BSN-DDC 基础网络

Solidity 合约详细设计

V2.0

北京红枣科技有限公司
2021 年 12 月

修改记录

日期	版本	修改说明	修改者
2021.12.01	v1.0	版本初始化	
2021.12.03	v1.0	1. 721 业务主合约所对应的数据合约及逻辑合约章节补充。 2. 1155 业务主合约所对应的数据合约及逻辑合约章节补充。 3. 计费合约所对应的数据合约及逻辑合约章节补充。 4. 权限合约所对应的数据合约及逻辑合约章节补充。	
2021.12.04	v1.0	添加获取符号方法	
2021.12.06	v1.0	1. 计费逻辑合约为充值账户充值添加事件。 2. 添加终端用户方法以及添加权限合约事件 3. 针对 721、1155 的事件部分定义统一	
2021.12.11	v1.0	1. 优化合约整体结构以及相关交易时序图; 2. 文档部分内容进行了统一;	
2021.12.13	v1.0	1. 721 添加 lastDDCID 方法 2. 1155 添加 lastDDCID 方法	
2021.12.16	v1.0	1. 细化了计费合约、权限合约、721 以及 1155 所对应的逻辑合约所对应的核心逻辑说明。	
2021.12.20	v1.0	1. 权限逻辑合约的部分接口添加了账户状态检查。	
2021.12.27	v1.0	1. 1155 业务主逻辑合约的销毁和批量销毁添加了权限控制逻辑。	
2021.12.30	v1.0	1. 1155 数据合约添加了 DDCID 列表字段以及 DDCID 是否存在接口。 2. 1155 逻辑合约生成 DDC、批量生成 DDC、安全转移、批量安全转移、冻结、解冻、销毁以及批	

		量销毁添加了 DDCID 是否存在检查，并对销毁和批量销毁的调用者、入参以及逻辑说明进行了更新。 3. 721 和 1155 业务主逻辑合约针对冻结和解冻扣除业务费的说明进行了删除。	
2022.1.10	V1.0	1. 721 业务主逻辑合约添加安全生成方法。 2. 1155 生成和批量生成添加元数据参数。 3. 721 和 1155 业务主逻辑合约账户授权查询添加参数检查说明。 4. 721 业务主逻辑合约安全转移元数据参数类型更新。 5. 1155 业务主逻辑合约销毁和安全销毁参数说明、参数以及核心逻辑进行了更新。	
2022.1.11	V1.0	1. 3.6 改为 UUPS 代理模式。	
2022.1.11	V1.0	1. 修改 3.1 的合约整体结构。 2. 合约整体结构调整，逻辑合约和数据合约进行了合并，并优化了所有方法的核心逻辑说明； 3. 计费合约删除了合约初始化方法，更新了删除 DDC 业务费计费规则方法和删除 DDC 业务主合约授权方法的方法名和事件定义，同时部分方法的参数名的也有细微的调整； 4. 权限合约删除了同属 leader 检验和 leader 检验方法，并添加了同平台检验方法，以及更新了账户状态检查方法和查询方法所对应的方法名，以及添加账户定义为平台方添加账户，添加终端用户定义为运营方添加账户。	
2022.1.17	V1.0	1. 1155 生成和批量生成方法名和名称进行重新定义，换成了安全生成和批量安全生成。	
2022.1.18	V1.0	1. 权限合约添加了账户 DID 授权数据结构以及跨平	

		台检验和跨平台授权方法。 2. 721 和 1155 添加了 URI 变更方法。 3. 721 和 1155 的 DDC 对应的转移、安全转移以及批量安全转移的核心逻辑对应的同平台检验进行了修订。 4. 1155 销毁和批量销毁添加 DDC 授权者也可以调用，同时添加了对 owner 参数进行检查。 5. 1155 的 URI 设置对应的逻辑进行了微调，并添加了 DDC 授权者也可以进行调用。	
2022.1.20	V1.0	1. DDC 业务费扣除事件通知添加 ddcId 字段。	
2022.1.21	V1.0	1. 跨平台授权中授权者和接收者账户添加了必须是平台方限制。	
2022.1.22	V1.0	1. 721 添加名称符号设置方法	
2022.2.15	V1.1	**计费合约**: 1. 修改了链账户余额查询方法所对应的关键字 constant 为 view。 2. 添加了批量充值和批量链账户余额查询接口。 3. 将普通用户描述改成了终端用户。 **权限合约**: 1. 数据结构添加了平台方 DID 集合字段。 2. 添加了平台方批量添加账户、运营方批量添加账户、平台方添加链账户开关设置以及平台方添加链账户开关查询接口。 3. 运营方添加账户添加了 DID 的逻辑控制，对输入参数的描述进行了更新。 4. 平台方添加链账户添加开关状态检验查询，对输入参数的描述进行了更新; **721 业务主合约** 1. 安全转移方法举例中附加数据缺少 memory 关键	

		字。 2. 生成、安全生成、授权、账户授权、安全转移、转移核心逻辑中指定的章节编号进行了更新。 **1155 业务主合约** 1. 安全生成、批量安全生成、账户授权、安全转移、批量安全转移核心逻辑中指定的章节编号进行了更新。 **公共修改点** 1. 除了查询类接口，去掉了返回参数。 2. 所有的批量接口，数组参数都添加了 memory 关键字。 3. 拥用者描述改成了拥有者;	
2022.2.23	V1.1	1. 修改了平台方或运营方批量添加账户的核心逻辑; 2. 更新了计费合约批量充值的事件名的定义;	
2022.2.28	V1.1	1. 所有查询接口返回都添加了 view 关键字，返回值为数组类型都添加 memory 关键字。 2. 更新了计费批量充值的核心逻辑的描述顺序。 3. 721 和 1155 的 URI 设置完善了部分逻辑。 4. 1155 的 DDCURI 设置、销毁和批量销毁去掉了对 owner 检查。	
2022.3.3	V1.1	1. 1155 的销毁和批量销毁对应的事件中 operator 和 owner 参数位置进行了调整。 2. 权限合约数据结构添加了平台方添加链账户开关字段以及添加了同步平台方 DID 接口。	
2022.3.30	V1.1	1. 1155 的批量销毁的核心逻辑部分检查进行了修订。 2. 1155 的批量安全生成方法命名定义错误，并进行了修正。	

		3. 1155 的 DDCURI 设置和最新 DDCID 查询章节序号进行了修改。 4. 权限合约数据结构中针对未补充的值进行了添加。 5. 权限合约针对运营方添加账户的核心逻辑中注意事项进行了更新。	
2022.3.30	v1.1	1. 计费合约的充值和批量充值的核心逻辑进行了优化; 2. 计费合约的批量充值；1155 合约的批量安全生成、批量安全转移以及批量销毁；权限合约的平台方批量添加账户和运营方批量添加账户的核心逻辑添加了对集合长度检查;	
2022.08.02	V2.0	1. 721 和 1155 添加了 DDC 锁定列表数据结构以及跨链锁定和解锁方法; 2. 721 对应的 DDC 授权、安全转移、转移、元交易转移以及 ddcURI 设置方法添加了对 DDC 是否锁定检查; 3. 1155 对应的安全转移、批量安全转移、元交易安全转移以及 ddcURI 设置方法添加了对 DDC 是否锁定检查;	
2022.08.03	V2.0	1. 721 合约数据结构添加了 Nonce 列表、域名分割符、授权类型哈希值列表，并添加了批量生成、批量安全生成、元交易生成、元交易安全生成、元交易批量生成、元交易批量安全生成、元交易销毁、元交易转移、元交易安全转移、Nonce 查询方法、授权哈希设置以及分隔符设置接口; 2. 1155 合约数据结构添加了添加了 Nonce 列表、域名分割符、授权类型哈希值列表字段，并添加了元交易安全生成、元交易批量安全生成、元交	

		易安全转移、元交易批量安全转移、元交易销毁、元交易批量销毁、Nonce 查询接口、授权哈希设置以及分隔符设置接口；	
2022.08.19	V2.0	1. 1155 合约数据结构添加 DDC 拥有者列表字段，并添加查询拥有者接口；	
2022.08.20	V2.0	1. 1155 数据结构补充 DDC 拥有者账户索引列表以及同步拥有者接口； 2. 计费合约的数据结构与合约代码保持了统一； 3. 计费合约和权限合约添加是否启动批量开关设置；	

目录

修改记录 2

1. 编写目的 8

2. 需求文档 8

3. 整体设计 9

3.1 合约整体结构 9

3.2 DDC 业务交易时序图 9

3.3 计费充值交易时序图 10

3.4 账户管理交易时序图 10

3.5 安全性设计说明 11

3.6 合约更新设计说明 11

4. 合约设计 11

4.1 BSN-DDC-计费合约 11

4.1.1 功能介绍 12

4.1.2 数据结构 12

4.1.3 API 定义 13

4.2 BSN-DDC-721 业务主合约 21
4.2.1 功能介绍 21
4.2.2 数据结构 21
4.2.3 API 定义 22
4.3 BSN-DDC-1155 业务主合约 44
4.3.1 功能介绍 44
4.3.2 数据结构 45
4.3.3 API 定义 46
4.4 BSN-DDC-权限合约 67
4.4.1 功能介绍 67
4.4.2 数据结构 67
4.4.3 API 定义 69

1. 编写目的

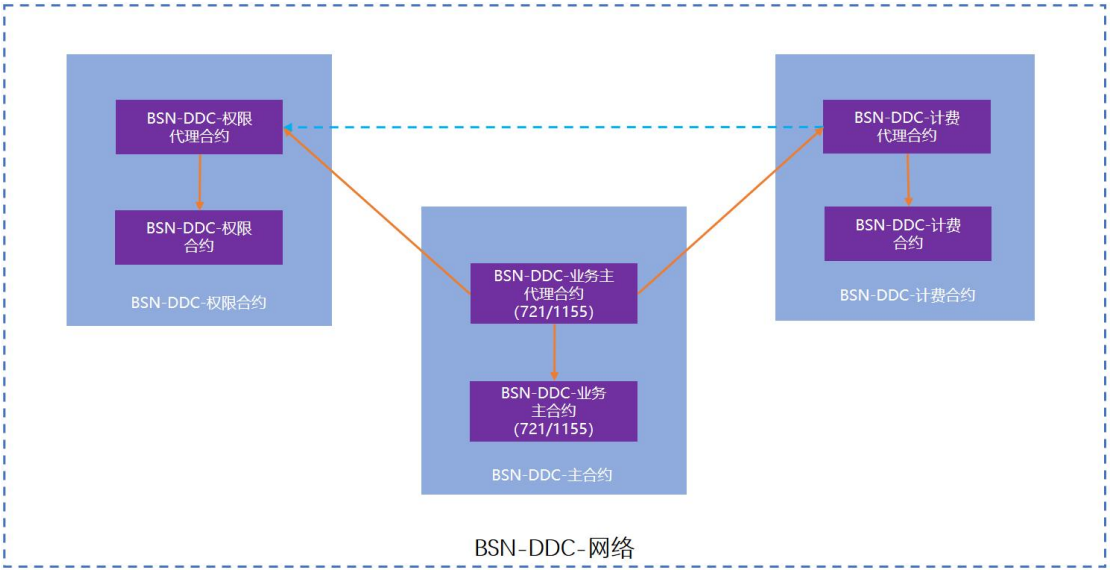
为了让项目组成员以及各开放链盟链框架方对 BSN-DDC 合约的整体设计有一个全面详细的了解，同时为项目的开发、测试、验证、交付等环节提供原始依据以及开发指导，特此整理 BSN-DDC 合约整体设计规范方案说明文档。

2. 需求文档

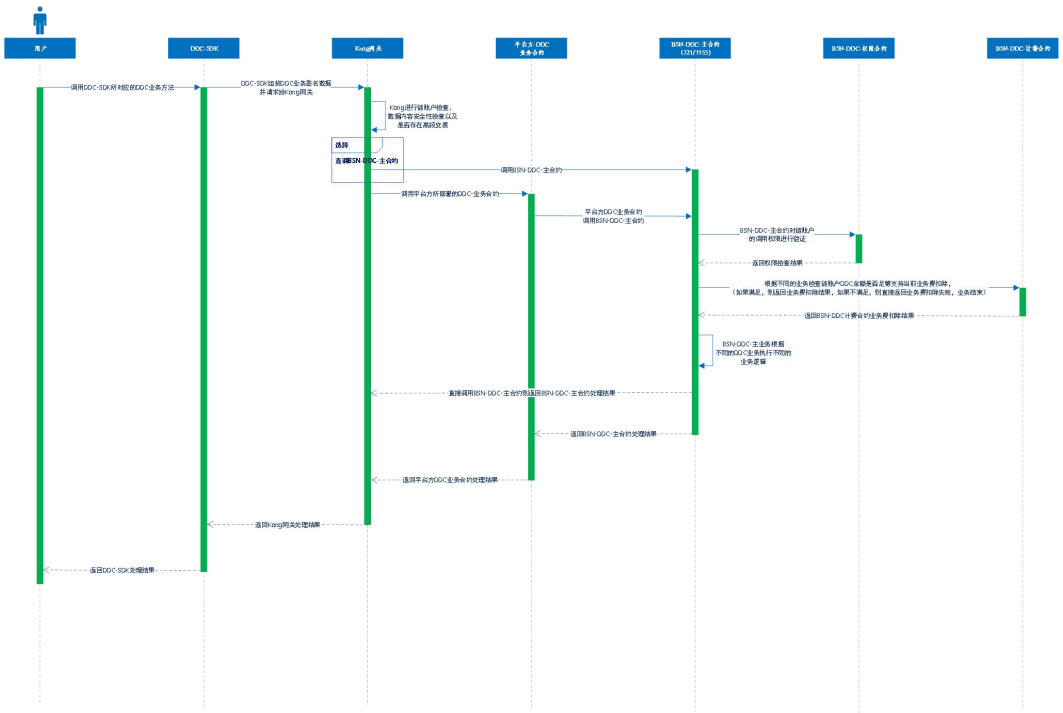
需求文档引用 BSN-DDC_需求说明书 V1.1.docx

3. 整体设计

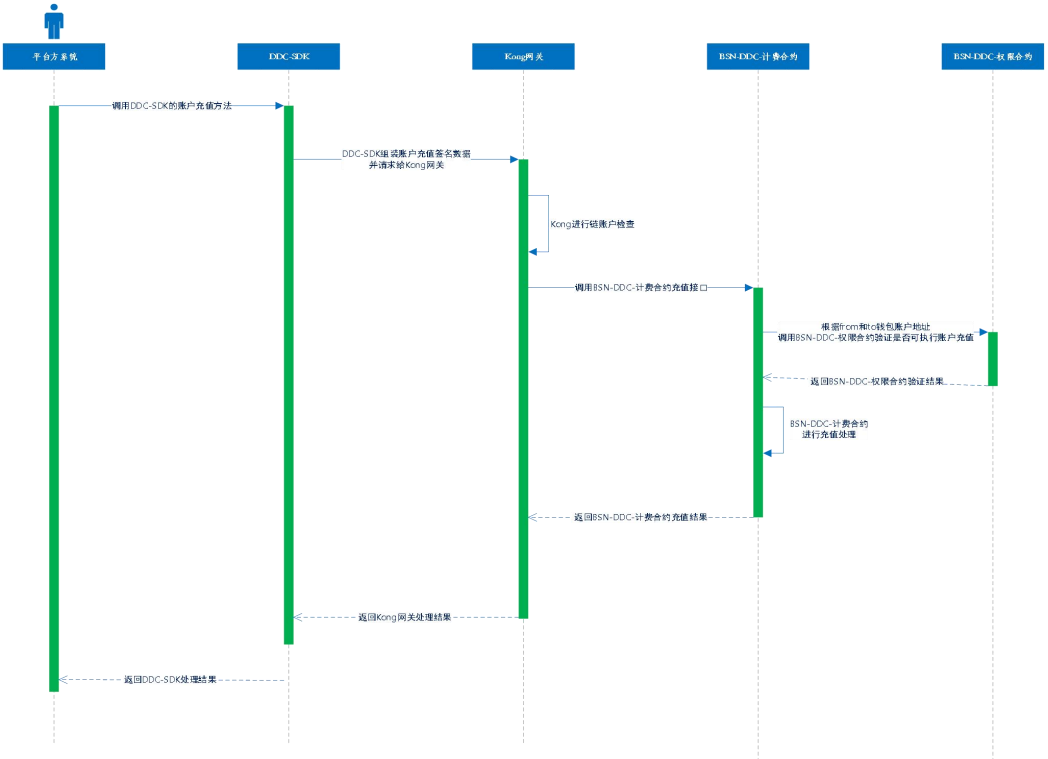
3.1 合约整体结构



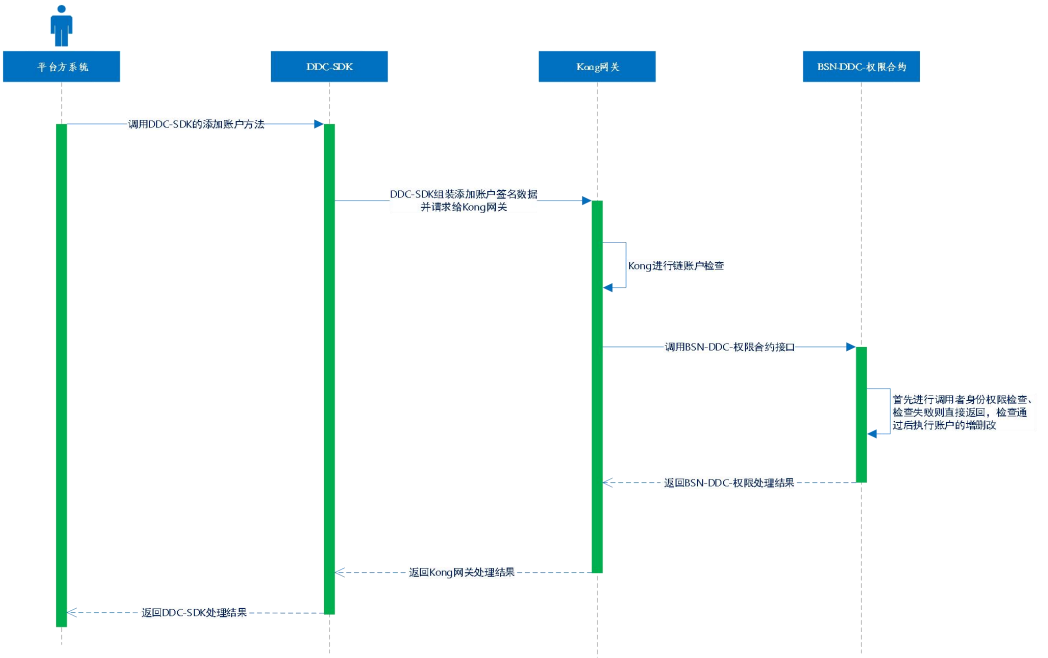
3.2 DDC 业务交易时序图



3.3 计费充值交易时序图



3.4 账户管理交易时序图



3.5 安全性设计说明

BSN-DDC 合约的整体设计目前采用 2 层设计模式，分别是代理合约和业务合约，业务合约只允许与之对应的代理合约进行调用，如：BSN-DDC-计费合约只允许 BSN-DDC-计费代理合约进行调用。

注：业务合约都需要定义相应的接口，业务处理在业务合约中进行实现。

3.6 合约更新设计说明

BSN-DDC 通过 UUPS（EIP-1822: Universal Upgradeable Proxy Standard）模式实现其业务合约的可升级。每一个 BSN-DDC 的业务合约都有一个代理合约，每个代理合约的 Owner 都归 BSN-DDC 运营方所有。平台方或平台方业务合约通过调用代理合约访问其对应的业务合约。

- 业务合约的修改如下：

1. 继承 UUPSUpgradeable。该类库实现了 UUPS 代理设计的可升级机制。
2. 添加初始化方法 initialize()。用于代理合约部署时调用以进行合约的初始化操作。

- 业务合约的部署过程如下：

1. 部署业务合约。
2. 部署代理合约。部署时构造传参写入业务合约地址、initialize 的方法签名，实现其与业务合约的映射以及初始化操作。

- 业务合约的升级过程如下：

1. 部署新版本的业务合约。
2. 调用当前代理合约中的 upgradeTo 方法。执行时传入新的业务合约地址，实现其与新版本业务合约的映射，达到升级的目的。

4. 合约设计

4.1 BSN-DDC-计费合约

业务费：用于定义 BSN-DDC 业务费所对应的费用计量单位名称。

4.1.1 功能介绍

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

各参与方的链上账户类型包含以下：

1. 运营方：运营方在计费合约中所对应的账户。
2. 平台方：平台方在计费合约中所对应的账户。
3. 终端用户：终端用户在计费合约中所对应的账户，每个终端用户只能所属于一个平台方。

4.1.2 数据结构

➢ 计费和账户存储结构

编号	字段名	字段	类型	备注
1.	总量	_total	uint256	业务费总量
2.	计费规则列表	_ddcFees	mapping(address=>FuncFee)	Key：业务主代理合约地址 Value：计费规则
3.	账户余额列表	_balances	mapping(address=>uint256)	Key：账户地址 Value：账户余额
4.	是否启用批量	_enableBatch	bool	
FuncFee				
1.	方法 sig 所对应的业务费	funcfee	mapping(bytes4 => uint32)	调用 BSN-DDC-业务主代理合约方法 sig 和对应的业务费
2.	业务主代理合约状态	used	bool	业务主代理合约的删除状态

➢ 用户账户

编号	字段名	字段	类型	备注
1.	账户地址	key	address	用户账户地址

2.	账户余额	value	uint32	用户账户余额

4.1.3 API 定义

4.1.3.1 充值

运营方、平台方用该 API 将账户的 DDC 业务费充值给下级用户。

- 输入参数：接收者账户、业务费额；
- 输出参数：无；
- 方法命名：recharge;
- 方法举例：recharge(address to,uint256 amount);
- 事件：Recharge(address indexed from,address indexed to,uint256 amount);
- 核心逻辑
  - 检查充值的业务费额是否等于 0，是则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查转出者账户与接收者账户是否相同，相同则返回提示信息；检查转出者账户状态是否活跃，不活跃则返回提示信息；
  - 检查接收者账户状态是否活跃，不活跃则返回提示信息；
  - 检查转者者账户和接收者账户类型是否为终端用户(禁止终端用户之间的业务费相互流转)或检查转出者账户和接收者账户类型是否为平台方且转出者账户 DID 和接收者账户 DID 不相同(禁止平台方之间的业务费相互流转)或接收者账户类型是运营方，如果满足任何一个则返回提示信息；
  - 检查转出者账户所对应的余额是否大于或等于充值的业务费额，不是则返回提示信息；
  - 所有检查通过后则保存充值结果，最后触发事件；
- 业务规则
  - 运营方可以充值给平台方以及终端用户；
  - 平台方可以充值所属自己平台的用户，不可以充值给其他平台方以及其他平台方所属的用户；

- 用户不能调用该方法充值给任何账户；

- 业务场景
  - 平台方在运营方处充值，运营方调用充值 API 给平台方充值；
  - 用户在某平台方充值，平台方调用充值 API 给该用户充值；

4.1.3.2 批量充值

运营方、平台方可以通过调用该 API 将账户的 DDC 业务费批量充值给下级用户。

- 输入参数：接收者账户集合、业务费额集合；
- 输出参数：无；
- 方法命名：rechargeBatch;
- 方法举例：rechargeBatch(address[] memory toList,uint256[] memory amounts);
- 事件：RechargeBatch(address indexed from,address[] toList,uint256[] amounts);
- 核心逻辑
  - 检查是否启用批量开关设置，是则返回提示信息；
  - 检查接收者账户集合长度是否为 0，是则返回提示信息；
  - 检查业务费集合长度是否为 0，是则返回提示信息；
  - 检查接收者账户集合长度与业务费集合长度是否相等，不是则返回提示信息；
  - 检查转出者账户状态是否活跃，不活跃则返回提示信息；
  - 批量循环挨个处理链账户业务费充值，处理逻辑如下：
    1. 根据索引检查充值的业务费额是否等于 0，是则返回提示信息；
    2. 根据索引检查接收者账户地址是否为 0 地址，是则返回提示信息；
    3. 根据索引检查接收者账户与转出者账户是否相同，相同则返回提示信息；
    4. 根据索引检查接收者账户状态是否活跃，不活跃则返回提示信息；

5. 根据索引检查转者者账户和接收者账户类型是否为终端用户(禁止终端用户之间的业务费相互流转)或检查转出者账户和接收者账户类型是否为平台方且转出者账户 DID 和接收者账户 DID 不相同(禁止平台方之间的业务费相互流转)或接收者账户类型是运营方，如果满足任何一个则返回提示信息；
6. 根据索引检查转出者账户所对应的余额是否大于或等于充值的业务费额，不是则返回提示信息；
7. 所有检查通过后则保存充值结果；

- 最后触发事件；

4.1.3.3 运营账户充值

在业务费总量不足以满足当前业务时，运营方可以充值业务费总量。

- 输入参数：充值业务费；
- 输出参数：无；
- 调用者：
- 方法命名：selfRecharge;
- 方法举例：selfRecharge(uint amount);
- 事件：Recharge(address indexed from,address indexed to,uint256 amount);
- 核心逻辑：
  - 检查充值业务费是否等于 0，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户所对应的身份是不是运营方，不是返回提示信息；
  - 所有检查通过后则为运营方账户添加相应的金额，最后触发事件（from 为空地址，to 为调用者地址）；
- 业务规则：只允许给运营方账户充值业务费总量；

4.1.3.4 DDC 业务费扣除

BSN-DDC-业务主代理合约内调用该 API，传入用户地址以及合约标识

（sig），将用户账户余额按照设置计费规则将费用转给运营账户。

- 输入参数：转出者账户，方法签名；
- 输出参数：无；
- 方法命名：pay;
- 方法举例：pay(address payer,bytes4 sig,uint256 ddcId);
- 事件：Pay(address indexed payer,address indexed payee,bytes4 sig,uint32 amount,uint256 ddcId);
- 核心逻辑：
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查业务主代理合约地址所对应的计费规则是否存在或是否可用，不存在则返回提示信息；
  - 检查业务主代理合约和方法 sig 所对应的业务费是否大于 0，不是则直接触发事件；
  - 检查转出者账户所对应的余额是否大于或等于所扣除的业务费，不是则返回提示信息；
  - 所有检查通过后则扣除转出者余额，并充值给业务主代理合约账户，最后触发事件；
- 业务规则：该方法只允许被添加的 DDC 业务主代理合约内部调用；

4.1.3.5 DDC 合约结算

运营账户调用该合约方法，对授权的 DDC 业务主代理合约账户发起结算。

- 输入参数：结算账户，结算金额；
- 输出参数：无；
- 方法命名：settlement;
- 方法举例：settlement(address ddcAddr,uint256 amount);
- 事件：Settlement(address indexed accAddr,address indexed ddcAddr,uint256 amount);
- 核心逻辑：
  - 检查结算金额是否大于 0，不是则返回提示信息；

- 检查结算账户是否为一个合约且结算账户是否为业务主代理合约且在计费规则中是否存在，不是则返回提示信息；
- 检查调用者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户所对应的身份是不是运营方，不是则返回提示信息；
- 检查结算账户所对应的余额是否大于或等于结算额，不是则返回提示信息；
- 所有检查通过后则进行合约结算，最后触发事件；

4.1.3.6 链账户余额查询

运营方、平台方或终端用户通过该 API 可以查询某一个用户的 DDC 业务费余额。

- 输入参数：账户地址；
- 输出参数：账户余额；
- 方法命名：balanceOf;
- 方法举例：balanceOf(address accAddr) view returns (uint256 balance);
- 核心逻辑：
  - 检查账户地址是否为 0 地址，是则返回提示信息；
  - 所有检查通过后则返回查询结果；

4.1.3.7 批量链账户余额查询

运营方、平台方或终端用户通过调用该 API 可以批量查询链账户所对应的 DDC 业务费余额。

- 输入参数：链账户集合；
- 输出参数：链账户余额集合；
- 方法命名：balanceOfBatch;
- 方法举例：balanceOfBatch(address[] memory accAddrs) view returns (uint256[] memory);
- 核心逻辑：

- 批量循环挨个查询链账户余额，根据索引检查链账户地址是否为 0 地址，是则返回提示信息，否则将挨个查询所对应的链账户余额逐个添加到链账户余额的集合列表，并进行返回；

4.1.3.8 设置 DDC 业务费计费规则

运营账户调用该 API 授权设置 DDC 合约收费标准，如果已经设置过计费，将会覆盖原有的计费规则。并且表示该 DDC 合约为授权合约，可以调用计费合约对调用者扣除费用。

- 输入参数：业务主代理合约地址、合约方法 sig、业务费额
- 输出参数：无
- 方法命名：setFee;
- 方法举例: setFee(address ddcAddr,byte4 sig,uint32 amount);
- 事件：SetFee(address ddcAddr,byte4 sig,uint amount);
- 合约逻辑：
  - 检查业务主代理合约地址是否为 0 地址，是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户所对应的身份是不是运营方，不是则返回提示信息；
  - 所有检查通过后则保存计费规则数据，最后触发事件；

4.1.3.9 删除 DDC 业务费计费规则

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

- 输入参数：DDC 业务合约地址
- 输出参数：无；
- 方法命名：delFee;
- 方法举例: delFee(address ddcAddr,bytes4 sig);
- 事件：DelFee(address ddcAddr,bytes4 sig);
- 合约逻辑：

- 检查业务主代理合约地址是否为 0 地址，是则返回提示信息；
- 检查调用者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户所对应的身份是不是运营方，不是则返回提示信息；
- 所有检查通过后则删除 DDC 合约计费规则数据，最后触发事件；

4.1.3.10 删除 DDC 业务主合约授权

运营方调用该接口删除 DDC 合约，表示该 DDC 合约不在被授权管理，同时也会删除该合约调用计费合约扣费的权限以及在计费合约中设置的计费规则。

- 输入参数：DDC 业务合约地址
- 输出参数：无；
- 方法命名：delDDC;
- 方法举例: delDDC (address ddcAddr);
- 事件：DelDDC(address ddcAddr);
- 合约逻辑：
  - 检查业务主代理合约地址是否为 0 地址，是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户所对应的身份是不是运营方，不是则返回提示信息；
  - 所有检查通过后则删除 DDC 业务主代理合约以及该 DDC 合约对应的所有计费规则数据，最后触发事件；

4.1.3.11 DDC 业务费计费规则查询

运营方、平台方或终端用户可以查询某一个 DDC 合约的各个 API 的费用。

- 输入参数：业务主代理合约地址；
- 输出参数：业务主代理合约对应的方法 Sig 业务费；
- 方法命名：queryFee;
- 方法举例：queryFee(address ddcAddr,bytes4 sig) view returns (uint amount);
- 核心逻辑：

- 检查业务主代理合约地址是否为 0 地址;
- 检查业务主代理合约地址所对应的计费规则是否存在或是否可用，不存在则返回提示信息;
- 所有检查通过后则返回查询结果;

4.1.3.12 业务费总量查询

运营方可以查询计费合约的链上的业务费总量。

- 输入参数：无;
- 输出参数：业务费总量;
- 方法命名：totalSupply;
- 方法举例：totalSupply() view returns (uint256 totalSupply);
- 核心逻辑：
  - 查询链上当前业务费总量;

4.1.3.13 启用批量设置

运营方可以通过此接口对是否启用批量开关进行设置。

- 输入参数：是否打开;
- 输出参数：无;
- 方法命名：setSwitcherStateOfBatch;
- 方法举例：setSwitcherStateOfBatch(bool isOpen);
- 事件：SetSwitcherStateOfBatch(address indexed operator,bool isOpen);
- 核心逻辑：
  - 检查调用者账户是否存在，不存在则返回提示信息;
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息;
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息;
  - 检查调用者账户类型是否为运营方，不是则返回提示信息;
  - 检查 isOpen 是否与当前开关状态一致，一致则返回提示信息;
  - 所有检查通过后则保存数据，最后触发事件;

4.2 BSN-DDC-721 业务主合约

4.2.1 功能介绍

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

4.2.2 数据结构

➢ DDC 存储结构

编号	字段名	字段	类型	备注
1.	DDC 的名称	_name	String	
2.	DDC 的符号	_symbol	string	
3.	DDC 的资源标识集合	_ddcURIs	mapping<uint256=>string>	Key: ddc 的唯一标识 Value:资源标识符
4.	DDC 的拥有者地址映射集合	_owners	mapping(uint256 => address)	Key: ddc 的唯一标识 Value:拥有者地址
5.	DDC 的余额映射集合	_balances	mapping(address => uint256)	Key: 拥有者地址 Value:ddc 的数量
6.	DDC 的授权者映射集合	_ddcApprovals	mapping(uint256 => address)	Key: ddc 的唯一标识 Value:被授权者的地址
7.	DDC 的操作者映射集合	_operatorApprovals	mapping(address => mapping(address => bool))	Key: ddc 的 owner 地址 Value:被授权者的地址以及是否授权
8.	DDC 黑名单列表	_blacklist	mapping<uint256=>bool>	Key: ddc 的唯一标识 Value:ddc 状态
9.	最新的 DDC 的 id	_lastDDCId	uint256	
10.	Nonce 列表	_nonces	mapping(address => uint256)	Key: 终端用户 Value: nonce 值
11.	授权类型哈希值列表	typeHashs	mapping(HashType=>bytes32)	Key: 授权哈希类型 Value: 授权哈希值

12.	域名分割符	DOMAIN_SEPARATOR	bytes32	
13.	DDC 锁定列表	_locklist	mapping(uint256=>bool)	Key: ddc 的唯一标识 Value:是否锁定

4.2.3 API 定义

4.2.3.1 生成

平台方、终端用户通过调用该 API 对 DDC 进行生成。

- 输入参数：接收者账户，资源标识符；
- 输出参数：
- 方法命名：mint;
- 方法举例：mint(address to,string memory ddcURI);
- 事件：Transfer(address(0),to,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有调用权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
  - 生成 DDCID，并检查 DDCID 是否已经存在，存在则返回提示信息；
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存生成的 DDC 数据，最后触发事件；

4.2.3.2 安全生成

平台方、终端用户通过调用该 API 进行 DDC 的安全生成。

- 输入参数：接收者账户，资源标识符，附加数据；
- 输出参数：
- 方法命名：safeMint;

- 方法举例：safeMint(address to,string memory ddcURI,bytes memory data);
- 事件：Transfer(address(0),to,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有调用权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
  - 生成 DDCID，并检查 DDCID 是否已经存在，存在则返回提示信息；
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并合约保存生成的 DDC 数据；
  - 触发事件，如果接收者账户对应的是一个合约，则需检查接收者账户是否接收 DDC;

4.2.3.3 DDC 授权

DDC 拥有者通过调用该 API 进行 DDC 的授权，发起者需要是 DDC 的拥有者。

- 输入参数：授权者账户，ddc 唯一标识；
- 输出参数：
- 方法命名：approve;
- 方法举例：approve(address to,uint256 ddcId);
- 事件：Approval(operator,to,ddcId);
- 核心逻辑：
  - 检查调用者状态是否可用，不可用则返回提示信息；
  - 检查调用者是否有调用权限，没有则返回提示信息；
  - 检查授权者账户地址是否为 0 地址，是则返回提示信息；
  - 检查授权者账户状态是否可用，不可用则返回提示信息；

- 检查调用者账户与授权者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
- 检查 DDCID 是否存在，不存在则返回提示信息；
- 检查 DDCID 是否被冻结，是则返回提示信息；
- 检查 DDCID 是否被锁定，是则返回提示信息；
- 检查 DDC 所对应的拥有者与授权账户是否属同一账户，属于则返回提示信息；
- 检查 DDC 所对应的拥有者与调用者账户是否属于同一账户或检查 DDC 所对应的拥有者账户是否授权给调用者账户，不是则返回提示信息；
- 所有检查通过后则调用计费合约支付 DDC 业务费，并合约保存 DDC 授权数据，最后触发事件；
- 注：针对 DDC 授权，授权范围仅限于当前拥有者所对应的某个 DDC，一旦后续进行转移或安全转移操作，则之前所拥有的授权会因转移或安全转移操作而失效（因拥有者发生改变）；

4.2.3.4 DDC 授权查询

运营方、平台方或者用户通过调用该 API 进行 DDC 的授权查询。

- 输入参数：ddc 唯一标识；
- 输出参数：授权者账户
- 方法命名：getApproved;
- 方法举例：getApproved(uint256 ddcId) view returns (address);
- 核心逻辑：
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 所有检查通过后则返回查询结果；

4.2.3.5 账户授权

DDC 拥有者通过调用该 API 进行账户授权，发起者需要是 DDC 的拥有方。

- 输入参数：授权者账户，授权标识

- 输出参数：
- 方法命名：setApprovalForAll;
- 方法举例：setApprovalForAll(address operator,bool approved);
- 事件：ApprovalForAll(owner,operator,approved);
- 核心逻辑：
  - 检查调用者状态是否可用，不可用则返回提示信息；
  - 检查调用者是否有调用权限，没有则返回提示信息；
  - 检查授权者账户地址是否为 0 地址，是则返回提示信息；
  - 检查授权者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户与授权者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
  - 检查调用者账户与授权者账户是否属于同一账户，属于则返回提示信息；
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存账户授权数据，最后触发事件；
  - 注：针对账户授权，授权范围仅限于当前拥有者所拥有的所有 DDC，一旦后续进行流转操作，则之前所拥有的授权会因流转操作而失效（因拥有者发生改变）；

4.2.3.6 账户授权验证

运营方、平台方或终端用户通过调用该 API 进行账户授权查询。

- 输入参数：拥有者账户，授权者账户
- 输出参数：bool 结果
- 方法命名：isApprovedForAll;
- 方法举例：isApprovedForAll(address owner,address operator) view returns (bool);
- 核心逻辑：
  - 检查拥有者账户或授权者账户是否为 0 地址，是则返回提示信息；
  - 所有检查通过后则根据拥有者账户以及授权者账户查询授权者账户是否被拥有者账户进行授权，最后返回验证结果；

4.2.3.7 安全转移

DDC 拥有者或 DDC 授权者通过调用该 API 进行 DDC 的安全转移。

- 输入参数：拥有者账户，接收者账户，ddc 唯一标识，附加数据;
- 输出参数:
- 方法命名：safeTransferFrom;
- 方法举例：safeTransferFrom(address from,address to,uint256 ddcId, bytes memory data);
- 事件：Transfer(from,to,ddcId);
- 核心逻辑:
  - 检查调用者账户状态是否可用，不可用则返回提示信息;
  - 检查调用者账户是否有权限，没有则返回提示信息;
  - 检查拥有者账户状态是否可用，不可用则返回提示信息;
  - 检查拥有者账户地址是否为 0 地址，是则返回提示信息;
  - 检查接收者账户状态是否可用，不可用则返回提示信息;
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息;
  - 检查 DDCID 是否存在，不存在则返回提示信息;
  - 检查 DDCID 是否被冻结，是则返回提示信息;
  - 检查 DDCID 是否被锁定，是则返回提示信息;
  - 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 和 4.4.3.14 章节;
  - 检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户，如果都不是则返回提示信息;
  - 检查 DDC 所对应的拥有者与所传拥有者账户参数是否属于同一账户，不属于则返回提示信息;
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存 DDC 安全转移数据和清除该 DDCID 授权列表;
  - 触发事件，如果接收者账户对应的是一个合约，则需检查接收者账户是否接收 DDC;

4.2.3.8 转移

DDC 拥有者或 DDC 授权者通过调用该 API 进行 DDC 的转移。

- 输入参数：拥有者账户，接收者账户，ddc 唯一标识；
- 输出参数：
- 方法命名：transferFrom;
- 方法举例：transferFrom(address from,address to,uint256 ddcId);
- 事件：Transfer(from,to,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查拥有者账户状态是否可用，不可用则返回提示信息；
  - 检查拥有者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDCID 是否被冻结，是则返回提示信息；
  - 检查 DDCID 是否被锁定，是则返回提示信息；
  - 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 和 4.4.3.14 章节；
  - 检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户，如果都不是则返回提示信息；
  - 检查 DDC 所对应的拥有者与所传拥有者账户参数是否属于同一账户，不属于则返回提示信息；
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存 DDC 转移数据和清除该 DDCID 授权列表，最后触发事件；

4.2.3.9 冻结

运营方通过调用该 API 进行 DDC 的冻结。

- 输入参数：ddc 唯一标识;
- 输出参数:
- 方法命名：freeze;
- 方法举例：freeze(uint256 ddcId);
- 事件：EnterBlacklist(sender,ddcId);
- 核心逻辑:
  - 检查调用者账户状态是否可用，不可用则返回提示信息;
  - 检查调用者账户是否有权限，没有则返回提示信息;
  - 检查调用者账户是否为运营方账户，不是则返回提示信息;
  - 检查 DDCID 是否存在，不存在则返回提示信息;
  - 检查 DDCID 是否被冻结，是则返回提示信息。
  - 所有检查通过后则将 DDCID 加入黑名单列表，最后触发事件;

4.2.3.10 解冻

运营方通过调用该 API 进行 DDC 的解冻。

- 输入参数：ddc 唯一标识
- 输出参数:
- 方法命名：unFreeze;
- 方法举例：unFreeze(uint256 ddcId);
- 事件：ExitBlacklist(sender,ddcId);
- 核心逻辑:
  - 检查调用者账户状态是否可用，不可用则返回提示信息;
  - 检查调用者账户是否有权限，没有则返回提示信息;
  - 检查调用者账户是否为运营方账户，不是则返回提示信息;
  - 检查 DDCID 是否存在，不存在则返回提示信息;
  - 检查 DDCID 是否被解冻，是则返回提示信息。

- 所有检查通过后则将 DDCID 从黑名单列表进行移除，最后触发事件；

4.2.3.11 销毁

DDC 拥有者或 DDC 授权者通过调用该 API 进行 DDC 的销毁。

- 输入参数：ddc 唯一标识；
- 输出参数：
- 方法命名：burn;
- 方法举例：burn(uint256 ddcId);
- 事件：Transfer(owner,address(0),ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户，如果都不是则返回提示信息；
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存 DDC 销毁数据和清除该 DDCID 授权列表，最后触发事件；

4.2.3.12 查询数量

运营方、平台方或终端用户通过调用该 API 进行查询当前账户拥有的 DDC 的数量。

- 输入参数：拥有者账户；
- 输出参数：ddc 的数量；
- 方法命名：balanceOf;
- 方法举例：balanceOf(address owner) view returns (uint256);
- 核心逻辑：
  - 检查拥有者账户所对应的地址是否为 0 地址，是则返回提示信息；

- 所有检查通过后则返回查询结果;

4.2.3.13 查询拥有者

运营方、平台方或终端用户通过调用该 API 进行查询当前 DDC 的拥有者。

- 输入参数：ddc 唯一标识;
- 输出参数：拥有者账户;
- 方法命名：ownerOf;
- 方法举例：ownerOf(uint256 ddcId) view returns (address);
- 核心逻辑：
  - 返回查询 DDC 所对应的拥有者账户;

4.2.3.14 获取名称

运营方、平台方或终端用户通过调用该 API 进行查询当前 DDC 的名称。

- 输入参数：无;
- 输出参数：ddc 的名称;
- 方法命名：name;
- 方法举例：name() view returns (string memory);
- 核心逻辑：
  - 返回全局名称;

4.2.3.15 获取符号

运营方、平台方或终端用户通过调用该 API 进行查询当前 DDC 的符号标识。

- 输入参数：无;
- 输出参数：ddc 的符号标识;
- 方法命名：symbol;
- 方法举例：symbol() view returns (string memory);
- 核心逻辑：

- 返回全局符号;

4.2.3.16 获取 ddcURI

运营方、平台方或终端用户通过调用该方法进行查询当前 DDC 的资源标识符。

- 输入参数：ddc 的唯一标识;
- 输出参数：返回 uri;
- 方法命名：ddcURI;
- 方法举例：ddcURI(uint256 ddcId) view returns (string memory);
- 核心逻辑：
  - 检查 DDCID 是否存在，不存在则返回提示信息;
  - 所有检查通过后则返回查询结果;

4.2.3.17 ddcURI 设置

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

- 输入参数：ddc 的唯一标识，资源标识符;
- 输出参数：
- 方法命名：setURI;
- 方法举例：setURI(uint256 ddcId,string memory ddcURI);
- 事件：SetURI(uint256 indexed ddcId,string ddcURI);
- 核心逻辑：
  - 检查调用者账户信息是否存在，不存在则返回提示信息;
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息;
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息;
  - 检查调用者账户是否有权限，没有则返回提示信息;
  - 检查 DDCID 是否存在，不存在则返回提示信息;
  - 检查 DDCID 是否被冻结，是则返回提示信息;

- 检查 DDCID 是否被锁定，是则返回提示信息；
- 检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户，如果都不是则返回提示信息；
- 检查 DDC 资源标识符参数是否为空值，是则返回提示信息；
- 检查根据 DDCID 参数查询所对应的资源标识符是否为空值，不是则返回提示信息；
- 所有检查通过后则保存资源标识符数据，最后触发事件；

4.2.3.18 名称符号设置

合约拥有者可以对 721 的名称以及符号进行初始化。

- 输入参数：名称，符号；
- 输出参数：
- 方法命名：setNameAndSymbol;
- 方法举例：setNameAndSymbol(string memory name_, string memory symbol_);
- 事件：SetNameAndSymbol(string name,string symbol);
- 核心逻辑：
  - 检查调用者账户是否为合约拥有者，不是则返回提示信息；
  - 所有检查通过后则保存名称和符号；最后触发事件；

4.2.3.19 最新 DDCID 查询

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。

- 输入参数：无；
- 输出参数：最新 DDCID；
- 方法命名：getLatestDDCId;
- 方法举例：getLatestDDCId() view returns (uint256);

- 核心逻辑：
  - 返回查询的最新 DDCID；

4.2.3.20 批量生成

平台方、终端用户通过调用该 API 对 DDC 进行批量生成。

- 输入参数：接收者账户，资源标识符集合；
- 输出参数：
- 方法命名：mintBatch；
- 方法举例：mintBatch(address to,string[] memory ddcURIs);
- 事件：TransferBatch(operator,address(0),to,ddcIds);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有调用权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
  - 批量循环挨个处理 DDC，并根据索引检查生成的每个 DDCID 是否存在，存在则返回提示信息，否则挨个调用计费合约支付 DDC 业务费和保存生成的 DDC 数据：
  - 最后触发事件；

4.2.3.21 批量安全生成

平台方、终端用户通过调用该 API 进行 DDC 的批量安全生成。

- 输入参数：接收者账户，资源标识符集合，附加数据；
- 输出参数：
- 方法命名：safeMintBatch;

- 方法举例：safeMintBatch(address to,string[] memory ddcURIs,bytes memory data);
- 事件：TransferBatch(operator,address(0),to,ddcIds);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有调用权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
  - 批量循环挨个处理 DDC，并根据索引检查生成的每个 DDCID 是否存在，存在则返回提示信息，否则挨个调用计费合约支付 DDC 业务费和保存生成的 DDC 数据:
  - 触发事件，并检查接收者账户是否对应的是一个合约，是则需检查接收者账户是否接收 DDC;

4.2.3.22 授权哈希设置

合约拥有者可以对元交易的授权类型哈希进行初始化。

- 输入参数：授权哈希类型，授权哈希值；
- 输出参数：
- 方法命名：setMetaTypeHashArgs;
- 方法举例：setMetaTypeHashArgs(HashType hashType,bytes32 hashValue);
- 核心逻辑：
  - 检查调用者是否为合约拥用者，不是则返回提示信息；
  - 所有检查通过后保存授权类型哈希；

4.2.3.23 分割符设置

合约拥有者可以对元交易的域名分割符进行设置。

- 输入参数：域名分割符；
- 输出参数：
- 方法命名：setMetaSeparatorArg;
- 方法举例：setMetaSeparatorArg(bytes32 separator);
- 核心逻辑：
  - 检查调用者是否为合约拥用者，不是则返回提示信息；
  - 所有检查通过后保存域名分割符；

4.2.3.24 元交易生成

终端用户通过授权平台方调用该 API 对 DDC 进行元交易生成。

- 输入参数：接收者账户，资源标识符、nonce 值、过期时间，签名值；
- 输出参数：
- 方法命名：metaMint;
- 方法举例：metaMint(address to,string memory ddcURI,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransfer(operator,address(0),to,ddcId);
- 核心逻辑：
  - 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获取签名者账户地址是否于 to 参数地址相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与接收者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有调用权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；

- 检查接收者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
- 生成 DDCID，并检查 DDCID 是否已经存在，存在则返回提示信息；
- 所有检查通过后则调用计费合约支付 DDC 业务费（扣除平台方账户业务费），并保存生成的 DDC 数据，最后触发事件；

4.2.3.25 元交易安全生成

终端用户通过授权平台方调用该 API 对 DDC 进行元交易安全生成。

- 输入参数：接收者账户，资源标识符，附加数据，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaSafeMint;
- 方法举例：metaSafeMint(address to,string memory ddcURI,bytes memory data,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransfer(operator,address(0),to,ddcId);
- 核心逻辑：
  - 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获取签名者地址是否于 to 参数地址相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与接收者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有调用权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；

- 检查接收者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
- 生成 DDCID，并检查 DDCID 是否已经存在，存在则返回提示信息；
- 所有检查通过后则调用计费合约支付 DDC 业务费（扣除平台方账户业务费），并合约保存生成的 DDC 数据；
- 触发事件，如果接收者账户对应的是一个合约，则需检查接收者账户是否接收 DDC;

4.2.3.26 元交易批量生成

终端用户通过授权平台方调用该 API 对 DDC 进行元交易批量生成。

- 输入参数：接收者账户，资源标识符集合，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaMintBatch;
- 方法举例：metaMintBatch(address to,string[] memory ddcURIs,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransferBatch(operator,address(0),to,ddcIds);
- 核心逻辑：
  - 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获取签名者地址是否于 to 参数地址相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与接收者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有调用权限，没有则返回提示信息；

- 检查接收者账户地址是否为 0 地址，是则返回提示信息；
- 检查接收者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
- 批量循环挨个处理 DDC，并根据索引检查生成的每个 DDCID 是否存在，存在则返回提示信息，否则挨个调用计费合约支付 DDC 业务费（扣除平台方账户业务费）和保存生成的 DDC 数据：
- 最后触发事件；

4.2.3.27 元交易批量安全生成

终端用户通过授权平台方调用该 API 进行 DDC 元交易批量安全生成。

- 输入参数：接收者账户，资源标识符集合，附加数据，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaSafeMintBatch;
- 方法举例：metaSafeMintBatch(address to,string[] memory ddcURIs,bytes memory data,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransferBatch(operator,address(0),to,ddcIds);
- 核心逻辑：
  - 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获取签名者地址是否于 to 参数地址相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与接收者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；

- 检查调用者账户是否有调用权限，没有则返回提示信息；
- 检查接收者账户地址是否为 0 地址，是则返回提示信息；
- 检查接收者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
- 批量循环挨个处理 DDC，并根据索引检查生成的每个 DDCID 是否存在，存在则返回提示信息，否则挨个调用计费合约支付 DDC 业务费（扣除平台方账户业务费）和保存生成的 DDC 数据:
- 触发事件，并检查接收者账户是否对应的是一个合约，是则需检查接收者账户是否接收 DDC;

4.2.3.28 元交易转移

DDC 拥有者或 DDC 授权者通过授权平台方调用该 API 进行 DDC 元交易转移。

- 输入参数：拥有者账户，接收者账户，ddc 唯一标识，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaTransferFrom;
- 方法举例：metaTransferFrom(address from,address to,uint256 ddcId,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransfer(operator,from,to,ddcId);
- 核心逻辑：
  - 检查 DDC 所对应的拥有者与签名者是否属于同一账户或 DDC 所对应的授权者与签名者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给签名者账户，如果都不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与拥有者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；

- 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
- 检查调用者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户是否有权限，没有则返回提示信息；
- 检查拥有者账户状态是否可用，不可用则返回提示信息；
- 检查拥有者账户地址是否为 0 地址，是则返回提示信息；
- 检查接收者账户状态是否可用，不可用则返回提示信息；
- 检查接收者账户地址是否为 0 地址，是则返回提示信息；
- 检查 DDCID 是否存在，不存在则返回提示信息；
- 检查 DDCID 是否被冻结，是则返回提示信息；
- 检查 DDCID 是否被锁定，是则返回提示信息；
- 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 和 4.4.3.14 章节；
- 检查 DDC 所对应的拥有者与签名者是否属于同一账户或 DDC 所对应的授权者与签名者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给签名者账户，如果都不是则返回提示信息；
- 检查 DDC 所对应的拥有者与所传拥有者账户参数是否属于同一账户，不属于则返回提示信息；
- 所有检查通过后则调用计费合约支付 DDC 业务费（扣除平台方账户业务费），并保存 DDC 转移数据和清除该 DDCID 授权列表，最后触发事件；

4.2.3.29 元交易安全转移

DDC 拥有者或 DDC 授权者通过授权平台方调用该 API 进行 DDC 元交易安全转移。

- 输入参数：拥有者账户，接收者账户，ddc 唯一标识，附加数据，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaSafeTransferFrom;

- 方法举例：metaSafeTransferFrom(address from,address to,uint256 ddcId,bytes memory data,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransfer(operator,from,to,ddcId);
- 核心逻辑：
  - 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获取签名者账户是否于拥有者账户或授权者相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与拥有者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查拥有者账户状态是否可用，不可用则返回提示信息；
  - 检查拥有者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDCID 是否被冻结，是则返回提示信息；
  - 检查 DDCID 是否被锁定，是则返回提示信息；
  - 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 和 4.4.3.14 章节；
  - 检查 DDC 所对应的拥有者与签名者是否属于同一账户或 DDC 所对应的授权者与签名者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给签名者账户，如果都不是则返回提示信息；
  - 检查 DDC 所对应的拥有者与所传拥有者账户参数是否属于同一账户，不属于则返回提示信息；

- 所有检查通过后则调用计费合约支付 DDC 业务费（扣除平台方账户业务费），并保存 DDC 安全转移数据和清除该 DDCID 授权列表；
- 触发事件，如果接收者账户对应的是一个合约，则需检查接收者账户是否接收 DDC;

4.2.3.30 元交易销毁

DDC 拥有者或 DDC 授权者通过授权平台方调用该 API 进行 DDC 元交易销毁。

- 输入参数：ddc 唯一标识，nonce 值、过期时间，签名值；
- 输出参数：
- 方法命名：metaBurn;
- 方法举例：metaBurn(uint256 ddcId,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransfer(operator,owner,address(0),ddcId);
- 核心逻辑：
  - 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获取签名者地址是否于拥有者或授权者账户相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与拥有者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；

- 检查 DDC 所对应的拥有者与签名者是否属于同一账户或 DDC 所对应的授权者与签名者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给签名者账户，如果都不是则返回提示信息；
- 所有检查通过后则调用计费合约支付 DDC 业务费（扣除平台方账户业务费），并保存 DDC 销毁数据和清除该 DDCID 授权列表，最后触发事件；

4.2.3.31 Nonce 查询

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。

- 输入参数：拥有者；
- 输出参数：无；
- 方法命名：getNonce;
- 方法举例：getNonce(address from) view returns (uint256);
- 核心逻辑：
  - 返回查询的签名者最新 nonce 值；

4.2.3.32 DDC 跨链锁定

平台方或终端用户通过 DDC 跨链应用合约调用该 API 进行 DDC 跨链锁定。

- 输入参数：DDC 唯一标识；
- 输出参数：
- 方法命名：lock;
- 方法举例：lock(uint256 ddcId);
- 事件：Locklist(operator,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；

- 检查 DDCID 是否存在，不存在则返回提示信息；
- 检查 DDCID 是否被冻结，是则返回提示信息；
- 检查 DDCID 是否被锁定，是则返回提示信息；
- 所有检查通过后则将 DDCID 加入冻结列表，最后触发事件；

4.2.3.33 DDC 跨链解锁

平台方或终端用户通过 DDC 跨链应用合约调用该 API 进行 DDC 跨链解锁。

- 输入参数：DDC 唯一标识；
- 输出参数：
- 方法命名：unlock;
- 方法举例：unlock(uint256 ddcId);
- 事件：UnLocklist(operator,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDCID 是否被冻结，是则返回提示信息；
  - 检查 DDCID 是否被解锁，是则返回提示信息；
  - 所有检查通过后则将 DDCID 移除冻结列表，最后触发事件；

4.3 BSN-DDC-1155 业务主合约

4.3.1 功能介绍

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

4.3.2 数据结构

➢ DDC 存储结构

编号	字段名	字段	类型	备注
1.	DDC 的资源标识集合	_ddcURIs	mapping(uint256=>string)	Key：ddc 的唯一标识 Value：资源标识符
2.	DDC 的余额映射集合	_balances	mapping(uint256 => mapping(address => uint256))	Key: ddc 的唯一标识 Value：拥有者的地址以及对应的数量
3.	DDC 的操作者映射集合	_operatorApprovals	mapping(address => mapping(address => bool))	Key: ddc 的 owner 地址 Value：授权者的地址以及是否授权
4.	DDC 黑名单列表	_blacklist	mapping(uint256=>bool)	Key: ddc 的唯一标识 Value：ddc 状态
5.	DDCID 列表	_ddcIds	mapping(uint256=>bool)	Key: ddc 的唯一标识 Value: ddcid 是否存在
6.	最新的 DDC 的 id	_lastDDCId	uint256	
7.	Nonce 列表	_nonces	mapping(address => uint256)	Key：终端用户 Value：nonce 值
8.	授权类型哈希值列表	typeHashs	mapping(HashType=>bytes32)	Key：授权哈希类型 Value：授权哈希值
9.	域名分割符	DOMAIN_SEPARATOR	bytes32	
10.	DDC 锁定列表	_locklist	mapping(uint256=>bool)	Key: ddc 的唯一标识 Value:是否锁定
11.	DDC 拥用者账户列表	_ddcOwners	mapping(uint256=>address[])	Key：ddc 的唯一标识 Value：DDC 所对应的拥有者账户列表
12.	DDC 拥有者账户索引列表	_ddcOwnerIndexs	mapping(uint256=>mapping(address=>uint256))	Key：ddc 的唯一标识 Value: Key 为拥有者账户，Value 为拥有者账户在 DDC 拥有者账户列表中的索引

4.3.3 API 定义

4.3.3.1 安全生成

平台方、终端用户通过调用该 API 进行 DDC 的安全创建。

- 输入参数：接收者账户，生成 DDC 对应数量，资源标识符，附加数据；
- 输出参数：
- 方法命名：safeMint;
- 方法举例：safeMint(address to,uint256 amount,string memory ddcURI,bytes memory data);
- 事件：TransferSingle(operator,address(0),to,ddcId,amount);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
  - 检查生成 DDC 所对应的数量是否大于 0，不是则返回提示信息；
  - 生成 DDCID，并检查 DDCID 是否已经存在，存在则返回提示信息；
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存生成的 DDC 数据；
  - 最后触发事件，如果接收者账户对应的是一个合约，则需检查接收者账户是否接收 DDC；

4.3.3.2 批量安全生成

平台方、终端用户通过调用该 API 进行 DDC 的批量安全创建。

- 输入参数：接收者账户，数量集合，资源标识符集合，附加数据；
- 输出参数：

- 方法命名：safeMintBatch;
- 方法举例：safeMintBatch(address to,uint256[] memory amounts,string[] memory ddcURIs,bytes memory data);
- 事件：TransferBatch(operator,address(0),to,ddcIds,amounts);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
  - 检查数量集合长度是否为 0，是则返回提示信息；
  - 检查资源标识符集合长度是否为 0，是则返回提示信息；
  - 检查数量集合与资源标识符集合长度是否相等，不相等则返回提示信息；
  - 批量循环挨个处理 DDC，并根据索引检查生成的每个 DDC 所对应的数量是否大于 0，不是则返回提示信息，根据索引检查生成的每个 DDCID 是否存在，存在则返回提示信息，否则挨个调用计费合约支付 DDC 业务费和保存生成的 DDC 数据；
  - 最后触发事件，并检查接收者账户是否对应的是一个合约，是则需检查接收者账户是否接收 DDC；

4.3.3.3 账户授权

DDC 拥有者通过调用该 API 进行账户授权，发起者需要是 DDC 的拥有者。

- 输入参数：授权者账户，授权标识
- 输出参数：
- 方法命名：setApprovalForAll;
- 方法举例：setApprovalForAll(address operator,bool approved);
- 事件：ApprovalForAll(owner,operator,approved);
- 核心逻辑：

- 检查调用者账户状态是否可用，不可用则返回提示信息;
- 检查调用者账户是否有权限，没有则返回提示信息;
- 检查授权者账户地址是否为 0 地址，是则返回提示信息;
- 检查授权者账户状态是否可用，不可用则返回提示信息;
- 检查调用者账户与授权者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节;
- 检查调用者账户与授权者账户是否属于同一账户，属于则返回提示信息;
- 所有检查通过后则调用计费合约支付 DDC 业务费，并保存账户授权数据，最后触发事件;
- 注：针对账户授权，授权范围仅限于当前拥有者所拥有的所有 DDC，一旦后续进行流转操作，则之前所拥有的授权会因流转操作而失效（因拥有者发生改变）；

4.3.3.4 账户授权验证

运营方、平台方或者终端用户通过调用该 API 进行 DDC 的授权查询。

- 输入参数：拥有者账户，授权者账户
- 输出参数：bool 结果
- 方法命名：isApprovedForAll;
- 方法举例：isApprovedForAll(address owner,address operator) view returns (bool);
- 核心逻辑：
  - 检查拥有者账户或授权者账户是否为 0 地址，是则返回提示信息;
  - 所有检查通过后则根据拥有者账户以及授权者账户查询授权者账户是否被拥有者账户进行授权，最后返回验证结果;

4.3.3.5 安全转移

DDC 拥有者或 DDC 授权者通过调用该 API 进行 DDC 的转移。

- 输入参数：拥有者账户，接受者账户，ddc 唯一标识，数量，附加数据;

- 输出参数：
- 方法命名：safeTransferFrom;
- 方法举例：safeTransferFrom(address from,address to,uint256 ddcId,uint256 amount,bytes memory data);
- 事件：TransferSingle(operator,from,to,ddcId,amount);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查拥有者账户状态是否可用，不可用则返回提示信息；
  - 检查拥有者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDCID 是否被冻结，是则返回提示信息；
  - 检查 DDCID 是否被锁定，是则返回提示信息；
  - 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 和 4.4.3.14 章节；
  - 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户，否则返回提示信息；
  - 检查安全转移 DDC 所对应的数量是否大于 0，不是则返回提示信息；
  - 检查拥有者账户对 DDC 所拥有的数量是否大于等于安全转移所需数量，否则返回提示信息；
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存 DDC 安全转移数据；
  - 最后触发事件，如果接收者账户对应的是一个合约，则需检查接收者账户是否接收 DDC；

4.3.3.6 批量安全转移

DDC 拥有者或 DDC 授权者通过调用该 API 进行 DDC 的批量转移。

- 输入参数：拥有者账户，接受者账户，ddc 唯一标识集合，数量集合，附加数据
- 输出参数：
- 方法命名：safeBatchTransferFrom;
- 方法举例：safeBatchTransferFrom(address from,address to,uint256[] memory ddcIds,uint256[] memory amounts,bytes memory data);
- 事件：TransferBatch(operator,from,to,ddcIds,amounts);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查拥有者账户状态是否可用，不可用则返回提示信息；
  - 检查拥有者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 和 4.4.3.14 章节；
  - 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户，否则返回提示信息；
  - 检查数量集合长度是否为 0，是则返回提示信息；
  - 检查 DDCID 集合长度是否为 0，是则返回提示信息；
  - 检查数量集合与 DDCID 集合长度是否相等，不相等则返回提示信息；
  - 批量循环挨个处理 DDC，并根据索引检查每个 DDCID 是否存在，不存在则返回提示信息，再根据索引检查 DDCID 是否被冻结，是则返回提示信息，再根据索引检查 DDCID 是否被锁定，是则返回提示信息，再根据索引获取需要安全转移的每个 DDC 所对应的数量是否大于 0，不是则返回提示信息，最后根据索引检查拥有者账户拥有 DDC 所对应的数量是否大于等于需要安合转移的 DDC 数量，不是则返回提示信息，是则挨个调用计费合约支付 DDC 业务费和保存 DDC 安全转移数据;
  - 最后触发事件，并检查接收者账户是否对应的是一个合约，是则需检查接收者账户是否接收 DDC；

4.3.3.7 冻结

运营方通过调用该 API 进行 DDC 的冻结。

- 输入参数：ddc 唯一标识
- 输出参数：
- 方法命名：freeze;
- 方法举例：freeze(uint256 ddcId);
- 事件：EnterBlacklist(sender,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查调用者账户是否为运营方账户，不是则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDCID 是否被冻结，是则返回提示信息；
  - 所有检查通过后则将 DDCID 加入黑名单列表，最后触发事件；

4.3.3.8 解冻

运营方通过调用该 API 进行 DDC 的解冻。

- 输入参数：ddc 唯一标识
- 输出参数：
- 方法命名：unFreeze;
- 方法举例：unFreeze(uint256 ddcId);
- 事件：ExitBlacklist(sender,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查调用者账户是否为运营方账户，不是则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDCID 是否被解冻，是则返回提示信息；

- 所有检查通过后则将 DDCID 从黑名单列表进行移除，最后触发事件；

4.3.3.9 销毁

DDC 拥有者或授权者通过调用该 API 进行 DDC 的销毁。

- 输入参数：DDC 拥有者账户，ddc 唯一标识
- 输出参数：
- 方法命名：burn;
- 方法举例：burn(address owner,uint256 ddcId);
- 事件：TransferSingle(operator,owner,address(0),ddcId,amount);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查拥有者账户对 DDC 所拥有的数量是否大于 0，不是则返回提示信息；
  - 检查拥有者账户与调用者账户是否属于同一账户或拥有者账户是否授权给调用者账户，否则返回提示信息。
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存 DDC 销毁数据，最后触发事件；

4.3.3.10 批量销毁

DDC 拥有者或授权者通过调用该 API 进行 DDC 的批量销毁。

- 输入参数：拥有者账户，ddc 唯一标识的集合
- 输出参数：
- 方法命名：burnBatch;
- 方法举例：burnBatch(address owner,uint256[] memory ddcIds);
- 事件：TransferBatch(operator,owner,address(0),ddcIds,amounts);
- 核心逻辑：

- 检查调用者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户是否有权限，没有则返回提示信息；
- 检查调用者账户与拥有者账户是否属于同一账户或拥有者账户是否将自己所有的 DDC 授权给调用者账户，不是则返回提示信息。
- 检查 DDCID 集合长度是否为 0，是则返回提示信息；
- 批量循环 DDCID 集合列表，并根据索引 DDCID 是否存在，不存在则返回提示信息，再根据索引获取 DDCID 检查拥有者账户对 DDC 所拥有的数量是否大于 0，不是则返回提示信息，是则调用计费合约支付 DDC 业务费和保存 DDC 销毁数据，
- 最后触发事件；

4.3.3.11 查询数量

运营方、平台方或终端用户通过调用该 API 进行查询当前账户拥有的 DDC 的数量。

- 输入参数：拥有者账户，ddc 唯一标识
- 输出参数：余额
- 方法命名：balanceOf;
- 方法举例：balanceOf(address owner,uint256 ddcId) view returns (uint256);
- 核心逻辑：
  - 检查拥有者账户所对应的地址是否为 0 地址，是则返回提示信息；
  - 所有检查通过后则返回查询结果；

4.3.3.12 批量查询数量

运营方、平台方或终端用户通过调用该 API 进行批量查询账户拥有的 DDC 的数量。

- 输入参数：拥有者账户集合，ddc 唯一标识的集合
- 输出参数：余额的集合
- 方法命名：balanceOfBatch;

- 方法举例：balanceOfBatch(address[] memory owners,uint256[] memory ddcIds) view returns (uint256[] memory);
- 核心逻辑：
  - 检查拥有者账户集合与 DDCID 集合长度是否相等，不相等则返回；
  - 批量循环 DDCID 集合列表，根据索引检查拥有者账户地址是否为 0 地址，是则返回提示信息，否则将挨个查询所对应的 DDC 数量逐个添加的数量集合列表，并进行返回；

4.3.3.13 获取 ddcURI

运营方、平台方或终端用户通过调用该 API 进行查询当前 DDC 的资源标识符。

- 输入参数：ddc 的唯一标识；
- 输出参数：返回 uri；
- 方法命名：ddcURI;
- 方法举例：ddcURI(uint256 ddcId) view returns (string memory);
- 核心逻辑：
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 所有检查通过后则返回查询结果；

4.3.3.14 ddcURI 设置

DDC 拥有者或授权者通过调用该方法对 DDC 的资源标识符进行设置。

- 输入参数：ddc 的唯一标识，资源标识符；
- 输出参数：
- 方法命名：setURI;
- 方法举例：setURI(address owner,uint256 ddcId,string memory ddcURI);
- 事件：SetURI(address indexed owner,uint256 indexed ddcId,string ddcURI);
- 核心逻辑：
  - 检查调用者账户信息是否存在，不存在则返回提示信息；

- 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
- 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
- 检查调用者账户是否有权限，没有则返回提示信息；
- 检查 DDCID 是否存在，不存在则返回提示信息；
- 检查 DDCID 是否被冻结，是则返回提示信息；
- 检查 DDCID 是否被锁定，是则返回提示信息；
- 检查 DDC 所对应的拥有者与调用者是否属于同一账户或 DDC 所对应的授权者与调用者账户是否属于同一账户或 DDC 所对应的拥有者账户是否授权给调用者账户，如果都不是则返回提示信息；
- 检查 DDC 资源标识符参数是否为空值，是则返回提示信息；
- 检查根据 DDCID 查询对应的资源标识符是否为空值，不是则返回提示信息；
- 所有检查通过后则保存资源标识符数据，最后触发事件；

4.3.3.15 最新 DDCID 查询

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。

- 输入参数：无；
- 输出参数：最新 DDCID；
- 方法命名：getLatestDDCId；
- 方法举例：getLatestDDCId() view returns (uint256);
- 核心逻辑：
  - 返回查询的最新 DDCID；

4.3.3.16 授权哈希设置

合约拥有者可以对元交易的授权类型哈希进行初始化。

- 输入参数：授权哈希类型，授权哈希值；
- 输出参数：

- 方法命名：setMetaTypeHashArgs;
- 方法举例：setMetaTypeHashArgs(HashType hashType,bytes32 hashValue);
- 核心逻辑：
  - 检查调用者是否为合约拥用者，不是则返回提示信息；
  - 所有检查通过后保存授权类型哈希；

4.3.3.17 分割符设置

合约拥有者可以对元交易的域名分割符进行设置。

- 输入参数：域名分割符；
- 输出参数：
- 方法命名：setMetaSeparatorArg;
- 方法举例：setMetaSeparatorArg(bytes32 separator);
- 核心逻辑：
  - 检查调用者是否为合约拥用者，不是则返回提示信息；
  - 所有检查通过后保存域名分割符；

4.3.3.18 元交易安全生成

平台方、终端用户通过授权平台方调用该 API 进行 DDC 元交易安全生成。

- 输入参数：接收者账户，生成 DDC 对应数量，资源标识符，附加数据，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaSafeMint;
- 方法举例：metaSafeMint(address to,uint256 amount,string memory ddcURI,bytes memory data,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransferSingle(operator,address(0),to,ddcId,amount);
- 核心逻辑：

- 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获取签名者地址是否于 to 参数地址相同，不是则返回提示信息；
- 检查签名者账户状态是否可用，不是则返回提示信息；
- 检查调用者与接收者是否属于同平台，不是则返回提示信息；
- 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
- 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
- 检查调用者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户是否有权限，没有则返回提示信息；
- 检查接收者账户地址是否为 0 地址，是则返回提示信息；
- 检查接收者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
- 检查生成 DDC 所对应的数量是否大于 0，不是则返回提示信息；
- 生成 DDCID，并检查 DDCID 是否已经存在，存在则返回提示信息；
- 所有检查通过后则调用计费合约支付 DDC 业务费（扣除平台方账户业务费），并保存生成的 DDC 数据；
- 最后触发事件，如果接收者账户对应的是一个合约，则需检查接收者账户是否接收 DDC；

4.3.3.19 元交易批量安全生成

平台方、终端用户通过授权平台方调用该 API 进行 DDC 元交易批量安全生成。

- 输入参数：接收者账户，数量集合，资源标识符集合，附加数据，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaSafeMintBatch;

- 方法举例：metaSafeMintBatch(address to,uint256[] memory amounts,string[] memory ddcURIs,bytes memory data,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransferBatch(operator,address(0),to,ddcIds,amounts);
- 核心逻辑：
  - 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获取签名者地址是否与 to 参数地址相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与接收者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户与接收者账户是否属于同平台，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 章节；
  - 检查数量集合长度是否为 0，是则返回提示信息；
  - 检查资源标识符集合长度是否为 0，是则返回提示信息；
  - 检查数量集合与资源标识符集合长度是否相等，不相等则返回提示信息；
  - 批量循环挨个处理 DDC，并根据索引检查生成的每个 DDC 所对应的数量是否大于 0，不是则返回提示信息，根据索引检查生成的每个 DDCID 是否存在，存在则返回提示信息，否则挨个调用计费合约支付 DDC 业务费和保存生成的 DDC 数据；
  - 最后触发事件，并检查接收者账户是否对应的是一个合约，是则需检查接收者账户是否接收 DDC；

4.3.3.20 元交易安全转移

DDC 拥有者或 DDC 授权者通过授权平台方调用该 API 进行 DDC 元交易安全转移。

- 输入参数：拥有者账户，接收者账户，ddc 唯一标识，数量，附加数据，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaSafeTransferFrom;
- 方法举例：metaSafeTransferFrom(address from,address to,uint256 ddcId,uint256 amount,bytes memory data,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransferSingle(operator,from,to,ddcId,amount);
- 核心逻辑：
  - 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获取签名者账户地址是否于拥有者或授权者账户相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与拥有者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查拥有者账户状态是否可用，不可用则返回提示信息；
  - 检查拥有者账户地址是否为 0 地址，是则返回提示信息；
  - 检查接收者账户状态是否可用，不可用则返回提示信息；
  - 检查接收者账户地址是否为 0 地址，是则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDCID 是否被冻结，是则返回提示信息；

- 检查 DDCID 是否被锁定，是则返回提示信息；
- 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 和 4.4.3.14 章节；
- 检查拥有者账户与签名者账户是否属于同一账户或拥有者账户是否授权给签名者账户，否则返回提示信息；
- 检查安全转移 DDC 所对应的数量是否大于 0，不是则返回提示信息；
- 检查拥有者账户对 DDC 所拥有的数量是否大于等于安全转移所需数量，否则返回提示信息；
- 所有检查通过后则调用计费合约支付 DDC 业务费，并保存 DDC 转移数据；
- 最后触发事件，如果接收者账户对应的是一个合约，则需检查接收者账户是否接收 DDC；

4.3.3.21 元交易批量安全转移

DDC 拥有者或 DDC 授权者通过授权平台方调用该 API 进行 DDC 元交易批量安全转移。

- 输入参数：拥有者账户，接受者账户，ddc 唯一标识集合，数量集合，附加数据，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaSafeBatchTransferFrom;
- 方法举例：metaSafeBatchTransferFrom(address from,address to,uint256[] memory ddclds,uint256[] memory amounts,bytes memory data,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransferBatch(operator,from,to,ddclds,amounts);
- 核心逻辑：
  - 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获取签名者账户地址是否于拥有者或授权者账户相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；

- 检查调用者与拥有者是否属于同平台，不是则返回提示信息；
- 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
- 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
- 检查调用者账户状态是否可用，不可用则返回提示信息；
- 检查调用者账户是否有权限，没有则返回提示信息；
- 检查拥有者账户状态是否可用，不可用则返回提示信息；
- 检查拥有者账户地址是否为 0 地址，是则返回提示信息；
- 检查接收者账户状态是否可用，不可用则返回提示信息；
- 检查接收者账户地址是否为 0 地址，是则返回提示信息；
- 检查拥有者账户与接收者账户是否属于同平台或跨平台授权，不是则返回提示信息，具体检查逻辑参考 4.4.3.13 和 4.4.3.14 章节；
- 检查拥有者账户与签名者账户是否属于同一账户或拥有者账户是否授权给签名者账户，否则返回提示信息；
- 检查数量集合长度是否为 0，是则返回提示信息；
- 检查 DDCID 集合长度是否为 0，是则返回提示信息；
- 检查数量集合与 DDCID 集合长度是否相等，不相等则返回提示信息；
- 批量循环挨个处理 DDC，并根据索引检查每个 DDCID 是否存在，不存在则返回提示信息，再根据索引检查 DDCID 是否被冻结，是则返回提示信息，再根据索引检查 DDCID 是否被锁定，是则返回提示信息，再根据索引获取需要安全转移的每个 DDC 所对应的数量是否大于 0，不是则返回提示信息，最后根据索引检查拥有者账户拥有 DDC 所对应的数量是否大于等于需要安合转移的 DDC 数量，不是则返回提示信息，是则挨个调用计费合约支付 DDC 业务费和保存 DDC 安全转移数据；最后触发事件，并检查接收者账户是否对应的是一个合约，是则需检查接收者账户是否接收 DDC；

4.3.3.22 元交易销毁

DDC 拥有者或授权者通过授权平台方调用该 API 进行 DDC 元交易销毁。

- 输入参数：DDC 拥有者账户，ddc 唯一标识，nonce 值，过期时间，签名值;
- 输出参数:
- 方法命名：metaBurn;
- 方法举例：metaBurn(address owner,uint256 ddcId,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransferSingle(operator,owner,address(0),ddcId,amount);
- 核心逻辑:
  - 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获取签名者账户地址是否于拥有者或授权者账户相同，不是则返回提示信息;
  - 检查签名者账户状态是否可用，不是则返回提示信息;
  - 检查调用者与拥有者是否属于同平台，不是则返回提示信息;
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息;
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息;
  - 检查调用者账户状态是否可用，不可用则返回提示信息;
  - 检查调用者账户是否有权限，没有则返回提示信息;
  - 检查 DDCID 是否存在，不存在则返回提示信息;
  - 检查拥有者账户对 DDC 所拥有的数量是否大于 0，不是则返回提示信息;
  - 检查拥有者账户与签名者账户是否属于同一账户或拥有者账户是否授权给签名者账户，否则返回提示信息。
  - 所有检查通过后则调用计费合约支付 DDC 业务费，并保存 DDC 销毁数据，最后触发事件;

4.3.3.23　元交易批量销毁

DDC 拥有者或授权者通过授权平台方调用该 API 进行 DDC 元交易批量销毁。

- 输入参数：拥有者账户，ddc 唯一标识的集合，nonce 值，过期时间，签名值；
- 输出参数：
- 方法命名：metaBurnBatch;
- 方法举例：metaBurnBatch(address owner,uint256[] memory ddclds,uint256 nonce,uint256 deadline,bytes memory sign);
- 事件：MetaTransferBatch(operator,owner,address(0),ddclds,amounts);
- 核心逻辑：
  - 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获取签名者账户地址是否于拥有者或授权者账户相同，不是则返回提示信息；
  - 检查签名者账户状态是否可用，不是则返回提示信息；
  - 检查调用者与拥有者是否属于同平台，不是则返回提示信息；
  - 检查过期时间是等于 0 或区块的时间戳是否小于等于过期时间，都不是则返回提示信息；
  - 检查 Nonce 值列表中所对应的签名者 nonce 值加 1 后与 nonce 值参数是否相等，不是则返回提示信息；
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查调用者账户与签名者账户是否属于同一账户或拥有者账户是否将自己所有的 DDC 授权给签名者账户，不是则返回提示信息；
  - 检查 DDCID 集合长度是否为 0，是则返回提示信息；
  - 批量循环 DDCID 集合列表，并根据索引 DDCID 是否存在，不存在则返回提示信息，再根据索引获取 DDCID 检查拥有者账户对 DDC 所拥有的数量是否大于 0，不是则返回提示信息，是则调用计费合约支付 DDC 业务费和保存 DDC 销毁数据，

- 最后触发事件;

4.3.3.24 Nonce 查询

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。

- 输入参数：拥有者;
- 输出参数：无;
- 方法命名：getNonce;
- 方法举例：getNonce(address from) view returns (uint256);
- 核心逻辑：
  - 返回查询的签名者最新 nonce 值;

4.3.3.25 DDC 跨链锁定

平台方或终端用户通过 DDC 跨链应用合约调用该 API 进行 DDC 跨链锁定。

- 输入参数：DDC 唯一标识;
- 输出参数：
- 方法命名：lock;
- 方法举例：lock(uint256 ddcId);
- 事件：Locklist(operator,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息;
  - 检查调用者账户是否有权限，没有则返回提示信息;
  - 检查 DDCID 是否存在，不存在则返回提示信息;
  - 检查 DDCID 是否被冻结，是则返回提示信息;
  - 检查 DDCID 是否被锁定，是则返回提示信息;
  - 检查 DDCID 所对应的拥有者账户列表长度是否等于 1，不是则返回提示信息;

- 所有检查通过后则将 DDCID 加入冻结列表，最后触发事件；

4.3.3.26 DDC 跨链解锁

平台方或终端用户通过 DDC 跨链应用合约调用该 API 进行 DDC 跨链解锁。

- 输入参数：DDC 唯一标识；
- 输出参数：
- 方法命名：unlock;
- 方法举例：unlock(uint256 ddcId);
- 事件：UnLocklist(operator,ddcId);
- 核心逻辑：
  - 检查调用者账户状态是否可用，不可用则返回提示信息；
  - 检查调用者账户是否有权限，没有则返回提示信息；
  - 检查 DDCID 是否存在，不存在则返回提示信息；
  - 检查 DDCID 是否被冻结，是则返回提示信息；
  - 检查 DDCID 是否被解锁，是则返回提示信息；
  - 所有检查通过后则将 DDCID 移除冻结列表，最后触发事件；

4.2.3.34 同步拥有者

运营方通过调用该 API 接口将旧的 DDC 所对应的拥有者列表信息进行同步。

- 输入参数：ddc 唯一标识列表、拥有者列表；
- 输出参数：无；
- 方法命名：syncDDCOwners;
- 方法举例：syncDDCOwners(uint256[] memory ddcIds,address[][] memory owners);
- 事件：SyncDDCOwners(address indexed operator,uint256[] ddcIds,address[][] owners);

- 核心逻辑：
  - 检查调用者账户信息是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户类型是否为运营方，不是则返回提示信息；
  - 检查 ddc 唯一标识列表和拥有者列表长度是否大于 0，不是则返回提示信息；
  - 检查 ddc 唯一标识列表和拥有者列表长度是否相等，不是则返回提示信息；
  - 批量循环挨个处理 DDC 信息，处理逻辑如下；
    1. 根据索引检查 ddc 唯一标识是否存在，不存在则返回提示信息；
    2. 根据索引检查 ddc 唯一标识在 DDC 拥用者账户列表中是否存在，存在则返回提示信息；
    3. 根据索引检查拥有者列表长度是否大于 0，不是则返回提示信息；
    4. 根据索引检查拥有者列表在 DDC 拥用者账户列表中是否存，存在则返回提示信息；
    5. 所有检查通过后则保存 DDC 拥用者账户信息。
  - 最后触发事件；

4.2.3.35 查询拥有者

DDC 跨链应用合约通过调用该方法查询 DDCID 所对应的拥有者账户列表。

- 输入参数：DDC 唯一标识；
- 输出参数：无；
- 方法命名：ownerOf;
- 方法举例：ownerOf(uint256 ddcId) view returns (address[] memory);
- 核心逻辑：
  - 返回根据 DDCID 所对应的拥有者列表；

4.4 BSN-DDC-权限合约

4.4.1 功能介绍

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

- 用户方权限：经与平台方绑定以后可进行整个 DDC 的生命周期的管理操作。
- 平台方权限：包含 DDC 的整个生命周期的管理以及本平台内用户方账户的增删改查操作。
- 运营方权限：包含用户方账户及权限、平台方账户及权限以及运营方账户和权限的增删改查操作，同时对 DDC 相关流转操作进行定价（DDC 发行、转移等操作需扣除相应的费用）以及对平台方账户及用户方账户的充提操作。

4.4.2 数据结构

- BSN-DDC-权限管理-账户信息

编号	字段名	字段	类型	备注
1.	私钥地址	account	address	DDC 用户链账户地址
2.	账户信息	_accountsInfo	AccountInfo	DDC 账户信息
AccountInfo				
1.	账户 DID	accountDID	string	DDC 账户对应的 DID 信息（普通用户可为空）
2.	账户名称	accountName	string	DDC 账户对应的账户信息
3.	账户角色	accountRole	enum	DDC 账户对应的身份信息。值包含： 0.Operator（运营方） 1.PlatformManager（平台方） 2.Consumer（用户方）
4.	账户上级管理者	leaderDID	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
5.	平台管理账户状态	platformState	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: 0.Frozen（冻结状态，无法进行 DDC 相关操作） 1.Active（活跃状态，可进行 DDC 相关操作）
6.	运营管理账户状态	operatorState	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: 0.Frozen（冻结状态，无法进行 DDC 相关操作） 1.Active（活跃状态，可进行 DDC 相关操作）
7.	冗余字段	field	string	冗余字段

➢ BSN-DDC-权限管理-方法信息

编号	字段名	字段	类型	备注
1.	账户角色	role	string	DDC 账户对应的身份信息。值包含: 0.Operator（运营方） 1.PlatformManager（平台方） 2.Consumer（用户方）
2.	合约方法	_funcAclList	FunAcl[]	合约方法
FunAcl				
1.	合约地址	contractAddress	address	DDC 合约地址
2.	方法列表	funList	string[]	DDC 角色对应可访问的方法列表
3.	索引集合	sigIndexList	mapping(bytes4 => uint256)	合约方法签名所对应的索引集合

➢ BSN-DDC-权限管理-合约索引集合

编号	字段名	字段	类型	备注
1.	索引集合	_contractIndexList	mapping(Role =>	账户类型所对应的合约索引集合

			mapping(address => uint256))	

- BSN-DDC-权限管理-账户 DID 授权集合

编号	字段名	字段	类型	备注
1.	账户 DID 授权集合	_didApprovals	mapping(string => mapping(string => bool))	账户 DID 授权集合 Key：授权者账户 DID Value：接收者账户 DID->是否授权

- BSN-DDC-权限管理-平台方 DID 集合

编号	字段名	字段	类型	备注
1.	平台方 DID 集合	_platformDIDs	mapping(string => bool)	平台方 DID 集合 Key：平台方 DID Value：是否存在

- BSN-DDC-权限管理-平台方添加链账户开关

编号	字段名	字段	类型	备注
1.	是否打开	_platformSwitcher	bool	

- BSN-DDC-权限管理-启用批量开关

编号	字段名	字段	类型	备注
1.	是否启用批量	_enableBatch	bool	

4.4.3 API 定义

4.4.3.1 添加运营账户

部署合约所对应的拥有者可以通过调用该 API 可以添加运营方账户。

- 输入参数：账户地址，账户名称，账户 DID；
- 输出参数：无；
- 方法命名：addOperator；
- 方法举例：addOperator(address operator,string memory accountName,string memory accountDID);

- 事件：AddAccount(address indexed caller,address indexed account);
- 核心逻辑：
  - 检查调用者是否为合约拥有者，不是则返回提示信息;
  - 检查账户 DID 长度是否为 0，是则返回提示信息;
  - 检查账户名称长度是否为 0，是则返回提示信息;
  - 检查添加的运营方账户是否存在，存在则返回提示信息;
  - 所有检查通过后则保存运营账户数据，最后触发事件;

4.4.3.2 平台方添加账户

平台方通过调用该 API 进行 DDC 账户信息的创建，address 为用户的链账户地址，AccountInfo 为该用户的详细信息，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。

- 输入参数：链账户地址，账户名称，账户 DID;
- 输出参数：无;
- 方法命名：addAccountByPlatform;
- 方法举例：addAccountByPlatform(address account,string memory accountName,string memory accountDID);
- 事件：AddAccount(address indexed caller,address indexed account);
- 核心逻辑：
  - 检查调用者账户信息是否存在，不存在则返回提示信息;
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息;
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息;
  - 检查调用者账户类型是否为平台方，不是则返回提示信息;
  - 检查调用者账户是否打开添加链账户开关控制，不是则返回提示信息;
  - 检查添加的账户名称长度是否为 0，是则返回提示信息;
  - 检查添加的账户是否已存在，存在则返回提示信息;
  - 所有检查通过后则保存账户数据，最后触发事件;

4.4.3.3 平台方批量添加账户

平台方通过调用该 API 进行 DDC 账户信息的创建，address 为用户的链账户地址，AccountInfo 为该用户的详细信息，上级角色可进行下级角色账户的操作，平台方通过该方法批量添加终端账户。

- 输入参数：链账户地址集合，账户名称集合，账户 DID 集合；
- 输出参数：无；
- 方法命名：addBatchAccountByPlatform;
- 方法举例：addBatchAccountByPlatform(address[] memory accounts,string[] memory accountNames,string[] memory accountDIDs);
- 事件： AddBatchAccount(address indexed operator,address[] accounts);
- 核心逻辑：
  - 检查是否启用批量开关设置，是则返回提示信息；
  - 检查调用者账户信息是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户类型是否为平台方，不是则返回提示信息；
  - 检查链账户地址集合长度是否为 0，是则返回提示信息；
  - 检查账户名称集合长度是否为 0，是则返回提示信息；
  - 检查账户 DID 集合长度是否为 0，是则返回提示信息；
  - 检查链账户地址集合长度、账户名称集合长度以及账户 DID 集合长度是否相等，不相等则返回提示信息；
  - 批量循环挨个处理链账户的添加，处理逻辑如下：
    1. 检查调用者账户是否打开添加链账户开关控制，不是则返回提示信息；
    2. 根据索引检查添加的账户名称长度是否为 0，是则返回提示信息；
    3. 根据索引检查添加的账户是否已存在，存在则返回提示信息；
    4. 所有检查通过后则保存账户数据；
  - 最后触发事件；

4.4.3.4 平台方添加链账户开关设置

运营方通过调用该 API 接口设置平台方添加链账户开关权限控制。

- 输入参数：是否打开；
- 输出参数：无；
- 方法命名：setSwitcherStateOfPlatform;
- 方法举例：setSwitcherStateOfPlatform(bool isOpen);
- 事件：SetSwitcherStateOfPlatform(address indexed operator,bool isOpen);
- 核心逻辑：
  - 检查调用者账户信息是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户类型是否为运营方，不是则返回提示信息；
  - 检查 isOpen 是否与当前开关状态一致，一致则返回提示信息；
  - 所有检查通过后则保存数据，最后触发事件;

4.4.3.5 平台方添加链账户开关查询

运营方、平台方或终端用户可以通过调用该 API 接口查询平台方添加链账户开关所对应的状态。

- 输入参数：无；
- 输出参数：bool 结果；
- 方法命名：switcherStateOfPlatform;
- 方法举例：switcherStateOfPlatform() view returns (bool);
- 核心逻辑：
  - 返回平台方添加链账户开关状态;

4.4.3.6 运营方添加账户

运营方通过调用该 API 可以直接对平台方或平台方的终端用户进行创建。

- 输入参数：链账户地址，账户名称、账户 DID，上级 DID；
- 输出参数：无；
- 方法命名：addAccountByOperator;
- 方法举例：addAccountByOperator(address account,string memory accountName,string memory accountDID,leaderDID);
- 事件：AddAccount(address indexed caller,address indexed account);
- 核心逻辑：
  - 检查调用者账户信息是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户类型是否为运营方，不是则返回提示信息；
  - 检查添加的账户名称长度是否为 0，是则返回提示信息；
  - 检查添加的账户是否存在，存在则返回提示信息；
  - 检查上级 DID 是否为空，如果不为空则检查上级 DID 所对应的平台方是否存在，不存在则返回提示信息；如果为空则检查账户 DID 是否为空，是则返回提示信息；
  - 所有检查通过后则保存账户数据，最后触发事件;
  - 注：如果添加的账户所对应的上级 DID 为空，则说明添加的账户为平台方；

4.4.3.7 运营方批量添加账户

运营方通过调用该 API 可以直接对平台方或平台方的终端用户进行创建。

- 输入参数：链账户地址集合，账户名称集合、账户 DID 集合，上级 DID 集合；
- 输出参数：无；
- 方法命名：addBatchAccountByOperator;
- 方法举例：addBatchAccountByOperator(address[] memory accounts,string[] memory accountNames,string[] memory accountDIDs,string[] memory leaderDIDs);

- 事件：AddBatchAccount(address indexed operator,address[] indexed accounts);
- 核心逻辑：
  - 检查是否启用批量开关设置，是则返回提示信息；
  - 检查调用者账户信息是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户类型是否为运营方，不是则返回提示信息；
  - 检查链账户地址集合长度是否为 0，是则返回提示信息；
  - 检查账户 DID 集合长度是否为 0，是则返回提示信息；
  - 检查上级 DID 集合长度是否为 0，是则返回提示信息；
  - 检查链账户地址集合，账户名称集合、账户 DID 集合以及上级 DID 集合长度是否相等，不相等则返回提示信息；
  - 批量循环挨个处理链账户的添加，处理逻辑如下：
    1. 根据索引检查添加的账户名称长度是否为 0，是则返回提示信息；
    2. 根据索引检查添加的账户是否存在，存在则返回提示信息；
    3. 根据索引检查上级 DID 是否为空，如果不为空则检查上级 DID 所对应的平台方是否存在，不存在则返回提示信息；如果为空则检查账户 DID 是否为空，是则返回提示信息；
    4. 根据索引所有检查通过后则保存账户数据；
  - 最后触发事件。注：如果添加的账户所对应的上级 DID 是否为空，则说明添加的账户为平台方；

4.4.3.8 更新账户状态

运营方或平台方通过该 API 进行 DDC 账户信息状态的更改，address 为用户的链账户地址，平台方可冻结/解冻用户的平台方状态标识，运营方可冻结/解冻用户和平台的平台方状态标识和运营方状态标识。

- 输入参数：账户地址，账户状态，是否更新平台方状态；
- 输出参数：无；

- 方法命名：updateAccountState;
- 方法举例：updateAccountState(address account,State state,bool changePlatformState);
- 事件：UpdateAccountState(address indexed account,IAuthorityData.State platformState,IAuthorityData.State operatorState);
- 核心逻辑：
  - 检查账户信息是否存在，不在在则返回提示信息；
  - 检查调用者账户信息是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查账户对应的上级 DID 与调用者所对应的账户 DID 是否相等或调用者账户类型是否等于运营方，不相等则返回提示信息；
  - 如果调用账户类型为运营方，则分支检查如下：
    1. 如果 changePlatformState 为 true，则需要检查 state 与账户所对应的平台方状态是否相等，相等则返回提示信息；
    2. 如果 changePlatformState 为 false，则需要检查 state 与账户所对应的运营方状态是否相等，相等则返回提示信息；
  - 如果调用账户类型为平台方，则需要检查账户的平台方状态是否与 state 是否相等，相等则返回提示信息；
  - 所有检查通过后则更新账户状态，最后触发事件；

4.4.3.9 查询账户

运营方、平台方以及终端用户通过该 API 进行 DDC 账户信息的查询，address 为用户的链账户地址，上级角色可进行下级角色账户的操作。

- 输入参数：账户地址；
- 输出参数：账户信息；
- 方法命名：getAccount;
- 方法举例：getAccount(address account) view returns (AccountInfo);
- 核心逻辑：

- 所有检查通过后则返回查询结果；

4.4.3.10 删除账户

运营方通过该 API 进行 DDC 账户信息的删除，address 为用户的链账户地址，上级角色可进行下级角色账户的操作，暂不对外开放调用。

- 输入参数：账户地址；
- 输出参数：
- 方法命名：delAccount;
- 方法举例：delAccount(address account);
- 事件：DelAccount(address indexed account);
- 核心逻辑：
  - 检查待删除账户是否已存在，不存在则返回提示信息;
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查待删除账户的上级是否与调用者账户是否匹配（通过该账户的上级 DID 和调用者所对应的账户 DID 进行匹配），不匹配则返回提示信息;
  - 所有检查通过后则删除账户数据，最后触发事件;

4.4.3.11 账户状态检验

DDC 业务主代理合约通过该 API 进行查询某个链账户地址对应的账户是否可用。

- 输入参数：账户地址；
- 输出参数：账户角色；
- 方法命名：accountAvailable;
- 方法举例：accountAvailable(address account) view returns (bool);
- 核心逻辑：
  - 调用数据合约校验该账户是否存在，不存在则返回提示信息；
  - 校验该账户的平台方状态是否活跃，不活跃则返回提示信息；

- 检查该账户的运营方状态是否活跃，不活跃则返回提示信息；
- 所有检查通过后则返回该账户状态是否可用；

4.4.3.12 角色断言

DDC 业务主代理合约通过该 API 进行对某个账户对应的角色进行断言。

- 输入参数：账户地址，账户角色；
- 输出参数：bool 结果；
- 方法命名：checkAvailableAndRole;
- 方法举例：checkAvailableAndRole(address account,Role role) view returns (bool);
- 核心逻辑：
  - 调用数据合约校验该账户是否存在，不存在则返回提示信息；
  - 校验该账户的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查该账户的运营方状态是否活跃，不活跃则返回提示信息；
  - 所有检查通过后则则返回该账户所对应的账户类型与传入的角色类型的比较结果；

4.4.3.13 同平台检验

DDC 业务主代理合约通过该 API 进行查询两个账户是否属于同平台账户。

- 输入参数：账户 1，账户 2；
- 输出参数：bool 结果；
- 方法命名：onePlatformCheck;
- 方法举例：onePlatformCheck(address acc1,address acc2) view returns (bool);
- 核心逻辑：
  - 检查账户 1 身份是否存在，不存在则返回提示信息；
  - 检查账户 1 对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查账户 1 对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查账户 2 身份是否存在，不存在则返回提示信息；

- 检查账户 2 对应的平台方状态是否活跃，不活跃则返回提示信息;
- 检查账户 2 对应的运营方状态是否活跃，不活跃则返回提示信息;
- 如果账户 1 所对应的账户类型为平台方且账户 2 所对应的账户类型为平台方，则检查账户 1 所对应的账户 DID 与账户 2 所对应的账户 DID 是否匹配且账户 1 所对应的上级 DID 与账户 2 所对应的上级 DID 是否匹配，如果不匹配则返回提示信息，匹配则返回 true;
- 如果账户 1 所对应的账户类型为平台方且账户 2 所对应的账户类型为终端用户，则检查账户 1 所对应的账户 DID 与账户 2 所对应的上级 DID 是否匹配，如果不匹配则返回提示信息，匹配则返回 true;
- 如果账户 1 所对应的账户类型为终端用户且账户 2 所对应的账户类型为平台方，则检查账户 1 所对应的上级 DID 与账户 2 所对应的账户 DID 是否匹配，如果不匹配则返回提示信息，匹配则返回 true;
- 如果账户 1 所对应的账户类型为终端用户且账户 2 所对应的账户类型为终端用户，则检查账户 1 所对应的上级 DID 与账户 2 所对应的上级 DID 是否匹配，如果不匹配则返回提示信息，匹配则返回 true;
- 其它条件则返回 false;

4.4.3.14 跨平台检验

DDC 业务主代理合约通过该 API 进行查询两个账户是否属于跨平台账户。

- 输入参数：授权者账户，接收者账户;
- 输出参数：bool 结果;
- 方法命名：crossPlatformCheck;
- 方法举例：crossPlatformCheck(address from,address to) view returns (bool);
- 核心逻辑：
  - 检查授权者账户是否存在，不存在则返回提示信息;
  - 检查授权者账户对应的平台方状态是否活跃，不活跃则返回提示信息;
  - 检查授权者账户对应的运营方状态是否活跃，不活跃则返回提示信息;
  - 检查接收者账户是否存在，不存在则返回提示信息;
  - 检查接收者账户对应的平台方状态是否活跃，不活跃则返回提示信息;

- 检查接收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；
- 如果授权者账户所对应的账户类型为平台方且接收者所对应的账户类型为平台方，则检查授权者账户所对应的上级 DID 与接收者账户所对应的上级 DID 是否相同且授权者账户所对应的账户 DID 与接收者账户所对应的账户 DID 是否不同，如果条件不满足则返回提示信息，条件满足则检查授权者账户 DID 和授权者账户 DID 在账户 DID 授权列表中是否存在，不存在则返回提示信息，存在则返回 true；
- 如果授权者所对应的账户类型为平台方且接收者所对应的账户类型为终端用户，则检查授权者所对应的账户 DID 与接收者所对应的上级 DID 在账户 DID 授权列表中是否存在，不存在则返回提示信息，存在则返回 true；
- 如果授权者所对应的账户类型为终端用户且接收者所对应的账户类型为平台方，则检查授权者所对应的上级 DID 与接收者所对应的账户 DID 在账户 DID 授权列表中是否存在，不存在则返回提示信息，存在则返回 true；
- 如果授权者所对应的账户类型为终端用户且接收者所对应的账户类型为终端用户，则检查授权者所对应的上级 DID 与接收者所对应的上级 DID 在账户 DID 授权列表中是否存在，不存在则返回提示信息，存在则返回 true；
- 其它条件则返回 false;

4.4.3.15 添加方法

运营方通过该 API 进行角色可调用方法的增加。

- 输入参数：账户角色，合约地址，方法名称；
- 输出参数：无；
- 方法命名：addFunction;
- 方法举例：addFunction(Role role,address contractAddress,byte4 sig);
- 事件：AddFunction(address indexed operator,Role indexed role,address contractAddress,byte4 sig)

- 核心逻辑：
  - 检查调用者身份是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者身份是否为运营方，不是则返回提示信息；
  - 所有检查通过后则保存代理合约所对应的方法数据，最后触发事件；

4.4.3.16 删除方法

运营方通过该 API 进行角色可调用方法的删除。

- 输入参数：账户角色，合约地址，方法名称；
- 输出参数：无；
- 方法命名：delFunction;
- 方法举例：delFunction(Role role,address contractAddress,bytes4 sig);
- 事件：DelFunction(address indexed operator,Role indexed role,address contractAddress,byte4 sig)
- 核心逻辑：
  - 检查调用者身份是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者身份是否为运营方，不是则返回提示信息；
  - 检查当前待删除的该方法是否存在，不存在则返回提示信息；
  - 所有检查通过后则删除代理合约所对应的方法数据，最后触发事件；

4.4.3.17 查询方法

运营方通过该 API 进行角色可调用方法的查询。

- 输入参数：账户角色，合约地址；
- 输出参数：方法列表；
- 方法命名：getFunctions;

- 方法举例：getFunctions(Role role,address contractAddress) view returns (bytes4[] memory);
- 核心逻辑：
  - 返回查询结果；

4.4.3.18 方法权限校验

DDC 业务主代理合约通过该 API 进行调用账户、被调用合约的被调用方法之间权限的校验。

- 输入参数：账户地址，合约地址，方法名称；
- 输出参数：bool 结果；
- 方法命名：hasFunctionPermission;
- 方法举例：hasFunctionPermission(address account,address contractAddress,bytes4 sig) view returns(bool);
- 核心逻辑：
  - 检查账户是否存在，不存在则返回提示信息；
  - 检查账户所对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查账户所对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 根据账户的所属类型及合约地址获取方法列表，并检查所传的方法签名在方法列表中是否存在，并返回检查结果；

4.4.3.19 跨平台授权

运营方通过调用该 API 进行跨平台之间的账户授权。

- 输入参数：授权者账户，接收者账户，授权标识；
- 输出参数：无；
- 方法命名：crossPlatformApproval;
- 方法举例：crossPlatformApproval(address from,address to,bool approved);
- 事件：CrossPlatformApproval(address indexed from,address indexed to, bool approved)；

- 核心逻辑：
  - 检查调用者账户信息是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户是否为运营方，不是则返回提示信息；
  - 检查授权者账户信息是否存在，不存在则返回提示信息；
  - 检查授权者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查授权者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查授权者账户是否为平台方，不是则返回提示信息；
  - 检查接收者账户是否存在，不存在则返回提示信息；
  - 检查接收者账户所对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查接收者账户所对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查接收者账户是否为平台方，不是则返回提示信息；
  - 检查授权者账户与接收者账户是否属于同平台，是则返回提示信息；
  - 所有检查通过后则保存跨平台授权结果，最后触发事件；

4.4.3.20 同步平台方 DID

运营方通过调用该 API 接口将旧平台方链账户所对应的 DID 同步存储到平台方 DID 集合。

- 输入参数：是否打开；
- 输出参数：无；
- 方法命名：syncPlatformDID;
- 方法举例：syncPlatformDID(string[] memory dids);
- 事件：SyncPlatformDID(address indexed operator,string[] dids);
- 核心逻辑：
  - 检查调用者账户信息是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户类型是否为运营方，不是则返回提示信息；

- 批量循环挨个处理链账户的添加，处理逻辑如下；

  6. 根据索引检查 DID 是否为空，是则返回提示信息;

  7. 根据索引检查 DID 在 DID 集合中是否存在，是则返回提示信息；

  8. 所有检查通过后则保存平台方 DID 数据。

- 最后触发事件；

4.4.3.21 启用批量设置

运营方可以通过此接口对是否启用批量开关进行设置。

- 输入参数：是否打开；
- 输出参数：无；
- 方法命名：setSwitcherStateOfBatch;
- 方法举例：setSwitcherStateOfBatch(bool isOpen);
- 事件：SetSwitcherStateOfBatch(address indexed operator,bool isOpen);
- 核心逻辑：
  - 检查调用者账户是否存在，不存在则返回提示信息；
  - 检查调用者账户对应的平台方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户对应的运营方状态是否活跃，不活跃则返回提示信息；
  - 检查调用者账户类型是否为运营方，不是则返回提示信息；
  - 检查 isOpen 是否与当前开关状态一致，一致则返回提示信息；
  - 所有检查通过后则保存数据，最后触发事件;