<name> to_list, std::vector<asset> value_list);
- 核心逻辑
- 检查接收者账户列表长度与业务费集合长度是否相等，不是则返回提示信息
 - 检查转出者账户状态是否活跃，不活跃则终止调用；
 - 批量循环挨个处理链账户业务费充值，处理逻辑如下：
 1. 检查转出者账户与接收者账户是否相同，相同则终止调用；
 2. 检查接收者账户状态是否活跃，不活跃则终止调用；
 3. 检查转出者账户所对应的账户类型是否为运营方账户(运营方给平台方或终端账户进行充值)或检查转出者账户对应的账户 DID 与接收者账户的上级 DID 是否相同(运营方给平台方进行充值或平台方给终端账户进行充值)或检查转出者账户与接收者账户所对应的账户 DID 相同且接收者账户所对应的账户类型不是终端客户(平台方自己的链账户之间的相互充值)，如果都不满足则终止调用；

- 4. 判断账户余额或者授权额度是否满足充值，不满足则终止调用；
 - 5. 否则调用数据合约保存充值结果，并触发事件；
- 业务规则
 - 运营方可以充值给平台方以及终端用户；
 - 平台方可以充值所属自己平台的用户，不可以充值给其他平台方以及其他平台方所属的用户；
 - 用户不能调用该方法充值给任何账户；
- 业务场景
 - 平台方在运营方处充值，运营方调用充值 API 给平台方充值；

用户在某平台方充值，平台方调用充值 API 给该用户充值

4.1.3.3 设置 DDC 业务费收费规则

由运营账户调用该 API 授权设置 DDC 合约收费标准，如果已经设置过计费，将会覆盖原有的计费规则。计费模块根据收费规则对调用者扣除费用。如果调用方法未设置计费规则，则不扣除手续费。

- 输入参数：发送账户，业务类型，合约方法名、对应价格（参考格式：1.0000 FEE）
- 方法命名：setfee
- 方法举例: setfee(name sender,uint64 business_type, name func_name, asset value);
- 模块逻辑：
 - 判断调用者身份，不是运营方，返回错误
 - 增加计费规则

4.1.3.4 删除 DDC 业务模块计费规则

运营方调用该接口删除设置的 DDC 合约计费规则，即便是删除了全部的计费规则也不表示不再授权 DDC 主模块。

- 输入参数：发送账户， 业务类型， 合约方法名
- 方法命名：deletefee
- 方法举例: deletefee(name sender, uint64 business_type, name func_name);
- 模块逻辑：
 - 判断调用者身份， 不是运营方， 返回错误
 - 删除 DDC 合约计费规则

4.1.3.5 删除 DDC 业务主模块授权

运营方调用该接口废除 DDC 计费规则， 废除后， 无法调用计费模块进行扣费。

- 输入参数：发送账户， 业务类型
- 方法命名：deleteddc
- 方法举例: deleteddc(name sender, uint64 business_type);
- 模块逻辑：
 - 判断调用者身份， 不是运营方， 返回错误
 - 删除 DDC 合约的授权。

4.1.3.6 运营账户业务费充值

在业务费总量不足以满足当前业务时， 运营方可以充值业务费总量。

- 输入参数：发送账户， 充值业务费数量（参考格式：1.0000 FEE）；
- 调用者：
- 方法命名：selfrecharge
- 方法举例：selfrecharge(name sender, asset value);
- 核心逻辑：
 - 检查调用者是不是运营账户， 如果不是返回错误
 - 检查充值业务费数量是否大于 0， 不是则返回错误
 - 为运营账户增加对应的金额；

- 业务规则：只允许给运营方账户充值业务费总量；

4.1.3.7 DDC 合约结算

运营账户调用该合约方法，对授权的 DDC 主模块账户发起
结算；

- 输入参数：发送账户，结算金额（参考格式：1.0000 FEE）；
- 方法命名：settlement；
- 方法举例：settlement(name sender, asset value);
- 核心逻辑：
 - 判断调用者(sender)是否是运营账户，否则返回异常；
 - 检查结算金额是否大于 0，不是则返回异常；
 - 按照结算金额将 DDC 合约费用，转账给调用的运营账户；
- 业务规则：该方法只允许运营账户调用；

4.1.3.8 充值(带留言)

- 运营方、平台方调用该 API 将账户的 DDC 业务费充值给下级用户。
- 输入参数：发送账户名，接收方账户名、充值额度(参考格式: 1.000 FEE)、留言；
 - 方法命名：recharge2；
 - 方法举例：recharge2(name from, name to, asset value, string memo);
 - 核心逻辑
 - 同 4.1.3.1 充值接口；
 - 业务规则
 - 同 4.1.3.1 充值接口；

- [业务场景](#)
 - [同 4.1.3.1 充值接口;](#)

4.1.3.2 批量充值

[运营方、平台方调用该 API 将账户的 DDC 业务费批量充值给下级用户。](#)

- [输入参数：发送账户名，接收方账户名列表、充值额度列表、留言;](#)
- [方法命名：rechargebat;](#)
- [方法举例：rechargebat\(name from, std::vector<name> to list, std::vector<asset> value list, string memo\);](#)

- [核心逻辑](#)
 - [同 4.1.3.2 批量充值接口;](#)

- [业务规则](#)
 - [同 4.1.3.2 批量充值接口;](#)

- [业务场景](#)
 - [同 4.1.3.2 批量充值接口;](#)

4.2BSN-DDC-721 业务主模块

4.2.1 功能介绍

BSN-DDC-721 业务主模块用于对外提供一整套完整的 721 所对应的 API 接口便于链账户调用，API 接口包括 BSN-DDC 的生成、转移、冻结、解冻以及销毁等功能。

4.2.2 数据结构

- DDC 全局表（ercglobal）

数据表信息：

编号	字段名	字段	类型	备注
1.	主键 id	primary	uint64	唯一性主键
2.	符号	symbol	string	bsn-ddc 符号
3.	名称	name	string	bsn-ddc 名称
4.	721 主键 序号	erc_721_key	uint64	721 自增主键 的当前位置
4.	1155 主键 序号	erc_1155_key	uint64	1155 自增主键 的当前位置

➤ ddc 信息表结构 (s21info)

数据表信息：

编号	字段名	字段	类型	备注
1.	ddc 的唯一性标识	ddc_id	uint64	自增主键， 此 ddc 的唯一 性标识
2.	ddc 的资源标识符	ddc_uri	string	此 ddc 资源标 识符
3.	ddc 发行者	issuer	name	此 ddc 的 的发行人
4.	状态	allowed	bool	此 ddc 是否 允 许 操 作 (false: 不 允 许， true: 允 许)
5.	ddc 名称	ddc_name	string	可为空， ddc 名称
6.	ddc 符号	ddc_symbol	string	可为空， ddc 符号

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	ddc_id	唯一性索引
2.	二级索引	issuer	name	issuer.value	发行者构成索引

➤ 用户 ddc 数量表(s21balance)

数据表信息：

编号	字段名	字段	类型	备注
1.	区块链账户名	owner	name	拥有者
2.	DDC 数量	balance	uint64	此用户拥有 ddc 的数量

索引信息：

编号	索引类型	索引名	索引类型	索引内容	索引内容
1.	主索引	primay	i64	owner.value	唯一性索引

➤ 用户表结构(s21account)

数据表信息：

编号	字段名	字段	类型	备注
3.	主键 id	primary	uint64	自增主键，从 0 开始
4.	ddc 唯一性标识	ddc_id	uint64	ddc 唯一性标识符
5.	区块链账户名	owner	name	拥有者

索引信息：

编号	索引类型	索引名	索引类型	索引内容	索引内容
1.	主索引	primay	i64	primary	唯一性索引
2.	二级索引	owner	name	owner.value	拥有者构成的索引
3.	二级索引	ddcid	i64	ddc_id	ddc_id 生成索引
4.	二级索引	idowner	i128	uint128(owner.value)<64 + ddc_id	账户名和 ddc_id 构成的索引

➤ ddc 授权表结构(s21ddcappr)

数据表信息：

编号	字段名	字段	类型	备注
1.	ddc 唯一性标识	ddc_id	uint64	ddc 唯一性标识符
3.	授权用户列表	approvals	set<name>	ddc 授权用户列表

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	ddc_id	唯一性索引

➤ 用户授权表结构(s21userappr)

数据表信息：

编号	字段名	字段	类型	备注
1.	主键 id	primary	uint64	自增主键，从 0 开始
2.	授权人	owner	name	授权人
3.	被授权人	account	name	被授权人
4.	是否授权	approved	bool	此用户目前是否被授权

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	primary	唯一性索引
2.	二级索引	owner	name	owner.value	拥有者索引
3.	二级索引	account	name	account.value	被授权人索引
4.	二级索引	owneraccount	i128	uint128(owner.value)<<64+account.value	授权人和被授权人构成的索引

4.2.3 API 定义

4.2.3.1 生成

平台方、终端用户通过调用该方法进行 DDC 的创建。

- 输入参数：发送方账户名,接收方账户名，资源标识符（可空），业务类型,备注;
- 方法命名：mint;
- 方法举例：mint(name sender, name to, uint64_t amount, string ddc_uri, uint64 business_type, string memo);
- 核心逻辑：
 - 检查调用者账户状态是否可用，不可用则调用失败;
 - 验证调用者账户是否有调用权限，没有则调用失败;
 - 检查接收者账户状态是否可用，不可用则调用失败;
 - 检查调用者账户与接收者账户是否属于同平台，不属于则调用失败
 - 验证 business_type 是否为 1
 - 支付 DDC 业务费
 - 保存记录;

4.3.3.2 批量生成

平台方、终端用户通过调用该方法进行 DDC 的批量创建。

- 输入参数：发送账户名，接收方账户名，数量集合，资源标识符集合，业务类型，备注;
- 输出参数：
- 方法命名：mintbatch;
- 方法举例：mintbatch(name from,name to, vector<uint64> amounts, vector<string> ddc_uris, uint64 business_type, string memo);
- 核心逻辑：

- 验证 business_type 是否为 1;
- 检查调用者账户状态是否可用, 不可用则调用失败;
- 验证调用者账户是否有权限, 没有则调用失败;
- 检查接收者账户状态是否可用, 不可用则调用失败;
- 检查调用者账户与接收者账户是否属于同平台, 不属于则调用失败;
- 检查数量集合与资源标识符集合长度是否相等, 不相等则调用失败;
- 批量循环挨个处理 DDC, 并根据索引检查生成的每个 DDCID 是否存在, 存在则调用失败, 否则挨个调用计费合约支付 DDC 业务费和保存生成的 DDC 数据:
- 按照条目数扣除 生成 需要的业务费;
- 保存记录

4.2.3.3 转移

DDC 拥有者或 DDC 授权者通过调用该方法进行 DDC 的转移。

- 输入参数: 拥有方账户名, 接收方账户名, ddc 唯一标识, 业务类型;
- 方法命名: transfer;
- 方法举例: transfer(name sender, name from, name to, uint64 ddc_id, uint64 amount, string memo, uint64 business_type);
- 核心逻辑:
 - 验证 business_type 是否为 1
 - 检查调用者账户状态是否可用, 不可用则调用失败;
 - 检查拥有者账户状态是否可用, 不可用则调用失败;
 - 检查接收者账户状态是否可用, 不可用则调用失败;
 - 验证调用者账户是否有权限, 没有则调用失败;
 - 检查 ddc_id 是否存在, 不存在则调用失败;
 - 检查 ddc_id 是否被冻结, 是则调用失败;

- 检查拥有者账户与接收者账户是否属于同平台，不属于则调用失败；
- 检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户，如果都不是则调用失败；
- 检查 DDC 所对应的拥有者与所传拥有者账户参数是否属于同一账户，不属于则调用失败；
- 支付 DDC 业务费，清除 ddc_id 授权列表；
- 更新记录

4.3.3.4 批量转移

DDC 拥有者或 DDC 授权者通过调用该方法进行 DDC 的批量转移。

- 输入参数：调用者，拥有者的账户名，接受者的账户名，ddc 唯一标识集合，数量集合，附加数据，业务类型；
- 输出参数：
- 方法命名：batchtrans
- 方法举例：batchtrans(name sender,name from, name to, vector<uint64> ddc_ids, vector<uint64> amount, string memo, uint64 business_type) ；
- 核心逻辑：
 - 验证 business_type 是否为 1；
 - 检查数量集合与资源标识符集合长度是否相等，不相等则调用失败；
 - 检查调用者账户状态是否可用，不可用则调用失败；
 - 验证调用者账户是否有权限，没有则调用失败；
 - 检查拥有者账户状态是否可用，不可用则调用失败；
 - 检查接收者账户状态是否可用，不可用则调用失败；
 - 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不属于则调用失败；

- 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户,否则调用失败;
- 批量循环挨个处理 DDC 的安全转移，处理逻辑如下:
 - 1.检查 ddc_id 是否存在，不存在则调用失败;
 - 2.检查 ddc_id 是否被冻结，是则调用失败;
 - 3.检查转移数量是否为 1，否则调用失败;
 - 4.检查拥有者与数据表中拥有者是否属于同一账户，不属于则调用失败
 - 5.扣除业务费;

4.2.3.5 冻结

运营方通过调用该方法进行 DDC 的冻结。

- 输入参数：发送者，ddc 唯一标识，业务类型;
- 方法命名：freeze;
- 方法举例：freeze(name sender, uint64 ddc_id, uint64 business_type);
- 核心逻辑：
 - 验证 business_type 是否为 1;
 - 检查调用者账户状态是否可用，不可用则调用失败;
 - 验证调用者账户是否有权限，没有则调用失败;
 - 检查调用者账户是否为运营方账户，不是则调用失败;
 - 检查 ddc_id 是否存在，不存在则调用失败;
 - 检查 ddc_id 是否被冻结，是则调用失败。
 - 更新状态为冻结

4.2.3.6 解冻

运营方通过调用该方法进行 DDC 的解冻。

- 输入参数：发送者，ddc 唯一标识，业务类型;
- 方法命名：unfreeze;

- 方法举例：unfreeze(name sender, uint64 ddc_id, uint64 business_type);
- 核心逻辑：
 - 验证 business_type 是否为 1;
 - 检查调用者账户状态是否可用，不可用则调用失败;
 - 验证调用者账户是否有权限，没有则调用失败;
 - 检查调用者账户是否为运营方账户，不是则调用失败;
 - 检查 ddc_id 是否存在，不存在则调用失败;
 - 检查 ddc_id 是否被解冻，是则调用失败。
 - 更新状态为解冻

4.2.3.7 销毁

拥有者或授权者通过调用该方法进行 DDC 的销毁。

- 输入参数：发送者，拥有者(721 此参数无效), ddc 唯一标识，业务类型;
- 输出参数：
- 方法命名：burn;
- 方法举例：burn(name sender, name owner, uint64 ddc_id, uint64 business_type)
- 核心逻辑：
 - 验证 business_type 是否为 1
 - 检查调用者账户状态是否可用，不可用则调用失败;
 - 检查拥有者账户状态是否可用，不可用则调用失败;
 - 验证调用者账户是否有权限，没有则调用失败;
 - 检查 DDCID 是否存在，不存在则调用失败;
 - 检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户，如果都不是则调用失败;
 - 支付 DDC 业务费，清除 ddc_id 授权列表;

■ 更新记录

4.2.3.8 批量销毁

拥有者或授权者通过调用该方法进行 DDC 的批量销毁。

- 输入参数：发送者, 拥有者, ddc 唯一标识集合, 业务类型;
- 输出参数:
- 方法命名: burnbatch;
- 方法举例: burnbatch(name sender, name owner, std::vector<uint64_t> ddc_ids, uint64_t business_type)
- 核心逻辑:
 - 验证 business_type 是否为 1
 - 检查调用者账户状态是否可用, 不可用则调用失败;
 - 检查拥有者账户状态是否可用, 不可用则调用失败;
 - 验证调用者账户是否有权限, 没有则调用失败;
 - 批量循环挨个处理 DDC 销毁, 处理逻辑如下:
 - 1.根据索引检查 DDCID 是否存在, 不存在则返回提示信息;
 - 2.根据索引检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户, 如果都不是则返回提示信息;
 - 3.所有检查通过后则调用计费合约支付 DDC 业务费, 并保存 DDC 销毁数据和清除该 DDCID 授权列表;

4.2.3.9 DDC 授权

DDC 拥有者通过该方法进行 DDC 的授权, 发起者需要是 ddc 的拥有方。

- 输入参数：发送者, 被授权方账户, ddc 唯一标识, 业务类型;
- 输出参数:

- 方法命名：approve;
- 方法举例：approve(name sender, name to, uint64 ddc_id, uint64 business_type);
- 核心逻辑：
 - 验证 business_type 是否为 1;
 - 检查调用者状态是否可用，不可用则调用失败;
 - 验证调用者是否有调用权限，没有则调用失败;
 - 检查授权者账户状态是否可用，不可用则调用失败;
 - 检查 ddc_id 是否存在，不存在则调用失败;
 - 检查 ddc_id 是否被冻结，是则调用失败;
 - 检查调用者账户与授权者账户是否属于同平台，不属于则调用失败;
 - 检查 DDC 所对应的拥有者与授权账户是否属同一账户，属于则调用失败;
 - 检查 DDC 所对应的拥有者与调用者账户是否属于同一账户或检查 DDC 所对应的拥有者账户是否授权给调用者账户，不是则调用失败
 - 支付 DDC 业务费
 - 保存记录
 - 注：针对 DDC 授权，授权范围仅限于当前拥有者所拥有的所有 DDC，一旦后续进行转移或安全转移操作，则之前所拥有的授权会因转移或安全转移操作而失效（因拥有者发生改变）；

4.2.3.10 DDC 批量授权

DDC 拥有者通过该方法进行 DDC 的授权，发起者需要是 ddc 的拥有方。

- 输入参数：发送者，被授权方账户, ddc 唯一标识，业务类型;
- 输出参数：
- 方法命名：approvebatch;

- 方法举例：approvebatch(name sender, name to, std::vector<uint64_t> ddc_ids, uint64_t business_type);
- 核心逻辑：
 - 验证 business_type 是否为 1；
 - 检查调用者状态是否可用，不可用则调用失败；
 - 验证调用者是否有调用权限，没有则调用失败；
 - 检查授权者账户状态是否可用，不可用则调用失败；
 - 检查调用者账户与授权者账户是否属于同平台，不属于则调用失败；
 - 批量循环挨个处理 DDC 授权，处理逻辑如下：
 1. 检查 ddc_id 是否存在，不存在则调用失败；
 2. 检查 ddc_id 是否被冻结，是则调用失败；
 3. 检查 DDC 所对应的拥有者与授权账户是否属同一账户，属于则调用失败；
 4. 检查 DDC 所对应的拥有者与调用者账户是否属于同一账户或检查 DDC 所对应的拥有者账户是否授权给调用者账户，不是则调用失败
 5. 支付 DDC 业务费，保存记录
 - 注：针对 DDC 授权，授权范围仅限于当前拥有者所拥有的所有 DDC，一旦后续进行转移或安全转移操作，则之前所拥有的授权会因转移或安全转移操作而失效（因拥有者发生改变）；

4.2.3.11 账户授权

- 平台方或者用户通过调用该方法进行账户的授权。
- 输入参数：发送方，被授权账户，授权标识，业务类型；
 - 输出参数：
 - 方法命名：approvalall;
 - 方法举例：approvalall(name sender, name to, bool approved, uint64 business_type);
 - 核心逻辑：

- 检查调用者状态是否可用，不可用则返回提示信息；
- 验证调用者是否有调用权限，没有则返回提示信息；
- 验证 business_type 是否为 1
- 检查授权者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户与授权者账户是否属于同平台，不属于则返回提示信息；
- 检查调用者账户与授权者账户是否属于同一账户，属于则返回提示信息；
- 支付 DDC 业务费
- 保存记录

4.2.3.12 URI 设置

DDC 拥有者或 DDC 授权者通过调用该方法对 DDC 的资源标识符进行设置。

- 输入参数：调用者，拥有者（ERC-721 此参数无效），ddc 唯一标识，资源标识符，业务类型；
- 输出参数：
- 方法命名：seturi；
- 方法举例：seturi(name sender, name owner, uint64_t ddc_id, std::string ddc_uri, uint64_t business_type);
- 核心逻辑：
 - 检查传入 ddc 资源标识符是否为空值，是则返回提示信息；
 - 验证 business_type 是否为 1；
 - 检查调用者账户状态是否可用，不可用则调用失败；
 - 检查调用者账户是否有权限，没有则调用失败；
 - 检查 ddc_id 是否存在，不存在则调用失败；
 - 检查 ddc_id 对应的资源标识符是否为空值，不是则调用失败；

- 检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户， 如果都不是则调用失败；
- 保存记录；

4.2.3.13 名称符号设置

合约拥用者可以对 721 的名称以及符号进行初始化。

- 输入参数： 名称， 符号；
- 输出参数：
- 方法命名： setnamesym;
- 方法举例： setnamesym(string name, string symbol);
- 核心逻辑： 无

4.3BSN-DDC-1155 业务主模块

4.3.1 功能介绍

BSN-DDC-1155 业务主模块用于对外提供一整套完整的 1155 所对应的 API 接口便于链账户调用， API 接口包括 BSN-DDC 的生成、批量生成、转移、批量转移、冻结、解冻以及销毁等功能。

4.3.2 数据结构

- ddc 信息表结构 (1155info)

数据表信息：

编号	字段名	字段	类型	备注
1.	ddc 的唯一性标识	ddc_id	uint64	自增主键，此 ddc 的唯一性标识

2.	ddc 的资源标识符	ddc_uri	string	此 ddc 资源标识符
3.	ddc 发行者	issuer	name	此 ddc 的 的发行人
4.	状态	allowed	bool	此 ddc 是否 允许操作 (false: 不 允许, true: 允许)
5.	ddc 数量	supply	uint64	此 ddc 的 数量
6.	ddc 名称	ddc_name	string	可为空, ddc 名称
7.	ddc 符号	ddc_symbol	string	可为空, ddc 符号

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	ddc_id	唯一性索引
2.	二级索引	issuer	name	issuer.value	拥有者构成索引

➤ 用户信息表结构（1155account）

数据表信息：

编号	字段名	字段	类型	备注
1.	主键 id	primary	uint64	自增主键, 从 0 开始
2.	区块链账户名	owner	name	拥有者
3.	ddc 唯一性标识	ddc_id	uint64	ddc 唯一性标 识符
4.	DDC 数量	quantity	uint64	此用户拥有 ddc 的数量

索引信息：

编号	索引类型	索引名	索引类型	索引内容	索引内容
1.	主索引	primay	i64	primary	唯一性索引

2.	二级索引	owner	name	owner.value	账户名构成的索引
3.	二级索引	ddcid	i64	ddc_id	ddc_id 生成索引
4.	二级索引	idowner	i128	uint128(owner.value) <<64 + ddc_id	账户名和 ddc_id 构成的索引

➤ 用户授权表结构(1155userappr)

数据表信息：

编号	字段名	字段	类型	备注
1.	主键 id	primary	uint64	自增主键，从 0 开始
2.	授权人	owner	name	授权人
3.	被授权人	account	name	被授权人
4.	是否授权	approved	bool	此用户目前是否被授权

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	primary	唯一性索引
2.	二级索引	owner	name	owner.value	拥有者索引
3.	二级索引	account	name	account.value	被授权人索引
4.	二级索引	owneraccount	i128	uint128(owner.value) <<64 + account.value	授权人和被授权人构成的索引

4.3.3 API 定义

4.3.3.1 生成

平台方、终端用户通过调用该方法进行 DDC 的创建。

- 输入参数：发送方账户名，接收方账户名，数量，资源标识符，业务类型，备注；
- 输出参数：
- 方法命名：mint；
- 方法举例：mint(name from,name to,uint64 amount, string ddc_uri, uint64 business_type, string memo)；
- 核心逻辑：
 - 检查调用者账户状态是否可用，不可用则返回提示信息；
 - 验证调用者账户是否有权限，没有则返回提示信息；
 - 验证 business_type 是否为 2
 - 检查接收者账户状态是否可用，不可用则返回提示信息；
 - 检查生成 DDC 所对应的数量是否大于 0，不是则返回提示信息；
 - 支付 DDC 业务费
 - 保存记录

4.3.3.2 批量生成

平台方、终端用户通过调用该方法进行 DDC 的批量创建。

- 输入参数：发送账户名，接收方账户名，数量集合，资源标识符集合，业务类型，备注；
- 输出参数：
- 方法命名：mintbatch；
- 方法举例：mintbatch(name from,name to, vector<uint64> amounts, vector<string> ddc_uris, uint64 business_type, string memo)；
- 核心逻辑：
 - 验证 business_type 是否为 2；
 - 检查调用者账户状态是否可用，不可用则调用失败；
 - 验证调用者账户是否有权限，没有则调用失败；
 - 检查接收者账户状态是否可用，不可用则调用失败；

- 检查调用者账户与接收者账户是否属于同平台，不属于则调用失败；
- 检查数量集合与资源标识符集合长度是否相等，不相等则调用失败；
- 按照条目数扣除 生成 需要的业务费；
- 保存记录

4.3.3.3 转移

DDC 拥用者或 DDC 授权者通过调用该方法进行 DDC 的转移。

- 输入参数：调用者，拥有者的账户名，接受者的账户名，ddc 唯一标识，数量，附加数据，业务类型；
- 输出参数：
- 方法命名：transfer；
- 方法举例：transfer(name sender, name from, name to, uint64 ddc_id, uint64 amount, string memo, uint64 business_type) ；
- 核心逻辑：
 - 验证 business_type 是否为 2
 - 检查调用者账户状态是否可用，不可用则调用失败；
 - 验证调用者账户是否有权限，没有则调用失败；
 - 检查拥有者账户状态是否可用，不可用则调用失败；
 - 检查接收者账户状态是否可用，不可用则调用失败；
 - 检查 ddc_id 是否存在，不存在则调用失败；
 - 检查 ddc_id 是否被冻结，是则调用失败；
 - 检查拥有者账户与接收者账户是否属于同平台，不属于则调用失败；
 - 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户,否则调用失败；
 - 查看转移数量是否大于 0，否则调用失败；

- 检查拥有者账户对 DDC 所拥有的数量是否大于等于转移数量, 否则调用失败;
- 支付 DDC 业务费;
- 更新记录

4.3.3.4 批量转移

DDC 拥有者或 DDC 授权者通过调用该方法进行 DDC 的批量转移。

- 输入参数：调用者，拥有者的账户名，接受者的账户名，ddc 唯一标识集合，数量集合，附加数据，业务类型;
- 输出参数：
- 方法命名：batchtrans
- 方法举例：batchtrans(name sender,name from, name to, vector<uint64> ddc_ids, vector<uint64> amount, string memo, uint64 business_type) ;
- 核心逻辑：
 - 验证 business_type 是否为 2
 - 检查调用者账户状态是否可用，不可用则调用失败;
 - 验证调用者账户是否有权限，没有则调用失败;
 - 检查拥有者账户状态是否可用，不可用则调用失败;
 - 检查接收者账户状态是否可用，不可用则调用失败;
 - 检查 ddc_id 是否存在，不存在则调用失败;
 - 检查 ddc_id 是否被冻结，是则调用失败;
 - 检查拥有者账户与接收者账户是否属于同平台，不属于则调用失败;
 - 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户,否则调用失败;
 - 检查数量集合与 ddc_id 集合长度是否相等，不相等则返回调用失败;

- 批量循环挨个处理 DDC，并根据索引检查每个 ddc_id 是否存在，是否被冻结，不存在或被冻结则返回调用失败；
- 检查每个 DDC 所对应的数量是否大于 0，不是则返回调用失败，最后根据索引检查拥有者账户拥有 DDC 所对应的数量是否大于等于需要转移的 DDC 数量
- 按照条目数扣除 转移 需要的业务费；
- 更新记录

4.3.3.5 冻结

运营方通过调用该方法进行 DDC 的冻结。

- 输入参数：发送者，ddc 唯一标识，业务类型；
- 输出参数：
- 方法命名：freeze；
- 方法举例：freeze(name sender, uint64 ddc_id, uint64 business_type);
- 核心逻辑：
 - 验证 business_type 是否为 2；
 - 检查调用者账户状态是否可用，不可用则调用失败；
 - 验证调用者账户是否有权限，没有则调用失败；
 - 检查调用者账户是否为运营方账户，不是则调用失败；
 - 检查 ddc_id 是否存在，不存在则调用失败；
 - 检查 ddc_id 是否被冻结，是则调用失败；
 - 更新状态为冻结

4.3.3.6 解冻

运营方通过调用该方法进行 DDC 的解冻。

- 输入参数：发送者, ddc 唯一标识，业务类型；
- 输出参数：
- 方法命名：unfreeze；

- 方法举例：unfreeze(name sender, uint64 ddc_id, uint64 business_type);
- 核心逻辑：
 - 验证 business_type 是否为 2;
 - 检查调用者账户状态是否可用，不可用则调用失败;
 - 验证调用者账户是否有限权，没有则调用失败;
 - 检查调用者账户是否为运营方账户，不是则调用失败;
 - 检查 ddc_id 是否存在，不存在则调用失败;
 - 检查 ddc_id 是否被解冻，是则调用失败。
 - 更新状态为解冻

4.3.3.7 销毁

拥有者或授权者通过调用该方法进行 DDC 的销毁。

- 输入参数：发送者，拥有者，ddc 唯一标识，业务类型;
- 输出参数:
- 方法命名：burn;
- 方法举例：burn(name sender, name owner, uint64 ddc_id, uint64 business_type);
- 核心逻辑：
 - 验证 business_type 是否为 2
 - 检查调用者账户状态是否可用，不可用则调用失败;
 - 检查拥有者账户状态是否可用，不可用则调用失败;
 - 验证调用者账户是否有限权，没有则调用失败;
 - 检查 ddc_id 是否存在，不存在则返回调用失败;
 - 检查拥有者账户是否拥有此 DDC，没有则调用失败;
 - 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户，否则调用失败;
 - 支付 DDC 业务费;
 - 更新记录

4.3.3.8 批量销毁

拥有者或授权者通过调用该方法进行 DDC 的销毁。

- 输入参数：发送者，拥有者，ddc 唯一标识的集合，业务类型；
- 输出参数：
- 方法命名：burnbatch；
- 方法举例：burnbatch(name sender, name owner, vector<uint64>
ddc_ids, uint64 business_type);
- 核心逻辑：
 - 验证 business_type 是否为 2
 - 检查调用者账户状态是否可用，不可用则调用失败；
 - 检查拥有者账户状态是否可用，不可用则调用失败；
 - 验证调用者账户是否有权限，没有则调用失败；
 - 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户，否则调用失败
 - 检查 ddc_id 是否存在，不存在则返回调用失败；
 - 批量循环 ddc_id 集合列表，依次检查拥有者账户是否拥有此 DDC，否则调用失败
 - 按照条目数扣除 转移 需要的业务费；
 - 更新记录

4.3.3.9 账户授权

平台方或者用户通过调用该方法进行账户的授权。

- 输入参数：发送方，被授权账户，授权标识，业务类型；
- 输出参数：
- 方法命名：approvalall；
- 方法举例：approvalall(name sender, name to, bool approved, uint64
business_type);
- 核心逻辑：

- 检查调用者状态是否可用，不可用则返回提示信息；
- 验证调用者是否有调用权限，没有则返回提示信息；
- 验证 business_type 是否为 2
- 检查授权者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户与授权者账户是否属于同平台，不属于则返回提示信息；
- 检查调用者账户与授权者账户是否属于同一账户，属于则返回提示信息；
- 支付 DDC 业务费；
- 保存记录；

4.3.3.10 URI 设置

DDC 拥有者或 DDC 授权者通过调用该方法对 DDC 的资源标识符进行设置。

- 输入参数：调用者，拥有者，ddc 唯一标识，资源标识符，业务类型；
- 输出参数：
- 方法命名：seturi；
- 方法举例：seturi(name sender, name owner, uint64_t ddc_id, std::string ddc_uri, uint64_t business_type);
- 核心逻辑：
 - 检查传入 ddc 资源标识符是否为空值，是则返回提示信息；
 - 验证 business_type 是否为 2；
 - 检查调用者账户状态是否可用，不可用则调用失败；
 - 检查调用者账户是否有权限，没有则调用失败；
 - 检查 ddc_id 是否存在，不存在则调用失败；
 - 检查 ddc_id 对应的资源标识符是否为空值，不是则调用失败；
 - 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户，否则调用失败；
 - 保存记录；

4.4BSN-DDC-权限模块

4.4.1 功能介绍

BSN-DDC-权限合约用以进行对某个账户进行角色判定的逻辑实现。权限角色分为三种：运营方、平台方、用户方，不同角色拥有的权限各不相同，一个现实中的实体可持有多套链账户用以对应 DDC 中的多个角色，如 BSN 可作为运营方存在于 DDC 中。三种角色分别为下级角色的 leader，可管理该下级账户。

- 用户方权限：经与平台方绑定以后可进行该平台内 DDC 的转移和查询的操作。
- 平台方权限：包含 DDC 的整个生命周期的管理以及本平台内用户方账户的增删改查操作。
- 运营方权限：包含用户方权限、平台方账户和权限以及运营方账户和权限的增删改查操作、DDC 相关流转操作的定价权（DDC 发行、转移等操作需扣除相应的费用）、对平台方账户以及用户方账户的充提查操作。

4.4.2 数据结构

- 账户权限表结构（permglobal）

数据表信息：

编号	字段名	字段	类型	备注
1.	主键 id	primary	uint64_t	唯一性主键索引
2.	平台方操作开关	is_platform_ope n	bool	是否开启平台方操作开关

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	primary	唯一性主键索引

➤ 账户权限表结构（permaccounts）

数据表信息：

编号	字段名	字段	类型	备注
1.	区块链账户名	account	name	DDC 用户链账户名
2.	账户 DID	account_did	string	DDC 账户对应的 DID 信息（普通用户可为空）
3.	账户名称	account_name	string	DDC 账户对应的账户名
4.	账户角色	account_role	enum	DDC 账户对应的身份信息。值包含： 1. Operator（运营方） 2. PlatformManager（平台方） 3. Consumer（用户方）
5.	账户上级管理者	leader_did	string	DDC 账户对应的上级管理员，账户角色为 Consumer 时必填。对于普通用户 Consumer 该值为平台管理者 PlatformManager
6.	平台管理账户状态	platform_state	enum	DDC 账户对应的当前账户状态（仅平台方可操作该状态）。值包含： 1. Frozen（冻结状态，无法进行 DDC 相关操作） 2. Active（活跃状态，可进行 DDC 相关操作）
7.	运营管理账户状态	operator_state	enum	DDC 账户对应的当前账户状态（仅运营方可操作该状态）。值包含：

				1. Frozen（冻结状态，无法进行 DDC 相关操作） Active（活跃状态，可进行 DDC 相关操作）
8.	冗余字段	field	string	冗余字段

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	account.value	区块链账户名形成的唯一性索引
2.	二级索引	accountdid	sha256	sha256(account_did)	账户 DID 生成的哈希索引
3.	二级索引	leaderdid	sha256	sha256(leader_did)	上级 DID 的哈希索引

➤ 方法权限表结构（permethods）

数据表域：数据表按照 business_type（ERC-721:1、ERC-1155:2） 进行分域，

即在 get_table_rows 操作时，scope 值填写 1 或 2

数据表信息：

编号	字段名	字段	类型	备注
1.	账户角色	role	uint8	DDC 账户对应的身份信息，值包含： Operator（运营方），值为 1 PlatformManager（平台方），值为 2 Consumer（用户方），值为 3
2.	方法列表	methods	set<name>	可访问的方法列表

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	role	角色形成的唯一性索引

➤ 跨平台授权表结构（permapprs）

数据表信息：

编号	字段名	字段	类型	备注
1.	主键 id	primary	uint64	唯一性 id
2.	授权 did	account_did	string	授权人的 account_did
3.	被授权 did 列表	did_approvals	set<string>	跨平台被授权人的 account_did 列表

索引信息：

编号	索引类型	索引名	索引类型	索引内容	备注
1.	主索引	primay	i64	primary	唯一性索引
2.	二级索引	accountdid	sha256	sha256(account_did)	授权 DID 生成的哈希索引

4.4.3 API 定义

4.4.3.1 添加运营账户

合约所有者通过调用该方法可以创建运营方账户。

- 输入参数：账号，账户名称，账户 DID；
- 输出参数：
- 方法命名：addoperator；
- 方法举例：addoperator(name operator_name,string account_name,string account_did)

- 核心逻辑：
 - 校验调用者是否为合约拥有者，否则调用失败；
 - 校验 account_did 和 account_name 不为空，若不满足则调用失败；
 - 校验该 operator 不存在，若不满足则调用失败；
 - 否则进行该信息的存储并调用成功；

4.4.3.2 运营方添加账户

运营方通过调用该方法进行平台方，终端用户的创建。

- 输入参数：发送者，账户，账号名，账号 did，上级 did；
- 方法命名：operatoradd；
- 方法举例：operatoradd(name sender, name account, string account_name, string account_did, string leader_did);
- 核心逻辑：
 - 检查调用者账户信息是否存在，不存在则返回；
 - 检查调用者账户对应的平台方状态是否活跃，不活跃则调用失败；
 - 检查调用者账户对应的运营方状态是否活跃，不活跃则调用失败；
 - 检查添加的账户是否存在，存在则调用失败；
 - 检查调用者账户类型是否为运营方，不是则调用失败；
 - 检查添加的账户名称长度是否为 0，是则返回调用失败；
 - 如果上级 did 为空(添加平台方)，检查此账户 did 对应的账户是否存在，如存在，检查账户角色是否为平台方且账户的上级 did 和调用者的账户 did 是否相等，否则调用失败
 - 如果上级 did 不为空(添加终端用户)，检查上级 did 对应的账户是否存在且角色是否为平台方，否则调用失败
 - 写入记录；

4.4.3.3 运营方批量添加账户

运营方通过调用该方法批量进行平台方，终端用户的创建。

- 输入参数：发送者，账户集合，账号名集合，账号 did 集合，上级 did 集合；
- 方法命名：operaddbat;
- 方法举例：operaddbat(name sender, std::vector<name> accounts, std::vector<std::string> account_names, std::vector<std::string> account_dids, std::vector<std::string> leader_dids);
- 核心逻辑：
 - 检查调用者账户信息是否存在，不存在则返回；
 - 检查调用者账户对应的平台方状态是否活跃，不活跃则调用失败；
 - 检查调用者账户对应的运营方状态是否活跃，不活跃则调用失败；
 - 检查调用者账户类型是否为运营方，不是则调用失败；
 - 检查账户集合，账户名称集合、账户 DID 集合以及上级 DID 集合长度是否相等
 - 批量循环挨个处理链账户的添加，处理逻辑如下：
 - 1.根据索引检查添加的账户名称长度是否为 0，是则返回提示信息；
 - 2.根据索引检查添加的账户是否存在，存在则返回提示信息；
 - 3.如果上级 did 为空(添加平台方),检查此账户 did 对应的账户是否存在，如存在，检查账户角色是否为平台方且账户的上级 did 和调用者的账户 did 是否相等，否则调用失败
 - 4.如果上级 did 不为空(添加终端用户),检查上级 did 对应的账户是否存在且角色是否为平台方，否则调用失败
 - 5.根据索引所有检查通过后则保存账户数据；

4.4.3.4 平台方添加账户

平台方通过调用该方法进行终端用户的创建。

- 输入参数：发送者，账户，账号名，账号 did；
- 方法命名：manageradd;

- 方法举例：manageradd(name sender, name account, string account_name, string account_did);
- 核心逻辑：
 - 检查调用者账户信息是否存在，不存在则返回提示信息；
 - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
 - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
 - 检查调用者账户类型是否为平台方，不是则返回提示信息；
 - 检查调用者账户是否打开添加链账户开关控制，不是则返回提示信息；
 - 检查添加的账户名称长度是否为 0，是则返回提示信息；
 - 检查添加的账户是否已存在，存在则返回提示信息；
 - 所有检查通过后则保存账户数据，最后触发事件；

4.4.3.5 平台方批量添加账户

平台方通过调用该方法进行终端用户的批量创建。

- 输入参数：发送者，账户集合，账号名集合，账号 did 集合；
- 方法命名：managaddbat;
- 方法举例：managaddbat(name sender, std::vector<name> accounts, std::vector<std::string> account_names, std::vector<std::string> account_dids);
- 核心逻辑：
 - 检查调用者账户信息是否存在，不存在则调用失败；
 - 检查链账户地址集合长度、账户名称集合长度以及账户 DID 集合长度是否相等，不相等则调用失败；
 - 检查调用者账户对应的平台方状态是否活跃，不活跃则调用失败；
 - 检查调用者账户对应的运营方状态是否活跃，不活跃则调用失败；

- 检查调用者账户类型是否为平台方，不是则调用失败；
- 批量循环挨个处理链账户的添加，处理逻辑如下：
 1. 检查调用者账户是否打开添加链账户开关控制，不是则调用失败；
 2. 根据索引检查添加的账户名称长度是否为 0，是则调用失败；
 3. 根据索引检查添加的账户是否已存在，存在则调用失败；
 4. 所有检查通过后则保存账户数据；

4.4.3.6 删除账户

通过该方法进行 DDC 账户信息的删除。account 为用户的链账户名。

上级角色可进行下级角色账户的操作。暂不对外开放调用。

- 输入参数：发送者,账户名；
- 方法命名：delaccount；
- 方法举例：delaccount(name sender, name account)；
- 核心逻辑：
 - 校验该 account 是否已存在，不存在则调用失败；
 - 校验该 account 的上级是否为调用者，若不满足则调用失败；
 - 否则进行该信息的删除，完成调用；

4.4.3.7 平台方添加链账户开关设置

运营方通过调用该 API 接口设置平台方添加链账户开关权限控制。

- 输入参数：是否打开；
- 输出参数：无；
- 方法命名：setswitcher；
- 方法举例：setswitcher(name sender, bool is_platform_open)；
- 核心逻辑：
 - 检查调用者账户信息是否存在，不存在则调用失败；
 - 检查调用者账户对应的平台方状态是否活跃，不活跃则调用失败；

- 检查调用者账户对应的运营方状态是否活跃，不活跃则调用失败；
- 检查调用者账户类型是否为运营方，不是则调用失败；
- 检查 is_platform_open 是否与当前开关状态一致，一致则调用失败；

4.4.3.8 更新账户状态

通过该方法进行 DDC 账户信息状态的更改。name 为用户的链账户名。上级角色可进行下级角色账户的操作。

- 输入参数：发送方，账户名，账户状态；
- 方法命名：updateacc;
- 方法举例：updateacc(name sender,name account, state sta)
- 核心逻辑：
 - 校验该账户是否已存在，不存在则调用失败；
 - 校验该 account 的上级是否为调用者，若不满足则调用失败；
 - 校验要操作的 account 状态是否可被该调用者操作；
 - 进行该信息状态的更新,完成调用；

4.4.3.9 添加方法

通过该方法进行角色可调用方法的增加。

- 输入参数：发送名，账户角色，业务类型，方法名称；
- 方法命名：addfunction;
- 方法举例：addfunction(name sender,uint8 account_role, uint64 business_type, name func_name);
- 核心逻辑：
 - 校验调用者身份是否为运营方，不为运营方则调用失败；
 - 校验该方法名是否存在，存在则调用失败；
 - 进行该信息的插入，调用成功。

4.4.3.10 删除方法

通过该方法进行角色可调用方法的删除。

- 输入参数：发送名,账户角色， 业务类型 方法名称；
- 输出参数：
- 方法命名：delfunction;
- 方法举例：delfunction(name sender,uint8 account_role, uint64 business_type, name func_name);
- 核心逻辑：
 - 校验调用者身份是否为运营方，不为运营方则调用失败；
 - 校验该方法名是否存在，不存在则调用失败；
 - 进行该信息的删除，完成调用。

4.4.3.11 跨平台授权

运营方通过调用该 API 进行跨平台之间的账户授权。

- 输入参数：发送账户,授权者账户， 接收者账户， 是否授权；
- 输出参数：
- 方法命名：crossappr;
- 方法举例：crossappr(name sender, name from, name to, bool approved);
- 核心逻辑：
 - 检查调用者账户信息是否存在，不存在则调用失败；
 - 检查调用者账户是否为运营方，不是则调用失败；
 - 检查调用者账户对应的平台方状态是否活跃，不活跃则调用失败；
 - 检查调用者账户对应的运营方状态是否活跃，不活跃则调用失败；
 - 检查授权者账户信息是否存在，不存在则调用失败；
 - 检查授权者账户对应的平台方状态是否活跃，不活跃则调用失败；
 - 检查授权者账户对应的运营方状态是否活跃，不活跃则调用失败；
 - 检查接收者账户信息是否存在，不存在则调用失败；

- 检查接收者账户对应的平台方状态是否活跃, 不活跃则调用失败;
- 检查接收者账户对应的运营方状态是否活跃, 不活跃则调用失败;
- 检查授权者账户与接收者账户是否属于同平台, 是则调用失败;
- 所有检查通过后则保存跨平台授权结果。

4.5BSN-DDC-元交易模块

4.5.1 功能介绍

在 BSN-DDC 合约中, 用户分为三个角色, 运营方、平台方和终端用户, 运营方为 BSN-DDC 的运营者, 平台方为入驻 BSN-DDC 合约中的发行方, 终端用户为平台方的用户。

BSN-DDC 的拥有者大部分为终端用户, 如后续发起转移、授权等操作, 需要终端用户拥有链资源和 DDC 合约中的手续费, 由于终端用户数量较多(在几十万级别), 为每个终端用户充值链资源(中移链为 CPU\NET\RAM)并充值手续费(DDC 合约中的手续费)的工作, 反复复杂、流程较长, 故提出元交易的设想, 解决这个问题。

元交易的目的是由平台方发起终端用户对 DDC 的转移和授权, 平台方付资源费和手续费。终端用户, 需要使用自己的私钥, 对自己拥有的 DDC 的转移、授权等交易进行签名, 把签名发给平台方, 由平台方发起这个元交易。

4.5.2 数据结构

➤ 元交易用户信息表 (metauserinfo)

数据表信息:

编号	字段名	字段	类型	备注
1.	中移链账户名	account	name	唯一性主键索引
2.	账户对应公钥	pubKey	string	EOS 格式公钥
3	nonce 值列表	nonceMap	map<uint64_t,	键为业务类型: 1 表示 ERC-721

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, (中文)

设置格式[1]: (中文)

设置格式[1]: (中文) 中文(简体)

			<u>uint64 t></u>	<u>2 表示 ERC-1155</u>
--	--	--	---------------------	----------------------

设置格式[1]: (中文) 中文(简体)

索引信息:

设置格式[1]: (中文) 中文(简体)

编号	索引类型	索引名	索引类型	索引内容	备注
<u>1.</u>	<u>主索引</u>	<u>account</u>	<u>i64</u>	<u>account.value</u>	<u>唯一性主键索引</u>

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, (中文)

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, (中文)

➤ 元交易初始化信息表 (metainitinfo)

数据表信息:

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, (中文)

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 英语(美国), (中文)

编号	字段名	字段	类型	备注
<u>1.</u>	<u>唯一性主键</u>	<u>primary</u>	<u>uint64 t</u>	<u>唯一性主键索引</u>
<u>2.</u>	<u>分隔符列表</u>	<u>separatorMap</u>	<u>map<uint64 t, string></u>	<u>键为业务类型:</u> <u>1 表示 ERC-721</u> <u>2 表示 ERC-1155</u>
<u>3.</u>	<u>授权哈希列表</u>	<u>hashTypeMap</u>	<u>map<uint8 t, string></u>	<u>键为哈希类型, 包含:</u> <u>1 mint (生成)</u> <u>2 mintbatch (批量生成)</u> <u>3 transfer (转移)</u> <u>4 batchtrans (批量转移)</u> <u>5 burn (销毁)</u> <u>6 burnbatch (批量销毁)</u>

设置格式[1]: (中文)

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, 五号, 字距调整: 1 磅

4.5.3 API 定义

设置格式[1]: 孤行控制

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 左

4.5.3.1 授权哈希设置

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号

合约部署方通过调用该方法可以设置每种交易对应的授权哈希值。

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

➤ 输入参数: 授权哈希类型, 授权哈希值;

➤ 输出参数:

设置格式[1]: 标题 4

➤ 方法命名: setmetathash;

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非倾斜


➤ 方法举例：`setmetathash (uint64_t hashType, string hashValue)`


➤ 核心逻辑：


■ 校验调用者账户是否为合约部署方，不是则返回提示信息；

■ 检查授权哈希值长度是否为 0，是则返回提示信息；

■ 校验通过后将授权哈希类型及授权哈希值存入元交易初始化信息表。

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 左, 缩进: 左侧: 37.5 毫米, 悬挂缩进: 5 

设置格式[1]: 项目符号和编号

4.5.3.2 分隔符设置

合约部署方通过调用该方法可以设置每种业务类型对应的域名分隔符。

➤ 输入参数：业务类型，域名分隔符；

➤ 输出参数：

➤ 方法命名：`setmetasepar` ；


➤ 方法举例：`setmetasepar (uint64_t business_type ,string separator);`


➤ 核心逻辑：


■ 校验调用者账户是否为合约部署方，不是则返回提示信息；

■ 检查域名分隔符长度是否为 0，是则返回提示信息；


■ 校验通过后将业务类型及域名分隔符存入元交易初始化信息表。


设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 左, 缩进: 左侧: 37.5 毫米, 悬挂缩进: 5 

设置格式[1]: 项目符号和编号

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

4.5.3.3 添加终端用户公钥

平台方或运营方通过调用该方法添加终端用户的账户名和公钥信息。


➤ 输入参数：调用者账户名称，中移链账户名称，中移链账户对应的公钥；


➤ 输出参数：


➤ 方法命名：`addmetauser` ；


➤ 方法举例：`addmetauser (name sender, name account, string pubKey);`


➤ 核心逻辑：


设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 


设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 


设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 


设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 


设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 L 

- 检查调用者账户信息是否存在，不存在则返回提示信息；
- 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
- 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
- 检查添加账户状态是否可用，不可用则调用失败；
- 检查调用者账户类型是否为平台方或运营方，不是则返回提示信息；
- 检查调用者账户与接收者账户是否属于同平台，不属于则调用失败；
- 检查添加的账户名称长度是否为 0，是则返回提示信息；
- 检查添加的公钥长度是否为 0，是则返回提示信息；
- 所有检查通过后则保存或更新账户数据至元交易用户信息表，并将交易顺序 nonce 值初始化都置为 1。

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示

设置格式[1]: 左, 缩进: 左侧: 37.5 毫米, 悬挂缩进: 5 毫米, 项目符号 + 级别: 1 + 对齐位置: 37.5 毫米 + 缩进位置: 44.9 毫米, 孤行控制

设置格式[1]: 项目符号和编号

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 非突出显示

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 非突出显示

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示

设置格式[1]: 左, 缩进: 左侧: 37.5 毫米, 悬挂缩进: 5 毫米, 行距: 1.5 倍行距, 项目符号 + 级别: 1 + 对齐位置: 37.5 毫米 + 缩进位置: 44.9 毫米, 孤行控制

4.5.3.4 元交易生成

平台方通过调用该方法进行 DDC 元交易的创建。

- 输入参数: 调用者账户名 (平台方), 发送者账户名 (终端用户), 接收方账户名, 数量, 资源标识符, 业务类型, 备注, nonce 值, 过期时间, 签名值;

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号

- 输出参数:

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非倾斜

- 方法命名: metamint;

- 方法举例: metamint (name sender, name from, name to, uint64_t amount, std::string ddc_uri, uint64_t business_type, std::string memo, uint64_t nonce, uint64_t deadline, signature signature);

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非倾斜, 非突出显示

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非倾斜

- 核心逻辑:

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示

- 校验调用者账户是否为平台方，不是则返回提示信息；
- 根据 nonce 值和过期时间以及预设的域名分隔符和授权哈希类型所对应的授权哈希值计算出哈希值，并根据哈希值和签名值计算得

设置格式[1]: 左, 缩进: 左侧: 35.3 毫米, 悬挂缩进: 5 毫米, 项目符号 + 级别: 2 + 对齐位置: 44.9 毫米 + 缩进位置: 52.3 毫米, 孤行控制

- 出公钥，校验计算得出的公钥与元交易用户信息表中 from 对应的公钥是否相等，不相等则返回提示信息；
- 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；（后期实现，需要对合约虚拟机改造，获取交易或者区块的时间戳）；
- 校验参数中业务类型是否符合规范，否则返回提示信息；
- 校验元交易用户信息表中 from 对应的该业务类型 nonce 值与传参 nonce 值参数是否相等，不是则返回提示信息，校验通过后为对应 nonce 值加 1；
- 校验通过后根据业务类型调用 mint_721 或 mint_1155 方法。

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示

4.5.3.5 元交易批量生成

- 平台方通过调用该方法进行 DDC 元交易的批量创建。
- 输入参数: 调用者账户名 (平台方), 发送者账户名 (终端用户), 接收方账户名, 数量集合, 资源标识符集合, 业务类型, 备注, nonce 值, 过期时间, 签名值;
- 输出参数:
- 方法命名: metamintbatch;
- 方法举例: metamintbatch (name sender, name from, name to, std::vector<uint64_t> amounts, std::vector<std::string> ddc_uris, uint64_t business_type, std::string memo, uint64_t nonce, uint64_t deadline, signature signature);
- 核心逻辑:
 - 校验调用者账户是否为平台方，不是则返回提示信息；
 - 根据 nonce 值和过期时间以及预设的域名分隔符和授权哈希类型所对应的授权哈希值计算出哈希值，并根据哈希值和签名值计算出公钥，校验计算得出的公钥与元交易用户信息表中 from 对应的公钥是否相等，不相等则返回提示信息；

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非倾斜

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非倾斜, 非突出显示

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非倾斜

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示, (中文)

设置格式[1]: 左, 缩进: 左侧: 37.5 毫米, 悬挂缩进: 5 毫米, 项目符号 + 级别: 1 + 对齐位置: 37.5 毫米 + 缩进位置: 44.9 毫米, 孤行控制

设置格式[1]: 项目符号和编号

- 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；（后期实现，需要对合约虚拟机改造，获取交易或者区块的时间戳）；
- 校验参数中业务类型是否符合规范，否则返回提示信息；
- 校验元交易用户信息表中from对应的该业务类型 nonce 值与传参 nonce 值参数是否相等，不是则返回提示信息，校验通过后为对应 nonce 值加 1；
- 校验通过后根据业务类型调用 mintbatch 721 或 mintbatch 1155 方法。

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示, (中文)

4.5.3.6 元交易转移

平台方通过调用该方法进行 DDC 元交易的转移。

- 输入参数: 调用者账户名 (平台方), 拥有者的账户名 (终端用户), 接受者的账户名, ddc 唯一标识, 数量, 附加数据, 业务类型, nonce 值, 过期时间, 签名值;
- 输出参数:
- 方法命名: metatransfer ;
- 方法举例: metatransfer (name sender, name from, name to, uint64 t ddc_id, uint64 t amount, std::string memo, uint64 t business_type, uint64 t nonce, uint64 t deadline, signature signature);
- 核心逻辑:

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非突出显示

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, 五号, 非倾斜

- 校验调用者账户是否为平台方，不是则返回提示信息；
- 检查根据 nonce 值和过期时间以及预设的域名分隔符和授权哈希类型所对应的授权哈希值计算出哈希值，并根据哈希值和签名值计算得出公钥，校验计算得出的公钥与元交易用户信息表中 from 对应的公钥是否相等，不相等则返回提示信息；

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

- 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；（后期实现，需要对合约虚拟机改造，获取交易或者区块的时间戳）；
- 校验参数中业务类型是否符合规范，否则返回提示信息；
- 校验元交易用户信息表中from对应的该业务类型 nonce 值与传参 nonce 值参数是否相等，不是则返回提示信息，校验通过后为对应 nonce 值加 1；
- 校验通过后根据业务类型调用 transfer_721 或 transfer_1155 方法。

4.5.3.7 元交易批量转移

平台方通过调用该方法进行 DDC 元交易的批量转移。

- 输入参数：调用者账户名（平台方），拥有者的账户名（终端用户），接受者的账户名，ddc 唯一标识集合，数量集合，附加数据，业务类型，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metatransbat；
- 方法举例：metatransbat (name sender, name from, name to, std::vector<uint64_t> ddc_ids, std::vector<uint64_t> amounts, std::string memo, uint64_t business_type, uint64_t nonce, uint64_t deadline, signature signature);
- 核心逻辑：
 - 校验调用者账户是否为平台方，不是则返回提示信息；
 - 检查根据 nonce 值和过期时间以及预设的域名分隔符和授权哈希类型所对应的授权哈希值计算出哈希值，并根据哈希值和签名值计算得出公钥，校验计算得出的公钥与元交易用户信息表中 from 对应的公钥是否相等，不相等则返回提示信息；
 - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；（后期实现，需要对合约虚拟机改造，获取交易或者区块的时间戳）；

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

- 校验参数中业务类型是否符合规范，否则返回提示信息；
- 校验元交易用户信息表中 from 对应的该业务类型 nonce 值与传参 nonce 值参数是否相等，不是则返回提示信息，校验通过后为对应 nonce 值加 1；
- 校验通过后根据业务类型调用 batchtrans_721 或 batchtrans_1155 方法。

4.5.3.8 元交易销毁

平台方通过调用该方法进行 DDC 元交易的销毁。

- 输入参数：调用者账户名（平台方），拥有者（终端用户），ddc 唯一标识，业务类型, nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaburn；
- 方法举例：metaburn (name sender, name owner, uint64_t ddc_id, uint64_t business_type, uint64_t nonce, uint64_t deadline, signature signature)
- 核心逻辑：

- 校验调用者账户是否为平台方，不是则返回提示信息；
- 根据 nonce 值和过期时间以及预设的域名分隔符和授权哈希类型所对应的授权哈希值计算出哈希值，并根据哈希值和签名值计算出公钥，校验计算得出的公钥与元交易用户信息表中 owner 对应的公钥是否相等，不相等则返回提示信息；
- 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；（后期实现，需要对合约虚拟机改造，获取交易或者区块的时间戳）；
- 校验参数中业务类型是否符合规范，否则返回提示信息；
- 校验元交易用户信息表中from对应的该业务类型 nonce 值与传参 nonce 值参数是否相等，不是则返回提示信息，校验通过后为对应 nonce 值加 1；
- 校验通过后根据业务类型调用 burn_721 或 burn_1155 方法。

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, (中文)

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

4.5.3.9 元交易批量销毁

平台方通过调用该方法进行 DDC 元交易的批量销毁。

➤ 输入参数：调用者账户名（平台方），拥有者（终端用户），ddc 唯一标识的集合，业务类型, nonce 值，过期时间，签名值；

➤ 输出参数：

➤ 方法命名：metaburnbatc；

➤ 方法举例：metaburnbatc (name sender, name owner, std::vector<uint64_t> ddc_ids, uint64_t business_type, uint64_t nonce, uint64_t deadline, signature signature);

➤ 核心逻辑：

■ 校验调用者账户是否为平台方，不是则返回提示信息；

■ 根据 nonce 值和过期时间以及预设的域名分隔符和授权哈希类型所对应的授权哈希值计算出哈希值，并根据哈希值和签名值计算出公钥，校验计算得出的公钥与元交易用户信息表中 owner 对应的公钥是否相等，不相等则返回提示信息；

■ 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；（后期实现，需要对合约虚拟机改造，获取交易或者区块的时间戳）；

■ 校验参数中业务类型是否符合规范，否则返回提示信息；

■ 校验元交易用户信息表中from对应的该业务类型 nonce 值与传参 nonce 值参数是否相等，不是则返回提示信息，校验通过后为对应 nonce 值加 1；

■ 校验通过后根据业务类型调用 burnbatch_721 或 burnbatch_1155 方法。

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light, (中文)

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light

设置格式[1]: 字体: (默认)等线 Light, (中文)等线 Light