DDC-SDK EOS

详细设计

文档信息

项目名称：BSN-DDC	项目编号:
项目负责人:	所属部门:
编 制 人：鲍宏飞	编制时间:
审 核 人：谢兆颜	审核时间:
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
2021.12.17	V1.0	初版编辑	鲍宏飞
2021.12.20	V1.1	1、管理费充值 api，修改接收方的账户类型为 eos 账户； 2、运营账户充值，增加了运营账户字段； 3、删除计费规则，增加了运营账户字段，业务类型字段，方法名字段。	鲍宏飞
2021.12.22	V1.2	1、补充区块查询功能介绍	鲍宏飞
2021.12.23	V1.3	1、补充签名事件&数据解析	鲍宏飞
2021.12.23	V1.4	1、补充出参字段名和字段	鲍宏飞
2021.12.31	V1.5	1、补充 SDK 使用说明； 2、附录补充区块信息示例和交易信息示例。	鲍宏飞
2021.1.14	V1.6	1、修改了错误的 sdk 接口； 2、修改了调用合约接口的合约接口名和合约入参。	鲍宏飞
2021.1.15	V1.7	1、修改销毁 api 描述	鲍宏飞
2021.1.20	V1.8	1、修改部分接口描述和事件数据解析 2、增加修改 URI 方法 3、增加跨平台授权的方法	鲍宏飞
2021.1.24	V1.9	1、增加符号名称设置方法 2、更改 SDK 初始化配置方法名	鲍宏飞
2022.3.16	V1.10	1、上链操作类接口返回交易 hash 2、费用管理添加了批量充值以及批量链账户余额查询方法。 3、权限管理添加了平台方批量添加账	张宇

		户、平台方添加链账户开关设置、平台方添加链账户开关查询、运营方批量添加链账户方法。 4、721 添加了 DDC 批量授权、批量转移、批量销毁、批量查询数量、最新 DDCID 查询。 5、1155 添加了最新 DDCID 查询。 6、SDK 接口增加了入参校验。 补充了新增接口的事件解析	
2022.3.25	V1.11	1、721 的生成、批量生成、安全生成、批量安全生成以及 1155 的安全生成以及批量安全生成去掉了对 DDCURI 参数检验 2、修改 1155 批量查询方法	张宇
2022.4.1	V1.12	1.修改计费合约充值接口入参校验方法 2.修改上链操作的方法返回值	张宇
2022.6.7	V1.13	1、新增查询指定账户 721 资产 2、新增查询指定账户 1155 资产	张宇
2022.6.20	V1.14	1 修改所有上链方法可通过 try catch 捕捉到异常信息; 2.抛出区块解析 getBlockInfo 的异常; 3.修改 permission 为可配置项; 4.readme 增加混淆说明 5.修改 PushedTransaction 类，新增 Processed 、 ActionTrace 、 Receipt 、 AccountRamDelta 等类用于接收返回的链上最新数据; 6.修改所有上链方法在上链之后同步返回结果（交易哈希，上链之后的链上最新数据）; 7.新增可配置项 EOS 区块解析链地址; 8.修改文档与 SDK 不一致描述; 9.在 SDK 中提供所有的合约方法名常量;	张宇
2022.6.22	V1.15	1.721 注释掉批量生成方法、批量转移方法、批量销毁方法、批量查询数量方法、批量查询拥有者方法、查询账户 721 资产方法; 2.1155 注释掉设置名称和符号方法、查询账户 1155 资产方法;	张宇
2022.8.4	V1.16	1. 721 的 mint 方法做 sdk 重载，增加 memo 字段 2. 1155 的 mint 方法做 sdk 重载，增加 memo 字段	张宇

		3. 721 的 transfer 方法做 sdk 重载，增加 memo 字段 4. 1155 的 transfer 方法做 sdk 重载，增加 memo 字段 5. 1155 的 batchtrans 方法做 sdk 重载，增加 memo 字段 6.充值(rechage)方法做 sdk 重载，增加 memo 字段 7.批量充值(rechargebat)方法做 sdk 重载，增加 memo 字段	
2022.11.11	V1.17	新增元交易相关方法: 1、授权哈希设置 2、分隔符设置 3、添加终端用户公钥 4、元交易生成 5、元交易批量生成 6、元交易转移 7、元交易批量转移 8、元交易销毁 9、元交易批量销毁	曹理辉
2022.12.05	V1.18	1、授权哈希值支持离线生成 2、域名分隔符支持离线生成	曹理辉

目录

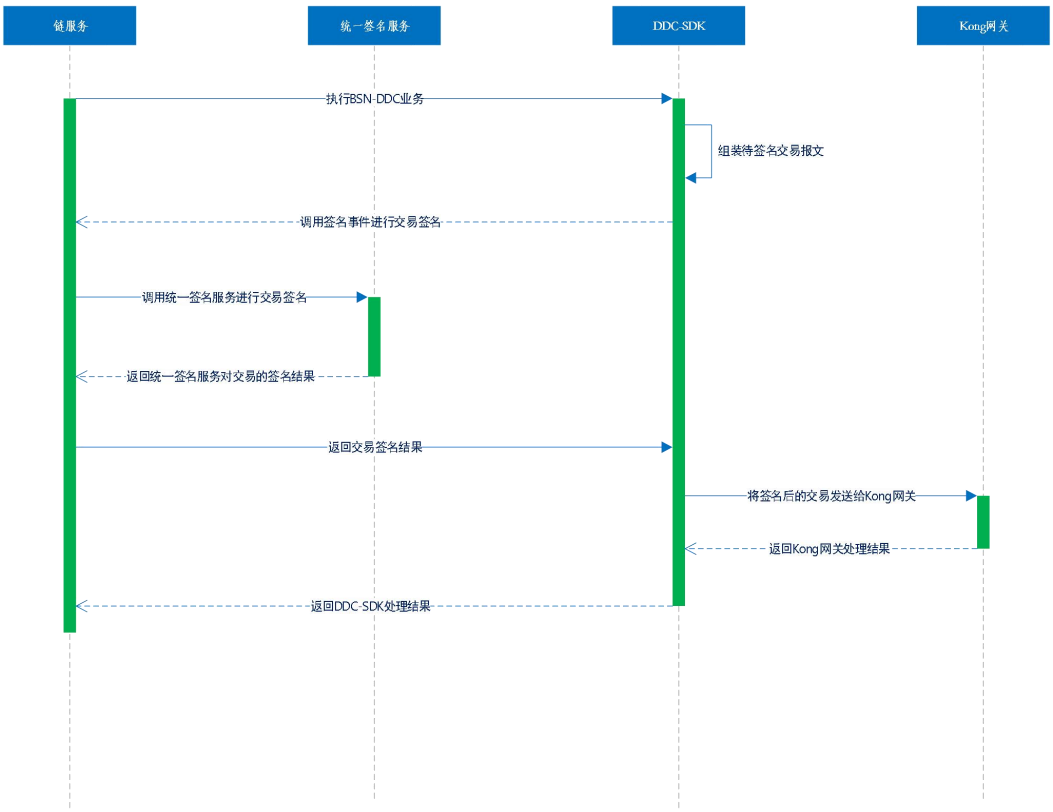
文档信息 2
修改记录 2
1 编写目的 6
2 整体设计 7
2.1 调用时序图 7
2.2 开发语言标准 7
2.3 参数格式标准 7
3 功能设计 8
3.1 DDC 8
3.1.1 BSN-DDC-权限管理 8
3.1.2 BSN-DDC-费用管理 24
3.1.3 BSN-DDC-721 33
3.1.4 BSN-DDC-1155 59
3.1.5 BSN-DDC-区块查询 85
3.1.6 BSN-DDC-签名事件 86
3.1.7 BSN-DDC-数据解析 86
4 SDK 使用说明 114
4.1 初始化配置 114
4.1.1 初始化 ChainConfig 类 114
4.1.2 初始化 MetaTrxConfig 类 114
4.2 API 调用 114
4.2.1 BSN-DDC-权限管理 114
4.2.2 BSN-DDC-费用管理 121
4.2.3 BSN-DDC-721 124
4.2.4 BSN-DDC-1155 132
5 附录 142
5.1 区块信息示例 142
5.2 交易信息示例 142

1 编写目的

为了让运营方或各平台方对 DDC-SDK 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 DDC-SDK 整体设计规范方案说明文档。在《BSN-DDC EOS 合约详细设计》的基础上，通过 Java SDK 实现对合约的远程调用。

2 整体设计

2.1 调用时序图



2.2 开发语言标准

目前使用 Java 语言开发 SDK。

2.3 参数格式标准

- 时间

格式为 yyyy-MM-dd HH:mm:ss 形式的字符串，例如: 2021-05-25 12:30:59 表示 2021 年 5 月 25 日 12 时 30 分 59 秒。

- 返回异常

当 SDK 处理功能逻辑出错时，会抛出相应的运行时异常，包含具体的错误信息。

3 功能设计

3.1 DDC

所有的 BSN-DDC 方法都需要调用签名事件对待签名交易进行签名，签名事件必须由业务调用者服务进行注册并实现交易签名的业务逻辑。

3.1.1 BSN-DDC-权限管理

对账户进行管理，包含了账户的添加、删除、查询及更新账户状态的操作。

3.1.1.1 添加运营账户

3.1.1.1.1 功能介绍

运营方通过调用该方法进行 DDC 运营账户信息的创建。

3.1.1.1.2 API 定义

- 方法定义：PushedTransaction addOperator(String operatorName, String accountName, String accountDID);
- EOS 合约方法：addoperator(name operator_name, std::string account_name, std::string account_did);
- 调用者：合约拥有者;
- 核心逻辑：

  1、验证 operatorName 为标准 eos name 格式，并且不是空地址;

  2、验证 accountName 不为空;
- 所在类库：com.bsn.eos.service.DDCPermissionService#addOperator
- 输入参数：

字段名	字段	类型	必传	备注

调用者	operatorName	String	是	调用者地址
账户名称	accountName	String	是	DDC 运营账户名称
账户 DID	accountDID	String	是	DDC 运营账户对应的 DID 信息

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.1

3.1.1.2 运营方添加平台方、终端账户

3.1.1.2.1 功能介绍

运营方可以通过调用该方法直接对平台方或平台方的终端用户进行创建。

3.1.1.2.2 API 定义

- 方法定义：PushedTransaction addAccountByOperator(String sender, String account, String accountName, String accountDID, String leaderDID);
- EOS 合约方法：operatoradd(name sender, name account, std::string account_name, std::string account_did, std::string leader_did);
- 调用者：运营方；
- 核心逻辑：

  1.检查 sender 为标准 eos name 格式，并且不是空地址；

  2.检查 account 为标准 eos name 格式，并且不是空地址；

  3.检查 accountName 不为空；
- 所在类库：com.bsn.eos.service.DDCPermissionService# addAccountByOperator

- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
账户地址	account	String	是	DDC 链账户地址
账户名称	accountName	String	是	DDC 账户对应的账户名称
账户 DID	accountDID	String	否	DDC 账户对应的 DID 信息
上级 DID	leaderDID	String	是	该普通账户对应的上级账户的 DID

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.2

3.1.1.3 运营方批量添加平台方、终端用户

3.1.1.3.1 功能介绍

运营方可以通过调用该方法直接对平台方或平台方的终端用户进行批量创建。

3.1.1.3.2 API 定义

- 方法定义：PushedTransaction addBatchAccountByOperator(String sender, List<String> accounts, List<String> accountNames, List<String> accountDIDs, List<String> leaderDIDs);
- EOS 合约方法：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);
- 调用者：运营方；

- 核心逻辑：

  1.检查 sender 为标准 eos name 格式，并且不是空地址；

  2.检查 accounts 集合大小必须大于 0；

  3.循环 accounts 集合，并根据索引检查 account 为标准 eos 账户格式，并且是否为活跃账户以及 accountName 不为空；

- 所在类库：com.bsn.eos.service.DDCPermissionService#addBatchAccountByOperator
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
账户地址列表	accounts	List<String>	是	DDC 链账户地址列表
账户名称列表	accountNames	List<String>	是	DDC 账户对应的账户名称列表
账户 DID 列表	accountDIDs	List<String>	否	DDC 账户对应的 DID 信息列表
平台方 DID 列表	leaderDIDs	List<String>	是	该普通账户对应的上级账户的 DID 列表

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.3

3.1.1.4 平台方添加终端账户

3.1.1.4.1 功能介绍

平台方可以通过调用该方法进行 DDC 账户信息的创建，上级角色可进行下级角色账户的操作，平台方通过该方法只能添加终端账户。

3.1.1.4.2 API 定义

- 方法定义：PushedTransaction addAccountByPlatform(String sender, String account, String accountName, String accountDID);
- EOS 合约方法：manageradd(name sender, name account, std::string account_name, std::string account_did);
- 调用者：平台方；
- 核心逻辑：

  1.检查 sender 为标准 eos name 格式，并且不是空地址；

  2.检查 account 为标准 eos name 格式，并且不是空地址；

  3.检查 accountName 不为空；
- 所在类库：com.bsn.eos.service.DDCPermissionService#addAccountByPlatform
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
账户地址	account	String	是	DDC 链账户地址
账户名称	accountName	String	是	DDC 账户对应的账户名称
账户 DID	accountDID	String	否	DDC 账户对应的 DID 信息

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.4

3.1.1.5　平台方批量添加终端账户

3.1.1.5.1 功能介绍

平台方可以通过调用该方法进行 DDC 链账户信息的批量创建，上级角色可进行下级角色账户的操作，平台方通过该方法只能添加终端账户。

3.1.1.5.2 API 定义

- 方法定义：PushedTransaction addBatchAccountByPlatform(String sender, List<String> accounts, List<String> accountNames, List<String> accountDIDs);
- EOS 合约方法：managaddbat(name sender, std::vector<name> accounts, std::vector<std::string> account_names, std::vector<std::string> account_dids);
- 调用者：平台方；
- 核心逻辑：

  1.检查 sender 为标准 eos name 格式，并且不是空地址；

  2.检查 accounts 集合大小必须大于 0；

  3.循环 accounts 集合，并根据索引检查 account 为标准 eos 账户格式，并且是否为活跃账户以及 accountName 不为空；
- 所在类库：com.bsn.eos.service.DDCPermissionService#addBatchAccountByPlatform
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
账户地址列表	accounts	List<String>	是	DDC 链账户地址列表
账户名称列表	accountNames	List<String>	是	DDC 账户对应的账户名称列表
账户 DID 列表	accountDIDs	List<String>	否	DDC 账户对应的 DID 信息列表

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.5

3.1.1.6 查询账户

3.1.1.6.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法进行 DDC 账户信息的查询。

3.1.1.6.2 API 定义

- 方法定义：AccountInfo getAccount(String account);
- EOS 合约方法：get_table_rows(name contract, name table);
- 调用者：运营方、平台方以及终端用户；
- 核心逻辑：

  1.检查 account 为标准 eos name 格式，并且不是空地址；
- 所在类库：com.bsn.eos.service.DDCPermissionService#getAccount
- 输入参数：

字段名	字段	类型	必传	备注
账户地址	account	String	是	DDC 用户链账户地址

- 输出参数：

字段名	字段	类型	必传	备注
账户 DID	accountDID	String	否	DDC 账户对应的 DID 信息（普通用户可为空）
账户名称	accountName	String	是	DDC 账户对应的账户名称
账户角色	accountRole	enum	是	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）。
账户上级管理者	leaderDID	String	是	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
平台管理账户状态	platformState	enum	是	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）。
运营管理账户状态	operatorState	enum	是	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）； 2.Active（活跃状态，可进行 DDC 相关操作）。
冗余字段	field	String	否	冗余字段

- 调用实例：见 4.2.1.6

3.1.1.7 更新账户状态

3.1.1.7.1 功能介绍

运营方或平台方通过调用该方法对终端用户进行 DDC 账户信息状态的更改。

3.1.1.7.2 API 定义

- 方法定义：PushedTransaction updateAccState(String sender, String account, AccountStateEnum state, boolean changePlatformState);

- EOS 合约方法：updateaccsta(name sender, name account, PlatformState state, bool change_platform_state);
- 调用者：运营方、平台方；
- 核心逻辑：

  1.检查 sender 为标准 eos name 格式，并且不是空地址；

  2.检查 account 为标准 eos name 格式，并且不是空地址；
- 所在类库：com.bsn.eos.service.DDCPermissionService#updateAccState
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
账户地址	account	String	是	DDC 用户链账户地址
状态	state	enum	是	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）
修改平台方状态	changePlatformState	Boolean	是	是否修改平台方状态(仅对运营方生效)

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.7

3.1.1.8 添加方法

3.1.1.8.1 功能介绍

运营方通过调用该方法添加 DDC 方法的访问权限。

3.1.1.8.2 API 定义

- 方法定义：PushedTransaction addFunction(String sender, RoleEnum accountRole, BusinessTypeEnum businessType, String funcName);
- EOS 合约方法：addfunction(name sender, AccountRole account_role, BusinessType business_type, name func_name);
- 调用者：运营方；
- 核心逻辑：
  1. 验证 accoun_role 为具有相关权限的角色；
  2. 验证 business_type 为可用类型，且不能为空；
  3. 添加方法；
- 所在类库：com.bsn.eos.service.DDCPermissionService#addFunction；
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
DDC 账户角色	accountRole	enum	是	添加方法的对应角色
业务类型	businessType	enum	是	1 表示 ERC-721 2 表示 ERC-1155
方法名	funcName	String	是	要添加的方法名

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.8

3.1.1.9 删除方法

3.1.1.9.1 功能介绍

运营方通过调用该方法删除 DDC 方法的访问权限。

3.1.1.9.2 API 定义

- 方法定义：PushedTransaction delFunction(String sender, RoleEnum accountRole, BusinessTypeEnum businessType, String funcName);
- EOS 合约方法：delfunction(name sender, AccountRole account_role, BusinessType business_type, name func_name);
- 调用者：运营方；
- 核心逻辑：
  1. 验证 accoun_role 为具有相关权限的角色；
  2. 验证 business_type 为可用类型，且不能为空；
  3. 删除方法；
- 所在类库：com.bsn.eos.service.DDCPermissionService#delFunction
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
DDC 账户角色	accountRole	enum	是	删除方法的对应角色
业务类型	businessType	enum	是	1 表示 ERC-721 2 表示 ERC-1155
方法名	funcName	String	是	要删除的方法名

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.9

3.1.1.10 跨平台授权

3.1.1.10.1 功能介绍

运营方可以通过调用该方法对 DDC 的跨平台操作进行授权。

3.1.1.10.2 API 定义

- 方法定义：PushedTransaction crossAppr(String sender, String from, String to, boolean approved);
- EOS 合约方法：crossappr(name sender, name from, name to, bool approved);
- 调用者：运营方；
- 核心逻辑：
  1. 检查 sender 为标准 eos name 格式，并且不是空地址；
  2. 检查 from 为标准 eos name 格式，并且不是空地址；
  3. 检查 to 为标准 eos name 格式，并且不是空地址；
- 所在类库：com.bsn.eos.service.DDCPermissionService#crossAppr
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
授权者	from	String	是	授权账户
接收者	to	String	是	被授权账户
授权标识	approved	boolen	是	是否授权标识

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.10

3.1.1.11 平台方添加链账户开关设置

3.1.1.11.1 功能介绍

运营方可以通过调用该方法设置平台方添加链账户开关。

3.1.1.11.2 API 定义

- 方法定义：PushedTransaction setSwitcherStateOfPlatform(String sender,boolean isOpen);
- EOS 合约方法：setswitcher(name sender, bool is_platform_open);
- 调用者：运营方；
- 核心逻辑：

1. 检查 sender 为标准 eos name 格式，并且不是空地址；

- 所在类库：com.bsn.eos.service.DDCPermissionService#setSwitcherStateOfPlatform
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
开关标识	isOpen	Boolean	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.11

3.1.1.12 平台方添加链账户开关查询

3.1.1.12.1 功能介绍

运营方可以通过调用该方法查询平台方添加链账户开关状态。

3.1.1.12.2 API 定义

- 方法定义：Boolean switcherStateOfPlatform();
- EOS 合约方法：is_platform_switcher_on();
- 调用者：运营方；
- 核心逻辑：无；
- 所在类库：com.bsn.eos.service.DDCPermissionService#switcherStateOfPlatform
- 输入参数：

字段名	字段	类型	必传	备注

- 输出参数：

字段名	字段	类型	必传	备注
开关状态		Boolean	是	

- 调用实例：见 4.2.1.12

3.1.1.13 授权哈希设置(元交易)

3.1.1.13.1 功能介绍

合约拥有者可以对元交易的授权类型哈希进行初始化。

3.1.1.13.2 API 定义

- 方法定义：PushedTransaction setMetaTHash(HashTypeEnum hashType, String func);
- EOS 合约方法：setmetathash (uint8_t hashType, string hashValue);
- 调用者：合约部署方；
- 核心逻辑：

1、校验调用者账户是否为合约部署方，不是则返回提示信息；

2、检查授权哈希值长度是否为 0，是则返回提示信息；

3、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；

- 所在类库：com.bsn.eos.service.DDCPermissionService#setMetaTHash
- 输入参数：

字段名	字段	类型	必传	备注
授权哈希类型	hashType	HashTypeEnum	是	MINT(1), MINTBATCH(2), TRANSFER(3), BATCHTRANS(4), BURN(5), BURNBATCH(6),
方法名及方法参数	func	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.13

3.1.1.14 分隔符设置(元交易)

3.1.1.14.1 功能介绍

合约拥有者可以对元交易的域名分隔符进行设置。

3.1.1.14.2 API 定义

- 方法定义：PushedTransaction setMetaSeparator(BusinessTypeEnum businessType, String ddcName, String chainId, String contractAddress);
- EOS 合约方法：setmetasepar (uint64_t business_type ,string separator);
- 调用者：合约部署方；
- 核心逻辑：

  1)校验调用者账户是否为合约部署方，不是则返回提示信息；
  2)检查域名分隔符长度是否为 0，是则返回提示信息；
  3)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；
- 所在类库：com.bsn.eos.service.DDCPermissionService#setMetaSeparator

- 输入参数：

字段名	字段	类型	必传	备注
业务类型	businessType	BusinessTypeEnum	是	ERC721(1), ERC1155(2);
DDC 名称	ddcName	String	是	
EOS 链 id	chainId	String	是	
DDC 合约部署账户	contractAddress	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.14

3.1.1.15 添加终端用户公钥(元交易)

3.1.1.15.1 功能介绍

运营方或平台方可以将终端用户的公钥存入元交易用户信息表。

3.1.1.15.2 API 定义

- 方法定义：PushedTransaction addMetaUser(String sender, String account, String pubKey);
- EOS 合约方法：addmetauser (name sender, name account, string pubKey);
- 调用者：运营方或平台方；
- 核心逻辑：

1)检查调用者账户信息是否存在，不存在则返回提示信息；
2)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；
3)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；
4)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；
5)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；
6)检查添加的账户名称长度是否为 0，是则返回提示信息；

7)检查添加的公钥长度是否为 0，是则返回提示信息；
8)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。

- 所在类库：com.bsn.eos.service.DDCPermissionService#addMetaUser
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
终端用户账户	account	String	是	
终端用户的公钥	pubKey	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.1.15

3.1.2 BSN-DDC-费用管理

3.1.2.1 充值

3.1.2.1.1 功能介绍

运营方或平台方可以通过调用该方法为所属同一方的同一级别账户或者下级账户进行充值；

3.1.2.1.2 API 定义

- 方法定义：

- PushedTransaction recharge(String sender,String to,String amount);
- PushedTransaction recharge(String sender,String to,String amount,String memo);

- EOS 合约方法：

- recharge(name from, name to, asset value);
- recharge2(name from, name to, asset value, string memo);

➢ 调用者：运营方、平台方；

➢ 核心逻辑：

1. 检查 sender 为标准 eos name 格式，并且不是空地址；
2. 检查 to 为标准 eos name 格式，并且不是空地址；
3. 检查 amount 必须大于零；

➢ 所在类库：com.bsn.eos.service.DDCFeeService#recharge

➢ 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
用户账户地址	to	String	是	充值账户的地址
金额	amount	String	是	转移金额
附加数据	memo	String	否	

➢ 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

➢ 调用实例：见 4.2.2.1

3.1.2.2 批量充值

3.1.2.2.1 功能介绍

运营方或平台方可以通过调用该方法为所属同一方的同一级别账户或者下级账户进行批量充值；

3.1.2.2.2 API 定义

➢ 方法定义：

- PushedTransaction rechargeBatch(String sender, List<String> toList, List<String> valueList);
- PushedTransaction rechargeBatch(String sender, List<String> toList, List<String> valueList, String memo);

➢ EOS 合约方法：

- rechargebat(name from, std::vector<name> to_list, std::vector<asset> value_list);
- rechargebat(name from, std::vector<name> to_list, std::vector<asset> value_list, string memo);

➢ 调用者：运营方、平台方；

➢ 核心逻辑：

1.检查 sender 为标准 eos name 格式，并且不是空地址；

2. 检查 toList 集合大小必须大于 0；

3. 循环 toList 集合，并根据索引检查 account 为标准 eos name 格式，并且不是空地址以及金额必须大于 0；

➢ 所在类库：com.bsn.eos.service.DDCFeeService#rechargebat

➢ 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
账户充值列表	toList	List<String>	是	充值账户的地址列表
金额列表	valueList	BigInteger	是	转移金额列表
附加数据	memo	String	否	

➢ 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

➢ 调用实例：见 4.2.2.2

3.1.2.3 链账户余额查询

3.1.2.3.1 功能介绍

运营方、平台方或终端用户可以调用该方法查询指定账户的余额。

3.1.2.3.2 API 定义

- 方法定义：String balanceOf(String account);
- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：运营方、平台方或终端用户;
- 核心逻辑：
  1. 检查 account 为标准 eos name 格式，并且不是空地址;
- 所在类库：com.bsn.eos.service.DDCFeeService#balanceOf
- 输入参数：

字段名	字段	类型	必传	备注
账户地址	account	String	是	查询的账户地址

- 输出参数：

字段名	字段	类型	必传	备注
业务费余额	amount	String	是	账户所对应的业务费余额

- 调用实例：见 4.2.2.3

3.1.2.4 批量链账户余额查询

3.1.2.4.1 功能介绍

运营方、平台方或终端用户可以调用方法批量查询账户的余额。

3.1.2.4.2 API 定义

- 方法定义：List<String> balanceOfBatch(List<String> accounts);

- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：运营方、平台方或终端用户；
- 核心逻辑：

  1.检查 accounts 集合大小必须大于 0；

  2.循环 accounts 集合，并根据索引检查 account 为标准标准 eos name 格式，并且不是空地址；
- 所在类库：com.bsn.eos.service.DDCFeeService#balanceOfBatch
- 输入参数：

字段名	字段	类型	必传	备注
账户地址列表	accounts	List<String>	是	查询的账户地址列表

- 输出参数：

字段名	字段	类型	必传	备注
业务费余额列表	amounts	List<String>	是	账户所对应的业务费余额

- 调用实例：见 4.2.2.4

3.1.2.5 DDC 计费规则查询

3.1.2.5.1 功能介绍

运营方、平台方或终端用户可以通过调用该方法查询指定的 DDC 业务合约的方法所对应的调用业务费用。

3.1.2.5.2 API 定义

- 方法定义：String queryFee(String sender,BusinessTypeEnum businessType,String funcName);
- EOS 合约方法：get_table_rows(name contract, name table);
- 调用者：运营方、平台方，终端用户；
- 核心逻辑：

  1、验证 sender 为标准 eos name 格式，并且不是空地址；
- 所在类库：com.bsn.eos.service.DDCFeeService#queryFee

- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
业务类型	businessType	enum	是	1 表示 ERC-721 2 表示 ERC-1155
方法名	funcName	String	是	EOS 合约方法

- 输出参数：

字段名	字段	类型	必传	备注
业务费	amount	BigInteger	是	查询的 DDC 合约业务费

- 调用实例：见 4.2.2.7

3.1.2.6 运营账户充值

3.1.2.6.1 功能介绍

运营方可以通过调用该方法为自己的账户增加业务费。

3.1.2.6.2 API 定义

- 方法定义：PushedTransaction selfRecharge(String sender,String amount);
- EOS 合约方法：selfrecharge(name sender, asset value);
- 调用者：运营方；
- 核心逻辑：
  1. 验证 sender 为标准 eos name 格式，并且不是空地址；
  2. 验证 amount 必须大于零；
- 所在类库：com.bsn.eos.service.DDCFeeService#selfRecharge
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址

业务费	amount	String	是	对运营方账户进行充值的业务费

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.2.5

3.1.2.7 设置 DDC 计费规则

3.1.2.7.1 功能介绍

运营方可以通过调用该方法设置指定的 DDC 主合约的方法调用费用。

3.1.2.7.2 API 定义

- 方法定义：PushedTransaction setFee(String sender, BusinessTypeEnum businessType, String funcName, String amount)
- EOS 合约方法：setfee(name sender, BusinessType business_type, name func_name, asset value);
- 调用者：运营方；
- 核心逻辑：

  1. 验证 sender 为标准 eos name 格式，并且不是空地址；
  2. 验证 amount 必须大于等于零；

- 所在类库：com.bsn.eos.service.DDCFeeService#setFee
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
业务类型	businessType	enum	是	1 表示 ERC-721 2 表示 ERC-1155

方法名	funcName	String	是	EOS 合约方法
业务费用	amount	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.2.6

3.1.2.8 删除 DDC 计费规则

3.1.2.8.1 功能介绍

运营方可以通过调用该方法删除指定的 DDC 主合约的方法调用费用。

3.1.2.8.2 API 定义

- 方法定义：PushedTransaction delFee(String sender, BusinessTypeEnum businessType, String funcName);
- EOS 合约方法：deletefee(name sender, BusinessType business_type, name func_name);
- 调用者：运营方；
- 核心逻辑：验证 sender 为标准 eos name 格式，并且不是空地址；
- 所在类库：com.bsn.eos.service.DDCFeeService#delFee
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
业务类型	businessType	enum	是	1 表示 ERC-721 2 表示 ERC-1155
方法名	funcName	String	是	EOS 合约方法

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.2.8

3.1.2.9 按合约删除 DDC 计费规则

3.1.2.9.1 功能介绍

运营方可以通过调用该方法删除指定的 DDC 业务合约授权。

3.1.2.9.2 API 定义

- 方法定义：PushedTransaction delDDC(String sender, BusinessTypeEnum businessType);
- EOS 合约方法：deleteddc(name sender, BusinessType business_type);
- 调用者：运营方；
- 核心逻辑：验证 sender 为标准 eos name 格式，并且不是空地址；
- 所在类库：com.bsn.eos.service.DDCFeeService#delDDC
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
业务类型	businessType	enum	是	1 表示 ERC-721 2 表示 ERC-1155

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.2.9

3.1.3 BSN-DDC-721

3.1.3.1 生成

3.1.3.1.1 功能介绍

平台方或终端用户可以通过调用该方法进行 DDC 的生成。

3.1.3.1.2 API 定义

- 方法定义：
- PushedTransaction safeMint(String from， String to, String ddcURI);
- PushedTransaction safeMint(String from， String to, String ddcURI, String memo);
- EOS 合约方法： mint(name sender, name to, uint64_t amount, std::string ddc_uri, uint64_t business_type, std::string memo);
- 调用者： 平台方、终端用户;
- 核心逻辑：

  1. 检查发送方、接收者账户地址信息是否为空;
  2. 检查发送方、接收者账户地址格式是否正确;

- 所在类库： com.bsn.eos.service.DDC721Service#safeMint
- 输入参数：

字段名	字段	类型	必传	备注
调用者	From	String	是	调用者地址
接收者账户	to	String	是	
DDC 资源标识符	ddcURI	String	是	
附加数据	memo	String	否	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	

执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.1

3.1.3.2 批量生成(暂不提供)

3.1.3.2.1 功能介绍

平台方或终端用户可以通过调用该方法进行 DDC 的批量生成。

3.1.3.2.2 API 定义

- 方法定义：PushedTransaction safeMintBatch(String from, String to, List<String> ddcURIs);
- EOS 合约方法：mintbatch(name sender, name to, std::vector<uint64_t> amounts, std::vector<std::string> ddc_uris, uint64_t business_type, std::string memo);
- 调用者：平台方、终端用户;
- 核心逻辑：

  1.检查 sender 为标准 eos name 格式，并且不是空地址;

  2.检查 to 为标准 eos name 格式，并且不是空地址;

  3.检查 ddcURIs 集合大小必须大于 0;
- 所在类库：com.bsn.eos.service.DDC721Service#safeMintBatch
- 输入参数：

字段名	字段	类型	必传	备注
调用者	from	String	是	调用者地址
接收者账户	to	String	是	
DDC 资源标识符	ddcURIs	List<String>	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.2

3.1.3.3 DDC 授权

3.1.3.3.1 功能介绍

DDC 拥有者可以通过调用该方法对 DDC 进行授权，发起者需要是 DDC 的拥有者。

3.1.3.3.2 API 定义

- 方法定义：PushedTransaction approve(String sender,String to,BigInteger ddcId);
- EOS 合约方法：approve(name sender, name to, uint64_t ddc_id, uint64_t business_type);
- 调用者：DDC 拥有者；
- 核心逻辑：
  1. 检查授权者账户地址信息是否为空；
  2. 检查授权者账户地址格式是否正确；
  3. 检查 ddcId 是否大于 0；
- 所在类库：com.bsn.eos.service.DDC721Service#approve
- 输入参数：

字段名	字段	类型	必传	备注
调用者	Sender	String	是	调用者地址
被授权账户	to	String	是	
DDC 唯一标识	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.10

3.1.3.4 DDC 批量授权(暂不提供)

3.1.3.4.1 功能介绍

DDC 拥有者可以通过调用该方法对 DDC 进行批量授权，发起者需要是 DDC 的拥有者。

3.1.3.4.2 API 定义

- 方法定义：PushedTransaction approveBatch(String sender, String to, List<BigInteger> ddclds);
- EOS 合约方法：approvebatch(name sender, name to, std::vector<uint64_t> ddc_ids, uint64_t business_type);
- 调用者：DDC 拥有者；
- 核心逻辑：

  1.检查 sender 为标准 eos name 格式，并且不是空地址；

  2.检查 to 为标准 eos name 格式，并且不是空地址；

  3.检查 ddclds 集合大小必须大于 0；

  4.循环 list 集合，并根据索引检查对应的 DDCID 必须大于 0；
- 所在类库：com.bsn.eos.service.DDC721Service#approveBatch
- 输入参数：

字段名	字段	类型	必传	备注
调用者	Sender	String	是	调用者地址
被授权账户	to	String	是	
DDC 唯一标识	ddclds	List<BigInteger>	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.11

3.1.3.5 DDC 授权查询

3.1.3.5.1 功能介绍

运营方、平台方或终端用户可以通过调用该方法对 DDC 进行授权查询。

3.1.3.5.2 API 定义

- 方法定义：Boolean getApproved(String sender,String to,BigInteger ddcId);
- EOS 合约方法：get_table_rows(name contract,uint64_t primary, uint64 ddc_id);
- 调用者：运营方、平台方或终端用户;
- 核心逻辑：检查 ddcId 的值是否大于 0;
- 所在类库：com.bsn.eos.service.DDC721Service#getApproved
- 输入参数：

字段名	字段	类型	必传	备注
调用者	Sender	String	是	调用者地址
被授权账户	to	String	是	
DDC 唯一标识	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
		Boolean	是	是否授权

- 调用实例：见 4.2.3.10

3.1.3.6 账户授权

3.1.3.6.1 功能介绍

DDC 拥有者可以通过调用该方法进行账户授权，发起者需要是 DDC 的拥有者。

3.1.3.6.2 API 定义

- 方法定义：PushedTransaction setApprovalForAll(String sender,String to,bool approved);
- EOS 合约方法：approvalall(name sender, name to, bool approved, uint64_t business_type);
- 调用者：DDC 拥有者;
- 核心逻辑：

  1、检查授权者账户地址信息是否为空;

  2、检查授权者账户地址格式是否正确;
- 所在类库：com.bsn.eos.service.DDC721Service#setApprovalForAll
- 输入参数：

字段名	字段	类型	必传	备注
调用者	Sender	String	是	调用者地址
被授权账户	to	String	是	
授权标识	approved	Boolean	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.12

3.1.3.7 账户授权查询

3.1.3.7.1 功能介绍

运营方、平台方或终端用户可以通过调用该方法进行账户授权查询。

3.1.3.7.2 API 定义

- 方法定义：Boolean isApprovedForAll(String sender,String to);

- EOS 合约方法：get_table_rows(name contract, name table, name account)
- 调用者：所有人；
- 核心逻辑：

  1、检查拥有者账户地址信息是否为空；

  2、检查拥有者账户地址格式是否正确；

  3、检查授权者账户地址信息是否为空；

  4、检查授权者账户地址格式是否正确；
- 所在类库：com.bsn.eos.service.DDC721Service#isApprovedForAll
- 输入参数：

字段名	字段	类型	必传	备注
调用者	Sender	String	是	调用者地址
被授权账户	to	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
授权标识	approved	Boolean	是	

- 调用实例：见 4.2.3.12

3.1.3.8 转移

3.1.3.8.1 功能介绍

DDC 的拥有者或授权者可以通过调用该方法对 DDC 进行安全转移。

3.1.3.8.2 API 定义

- 方法定义：

- PushedTransaction transferFrom(String sender,String from, String to, BigInteger ddcId);
- PushedTransaction transferFrom(String sender,String from, String to, BigInteger ddcId, String memo);

- EOS 合约方法：transfer(name sender, name from, name to, uint64_t ddc_id, uint64_t

amount, std::string memo, uint64_t business_type);

- 调用者：DDC 拥有者、DDC 授权者；
- 核心逻辑：
  1. 检查拥有者账户地址信息是否为空；
  2. 检查拥有者账户地址格式是否正确；
  3. 检查接收者账户地址信息是否为空；
  4. 检查接收者账户地址格式是否正确；
  5. 检查 ddcId 的数值是否大于 0；
- 所在类库：com.bsn.eos.service.DDC721Service#transferFrom
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	from	String	是	
接收者账户	to	String	是	
DDC 唯一标识	ddcId	BigInteger	是	
附加数据	memo	String	否	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.3

3.1.3.9 批量转移(暂不提供)

3.1.3.9.1 功能介绍

DDC 的拥有者或授权者可以通过调用该方法对 DDC 进行批量转移。

3.1.3.9.2 API 定义

- 方法定义：PushedTransaction batchTransferFrom(String sender, String from, String to, List<BigInteger> ddcIds);
- EOS 合约方法：batchtrans(name sender, name from, name to, std::vector<uint64_t> ddc_ids, std::vector<uint64_t> amounts, std::string memo, uint64_t business_type);
- 调用者：DDC 拥有者、DDC 授权者;
- 核心逻辑：
  1. 检查 sender 为标准 address 格式，且不能为 0 地址;
  2. 检查 from 为标准 address 格式，且不能为 0 地址;
  3. 检查 to 为标准 address 格式，且不能为 0 地址;
  4. 检查 ddcIds 集合大小必须大于 0;
  5. 循环 ddcIds 集合，并根据索引检查对应的 DDCID 数值必须大于 0;
- 所在类库：com.bsn.eos.service.DDC721Service#batchTransferFrom
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	from	String	是	
接收者账户	to	String	是	
DDC 唯一标识集合	ddcIds	List<BigInteger>	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.4

3.1.3.10 冻结

3.1.3.10.1 功能介绍

运营方可以通过调用该方法对 DDC 进行冻结。

3.1.3.10.2 API 定义

- 方法定义：PushedTransaction freeze (String sender， BigInteger ddcId);
- EOS 合约方法： freeze(name sender, uint64_t ddc_id, uint64_t business_type);
- 调用者： 运营方;
- 核心逻辑:
  1. 检查 sender 为标准 eos name 格式，并且不是空地址;
  2. 检查 DDCID 数值是否大于 0;
- 所在类库： com.bsn.eos.service.DDC721Service#freeze
- 输入参数:

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
DDC 唯一标识	ddcId	BigInteger	是	DDC 唯一标识

- 输出参数:

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例： 见 4.2.3.5

3.1.3.11 解冻

3.1.3.11.1 功能介绍

运营方可以通过调用该方法对 DDC 进行解冻。

3.1.3.11.2 API 定义

- 方法定义：PushedTransaction unFreeze(String sender，BigInteger ddcId);
- EOS 合约方法：unfreeze(name sender, uint64_t ddc_id, uint64_t business_type);
- 调用者：运营方;
- 核心逻辑：
  1. 检查 sender 为标准 eos name 格式，并且不是空地址;
  2. 检查 DDCID 数值是否大于 0;
- 所在类库：com.bsn.eos.service.DDC721Service#unFreeze
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
DDC 唯一标识	ddcId	BigInteger	是	DDC 唯一标识

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.6

3.1.3.12 销毁

3.1.3.12.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过调用该方法对 DDC 进行销毁。

3.1.3.12.2 API 定义

- 方法定义：PushedTransaction burn(String sender，String owner，BigInteger ddcId);
- EOS 合约方法：burn(name sender, uint64_t ddc_id, uint64_t business_type);
- 调用者：DDC 拥有者、DDC 授权者;
- 核心逻辑：

1. 检查 sender 为标准 eos name 格式，并且不是空地址；
2. 检查 DDCID 数值是否大于 0;

- 所在类库：com.bsn.eos.service.DDC721Service#burn
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者	owner	String	是	
DDC 唯一标识	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.7

3.1.3.13 批量销毁(暂不提供)

3.1.3.13.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过调用该方法对 DDC 进行批量销毁。

3.1.3.13.2 API 定义

- 方法定义：PushedTransaction burnBatch(String sender, String owner, List<BigInteger> ddcIds);
- EOS 合约方法：burnbatch(name sender, name owner, std::vector<uint64_t> ddc_ids, uint64_t business_type);
- 调用者：DDC 拥有者、DDC 授权者;
- 核心逻辑：

1.检查 sender 为标准 eos name 格式，并且不是空地址；

2.检查 ddcIds 集合大小必须大于 0;

3.循环 ddclds 集合，并根据索引检查对应的 DDCID 数值必须大于 0;

- 所在类库：com.bsn.eos.service.DDC721Service#burnBatch
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者	owner	String	是	
DDCID 集合	ddclds	List<BigInteger>	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.8

3.1.3.14 查询数量

3.1.3.14.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法进行查询当前账户拥有的 DDC 的数量。

3.1.3.14.2 API 定义

- 方法定义：BigInteger balanceOf(String owner);
- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：运营方、平台方以及终端用户;
- 核心逻辑：

1. 检查拥有者账户地址信息是否为空;
2. 检查拥有者账户地址格式是否正确;

- 所在类库：com.bsn.eos.service.DDC721Service#balanceOf
- 输入参数：

字段名	字段	类型	必传	备注
拥有者账户	owner	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
DDC 的数量	balance	BigInteger	是	

- 调用实例：见 4.2.3.9

3.1.3.15 批量查询数量(暂不提供)

3.1.3.15.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法进行批量查询账户列表拥有的 DDC 的数量。

3.1.3.15.2 API 定义

- 方法定义：List<BigInteger> balanceOfBatch(List<String> owners);
- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：运营方、平台方以及终端用户；
- 核心逻辑：

  1.检查 owners 集合大小必须大于 0；

  2.循环 owners 集合，并根据索引检查拥有者账户地址为标准 eos name 格式，并且不是空地址；

- 所在类库：com.bsn.eos.service.DDC721Service#balanceOfBatch
- 输入参数：

字段名	字段	类型	必传	备注
拥有者账户集合	owners	List<String>	是	

- 输出参数：

字段名	字段	类型	必传	备注
DDC 的数量集		List<BigInteger>	是	

合				

- 调用实例：见 4.2.3.9

3.1.3.16 查询拥有者

3.1.3.16.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法查询当前 DDC 的 拥有者。

3.1.3.16.2 API 定义

- 方法定义：String ownerOf(BigInteger ddcId);
- EOS 合约方法：get_table_rows(name contract, name table, uint64_t ddc_id);
- 调用者：运营方、平台方以及终端用户;
- 核心逻辑：检查 ddcId 的数值是否大于 0;
- 所在类库：com.bsn.eos.service.DDC721Service#ownerOf
- 输入参数：

字段名	字段	类型	必传	备注
DDC 唯一标识	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
拥有者账户	owner	String	是	

- 调用实例：见 4.2.3.9

3.1.3.17 批量查询拥有者(暂不提供)

3.1.3.17.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法查询 DDC 列表 的 拥有者。

3.1.3.17.2 API 定义

- 方法定义：List<String> ownerOfBatch(List<BigInteger> ddcIds);
- EOS 合约方法：get_table_rows(name contract, name table, uint64_t ddc_id);
- 调用者：运营方、平台方以及终端用户；
- 核心逻辑：

  1.检查 ddcIds 集合大小必须大于 0；

  2.循环 ddcIds 集合，并根据索引检查 DDCID 数值必须大于 0；
- 所在类库：com.bsn.eos.service.DDC721Service#ownerOfBatch
- 输入参数：

字段名	字段	类型	必传	备注
DDCID 集合	ddcIds	List<BigInteger>	是	

- 输出参数：

字段名	字段	类型	必传	备注
拥有者账户集合	Owners	List<String>	是	

- 调用实例：见 4.2.3.9

3.1.3.18 获取名称

3.1.3.18.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法查询当前 DDC 的 名称。

3.1.3.18.2 API 定义

- 方法定义：String name(BigInteger ddcId);
- EOS 合约方法：get_table_rows(name contract,uint64_t primary, uint64_t ddc_id);
- 调用者：运营方、平台方以及终端用户；
- 所在类库：com.bsn.eos.service.DDC721Service#name
- 输入参数：

字段名	字段	类型	必传	备注
DDC 唯一标识	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
DDC 名称	ddcName	String	是	

- 调用实例：见 4.2.3.9

3.1.3.19 获取符号

3.1.3.19.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法查询当前 DDC 的 符号标识。

3.1.3.19.2 API 定义

- 方法定义：String symbol(BigInteger ddcId);
- EOS 合约方法：get_table_rows(name contract,uint64_t primary, uint64_t ddc_id);
- 调用者：运营方、平台方以及终端用户;
- 所在类库：com.bsn.eos.service.DDC721Service#symbol
- 输入参数：

字段名	字段	类型	必传	备注
DDC 唯一标识	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
DDC 运营方符号	symbol	String	是	DDC 运营方符号

- 调用实例：见 4.2.3.9

3.1.3.20 获取 DDCURI

3.1.3.20.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法查询当前 DDC 的 资源标识符。

3.1.3.20.2 API 定义

- 方法定义：String ddcURI(BigInteger ddcId);
- EOS 合约方法：get_table_rows(name contract,uint64_t primary, uint64_t ddc_id);;
- 调用者：运营方、平台方以及终端用户;
- 核心逻辑：检查 ddcId 的数值是否大于 0;
- 所在类库：com.bsn.eos.service.DDC721Service#ddcURI
- 输入参数：

字段名	字段	类型	必传	备注
DDC 唯一标识	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
DDC 资源标识符	ddcURI	String	是	

- 调用实例：见 4.2.3.9

3.1.3.21 设置 DDCURI

3.1.3.21.1 功能介绍

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。

3.1.3.21.2 API 定义

- 方法定义：PushedTransaction setURI(String sender,String owner,BigInteger ddcId,String ddcUri);

- EOS 合约方法： seturi(name sender, name owner, uint64_t ddc_id, std::string ddc_uri, uint64_t business_type);
- 调用者：DDC 拥有者或 DDC 授权者；
- 核心逻辑：
  1. 检查 DDCID 对应的 DDC 资源；
  2. 检查调用者身份和信息
  3. 触发事件；
- 所在类库：com.bsn.eos.service.DDC721Service#setURI
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者	owner	String	是	拥有者账号
DDC 唯一标识	ddcId	BigInteger	是	DDC 唯一标识
DDC 资源标识符	ddcUri	String	是	DDC 资源标识符

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.13

3.1.3.22 符号名称设置

3.1.3.22.1 功能介绍

合约拥有者调用该方法对 721 的名称及符号进行设置。

3.1.3.22.2 API 定义

- 方法定义：PushedTransaction setNameAndSymbol(String name, String symbol);

- EOS 合约方法：setnamesym(std::string name, std::string symbol);
- 调用者：合约拥有者
- 核心逻辑：
  1. 检查调用者身份和信息
  2. 触发事件;
- 所在类库：com.bsn.eos.service.DDC721Service#setNameAndSymbol
- 输入参数：

字段名	字段	类型	必传	备注
DDC 名称	name	String	是	DDC 名称
DDC 符号	symbol	String	是	DDC 符号

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.14

3.1.3.23 最新 DDCID 查询

3.1.3.23.1 功能介绍

运营方、平台方以及终端用户通过调用该方法对当前最新 DDCID 进 行查询。

3.1.3.23.2 API 定义

- 方法定义：BigInteger getLatestDDCId();
- EOS 合约方法：get_table_rows(name contract,uint64_t primary, uint64_t ddc_id);
- 调用者：运营方、平台方以及终端用户;
- 核心逻辑：
- 所在类库：com.bsn.eos.service.DDC721Service#getLatestDDCId
- 输入参数：

字段名	字段	类型	必传	备注

- 输出参数：

字段名	字段	类型	必传	备注
最新 DDCID		BigInteger	是	

- 调用实例：见 4.2.3.15

3.1.3.24 查询指定账户 721 资产(暂不提供)

3.1.3.24.1 功能介绍

运营方、平台方以及终端用户通过调用该方法查询指定账户 DDC721 资产。

3.1.3.24.2 API 定义

- 方法定义：List<String> assetsOf721(String owner);
- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：运营方、平台方以及终端用户;
- 核心逻辑：

  1.检查账户名不为空;
- 所在类库：com.bsn.eos.service.DDC721Service#assetsOf721
- 输入参数：

字段名	字段	类型	必传	备注
账户名	owner	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
账户拥有 DDCID 列表	assets	List<String>	是	

- 调用实例：见 4.2.3.16

3.1.3.25 Nonce 查询(元交易)

3.1.3.25.1 功能介绍

通过调用该方法对签名者账户所对应的最新 nonce 值进行查询，注：此查询只适用于发起元交易处理业务所对应的 nonce 值查询。

3.1.3.25.2 API 定义

- 方法定义：BigInteger getNonce(String from);
- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：所有用户;
- 核心逻辑：无
- 所在类库：com.bsn.eos.service.DDC721Service#getNonce
- 输入参数：

字段名	字段	类型	必传	备注
DDC 拥有者	from	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
最新 Nonce 值		BigInteger	是	

- 调用实例：见 4.2.3.17

3.1.3.26 元交易生成

3.1.3.26.1 功能介绍

终端用户可以通过授权平台方调用该方法对 DDC 进行元交易生成，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.3.26.2 API 定义

- 方法定义：PushedTransaction metaSafeMint(String sender, String from, String to, String

ddcURI, String memo, BigInteger nonce, BigInteger deadline, String signature);

- EOS 合约方法： 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);
- 调用者：平台方；
- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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；

  3)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）

  4)校验参数中业务类型是否符合规范，否则返回提示信息；

  5)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；
- 所在类库：com.bsn.eos.service.DDC721Service#metaSafeMint
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
发送方账户	from	String	是	
接收者账户	to	String	是	
DDC 资源标识符	ddcURI	String	是	
备注	memo	String	是	
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.18

3.1.3.27 元交易转移

3.1.3.27.1 功能介绍

DDC 拥有者或 DDC 授权者通过授权平台方调用该方法对 DDC 进行元交易转移，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.3.27.2 API 定义

- 方法定义：PushedTransaction metaSafeTransferFrom(String sender, String from, String to, BigInteger ddcId, String memo, BigInteger nonce, BigInteger deadline, String signature);
- EOS 合约方法：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);
- 调用者：平台方；
- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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；

  3)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）

  4)校验参数中业务类型是否符合规范，否则返回提示信息；

  5)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；

- 所在类库：com.bsn.eos.service.DDC721Service#metaSafeTransferFrom
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
发送方账户	from	String	是	
接收者账户	to	String	是	
DDC 唯一标识	ddcId	BigInteger	是	
备注	memo	String	是	
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.19

3.1.3.28 元交易销毁

3.1.3.28.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过授权平台方调用该方法对 DDC 进行元交易销毁，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.3.28.2 API 定义

- 方法定义：PushedTransaction metaBurn(String sender, String owner, BigInteger ddcId, BigInteger nonce, BigInteger deadline, String signature);
- EOS 合约方法：metaburn (name sender, name owner, uint64_t ddc_id, uint64_t

business_type, uint64_t nonce, uint64_t deadline, signature signature);

- 调用者：平台方；
- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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 owner 对应的公钥是否相等，不相等则返回提示信息；

  3)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）

  4)校验参数中业务类型是否符合规范，否则返回提示信息；

  5)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；

- 所在类库：com.bsn.eos.service.DDC721Service#metaBurn
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	owner	String	是	
DDC 唯一标识	ddcId	BigInteger	是	
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.3.20

3.1.4 BSN-DDC-1155

3.1.4.1 生成

3.1.4.1.1 功能介绍

平台方或终端用户可以通过调用该方法对 DDC 进行安全生成。

3.1.4.1.2 API 定义

- 方法定义：
- PushedTransaction safeMint(String from， String to,BigInteger amount,String ddcURI);
- PushedTransaction safeMint(String from, String to,BigInteger amount,String ddcURI,String memo);
- EOS 合约方法： mint(name sender, name to, uint64_t amount, std::string ddc_uri, uint64_t business_type, std::string memo);
- 调用者： 平台方、终端用户；
- 核心逻辑：
  1. 检查接收者账户地址信息是否为空；
  2. 检查接收者账户地址格式是否正确；
  3. 检查需要生成的 DDC 数量是否大于 0；
- 所在类库： com.bsn.eos.service.DDC1155Service#safeMint
- 输入参数：

字段名	字段	类型	必传	备注
调用者	From	String	是	调用者地址
接收者账户	to	String	是	
DDC 数量	amount	BigInteger	是	
DDCURI	ddcURI	String	是	
附加数据	memo	String	否	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.10

3.1.4.2 批量生成

3.1.4.2.1 功能介绍

平台方或终端用户可以通过调用该方法对 DDC 进行批量安全生成。

3.1.4.2.2 API 定义

- 方法定义：
- PushedTransaction safeMintBatch(String from , String to,List<BigInteger> amounts,List<String> ddcURIs);
- PushedTransaction safeMintBatch(String from , String to,List<BigInteger> amounts,List<String> ddcURIs,String memo);
- EOS 合约方法： mintbatch(name sender, name to, std::vector<uint64_t> amounts, std::vector<std::string> ddc_uris, uint64_t business_type, std::string memo);
- 调用者：平台方，终端用户；
- 核心逻辑：
  1. 检查接收者账户地址信息是否为空；
  2. 检查接收者账户地址格式是否正确；
  3. 检查生成的 DDC 数量集合大小是否大于 0；
  4. 检查生成的 DDC 数量集合中每个 DDC 数量是否大于 0；
  5. 检查生成的 DDCURI 集合大小是否大于 0；
  6. 检查生成的 DDC 数量集合与 DDCURI 集合的大小是否相等；
- 所在类库：com.bsn.eos.service.DDC1155Service#mintBatch

- 输入参数：

字段名	字段	类型	必传	备注
调用者	From	String	是	调用者账户
接收者账户	to	String	是	
DDC 数量集合	amounts	List<BigInteger>	是	
DDCURI 集合	ddcURIs	List<String>	是	
附加数据	memo	String	否	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.2

3.1.4.3 账户授权

3.1.4.3.1 功能介绍

DDC 拥有者可以通过调用该方法进行账户授权，发起者需要是 DDC 的拥有者。

3.1.4.3.2 API 定义

- 方法定义：PushedTransaction setApprovalForAll(String sender,String to,Boolean approved);
- EOS 合约方法：approvalall(name sender, name to, bool approved, uint64 business_type);
- 调用者：DDC 拥有者；
- 核心逻辑：
  1. 检查授权者账户地址信息是否为空；
  2. 检查授权者账户地址格式是否正确；
- 所在类库：com.bsn.eos.service.DDC1155Service#setApprovalForAll

- 输入参数：

字段名	字段	类型	必传	备注
调用者	Sender	String	是	调用者地址
被授权账户	to	String	是	
授权标识	approved	Boolean	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.11

3.1.4.4 账户授权查询

3.1.4.4.1 功能介绍

运营方、平台方或终端用户可以通过调用该方法进行账户授权查询。

3.1.4.4.2 API 定义

- 方法定义：Boolean isApprovedForAll(String sender,String to);
- EOS 合约方法：get_table_rows(name contract, name table, name account)
- 调用者：所有人；
- 核心逻辑：
  1. 检查拥有者账户地址信息是否为空；
  2. 检查拥有者账户地址格式是否正确；
  3. 检查授权者账户地址信息是否为空；
  4. 检查授权者账户地址格式是否正确；
- 所在类库：com.bsn.eos.service.DDC1155Service#isApprovedForAll
- 输入参数：

字段名	字段	类型	必传	备注
拥有者账户	Sender	String	是	
被授权账户	to	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
是否成功	isSuccess	Boolean	是	true/false

- 调用实例：见 4.2.4.11

3.1.4.5 转移

3.1.4.5.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过调用该方法对 DDC 进行安全转移。

3.1.4.5.2 API 定义

- 方法定义：

- PushedTransaction transferFrom(String sender,String from,String to,BigInteger ddcId,BigInteger amount);
- PushedTransaction transferFrom(String sender,String from,String to,BigInteger ddcId,BigInteger amount,String memo);

- EOS 合约方法：transfer(name from, name to, uint64 ddc_id, uint64 amount, string memo, uint64 business_type);
- 调用者：DDC 拥有者、DDC 授权者;
- 核心逻辑：

1. 检查拥有者账户地址信息是否为空;
2. 检查拥有者账户地址格式是否正确;
3. 检查接收者账户地址信息是否为空;
4. 检查接收者账户地址格式是否正确;
5. 检查 DDCID 数值是否大于 0;

6. 检查 DDC 转移所对应的数量是否大于 0;

- 所在类库：com.bsn.eos.service.DDC1155Service#transferFrom
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	from	String	是	
接收者账户	to	String	是	
DDCID	ddcId	BigInteger	是	
数量	amount	BigInteger	是	DDCID 所对应的数量
附加数据	memo	String	否	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.3

3.1.4.6 批量转移

3.1.4.6.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过调用该方法对 DDC 进行批量安全转移。

3.1.4.6.2 API 定义

- 方法定义：

- PushedTransaction safeBatchTransferFrom(String from,String to,List<Integer> ddcIds,List<Integer>amounts);
- PushedTransaction safeBatchTransferFrom(String from,String to,List<Integer> ddcIds,List<Integer>amounts，String memo);

- EOS 合约方法：batchtrans(name sender, name from, name to, std::vector<uint64_t> ddc_ids, std::vector<uint64_t> amounts, std::string memo, uint64_t business_type);
- 调用者：DDC 拥有者、DDC 授权者；
- 核心逻辑：
  1. 检查拥有者账户地址信息是否为空；
  2. 检查拥有者账户地址格式是否正确；
  3. 检查接收者账户地址信息是否为空；
  4. 检查接收者账户地址格式是否正确；
  5. 检查转移的 ddclds 集合大小是否大于 0；
  6. 检查转移的 ddclds 集合中每个 DDCID 是否大于 0；
  7. 检查转移的 amounts 集合中每个 DDC 数量是否大于 0；
- 所在类库：com.bsn.eos.service.DDC1155Service#safeBatchTransferFrom
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	from	String	是	
接收者账户	to	String	是	
拥有者 DDCID 集合	ddclds	List<Integer>	是	
转移数量	amounts	List<Integer>	是	
附加数据	memo	String	否	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.4

3.1.4.7 冻结

3.1.4.7.1 功能介绍

运营方可以通过调用该方法对 DDC 进行冻结。

3.1.4.7.2 API 定义

- 方法定义：PushedTransaction freeze(String sender， BigInteger ddcId);
- EOS 合约方法： freeze(name sender, uint64_t ddc_id, uint64_t business_type);
- 调用者：运营方;
- 核心逻辑：检查 DDCID 数值是否大于 0;
- 所在类库：com.bsn.eos.service.DDC1155Service#freeze
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
DDC 唯一标识	ddcId	BigInteger	是	DDC 唯一标识

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.5

3.1.4.8 解冻

3.1.4.8.1 功能介绍

运营方可以通过调用该方法对 DDC 进行解冻。

3.1.4.8.2 API 定义

- 方法定义：PushedTransaction unFreeze(String sender，BigInteger ddcId);
- EOS 合约方法：unfreeze(name sender, uint64_t ddc_id, uint64_t business_type);
- 调用者：运营方；
- 核心逻辑：检查 DDCID 数值是否大于 0;
- 所在类库：com.bsn.eos.service.DDC1155Service#unFreeze
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
DDC 唯一标识	ddcId	BigInteger	是	DDC 唯一标识

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.6

3.1.4.9 销毁

3.1.4.9.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过调用该方法对 DDC 进行销毁。

3.1.4.9.2 API 定义

- 方法定义：PushedTransaction burn(String sender，String owner，BigInteger ddcId);
- EOS 合约方法：burn(name sender, uint64_t ddc_id, name owner, uint64_t business_type);
- 调用者：DDC 拥有者、DDC 授权者；
- 核心逻辑：

1. 检查拥有者账户地址信息是否为空;

2. 检查拥有者账户地址格式是否正确;

3. 检查需要销毁的 DDCID 集合长度是否大于 0;

- 所在类库：com.bsn.eos.service.DDC1155Service#burn
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	Owner	String	是	
DDCID	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.7

3.1.4.10 批量销毁

3.1.4.10.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过调用该方法对 DDC 进行批量销毁。

3.1.4.10.2 API 定义

- 方法定义：PushedTransaction burnBatch(String sender，String owner，List<BigInteger> ddcIds);
- EOS 合约方法： burnbatch(name sender, std::vector<uint64_t> ddc_ids, name owner, uint64_t business_type);
- 调用者：DDC 拥用者、DDC 授权者;
- 核心逻辑：

1. 检查拥有者账户地址信息是否为空;

2. 检查拥有者账户地址格式是否正确;

3. 检查需要销毁的 DDCID 集合大小是否大于 0;

4. 检查需要销毁的 DDCID 集合中每个 DDCID 数值是否大于 0;

- 所在类库：com.bsn.eos.service.DDC1155Service#burnBatch
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	Owner	String	是	
DDCID 集合	ddclds	List<BigInteger>	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.8

3.1.4.11 查询数量

3.1.4.11.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法进行查询当前账户拥有的 DDC 的数量。

3.1.4.11.2 API 定义

- 方法定义：BigInteger balanceOf(String owner,BigInteger ddcId);
- EOS 合约方法：get_table_rows(name contract, name table, name account)
- 调用者：运营方、平台方以及终端用户;
- 核心逻辑：

1. 检查拥有者账户地址信息是否为空;

2. 检查拥有者账户地址格式是否正确;

3. 检查 DDCID 集合长度是否大于 0;

- 所在类库：com.bsn.eos.service.DDC1155Service#balanceOf
- 输入参数：

字段名	字段	类型	必传	备注
拥有者账户	owner	String	是	
DDCID	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
数量		BigInteger	是	拥有者账户所对应的 DDCID 所拥用的数量

- 调用实例：见 4.2.4.9

3.1.4.12 批量查询数量

3.1.4.12.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法进行批量查询账户拥有的 DDC 的数量。

3.1.4.12.2 API 定义

- 方法定义： List<BigInteger> balanceOfBatch(Multimap<String,BigInteger> ddcs);
- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：运营方、平台方以及终端用户;
- 核心逻辑：

1. 检查 ddcs 集合大小是否大于 0;
2. 检查 ddcs 集合中拥有者账户地址信息是否为空;
3. 检查 ddcs 集合中拥有者账户地址格式是否正确;
4. 检查 ddcs 集合中每个 DDCID 数值是否大于 0;

- com.bsn.eos.service.DDC1155Service#balanceOfBatch
- 输入参数：

字段名	字段	类型	必传	备注
拥有者 DDCID 集合	ddcs	Multimap<String,BigInteger>	是	

- 输出参数：

字段名	字段	类型	必传	备注
数量集合		List<BigInteger>	是	拥有者账户所对应的每个 DDCID 所拥用的数量

- 调用实例：见 4.2.4.10

3.1.4.13 获取 DDCURI

3.1.4.13.1 功能介绍

运营方、平台方以及终端用户可以通过调用该方法进行查询当前 DDC 的资源标识符。

3.1.4.13.2 API 定义

- 方法定义：String ddcURI(BigInteger ddcId);
- EOS 合约方法：get_table_rows(name contract,uint64_t primary, uint64_t ddc_id);
- 调用者：运营方、平台方以及终端用户；
- 核心逻辑：检查 DDCID 数值是否大于 0；
- 所在类库：com.bsn.eos.service.DDC1155Service#ddcURI
- 输入参数：

字段名	字段	类型	必传	备注
DDCID	ddcId	BigInteger	是	

- 输出参数：

字段名	字段	类型	必传	备注
DDC 资源标识符	ddcURI	String	是	

- 调用实例：见 4.2.4.12

3.1.4.14 设置 DDCURI

3.1.4.14.1 功能介绍

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。

3.1.4.14.2 API 定义

- 方法定义：PushedTransaction setURI(String sender,String owner,BigInteger ddcId,String ddcUri);
- EOS 合约方法：seturi(name sender, name owner, uint64_t ddc_id, std::string ddc_uri, uint64_t business_type);
- 调用者：DDC 拥有者、DDC 授权者；
- 核心逻辑：
  1. 检查 DDCID 对应的 DDC 资源；
  2. 检查调用者身份和信息
  3. 触发事件；
- 所在类库：com.bsn.eos.service.DDC1155Service#setURI
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
DDC 拥有者	owner	String	是	
DDC 唯一标识	ddcId	BigInteger	是	
DDC 资源标识符	ddcUri	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.13

3.1.4.15 符号名称设置(暂不提供)

3.1.4.15.1 功能介绍

合约拥有者调用该方法对 721 的名称及符号进行设置。

3.1.4.15.2 API 定义

- 方法定义：PushedTransaction setNameAndSymbol(String name, String symbol);
- EOS 合约方法：setnamesym(std::string name, std::string symbol);
- 调用者：合约拥有者
- 核心逻辑：
  1. 检查调用者身份和信息
  2. 触发事件;
- 所在类库：com.bsn.eos.service.DDC1155Service#setNameAndSymbol
- 输入参数：

字段名	字段	类型	必传	备注
DDC 名称	name	String	是	
DDC 符号	symbol	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.14

3.1.4.16 最新 DDCID 查询

3.1.4.16.1 功能介绍

运营方、平台方以及终端用户通过调用该方法对当前最新 DDCID 进行查询。

3.1.4.16.2 API 定义

- 方法定义：BigInteger getLatestDDCId();
- EOS 合约方法：get_table_rows(name contract,uint64_t primary, uint64_t ddc_id);
- 调用者：运营方、平台方以及终端用户;
- 核心逻辑：
- 所在类库：com.bsn.eos.service.DDC1155Service#getLatestDDCId
- 输入参数：

字段名	字段	类型	必传	备注

- 输出参数：

字段名	字段	类型	必传	备注
最新 DDCID		BigInteger	是	

- 调用实例：见 4.2.4.15

3.1.4.17 查询指定账户 1155 资产(暂不提供)

3.1.4.17.1 功能介绍

运营方、平台方以及终端用户通过调用该方法查询指定账户 DDC1155 资产。

3.1.4.17.2 API 定义

- 方法定义：Map<String, BigInteger> assetsOf1155(String owner);
- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：所有人;
- 核心逻辑：

  1. 检查账户名不为空;
- 所在类库：com.bsn.eos.service.DDC1155Service#assetsOf1155
- 输入参数：

字段名	字段	类型	必传	备注
账户名	owner	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
账户拥有 DDCID 及对应数量集合	assets	Map	是	

- 调用实例：见 4.2.4.16

3.1.4.18 Nonce 查询(元交易)

3.1.4.18.1 功能介绍

通过调用该方法对签名者账户所对应的最新 nonce 值进行查询，注：此查询只适用于发起元交易处理业务所对应的 nonce 值查询。

3.1.4.18.2 API 定义

- 方法定义：BigInteger getNonce(String from);
- EOS 合约方法：get_table_rows(name contract, name table, name account);
- 调用者：所有用户;
- 核心逻辑：无
- 所在类库：com.bsn.eos.service.DDC1155Service#getNonce

- 输入参数：

字段名	字段	类型	必传	备注
DDC 拥有者	from	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
最新 Nonce 值		BigInteger	是	

- 调用实例：见 4.2.4.17

3.1.4.19 元交易生成

3.1.4.19.1 功能介绍

终端用户可以通过授权平台方调用该方法对 DDC 进行元交易生成，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.4.19.2 API 定义

- 方法定义：PushedTransaction metaSafeMint(String sender, String from, String to, BigInteger amount, String ddcURI, String memo, BigInteger nonce, BigInteger deadline, String signature);
- EOS 合约方法：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);
- 调用者：平台方；
- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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；

  3)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）

4)校验参数中业务类型是否符合规范，否则返回提示信息；

5)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；

- 所在类库：com.bsn.eos.service.DDC1155Service#metaSafeMint
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
发送方账户	from	String	是	
接收者账户	to	String	是	
数量	amount	BigInteger	是	
DDC 资源标识符	ddcURI	String	是	
备注	memo	String	是	
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.18

3.1.4.20 元交易批量生成

3.1.4.20.1 功能介绍

终端用户可以通过授权平台方调用该方法对 DDC 进行元交易批量安全生成，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.4.20.2 API 定义

- 方法定义：PushedTransaction metaSafeMintBatch(String sender, String from, String to, List<BigInteger> amounts, List<String> ddcURIs, String memo, BigInteger nonce, BigInteger deadline, String signature);
- EOS 合约方法：metamintbatc(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);
- 调用者：平台方；
- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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；

  3)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）

  4)校验参数中业务类型是否符合规范，否则返回提示信息；

  5)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；
- 所在类库：com.bsn.eos.service.DDC1155Service#metaSafeMintBatch
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
发送方账户	from	String	是	
接收者账户	to	String	是	
数量列表	amounts	List<BigInteger>	是	
DDC 资源标识符列表	ddcURIs	List<String>	是	

备注	memo	String	是	
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.19

3.1.4.21 元交易转移

3.1.4.21.1 功能介绍

DDC 拥有者或 DDC 授权者通过授权平台方调用该方法对 DDC 进行元交易转移，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.4.21.2 API 定义

- 方法定义：PushedTransaction metaSafeTransferFrom(String sender, String from, String to, BigInteger ddcId, BigInteger amount, String memo, BigInteger nonce, BigInteger deadline, String signature);
- EOS 合约方法：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);
- 调用者：平台方；
- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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；

3)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）

4)校验参数中业务类型是否符合规范，否则返回提示信息；

5)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；

- 所在类库：com.bsn.eos.service.DDC1155Service#metaSafeTransferFrom
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
发送方账户	from	String	是	
接收者账户	to	String	是	
DDC 唯一标识	ddcId	BigInteger	是	
数量	amount	BigInteger	是	
备注	memo	String	是	
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.20

3.1.4.22 元交易批量转移

3.1.4.22.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过授权平台方调用该方法对 DDC 进行元交易批量转移，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.4.22.2 API 定义

- 方法定义：PushedTransaction metaSafeBatchTransferFrom(String sender, String from, String to, List<BigInteger> ddcIds, List<BigInteger> amounts, String memo, BigInteger nonce, BigInteger deadline, String signature);
- EOS 合约方法：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);
- 调用者：平台方；
- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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；

  3)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）

  4)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；
- 所在类库：com.bsn.eos.service.DDC1155Service#metaSafeBatchTransferFrom
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址

发送方账户	from	String	是	
接收者账户	to	String	是	
DDC 唯一标识列表	ddcIds	List<BigInteger>	是	
数量列表	amounts	List<BigInteger>	是	
备注	memo	String	是	
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.21

3.1.4.23 元交易销毁

3.1.4.23.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过授权平台方调用该方法对 DDC 进行元交易销毁，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.4.23.2 API 定义

- 方法定义： PushedTransaction metaBurn(String sender, String owner, BigInteger ddcId, BigInteger nonce, BigInteger deadline, String signature);
- EOS 合约方法： metaburn (name sender, name owner, uint64_t ddc_id, uint64_t business_type, uint64_t nonce, uint64_t deadline, signature signature);
- 调用者：平台方；

- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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 owner 对应的公钥是否相等，不相等则返回提示信息；

  3)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）

  4)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；

- 所在类库：com.bsn.eos.service.DDC1155Service#metaBurn
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	owner	String	是	
DDC 唯一标识	ddcId	BigInteger	是	
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.22

3.1.4.24 元交易批量销毁

3.1.4.24.1 功能介绍

DDC 拥有者或 DDC 授权者可以通过授权平台方调用该方法对 DDC 进行元交易批量销毁，注：调用此接口所消耗的能量值和业务费对应的是平台方账户。

3.1.3.24.2 API 定义

- 方法定义：PushedTransaction metaBurnBatch(String sender, String owner, List<BigInteger> ddcIds, BigInteger nonce, BigInteger deadline, String signature);
- EOS 合约方法：metaburnbatc(name sender, name owner, std::vector<uint64_t> ddc_ids, uint64_t business_type, uint64_t nonce, uint64_t deadline, signature signature);
- 调用者：平台方；
- 核心逻辑：

  1)校验调用者账户是否为平台方，不是则返回提示信息；

  2)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 owner 对应的公钥是否相等，不相等则返回提示信息；

  3)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）

  4)校验参数中业务类型是否符合规范，否则返回提示信息；

  5)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；
- 所在类库：com.bsn.eos.service.DDC1155Service#metaBurnBatch
- 输入参数：

字段名	字段	类型	必传	备注
调用者	sender	String	是	调用者地址
拥有者账户	owner	String	是	
DDC 唯一标识列	ddcIds	List<BigInte	是	

表		ger>		
Nonce 值	nonce	BigInteger	是	
过期时间	deadline	BigInteger	是	
签名值	signature	String	是	

- 输出参数：

字段名	字段	类型	必传	备注
交易哈希	transactionId	String	是	
执行信息	processed	Processed	是	

- 调用实例：见 4.2.4.23

3.1.5 BSN-DDC-区块查询

3.1.5.1 获取区块信息

3.1.5.1.1 功能介绍

运营方或平台方根据区块高度对区块信息进行查询，并解析区块数据返回给运营方或平台方。

3.1.5.1.2 API 定义

- 方法定义：Object getBlockInfo(BigInteger blockNum,List<String> actionList);
- EOS RPC 方法：cleos -u https://api.testnet.eos.io get block 48351112;
- 调用者：运营方、平台方;
- 核心逻辑：根据对应区块号，sdk 进行解析，若未提供 action，就将区块中所有 action 信息返回；若提供了 action，将区块中对应 action 中的信息返回，返回数据格式是 Java 数据对象。
- 输入参数：

字段名	字段	类型	必传	备注

区块号	blockNum	BigInteger	是	
Action 列表	actionList	List<String>	是	传入空的 List，返回合约中所有 Action 信息

- 输出参数：根据不同的 Action，返回不同的数据对象，具体 action 对应的事件数据，见 3.1.7 节 BSC-DDC-数据解析。

3.1.6 BSN-DDC-签名事件

3.1.6.1 功能介绍

此事件是通用事件，所有的上链待签名交易报文需调用此事件进行签名，业务调用方需要注册此签名事件，并在实现的签名事件中实现签名逻辑，并将最终签名后的结果返回给 DDC-SDK。

3.1.6.2 事件定义

- 输入参数：签名事件类
- 输出参数：签名结果
- String signEvent(SignEvent event)。

3.1.7 BSN-DDC-数据解析

3.1.7.1 权限数据

3.1.7.1.1 添加账户开关设置

3.1.7.1.1.1 功能说明

用于对 BSN-DDC-权限合约进行平台添加链账户开关设置所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.1.1.2 合约事件

- setswitcher(name sender, bool is_platform_open);

3.1.7.1.1.3 数据结构

字段名	字段	类型	备注
签名者	sender	String	
是否开通	isOpen	Boolean	

3.1.7.1.2 添加账户

3.1.7.1.2.1 功能说明

用于对 BSN-DDC-权限合约进行添加账户所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.1.2.2 合约事件

- addaccount(name sender, account_info info)

3.1.7.1.2.3 数据结构

字段名	字段	类型	备注
区块链账户名	account	name	DDC 用户链账户名
账户 DID	account_did	string	DDC 账户对应的 DID 信息（普通用户可为空）
账户名称	account_name	string	DDC 账户对应的账户名
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
冗余字段	field	string	冗余字段

3.1.7.1.3 批量添加账户

3.1.7.1.3.1 功能说明

用于对 BSN-DDC-权限合约进行批量平台方或运营方批量添加账户所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.1.3.2 合约事件

- addbatchaccount(name sender, account_info info)

3.1.7.1.3.3 数据结构

字段名	字段	类型	备注
区块链账户名	account	name	DDC 用户链账户名
账户 DID	account_did	string	DDC 账户对应的 DID 信息（普通用户可为空）
账户名称	account_name	string	DDC 账户对应的账户名
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
冗余字段	field	string	冗余字段

3.1.7.1.4 更新账户状态

3.1.7.1.4.1 功能说明

用于对 BSN-DDC-权限合约进行更新账户状态所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.1.4.2 合约事件

- updateacc(name sender,name account, PlatformState state, bool change_platform_state)

3.1.7.1.4.3 数据结构

字段名	字段	类型	备注
调用者	sender	String	是
区块链账户名	account	name	DDC 用户链账户名
平台管理账户状态	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）。
运营管理账户状态	change_platform_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）； 2．Active（活跃状态，可进行 DDC 相关操作）。

3.1.7.1.5 授权哈希设置

3.1.7.1.5.1 功能说明

用于对 BSN-DDC-权限合约进行授权哈希设置所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.1.5.2 合约事件

- setmetathash (uint8_t hashType, string hashValue)

3.1.7.1.5.3 数据结构

字段名	字段	类型	备注

唯一性主键	Primary	uint64_t	智能合约内部自动生成，业务方无须关心
分隔符列表	separatorMap	map<uint64_t, string>	1-ERC721 2-ERC1155
授权哈希列表	hashTypeMap	map<uint64_t, string>	MINT(1), MINTBATCH(2), TRANSFER(3), BATCHTRANS(4), BURN(5), BURNBATCH(6),

3.1.7.1.6 分隔符设置

3.1.7.1.6.1 功能说明

用于对 BSN-DDC-权限合约进行分隔符设置所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.1.6.2 合约事件

- setmetasepar (uint64_t business_type ,string separator)

3.1.7.1.6.3 数据结构

字段名	字段	类型	备注
唯一性主键	Primary	uint64_t	智能合约内部自动生成，业务方无须关心
分隔符列表	separatorMap	map<uint64_t, string>	1-ERC721 2-ERC1155
授权哈希列表	hashTypeMap	map<uint64_t, string>	MINT(1), MINTBATCH(2), TRANSFER(3), BATCHTRANS(4), BURN(5), BURNBATCH(6),

3.1.7.1.7 添加终端用户公钥

3.1.7.1.7.1 功能说明

用于对 BSN-DDC-权限合约进行添加终端用户公钥所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.1.7.2 合约事件

- addmetauser (name sender, name account, string pubKey)

3.1.7.1.7.3 数据结构

字段名	字段	类型	备注
终端用户账户	account	name	主键索引
账户对应的公钥	pubKey	string	
nonce 值列表	nonceMap	map<uint64_t, uint64_t>	1-ERC721 2-ERC1155

3.1.7.2 充值数据

3.1.7.2.1 充值

3.1.7.2.1.1 功能说明

用于对 BSN-DDC-计费合约进行充值所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.2.1.2 合约事件

- recharge(name from, name to, asset value);

3.1.7.2.1.3 数据结构

字段名	字段	类型	备注

充值账户地址	from	String	充值账户地址
被充值账户地址	to	String	被充值账户的地址
充值额度	value	asset	充值额度

3.1.7.2.2 DDC 业务费扣除

3.1.7.2.2.1 功能说明

用于对 BSN-DDC-计费合约进行 DDC 业务费扣除所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.2.2.2 合约事件

- receiptpay(name account, BusinessType business_type, name func_name, asset fee, asset balance);

3.1.7.2.2.3 数据结构

字段名	字段	类型	备注
支付人	account	name	支付人账号
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
调用方法	func_name	name	调用方法
调用花费手续费	fee	asset	调用花费手续费
调用后余额	balance	asset	调用后余额

3.1.7.2.3 设置 DDC 计费规则

3.1.7.2.3.1 功能说明

用于对 BSN-DDC-计费合约进行设置 DDC 计费规则所产生的区块中的事件进行解析，

并组装成所对应的数据结构。

3.1.7.2.3.2 合约事件

- setfee(name sender, BusinessType business_type, name func_name, asset value);

3.1.7.2.3.3 数据结构

字段名	字段	类型	备注
运营方账户	sender	name	运营方账户
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
方法名	func_name	String	EOS 合约方法
业务费用	value	BigInteger	

3.1.7.2.4 删除 DDC 计费规则

3.1.7.2.4.1 功能说明

用于对 BSN-DDC-计费合约进行删除 DDC 计费规则所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.2.4.2 合约事件

- deletefee(name sender, BusinessType business_type, name func_name);

3.1.7.2.4.3 数据结构

字段名	字段	类型	备注
运营方账户	sender	name	运营方账户
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
方法名	func_name	String	EOS 合约方法

3.1.7.2.5 按合约删除 DDC 计费规则

3.1.7.2.5.1 功能说明

用于对 BSN-DDC-计费合约进行按合约删除 DDC 计费规则所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.2.5.2 合约事件

- deleteddc(name sender, BusinessType business_type)

3.1.7.2.5.3 数据结构

字段名	字段	类型	备注
运营方账户	sender	name	运营方账户
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.3 BSN-DDC-721 数据

3.1.7.3.1 生成

3.1.7.3.1.1 功能说明

用于对 BSN-DDC-721 业务主逻辑合约进行 DDC 生成所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.3.1.2 合约事件

- mint(name sender, name to, uint64_t amount, std::string ddc_uri, uint64_t business_type, std::string memo);
- receiptmint(name sender, name to, uint64_t ddc_id, std::string ddc_uri, uint8_t allowed,

uint64_t amount, uint64_t business_type, std::string ddc_name, std::string ddc_symbol);

3.1.7.3.1.3 数据结构

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
发送数量	amount	uint64_t	ERC-721 忽略此数据
DDC 资源标识符	ddcURI	String	可为空
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
留言	memo	String	

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
DDC 唯一标识	ddc_id	uint64_t	
DDC 资源标识符	Ddc_uri	String	
是否进行授权	allowed	uint8_t	
增发数量	amount	uint8_t	
DDC 类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
DDC 名称	ddc_name	String	
DDC 符号	ddc_symbol	String	

3.1.7.3.2 转移

3.1.7.3.2.1 功能说明

用于对 BSN-DDC-721 业务主逻辑合约进行 DDC 转移/安全转移所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.3.2.2 合约事件

- transfer(name sender, name from, name to, uint64_t ddc_id, uint64_t amount, std::string memo, uint64_t business_type);

3.1.7.3.2.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
转出者账户	from	String	
接收者账户	to	String	
DDC 唯一标识	ddcId	uint64_t	
数量	amount	uint64_t	
留言	memo	String	
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.3.3 冻结

3.1.7.3.3.1 功能说明

用于对 BSN-DDC-721 业务主逻辑合约进行 DDC 解冻所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.3.3.2 合约事件

- freeze(name sender, uint64 ddc_id, uint64 business_type)

3.1.7.3.3.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
DDC 唯一标识	ddcId	BigInteger	DDC 唯一标识
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.3.4 解冻

3.1.7.3.4.1 功能说明

用于对 BSN-DDC-721 业务主逻辑合约进行 DDC 解冻所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.3.4.2 合约事件

- unfreeze(name sender, uint64 ddc_id, uint64 business_type)

3.1.7.3.4.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
DDC 唯一标识	ddcId	BigInteger	DDC 唯一标识
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.3.5 销毁

3.1.7.3.5.1 功能说明

用于对 BSN-DDC-721 业务主逻辑合约进行 DDC 销毁所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.3.5.2 合约事件

- burn(name sender, name owner, uint64_t ddc_id, uint64_t business_type);

3.1.7.3.5.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
拥有者	owner	name	拥有者
DDC 唯一标识	ddcId	uint64_t	DDC 唯一标识
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.3.6 元交易生成

3.1.7.3.6.1 功能说明

用于对 BSN-DDC-721 业务合约进行 DDC 元交易安全生成所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.3.6.2 合约事件

- 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);
- receiptmint(name sender, name to, uint64_t ddc_id, std::string ddc_uri, uint8_t allowed, uint64_t amount, uint64_t business_type, std::string ddc_name, std::string ddc_symbol);

3.1.7.3.6.3 数据结构

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
发送数量	amount	uint64_t	ERC-721 忽略此数据
DDC 资源标识符	ddcURI	String	可为空
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
留言	memo	String	

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
DDC 唯一标识	ddc_id	uint64_t	
DDC 资源标识符	Ddc_uri	String	
是否进行授权	allowed	uint8_t	
增发数量	amount	uint8_t	
DDC 类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
DDC 名称	ddc_name	String	
DDC 符号	ddc_symbol	String	

3.1.7.3.7 元交易转移

3.1.7.3.7.1 功能说明

用于对 BSN-DDC-721 业务主逻辑合约进行 DDC 元交易安全转移所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.3.7.2 合约事件

- 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);

3.1.7.3.7.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
转出者账户	from	String	
接收者账户	to	String	
DDC 唯一标识	ddcId	uint64_t	
数量	amount	uint64_t	
留言	memo	String	
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.3.8 元交易销毁

3.1.7.3.8.1 功能说明

用于对 BSN-DDC-721 业务主逻辑合约进行 DDC 元交易销毁所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.3.8.2 合约事件

- metaburn(name sender, name owner, uint64_t ddc_id, uint64_t business_type, uint64_t nonce, uint64_t deadline, signature signature);

3.1.7.3.8.3 数据结构

字段名	字段	类型	备注

调用者	sender	name	调用者地址
拥有者	owner	name	拥有者
DDC 唯一标识	ddcId	uint64_t	DDC 唯一标识
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4 BSN-DDC-1155 数据

3.1.7.4.1 生成

3.1.7.4.1.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 生成所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.1.2 合约事件

- mint(name sender, name to, uint64_t amount, std::string ddc_uri, uint64_t business_type, std::string memo);

3.1.7.4.1.3 数据结构

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
发送数量	amount	uint64_t	ERC-721 忽略此数据
DDC 资源标识符	ddcURI	String	可为空
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
留言	memo	String	

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
DDC 唯一标识	ddc_id	uint64_t	
DDC 资源标识符	Ddc_uri	String	
是否进行授权	allowed	uint8_t	
增发数量	amount	uint8_t	
DDC 类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
DDC 名称	ddc_name	String	
DDC 符号	ddc_symbol	String	

3.1.7.4.2 批量生成

3.1.7.4.2.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 批量生成所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.2.2 合约事件

- mintbatch(name sender, name to, std::vector<uint64_t> amounts, std::vector<std::string> ddc_uris, uint64_t business_type, std::string memo);
- receiptmint(name sender, name to, uint64_t ddc_id, std::string ddc_uri, uint8_t allowed, uint64_t amount, uint64_t business_type, std::string ddc_name, std::string ddc_symbol);

3.1.7.4.2.3 数据结构

字段名	字段	类型	备注
调用者	Sender	String	调用者地址

接收者账户	to	String	
DDC 数量集合	amounts	List<uint64_t>	
DDCURI 集合	ddcURIs	List<String>	可为空
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
留言	memo	String	

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
DDC 唯一标识	ddc_id	uint64_t	
DDC 资源标识符	Ddc_uri	String	
是否进行授权	allowed	uint8_t	
增发数量	amount	uint8_t	
DDC 类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
DDC 名称	ddc_name	String	
DDC 符号	ddc_symbol	String	

3.1.7.4.3 转移

3.1.7.4.3.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 安全转移所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.3.2 合约事件

- transfer(name sender, name from, name to, uint64_t ddc_id, uint64_t amount, std::string

memo, uint64_t business_type);

3.1.7.4.3.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
转出者账户	from	String	
接收者账户	to	String	
DDC 唯一标识	ddcId	uint64_t	
数量	amount	uint64_t	
留言	memo	String	
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.4 批量转移

3.1.7.4.4.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 批量安全转移所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.4.2 合约事件

- batchtrans(name sender, name from, name to, std::vector<uint64_t> ddc_ids, std::vector<uint64_t> amounts, std::string memo, uint64_t business_type);

3.1.7.4.4.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
转出者账户	from	String	

接收者账户	to	String	
DDC 标识符集合	ddcId	List< uint64_t >	
转移数量集合	amount	List< uint64_t >	
留言	memo	String	
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.5 冻结

3.1.7.4.5.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 冻结所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.5.2 合约事件

- freeze(name sender, uint64 ddc_id, uint64 business_type)

3.1.7.4.5.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
DDC 唯一标识	ddcId	BigInteger	DDC 唯一标识
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.6 解冻

3.1.7.4.6.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 解冻所产生的区块中的事件进行解

析，并组装成所对应的数据结构。

3.1.7.4.6.2 合约事件

- unfreeze(name sender, uint64 ddc_id, uint64 business_type)

3.1.7.4.6.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
DDC 唯一标识	ddcId	BigInteger	DDC 唯一标识
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.7 销毁

3.1.7.4.7.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 销毁所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.7.2 合约事件

- burn(name sender, name owner, uint64_t ddc_id, uint64_t business_type);

3.1.7.4.7.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
拥有者	owner	name	拥有者
DDC 唯一标识	ddcId	uint64_t	DDC 唯一标识
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.8 批量销毁

3.1.7.4.8.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 批量销毁所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.8.2 合约事件

- burnbatch(name sender, name owner, std::vector<uint64_t> ddc_ids, uint64_t business_type);

3.1.7.4.8.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
拥有者	owner	name	拥有者
DDC 标识集合	ddclds	List<uint64_t>	DDC 唯一标识集合
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.9URI 设置

3.1.7.4.9.1 功能说明

用于对 BSN-DDC-1155 业务合约进行 DDC 资源标识符设置所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.9.2 合约事件

- seturi(name sender, name owner, uint64_t ddc_id, std::string ddc_uri, uint64_t

business_type);

3.1.7.4.9.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
拥有者	owner	name	拥有者
DDC 标识	ddcId	List<uint64_t>	DDC 唯一标识集合
DDC 资源标识符	ddcURI	S	
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.10 元交易生成

3.1.7.4.10.1 功能说明

用于对 BSN-DDC-1155 业务合约进行 DDC 元交易安全生成所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.10.2 合约事件

- 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);
- receiptmint(name sender, name to, uint64_t ddc_id, std::string ddc_uri, uint8_t allowed, uint64_t amount, uint64_t business_type, std::string ddc_name, std::string ddc_symbol);

3.1.7.4.10.3 数据结构

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	

发送数量	amount	uint64_t	ERC-721 忽略此数据
DDC 资源标识符	ddcURI	String	可为空
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
留言	memo	String	

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
DDC 唯一标识	ddc_id	uint64_t	
DDC 资源标识符	Ddc_uri	String	
是否进行授权	allowed	uint8_t	
增发数量	amount	uint8_t	
DDC 类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
DDC 名称	ddc_name	String	
DDC 符号	ddc_symbol	String	

3.1.7.4.11 元交易批量生成

3.1.7.4.11.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 元交易批量生成所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.11.2 合约事件

- metamintbatc(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);

- receiptmint(name sender, name to, uint64_t ddc_id, std::string ddc_uri, uint8_t allowed, uint64_t amount, uint64_t business_type, std::string ddc_name, std::string ddc_symbol);

3.1.7.4.11.3 数据结构

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
DDC 数量集合	amounts	List<uint64_t>	
DDCURI 集合	ddcURIs	List<String>	可为空
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
留言	memo	String	

字段名	字段	类型	备注
调用者	Sender	String	调用者地址
接收者账户	to	String	
DDC 唯一标识	ddc_id	uint64_t	
DDC 资源标识符	Ddc_uri	String	
是否进行授权	allowed	uint8_t	
增发数量	amount	uint8_t	
DDC 类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155
DDC 名称	ddc_name	String	
DDC 符号	ddc_symbol	String	

3.1.7.4.12 元交易转移

3.1.7.4.12.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 元交易安全转移所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.12.2 合约事件

- 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);

3.1.7.4.12.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
转出者账户	from	String	
接收者账户	to	String	
DDC 唯一标识	ddcId	uint64_t	
数量	amount	uint64_t	
留言	memo	String	
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.13 元交易批量转移

3.1.7.4.13.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 元交易批量安全转移所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.13.2 合约事件

- 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);

3.1.7.4.13.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
转出者账户	from	String	
接收者账户	to	String	
DDC 标识符集合	ddcId	List< uint64_t >	
转移数量集合	amount	List< uint64_t >	
留言	memo	String	
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.14 元交易销毁

3.1.7.4.14.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 元交易销毁所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.14.2 合约事件

- metaburn(name sender, name owner, uint64_t ddc_id, uint64_t business_type, uint64_t nonce, uint64_t deadline, signature signature);

3.1.7.4.14.3 数据结构

字段名	字段	类型	备注

调用者	sender	name	调用者地址
拥有者	owner	name	拥有者
DDC 唯一标识	ddcId	uint64_t	DDC 唯一标识
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

3.1.7.4.15 元交易批量销毁

3.1.7.4.15.1 功能说明

用于对 BSN-DDC-1155 业务主逻辑合约进行 DDC 元交易批量销毁所产生的区块中的事件进行解析，并组装成所对应的数据结构。

3.1.7.4.15.2 合约事件

- metaburnbatc(name sender, name owner, std::vector<uint64_t> ddc_ids, uint64_t business_type, uint64_t nonce, uint64_t deadline, signature signature);

3.1.7.4.15.3 数据结构

字段名	字段	类型	备注
调用者	sender	name	调用者地址
拥有者	owner	name	拥有者
DDC 标识集合	ddcIds	List<uint64_t>	DDC 唯一标识集合
业务类型	business_type	uint64_t	1 表示 ERC-721 2 表示 ERC-1155

4 SDK 使用说明

4.1 初始化配置

4.1.1 初始化 ChainConfig 类

1. 通过 setGatewayUrl 配置 EOS 链地址
2. 通过 setDdcContractAndAccount 配置部署 DDC 合约账户名
3. 通过 setPk 配置平台方账户私钥
4. 通过 setPermission 配置调用权限名（默认为 active）
5. 通过 setBlockGatewayUrl 配置区块解析链地址（默认为 1 设置的 gatewayUrl 地址）
6. 通过 setLever 配置日志打印级别（默认为 NONE）

4.1.2 初始化 MetaTrxConfig 类

1. 通过 setChainId 配置 EOS 链 ID（适用于元交易）

4.2 API 调用

4.2.1 BSN-DDC-权限管理

4.2.1.1 添加运营账户

调用方法实例：

```
/**
 * 添加运营账户
 */
@Test
public void addOperator() {
    //使用合约账户私钥
    ChainConfig.setPk("5Hz8ZQ4dYSwwijhFTTnAgkdV3MJCHyZgw7i9ApxJ5Fdq8bs7dFD");

    DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
    PushedTransaction pt = ddcPermissionService.addOperator(accountList.get(0), accountList.get(0), didList.get(0));
    System.out.println(JSONUtil.toJsonStr(pt));
```

```
}
```

4.2.1.2 运营方添加平台方、终端用户

调用方法实例：

```
/**
 * 运营方添加平台方、终端用户
 */
@Test
public void addPlatformByOperator() {
    //使用运营方账户私钥
    ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

    DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
    //leaderDID 为空添加平台方账户
    PushedTransaction pt1 = ddcPermissionService.addAccountByOperator(accountList.get(0), accountList.get(1), accountList.get(1), didList.get(1), null);

    PushedTransaction pt2 = ddcPermissionService.addAccountByOperator(accountList.get(0), accountList.get(2), accountList.get(2), didList.get(2), didList.get(1));
    PushedTransaction pt3 = ddcPermissionService.addAccountByOperator(accountList.get(0), accountList.get(3), accountList.get(3), didList.get(3), didList.get(1));
    System.out.println(JSONUtil.toJsonStr(pt1));
    System.out.println(JSONUtil.toJsonStr(pt2));
    System.out.println(JSONUtil.toJsonStr(pt3));
}
```

4.2.1.3 运营方批量添加平台方、终端账户

调用方法实例：

```
/**
 * 运营方批量添加平台方和终端用户
 */
@Test
public void addBatchAccountByOperator() {
    //使用运营方账户私钥
    ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

    DDCPermissionServiceImpl ddcPermissionService = new DDCPermissionServiceImpl();
```

```
9.      List<String> list = Arrays.asList("ddc.con1", "ddc.con2");
10.     List<String> accountdids = Arrays.asList("bsn:ddc:did333", "bsn:ddc:did444");
11.     List<String> leaderdids = Arrays.asList("cm:ddc:manager1", "cm:ddc:manager1");
12.     PushedTransaction pt = ddcPermissionService.addBatchAccountByOperator("ddc.operator", list, list, accountdids, leaderdids);
13.     System.out.println(JSONUtil.toJsonStr(pt));
14. }
```

4.2.1.4 平台方添加终端账户

调用方法实例：

```
1.  /**
2.   * 平台方添加终端账户
3.   */
4.  @Test
5.  public void addAccountByPlatform() {
6.      //设置终端用户账户
7.      String normAccount = "testnew2";
8.      DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
9.      PushedTransaction pt = ddcPermissionService.addAccountByPlatform(accountList.get(1), normAccount, normAccount, "bsn:ddc:did123");
10.     System.out.println(JSONUtil.toJsonStr(pt));
11. }
```

4.2.1.5 平台方批量添加终端账户

调用方法实例：

```
1.  /**
2.   * 平台方批量添加终端账户
3.   */
4.  @Test
5.  public void addBatchAccountByPlatform() {
6.      //设置普通链账户列表
7.      List<String> list = Arrays.asList("testnew3", "testnew4");
8.      //设置普通链账户 DID 列表
9.      List<String> didList = Arrays.asList("bsn:ddc:did124", "bsn:ddc:did125");

10.     DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
```

```
        PushedTransaction pt = ddcPermissionService.addBatchAccountByPlatform(accountList.get(1), list, list, didList);
        System.out.println(JSONUtil.toJsonStr(pt));
    }
```

4.2.1.6 查询账户

调用方法实例：

```
    /**
     * 查询账户
     */
    @Test
    public void getAccount() {
        DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
        System.out.println(ddcPermissionService.getAccount(accountList.get(1)));

    }
```

4.2.1.7 更新账户状态

调用方法实例：

```
    /**
     * 更新账户状态
     */
    @Test
    public void updateAccState() {
        DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
        PushedTransaction pt = ddcPermissionService.updateAccState(accountList.get(1), accountList.get(2), AccountStateEnum.Frozen, false);
        System.out.println(JSONUtil.toJsonStr(pt));
    }
```

4.2.1.8 添加方法

调用方法实例：

```
    /**
     * 添加 721 方法
     */
    @Test
    public void add721Function() {
```

```
6.      //使用运营方账户私钥
7.      ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

8.      DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl
    ();
9.      ddcPermissionService.addFunction(accountList.get(0), RoleEnum.Operator,
    BusinessTypeEnum.ERC721, "freeze");
10.     ddcPermissionService.addFunction(accountList.get(0), RoleEnum.Operator,
    BusinessTypeEnum.ERC721, "unfreeze");
11. }
```

4.2.1.9 删除方法

调用方法实例:

```
1.  /**
2.   * 删除方法
3.   */
4.  @Test
5.  public void delFunction() {
6.      //使用运营方账户私钥
7.      ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

8.      DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl
    ();
9.      PushedTransaction pt = ddcPermissionService.delFunction(accountList.get(
    0), RoleEnum.Consumer, BusinessTypeEnum.ERC721, "approvalall");
10.     System.out.println(JSONUtil.toJsonStr(pt));
11. }
```

4.2.1.10 跨平台授权

调用方法实例:

```
1.  /**
2.   * 跨平台授权
3.   */
4.  @Test
5.  public void crossAppr() {
6.      //使用运营方账户私钥
7.      ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

8.      DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl
    ();
```

```
9.      PushedTransaction pt = ddcPermissionService.crossAppr(accountList.get(0),
    accountList.get(0), accountList.get(2), true);
10.     System.out.println(JSONUtil.toJsonStr(pt));
11. }
```

4.2.1.11 平台方添加链账户设置

调用方法实例:

```
1.  /**
2.   * 平台方添加链账户开关设置
3.   */
4.  @Test
5.  public void setSwitcherStateOfPlatform() {
6.      //使用运营方账户私钥
7.      ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

8.      DDCPermissionServiceImpl ddcPermissionService = new DDCPermissionService
    Impl();
9.      PushedTransaction pt = ddcPermissionService.setSwitcherStateOfPlatform(a
    ccountList.get(0), true);
10.     System.out.println(JSONUtil.toJsonStr(pt));
11. }
```

4.2.1.12 平台方添加链账户查询

调用方法实例:

```
1.  /**
2.   * 查询平台方添加链账户开关状态
3.   */
4.  @Test
5.  public void switcherStateOfPlatform() {
6.      DDCPermissionServiceImpl ddcPermissionService = new DDCPermissionService
    Impl();
7.      Boolean aBoolean = ddcPermissionService.switcherStateOfPlatform();
8.      System.out.println(aBoolean);
9.  }
```

4.2.1.13 授权哈希设置

调用方法实例:

```
1.  /**
2.   * 授权哈希设置
```

```
 */
@Test
public void setMetaTHash() {
    //使用合约账户私钥
    ChainConfig.setPk("5Hz8ZQ4dYSwwijhFTTnAgkdV3MJCHyZgw7i9ApxJ5Fdq8bs7dFD");

    DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
    PushedTransaction pt1 = ddcPermissionService.setMetaTHash(HashTypeEnum.MINT,MetaTrxConfig.META_SAFE_MINT);
    System.out.println(JSONUtil.toJsonStr(pt1));
}
```

4.2.1.14 分隔符设置

调用方法实例:

```
/**
 * 分隔符设置
 */
@Test
public void setMetaSeparator() {
    //使用合约账户私钥
    ChainConfig.setPk("5Hz8ZQ4dYSwwijhFTTnAgkdV3MJCHyZgw7i9ApxJ5Fdq8bs7dFD");

    String chainId = "894f2cc2fba7696ea971df449aba54da4a4240f40a1909539023888134e8b433";
    String contractAddress = "ddccontract1";
    DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
    PushedTransaction pt1 = ddcPermissionService.setMetaSeparator(BusinessTypeEnum.ERC721, MetaTrxConfig.DDC721_NAME, chainId, contractAddress);
    System.out.println(JSONUtil.toJsonStr(pt1));
    PushedTransaction pt2 = ddcPermissionService.setMetaSeparator(BusinessTypeEnum.ERC1155, MetaTrxConfig.DDC1155_NAME, chainId, contractAddress);
    System.out.println(JSONUtil.toJsonStr(pt2));
}
```

4.2.1.15 添加终端用户公钥

调用方法实例:

```
/**
 * 添加终端用户公钥
 */
```

```
@Test
public void addMetaUser() {
    DDCPermissionService ddcPermissionService = new DDCPermissionServiceImpl();
    PushedTransaction pt1 = ddcPermissionService.addMetaUser(accountList.get(1), accountList.get(2), "EOS6R3jYqb3uZsVgzJz4HVLcYV94CLkr3u9unEzm5rpuAqUos7fqK");
    System.out.println(JSONUtil.toJsonStr(pt1));
    PushedTransaction pt2 = ddcPermissionService.addMetaUser(accountList.get(1), accountList.get(3), "EOS5DQMoqswknpe5qXsMt3M4su1wK38Mj7Rzc5jxs1Ak5jq7BF623");
    System.out.println(JSONUtil.toJsonStr(pt2));
}
```

4.2.2 BSN-DDC-费用管理

4.2.2.1 充值

调用方法实例:

```
/**
 * 终端用户账户充值
 */
@Test
public void recharge() {
    DDCFeeService ddcFeeService = new DDCFeeServiceImpl();
    PushedTransaction pt = ddcFeeService.recharge(accountList.get(1), accountList.get(2), "1.0000 FEE", "memo0001");
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.2.2 批量充值

调用方法实例:

```
/**
 * 批量充值
 */
@Test
public void rechargebat() {
    //设置终端用户列表、充值金额列表
    List<String> toList = Arrays.asList(accountList.get(2), accountList.get(3));
    List<String> valueList = Arrays.asList("0.0001 FEE", "0.0001 FEE");
```

```
    DDCFeeServiceImpl ddcFeeService = new DDCFeeServiceImpl();
    PushedTransaction pt = ddcFeeService.rechargeBatch(accountList.get(1), toList, valueList);
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.2.3 链账户余额查询

调用方法实例:

```
/**
 * 链账户余额查询
 */
@Test
public void balanceOf() {
    //设置平台方账户
    String platformAccount = accountList.get(1);
    DDCFeeService ddcFeeService = new DDCFeeServiceImpl();
    System.out.println(ddcFeeService.balanceOf(platformAccount));
}
```

4.2.2.4 链账户余额批量查询

调用方法实例:

```
/**
 * 批量链账户余额查询
 */
@Test
public void balanceOfBatch() {
    //设置终端用户列表
    List<String> toList = Arrays.asList(accountList.get(2), accountList.get(3));
    DDCFeeServiceImpl ddcFeeService = new DDCFeeServiceImpl();
    System.out.println(ddcFeeService.balanceOfBatch(toList));
}
```

4.2.2.5 运营账户充值

调用方法实例:

```
/**
 * 运营方账户充值
 */
@Test
```

```
5.  public void selfRecharge() {
6.      //使用运营方账户私钥
7.      ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

8.      DDCFeeService ddcFeeService = new DDCFeeServiceImpl();
9.      PushedTransaction pt = ddcFeeService.selfRecharge(accountList.get(0), "1
    0000.0000 FEE");
10.     System.out.println(JSONUtil.toJsonStr(pt));
11. }
```

4.2.2.6 设置 DDC 计费规则

调用方法实例:

```
1.  /**
2.   * 设置 DDC721 计费规则
3.   */
4.  @Test
5.  public void set721Fee() {
6.      //使用运营方账户私钥
7.      ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

8.      DDCFeeService ddcFeeService = new DDCFeeServiceImpl();
9.      ddcFeeService.setFee(accountList.get(0), BusinessTypeEnum.ERC721, "mint",
     "0.0001 FEE");
10.     ddcFeeService.setFee(accountList.get(0), BusinessTypeEnum.ERC721, "trans
    fer", "0.0001 FEE");
11.     ddcFeeService.setFee(accountList.get(0), BusinessTypeEnum.ERC721, "burn",
     "0.0001 FEE");
12. }
```

4.2.2.7 查询 DDC 计费规则

调用方法实例:

```
1.  /**
2.   * 查询 DDC 计费规则
3.   */
4.  @Test
5.  public void queryFee() {
6.      DDCFeeService ddcFeeService = new DDCFeeServiceImpl();
7.      System.out.println(ddcFeeService.queryFee(accountList.get(0), BusinessTy
    peEnum.ERC721, "seturi"));
8.  }
```

4.2.2.8 删除 DDC 计费规则

调用方法实例：

```
/**
 * 删除 DDC 计费规则
 */
@Test
public void delFee() {
    //使用运营方账户私钥
    ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

    DDCFeeService ddcFeeService = new DDCFeeServiceImpl();
     PushedTransaction pt = ddcFeeService.delFee(accountList.get(0), BusinessTypeEnum.ERC1155, "mint");
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.2.9 按合约删除 DDC 计费规则

方法调用实例：

```
/**
 * 按合约删除 DDC 计费规则
 */
@Test
public void delDDC() {
    //使用运营方账户私钥
    ChainConfig.setPk("5JGBDss5JCkCqGf4Bi1LqNA7qmTebFwxP9TXPP8bX4wtJ8rWKTS");

    DDCFeeService ddcFeeService = new DDCFeeServiceImpl();
     PushedTransaction pt = ddcFeeService.delDDC(accountList.get(0), BusinessTypeEnum.ERC721);
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.3 BSN-DDC-721

4.2.3.1 生成

调用方法实例

```
/**
 * 721 生成
```

```
 */
@Test
public void mint721() {
    DDC721Service ddc721Service = new DDC721ServiceImpl();
     PushedTransaction pt = ddc721Service.safeMint(accountList.get(1), accountList.get(2), "https://bitnodes.io/0001", UUID.randomUUID().toString());
    System.out.println(JSONUtil.parse(pt));
}
```

4.2.3.2 批量生成

调用方法实例

```
/**
 * 721 批量生成
 * */
@Test
public void safeMintBatch(){
    DDC721ServiceImpl ddc721Service = new DDC721ServiceImpl();
     PushedTransaction pt = ddc721Service.safeMintBatch("testmxmwapa1", "testmxmwapa1", Arrays.asList("https://bitnodes.io/11", "https://bitnodes.io/22"), "hello");
   System.out.println(JSONUtil.parse(pt));
}
```

4.2.3.3 转移

调用方法实例:

```
/**
 * 721 转移
 */
@Test
public void transferFrom721() {
    ChainConfig.setPk("5JAT6ZYDhvvWVP2UMat84BhCyu5U3PYMYtkzqXEgoPoyhdJJfV1");

    DDC721Service ddc721Service = new DDC721ServiceImpl();
     PushedTransaction pt = ddc721Service.transferFrom(accountList.get(2), accountList.get(2), accountList.get(1), BigInteger.valueOf(4), UUID.randomUUID().toString());
    System.out.println(JSONUtil.parse(pt));
}
```

4.2.3.4 批量转移

调用方法实例:

```
/**
 * 721 批量转移
 */
@Test
public void batchTransferFrom721() {
    DDC721Service ddc721Service = new DDC721ServiceImpl();
    PushedTransaction pt = ddc721Service.batchTransferFrom(accountList.get(1), accountList.get(1), accountList.get(0), Arrays.asList(BigInteger.valueOf(1), BigInteger.valueOf(2)));
   System.out.println(JSONUtil.parse(pt));
}
```

4.2.3.5 冻结

调用方法实例:

```
/**
 * 721 冻结
 */
@Test
public void freeze721() {
    //使用运营方账户私钥
    ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

    DDC721Service ddc721Service = new DDC721ServiceImpl();
     PushedTransaction pt = ddc721Service.freeze(accountList.get(0), BigInteger.valueOf(2));
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.3.6 解冻

调用方法实例:

```
/**
 * 721 解冻
 */
@Test
public void unFreeze721() {
    //使用运营方账户私钥
```

```
7.      ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

8.      DDC721Service ddc721Service = new DDC721ServiceImpl();
9.      PushedTransaction pt = ddc721Service.unFreeze(accountList.get(0), BigInt
    eger.valueOf(2));
10.     System.out.println(JSONUtil.toJsonStr(pt));
11. }
```

4.2.3.7 销毁

调用方法实例：

```
1.  /**
2.   * 721 销毁
3.   */
4.  @Test
5.  public void burn721() {
6.      DDC721Service ddc721Service = new DDC721ServiceImpl();
7.      PushedTransaction pt = ddc721Service.burn(accountList.get(1), accountLis
    t.get(1), BigInteger.valueOf(4));
8.      System.out.println(JSONUtil.parse(pt));
9.  }
```

4.2.3.8 批量销毁

调用方法实例：

```
1.  /**
2.   * 721 批量销毁
3.   */
4.  @Test
5.  public void burnBatch721() {
6.      DDC721Service ddc721Service = new DDC721ServiceImpl();
7.      PushedTransaction pt  = ddc721Service.burnBatch(accountList.get(0), acco
    untList.get(0), Arrays.asList(BigInteger.valueOf(5), BigInteger.valueOf(6), B
    igInteger.valueOf(7)));
8.      System.out.println(JSONUtil.parse(pt));
9.  }
```

4.2.3.9 查询数量、拥有者、名称、符号、URI 以及批量查询

调用方法实例：

```
1.  /**
2.   * 721 查询数量、拥有者、名称、符号、DDCURI
```

```
 */
@Test
public void balanceOf721() {
    DDC721Service ddc721Service = new DDC721ServiceImpl();
    System.out.println(ddc721Service.balanceOf(accountList.get(1)));
    System.out.println(ddc721Service.ownerOf(BigInteger.valueOf(3)));
    System.out.println(ddc721Service.name(BigInteger.valueOf(3)));
    System.out.println(ddc721Service.symbol(BigInteger.valueOf(3)));
    System.out.println(ddc721Service.ddcURI(BigInteger.valueOf(3)));
}
```

4.2.3.10 DDC 授权及查询

方法调用实例:

```
/**
 * 721 DDC 授权
 */
@Test
public void approve721() {
    ChainConfig.setPk("5JAT6ZYDhvvWVP2UMat84BhCyu5U3PYMYtkzqXEgoPoyhdJJfV1");

    DDC721Service ddc721Service = new DDC721ServiceImpl();
     PushedTransaction pt = ddc721Service.approve(accountList.get(2), accountList.get(1), BigInteger.valueOf(4));
    Map<String, Object> trxMap = ChainUtil.getInstance().parseTrxResp(pt);
    System.out.println(JSONUtil.parse(trxMap));
}
/**
 * 721 DDC 授权查询
 */
@Test
public void getApproved721() {
    DDC721Service ddc721Service = new DDC721ServiceImpl();
    boolean flag = ddc721Service.getApproved(accountList.get(2), accountList.get(1), BigInteger.valueOf(4));
    System.out.println(flag);
}
```

4.2.3.11 DDC 批量授权及查询

方法调用实例:

```
/**
 * 721 DDC 批量授权及查询
```

```
 */
@Test
public void approveBatch721() {
    DDC721Service ddc721Service = new DDC721ServiceImpl();
     ddc721Service.approveBatch(accountList.get(0), accountList.get(1), Arrays.asList(BigInteger.valueOf(16),BigInteger.valueOf(17)));
     System.out.println(ddc721Service.getApproved(accountList.get(1), accountList.get(16), BigInteger.valueOf(17)));
}
```

4.2.3.12 账户授权及查询

调用方法实例:

```
/**
 * 721 账户授权
 */
@Test
public void approveall721() {
    ChainConfig.setPk("5JAT6ZYDhvvWVP2UMat84BhCyu5U3PYMYtkzqXEgoPoyhdJJfV1");

    DDC721Service ddc721Service = new DDC721ServiceImpl();
    PushedTransaction pt = ddc721Service.setApprovalForAll(accountList.get(2), accountList.get(1), true);
    System.out.println(JSONUtil.parse(pt ));
}
/**
 * 721 账户授权查询
 */
@Test
public void isApprovedForAll721() {
    DDC721Service ddc721Service = new DDC721ServiceImpl();
     boolean flag = ddc721Service.isApprovedForAll(accountList.get(2), accountList.get(1));
    System.out.println(flag);
}
```

4.2.3.13 设置 DDCURI

调用方法实例:

```
/**
 * 721 设置 DDCURI
 */
@Test
```

```
5.  public void setUri() {
6.      DDC721Service ddc721Service = new DDC721ServiceImpl();
7.      PushedTransaction pt = ddc721Service.setURI(accountList.get(1), accountL
    ist.get(1), BigInteger.valueOf(5), "https://bitnodes.io/0001");
8.      System.out.println(JSONUtil.toJsonStr(pt));
9.  }
```

4.2.3.14 符号名称设置

```
1.  /**
2.   * 721 符号名称设置
3.   */
4.  @Test
5.  public void setNameSym() {
6.      //使用合约账户私钥
7.      ChainConfig.setPk("5Hz8ZQ4dYSwwijhFTTnAgkdV3MJCHyZgw7i9ApxJ5Fdq8bs7dFD");

8.      DDC721Service ddc721Service = new DDC721ServiceImpl();
9.      PushedTransaction pt = ddc721Service.setNameAndSymbol("ddcname", "ddcsym
    bol");
10.     System.out.println(JSONUtil.toJsonStr(pt));
11. }
```

4.2.3.15 最新 DDCID 查询

调用方法实例

```
1.  /**
2.   * 721 最新 DDCID 查询
3.   */
4.  @Test
5.  public void getLatestDDCId() {
6.      DDC721ServiceImpl ddc721Service = new DDC721ServiceImpl();
7.      System.out.println(ddc721Service.getLatestDDCId());
8.  }
```

4.2.3.16 查询指定账户 721 资产

调用方法实例

```
1.  @Test
2.  public void assetOf721() {
3.      //设置平台方账户
4.      String platformAccount = "platform";
```

```
5.      DDC721ServiceImpl ddc721Service = new DDC721ServiceImpl();
6.      System.out.println(ddc721Service.assetsOf721(platformAccount));
7.  }
```

4.2.3.17 Nonce 查询

调用方法实例

```
1.  @Test
2.  public void getNonce() {
3.      DDC721Service ddc721Service = new DDC721ServiceImpl();
4.      BigInteger nonce = ddc721Service.getNonce(accountList.get(2));
5.      System.out.println("nonce: " + nonce);
6.  }
```

4.2.3.18 元交易生成

调用方法实例

```
1.  @Test
2.  public void metaSafeMint721() {
3.      String ddcURI = "https://bitnodes.io/0001";
4.      String memo = "memo0001";
5.      BigInteger deadline = BigInteger.valueOf(1671096761);
6.      DDC721Service ddc721Service = new DDC721ServiceImpl();
7.      BigInteger nonce = ddc721Service.getNonce(accountList.get(2));
8.      DDC721MetaTransaction metaTransaction = new DDC721MetaTransaction();
9.      String digest = metaTransaction.getSafeMintDigest(accountList.get(1), ac
    countList.get(2), accountList.get(2), ddcURI, memo, nonce, deadline);
10.     String signature = metaTransaction.generateSignature(MetaTrxConfig.getPr
    ivateKey(), digest.getBytes(StandardCharsets.UTF_8));
11.     PushedTransaction pt = ddc721Service.metaSafeMint(accountList.get(1), ac
    countList.get(2), accountList.get(2), ddcURI, memo, nonce, deadline, signatur
    e);
12.     System.out.println(JSONUtil.toJsonStr(pt));
13. }
```

4.2.3.19 元交易转移

调用方法实例

```
1.  @Test
2.  public void metaSafeTransferFrom721() {
3.      BigInteger ddcId = BigInteger.valueOf(1);
4.      String memo = "memo0001";
```

```
5.      BigInteger deadline = BigInteger.valueOf(1671096761);
6.      DDC721Service ddc721Service = new DDC721ServiceImpl();
7.      BigInteger nonce = ddc721Service.getNonce(accountList.get(2));
8.      DDC721MetaTransaction metaTransaction = new DDC721MetaTransaction();
9.      String digest = metaTransaction.getSafeTransferFromDigest(accountList.ge
    t(1), accountList.get(2), accountList.get(2), ddcId, memo, nonce, deadline);
10.     String signature = metaTransaction.generateSignature(MetaTrxConfig.getPr
    ivateKey(), digest.getBytes(StandardCharsets.UTF_8));
11.     PushedTransaction pt = ddc721Service.metaSafeTransferFrom(accountList.ge
    t(1), accountList.get(2), accountList.get(2), ddcId, memo, nonce, deadline, s
    ignature);
12.     System.out.println(JSONUtil.toJsonStr(pt));
13. }
```

4.2.3.19 元交易销毁

调用方法实例

```
1.  @Test
2.  public void metaBurn721() {
3.      BigInteger ddcId = BigInteger.valueOf(1);
4.      BigInteger deadline = BigInteger.valueOf(1671096761);
5.      DDC721Service ddc721Service = new DDC721ServiceImpl();
6.      BigInteger nonce = ddc721Service.getNonce(accountList.get(2));
7.      DDC721MetaTransaction metaTransaction = new DDC721MetaTransaction();
8.      String digest = metaTransaction.getBurnDigest(accountList.get(1), accoun
    tList.get(2), ddcId, nonce, deadline);
9.      String signature = metaTransaction.generateSignature(MetaTrxConfig.getPr
    ivateKey(), digest.getBytes(StandardCharsets.UTF_8));
10.     PushedTransaction pt = ddc721Service.metaBurn(accountList.get(1), accoun
    tList.get(2), ddcId, nonce, deadline, signature);
11.     System.out.println(JSONUtil.toJsonStr(pt));
12. }
```

4.2.4 BSN-DDC-1155

4.2.4.1 生成

调用方法实例

```
1.  /**
2.   * 1155 生成
3.   */
```

```
@Test
public void mint1155() throws InterruptedException {
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    PushedTransaction pt = ddc1155Service.safeMint(accountList.get(1), accountList.get(2), BigInteger.valueOf(10), "https://bitnodes.io/0001", UUID.randomUUID().toString());
    System.out.println(JSONUtil.parse(pt));
}
```

4.2.4.2 批量生成

调用方法实例

```
/**
 * 1155 批量生成
 */
@Test
public void mintBatch() throws InterruptedException {
    List<BigInteger> amounts = Arrays.asList(BigInteger.valueOf(10), BigInteger.valueOf(20));
    List<String> ddcURIS = Arrays.asList("https://bitnodes.io/0001", "https://bitnodes.io/0002");
    DDC1155ServiceImpl ddc1155Service = new DDC1155ServiceImpl();
    PushedTransaction pt = ddc1155Service.safeMintBatch(accountList.get(1), accountList.get(2), amounts, ddcURIS, UUID.randomUUID().toString());
    System.out.println(JSONUtil.parse(pt));
}
```

4.2.4.3 转移

调用方法实例：

```
/**
 * 1155 转移
 */
@Test
public void transferFrom1155() throws InterruptedException {
    ChainConfig.setPk("5JAT6ZYDhvvWVP2UMat84BhCyu5U3PYMYtkzqXEgoPoyhdJJfV1");

    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    PushedTransaction pt = ddc1155Service.transferFrom(accountList.get(2), accountList.get(2), accountList.get(1), BigInteger.valueOf(1), BigInteger.valueOf(2), UUID.randomUUID().toString());
    System.out.println(JSONUtil.parse(pt));
}
```

4.2.4.4 批量转移

调用方法实例:

```
/**
 * 1155 批量转移
 */
@Test
public void safeBatchTransferFrom() throws InterruptedException {
    ChainConfig.setPk("5JAT6ZYDhvvWVP2UMat84BhCyu5U3PYMYtkzqXEgoPoyhdJJfV1");

    List<BigInteger> ddcIds = Arrays.asList(BigInteger.valueOf(8), BigInteger.valueOf(9));
    List<BigInteger> amounts = Arrays.asList(BigInteger.valueOf(2), BigInteger.valueOf(3));
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    PushedTransaction pt = ddc1155Service.safeBatchTransferFrom(accountList.get(2), accountList.get(2), accountList.get(1), ddcIds, amounts, UUID.randomUUID().toString());
    System.out.println(JSONUtil.parse(pt));
}
```

4.2.4.5 冻结

调用方法实例:

```
/**
 * 1155 冻结
 */
@Test
public void freeze1155() {
    //使用运营方账户私钥
    ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    PushedTransaction pt = ddc1155Service.freeze(accountList.get(0), BigInteger.valueOf(10));
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.4.6 解冻

调用方法实例:

```
/**
```

```
 * 1155 解冻
 */
@Test
public void unFreeze1155() {
    //使用运营方账户私钥
    ChainConfig.setPk("5KadYhvnCZXJNCBurWpyDRDfJzyM83m4hzRb4mujRZm1pLsseZH");

    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    PushedTransaction pt = ddc1155Service.unFreeze(accountList.get(0), BigInteger.valueOf(10));
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.4.7 销毁

调用方法实例：

```
/**
 * 1155 销毁
 */
@Test
public void burn1155() throws InterruptedException {
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    PushedTransaction pt = ddc1155Service.burn(accountList.get(1), accountList.get(1), BigInteger.valueOf(1));
    System.out.println(JSONUtil.parse(pt));
}
```

4.2.4.8 批量销毁

调用方法实例：

```
/**
 * 1155 批量销毁
 */
@Test
public void burnBatch1155() throws InterruptedException {
    List<BigInteger> ddcIds = Arrays.asList(BigInteger.valueOf(8), BigInteger.valueOf(9));
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    PushedTransaction pt = ddc1155Service.burnBatch(accountList.get(1), accountList.get(1), ddcIds);
    System.out.println(JSONUtil.parse(pt));
}
```

4.2.4.9 查询数量

调用方法实例：

```
/**
 * 1155 查询数量
 */
@Test
public void balanceOf1155() {
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
     System.out.println(ddc1155Service.balanceOf(accountList.get(2), BigInteger.valueOf(10)));
}
```

4.2.4.10 批量查询数量

方法调用实例：

```
/**
 * 1155 批量查询数量
 */
@Test
public void balanceOfBatch1155() {
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
     ArrayListMultimap<String, BigInteger> ddcs = ArrayListMultimap.create();

    ddcs.put(accountList.get(2), BigInteger.valueOf(10));
    ddcs.put(accountList.get(2), BigInteger.valueOf(11));
    System.out.println(ddc1155Service.balanceOfBatch(ddcs));
}
```

4.2.4.11 账户授权及查询

调用方法实例：

```
/**
 * 1155 账户授权
 */
@Test
public void approveall1155() {
    ChainConfig.setPk("5JAT6ZYDhvvWVP2UMat84BhCyu5U3PYMYtkzqXEgoPoyhdJJfV1");

    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
     PushedTransaction pt = ddc1155Service.setApprovalForAll(accountList.get(2), accountList.get(1), false);
```

```
        System.out.println(JSONUtil.parse(pt));
    }
    /**
     * 1155 账户授权查询
     */
    @Test
    public void isApprovedForAll1155() {
        DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
        boolean flag = ddc1155Service.isApprovedForAll(accountList.get(2), accou
ntList.get(1));
        System.out.println(flag);
    }
```

4.2.4.12 获取 DDCURI

调用方法实例：

```
    /**
     * 1155 获取 DDCURI
     */
    @Test
    public void ddcURI() {
        DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
        System.out.println(ddc1155Service.ddcURI(BigInteger.valueOf(10)));
    }
```

4.2.4.13 设置 DDCURI

调用方法实例：

```
    /**
     * 1155 设置 DDCURI
     */
    @Test
    public void setUri() {
        DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
        ddc1155Service.setURI(accountList.get(1), accountList.get(1), BigInteger.
valueOf(2), "https://bitnodes.io/0001");
    }
```

4.2.4.14 符号名称设置

方法调用实例：

```
    @Test
```

```
2.  public void setNameSym() {
3.      DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
4.      ddc721Service.setNameAndSymbol("ddcname", "ddcsymbol");
5.  }
```

4.2.4.15 最新 DDCID 查询

方法调用实例:

```
1.  /**
2.   * 1155 最新 DDCID 查询
3.   */
4.  @Test
5.  public void getLatestDDCId() {
6.      DDC1155ServiceImpl ddc1155Service = new DDC1155ServiceImpl();
7.      System.out.println(ddc1155Service.getLatestDDCId());
8.  }
```

4.2.4.16 查询指定账户 1155 资产

方法调用实例:

```
1.  @Test
2.  public void assetsOf1155() {
3.  String platformAccount = "platform";
4.      DDC1155ServiceImpl ddc1155Service = new DDC1155ServiceImpl();
5.      System.out.println(ddc1155Service.assetsOf1155(platformAccount));
6.  }
```

4.2.4.17 Nonce 查询

调用方法实例

```
1.  @Test
2.  public void getNonce() {
3.      DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
4.      BigInteger nonce = ddc1155Service.getNonce(accountList.get(2));
5.      System.out.println("nonce: " + nonce);
6.  }
```

4.2.4.18 元交易生成

调用方法实例

```
1.  @Test
```

```
2.  public void metaSafeMint1155() {
3.      BigInteger amount = BigInteger.valueOf(10);
4.      String ddcURI = "https://bitnodes.io/0001";
5.      String memo = "memo0001";
6.      BigInteger deadline = BigInteger.valueOf(1671096761);
7.      DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
8.      BigInteger nonce = ddc1155Service.getNonce(accountList.get(2));
9.      DDC1155MetaTransaction metaTransaction = new DDC1155MetaTransaction();
10.     String digest = metaTransaction.getSafeMintDigest(accountList.get(1), ac
    countList.get(2), accountList.get(2), amount, ddcURI, memo, nonce, deadline);

11.     String signature = metaTransaction.generateSignature(MetaTrxConfig.getPr
    ivateKey(), digest.getBytes(StandardCharsets.UTF_8));
12.     PushedTransaction pt = ddc1155Service.metaSafeMint(accountList.get(1), a
    ccountList.get(2), accountList.get(2), amount, ddcURI, memo, nonce, deadline,
     signature);
13.     System.out.println(JSONUtil.toJsonStr(pt));
14. }
```

4.2.4.19 元交易批量生成

调用方法实例

```
1.  @Test
2.  public void metaSafeMintBatch1155() {
3.      List<BigInteger> amounts = Arrays.asList(BigInteger.valueOf(20), BigInte
    ger.valueOf(30));
4.      List<String> ddcURIs = Arrays.asList("http://ddcUrl1", "http://ddcUrl2");

5.      String memo = "memo0001";
6.      BigInteger deadline = BigInteger.valueOf(1671096761);
7.      DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
8.      BigInteger nonce = ddc1155Service.getNonce(accountList.get(2));
9.      DDC1155MetaTransaction metaTransaction = new DDC1155MetaTransaction();
10.     String digest = metaTransaction.getSafeMintBatchDigest(accountList.get(1)
    , accountList.get(2), accountList.get(2), amounts, ddcURIs, memo, nonce, dead
    line);
11.     String signature = metaTransaction.generateSignature(MetaTrxConfig.getPr
    ivateKey(), digest.getBytes(StandardCharsets.UTF_8));
12.     PushedTransaction pt = ddc1155Service.metaSafeMintBatch(accountList.get(
    1), accountList.get(2), accountList.get(2), amounts, ddcURIs, memo, nonce, de
    adline, signature);
13.     System.out.println(JSONUtil.toJsonStr(pt));
14. }
```

4.2.4.20 元交易转移

调用方法实例

```
@Test
public void metaSafeTransferFrom1155() {
    BigInteger ddcId = BigInteger.valueOf(1);
    BigInteger amount = BigInteger.valueOf(2);
    String memo = "memo0001";
    BigInteger deadline = BigInteger.valueOf(1671096761);
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    BigInteger nonce = ddc1155Service.getNonce(accountList.get(2));
    DDC1155MetaTransaction metaTransaction = new DDC1155MetaTransaction();
    String digest = metaTransaction.getSafeTransferFromDigest(accountList.get(1), accountList.get(2), accountList.get(2), ddcId, amount, memo, nonce, deadline);
    String signature = metaTransaction.generateSignature(MetaTrxConfig.getPrivateKey(), digest.getBytes(StandardCharsets.UTF_8));
    PushedTransaction pt = ddc1155Service.metaSafeTransferFrom(accountList.get(1), accountList.get(2), accountList.get(2), ddcId, amount, memo, nonce, deadline, signature);
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.4.21 元交易批量转移

调用方法实例

```
@Test
public void metaSafeBatchTransferFrom1155() {
    List<BigInteger> ddcIds = Arrays.asList(BigInteger.valueOf(2), BigInteger.valueOf(3));
    List<BigInteger> amounts = Arrays.asList(BigInteger.valueOf(2), BigInteger.valueOf(3));
    String memo = "memo0001";
    BigInteger deadline = BigInteger.valueOf(1671096761);
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    BigInteger nonce = ddc1155Service.getNonce(accountList.get(2));
    DDC1155MetaTransaction metaTransaction = new DDC1155MetaTransaction();
    String digest = metaTransaction.getSafeBatchTransferFromDigest(accountList.get(1), accountList.get(2), accountList.get(2), ddcIds, amounts, memo, nonce, deadline);
    String signature = metaTransaction.generateSignature(MetaTrxConfig.getPrivateKey(), digest.getBytes(StandardCharsets.UTF_8));
```

```
    PushedTransaction pt = ddc1155Service.metaSafeBatchTransferFrom(accountList.get(1), accountList.get(2), accountList.get(2), ddcIds, amounts, memo, nonce, deadline, signature);
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.4.22 元交易销毁

调用方法实例

```
@Test
public void metaBurn1155() {
    BigInteger ddcId = BigInteger.valueOf(4);
    BigInteger deadline = BigInteger.valueOf(1671096761);
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    BigInteger nonce = ddc1155Service.getNonce(accountList.get(2));
    DDC1155MetaTransaction metaTransaction = new DDC1155MetaTransaction();
    String digest = metaTransaction.getBurnDigest(accountList.get(1), accountList.get(2), ddcId, nonce, deadline);
    String signature = metaTransaction.generateSignature(MetaTrxConfig.getPrivateKey(), digest.getBytes(StandardCharsets.UTF_8));
    PushedTransaction pt = ddc1155Service.metaBurn(accountList.get(1), accountList.get(2), ddcId, nonce, deadline, signature);
    System.out.println(JSONUtil.toJsonStr(pt));
}
```

4.2.4.23 元交易批量销毁

调用方法实例

```
@Test
public void metaBurnBatch1155() {
    List<BigInteger> ddcIds = Arrays.asList(BigInteger.valueOf(5), BigInteger.valueOf(6));
    BigInteger deadline = BigInteger.valueOf(1671096761);
    DDC1155Service ddc1155Service = new DDC1155ServiceImpl();
    BigInteger nonce = ddc1155Service.getNonce(accountList.get(2));
    DDC1155MetaTransaction metaTransaction = new DDC1155MetaTransaction();
    String digest = metaTransaction.getBurnBatchDigest(accountList.get(1), accountList.get(2), ddcIds, nonce, deadline);
    String signature = metaTransaction.generateSignature(MetaTrxConfig.getPrivateKey(), digest.getBytes(StandardCharsets.UTF_8));
    PushedTransaction pt = ddc1155Service.metaBurnBatch(accountList.get(1), accountList.get(2), ddcIds, nonce, deadline, signature);
    System.out.println(JSONUtil.toJsonStr(pt));
```

```
12. }
```

5 附录

5.1 区块信息示例

- **EOS**

参考:

https://bloks.io/block/223545017

5.2 交易信息示例

- **EOS**

参考:

https://bloks.io/transaction/799a4a9271b67329dd8e49eea882c0bbb1dc704c2fa9686600b934db5bffadb1?tab=raw

```
- {
    "status": "executed",
    "cpu_usage": 1174,
    "net_usage": 144,
    "id": "799a4a9271b67329dd8e49eea882c0bbb1dc704c2fa9686600b934db5bffadb1",
    "block_time": "2021-12-31T07:57:47.500",
    "block_num": 223545017,
    "delay_sec": undefined,
    "expiration": undefined,
  -  "actions": [
   -    {
     -      "account_ram_deltas": [
            ],
     -      "act": {
               "account": "push.sx",
      +        "authorization": [ /* 2 items */ ],
      +        "data": { /* 2 items */ },
               "hex_data": "40c255d3495d51c70500000000000000",
               "name": "mine"
            },
            "action_ordinal": 1,
            "block_num": 223545017,
            "block_time": "2021-12-31T07:57:47.500",
            "closest_unnotified_ancestor_action_ordinal": 0,
            "context_free": false,
            "creator_action_ordinal": 0,
            "elapsed": 257,
            "producer_block_id": "0d5306b91066579c88425d869d5e3380b3b44ba0f3a98fbed6a87ec0c3cc11e6",
     -      "receipt": {
               "abi_sequence": 29,
               "act_digest": "0f5509389f27e7c95c8c805ddd437468d5037c7b6e41033d27a822a50e060cf6",
      +        "auth_sequence": [ /* 2 items */ ],
               "code_sequence": 162,
               "global_sequence": 355140632570,
               "receiver": "push.sx",
               "recv_sequence": 131468092
            },
            "receiver": "push.sx",
            "trx_id": "799a4a9271b67329dd8e49eea882c0bbb1dc704c2fa9686600b934db5bffadb1",
     -      "inline_traces": [
      +        { /* 14 items */ },
      +        { /* 13 items */ },
```