dGoods：虚拟物品代币规范 v0.1

Cameron Thacker*　　Stephen Cunningham*　　Rudy Koch*
John Linden*
译：TokenPocket

1. 介绍

dGoods 是 EOS 区块链上的一个开源和免费的标准，适用于现实与虚拟物品的数字表示。虽然我们最初的重心是做游戏的虚拟资产，但我们正在将其范围扩大至其他类型的数字货品。事实上，我们后来为其他资产编写元数据标准，包括票据，3D / 2D 游戏资产，艺术品和珠宝等。我们相信，当在 EOS 区块链上有需要支持非同质化，半同质化或同质化的代币需求时，该规范可以成为标准。

假设一款游戏或一个公司想要基于一个主代币下创建许多不同的代币， 这些代币中的可能部分是同质化的或半同质化的，而其余的是非同质化的。真正将此标准与其他现有的 NFT 标准区分开来的是代币的分层命名结构模式。同质化代币都可以通过 符号:类别:名字 进行识别，而 NFT 还有一个额外的 token_id 与之关联。这个代币提供了一种崭新的组织方式，它也让钱包和 dApps 通过类别和名字进行代币展示，并且提供了搜索和过滤功能。

我们的设计理念是轻量化和灵活化，并通过提供强大的虚拟物品标准，以支持多元化和创新的开发社区。因此，该规范的大部分内容都是标准 EOS 代币的逻辑扩展，并增加了重要的功能改进，使项目方可以轻松集成和显示虚拟物品。在以下部分中，我们将介绍实现此规范所需的最小方法集和功能集，以及表结构，最后展示一些元数据模板的示例。

2. 必选函数

CREATE : create 函数将实例化一个代币。它是在任何代币进行发行前，或者设置诸如代币类别、名字、最大供应量、谁有权限进行发行代币，是否是非同质化的等等属性前，必须执行的一个方法。此外，第一次调用该方法时，代币符号会被记录，并且在后续调用时，必须指定改符号。符号必须是大写 A-Z 组成，请最长为 7 位字符。Name 类型是一个 a-z, 1-5 组成的最长 12 位字符的字符串。

ACTION create(name issuer, string symbol, name category, name token_name, bool fungible, bool burnable, bool transferable, int64_t max_supply);

ISSUE : issue 函数可以发行一个代币，并且将所有权指定给账户名’ to’ . 对于一个有效的函数调用，代币的符号、类别、名字都必须都先被创建好了。如果是非同质化或半同质化的代币，则数量必须等于 1，如果是同质化代币，则数量必须是大于等于 0.0001

ACTION issue(name to, string symbol, name category, name token_name, double quantity, string metadata_uri, string memo);

PAUSEXFER : 暂停所有代币的所有转账，仅有合约可以调用该函数。如果 pause 参数值为 true，则暂停转账；否则，不暂停。

ACTION pausexfer(bool pause);

BURNNFT：该函数可用于销毁第三方代币并且释放内存，只有代币的拥有者 (owner) 可以调用该方法，并且 burnable 参数值必须为 true

---

ACTION burnnft(name owner, vector<uint64_t> tokeninfo_ids);

BURN：该方法用于销毁同质化代币，并且在所有代币都销毁后释放内存 (RAM)，只有拥有者 (owner) 可以调用该方法，并且 burnable 参数值必须为 true

---

ACTION burn(name owner, uint64_t global_id, double quantity);

TRANSFERNFT：用于非同质化代币的转账。并且允许通过传递代币的 id 列表进行批量转账。仅有代币拥有者 (owner) 可以调用改函数并且 transferable 参数值必须为 true。

ACTION transfernft(name from, name to, vector<uint64_t> tokeninfo_ids, string memo);

TRANSFER：该标准的转账方法仅用于同质化代币；区别于将每个代币都一一区分，这里转账的代币由一个 global_id 进行区分，并且后面跟着希望发送的代币数量。

ACTION transfer(name from, name to, uint64_t global_id, double quantity);

3. 代币数据

3.1 符号信息表

为合约管理符号信息的单例，并在第一次调用 create 函数的时候设置好。global_id 在每次成功调用 create 函数时都会自增。

```
// scope is self
TABLE symbolinfo {
   string symbol;
   uint64_t global_id;
};
```

3.2 代币数据表

确保只会有一个类别和代币名称对，存储一个代币是否是同质化的或者是否可以销毁，还有当前的供应量和最大的供应量是多少。这些信息都是代币被创建的时候写入的。

```
// scope is category, then token_name is unique
TABLE tokenstats {

uint64_t global_id;

 name issuer;

string symbol;

name category;

name token_name;

bool fungible;

bool burnable;

bool transferable;
```

```
double current_supply;

uint64_t max_supply;

uint64_t primary_key() const { return token_name.value; }

};
```

3.3 代币信息表

这是非同质化代币和半同质化代币的全局列表；辅助索引提供了按照所有者 (owner)，类别 (category) 或者代币名称 (token name) 进行搜索。

```
// scope is self
TABLE tokeninfo {
   uint64_t id;
   uint64_t global_id;
   name owner;
   name category;
   name token_name;
   uint64_t serial_number;
   string metadata_uri;

   uint64_t primary_key() const { return id; }
   uint64_t get_owner() const { return owner.value; }
   uint64_t get_token_name() const { return token_name.value; }
   uint64_t get_category() const { return category.value; }
};
```

3.4 分类表

存储所有的分类名称，以便进行快查

```
// scope is self
```

```
TABLE categoryinfo {
  name category;

  uint64_t primary_key() const { return category.value; }
};
```

3.5 账户表

账户表存储了账户的同质化代币，以及对应账户拥有指定类型 NFT 的数量

```
// scope is owner
TABLE account {
   uint64_t global_id;
    name category;
    name token_name;
    double amount;

    uint64_t primary_key() const { return global_id; }
};
```

4.元数据模板

为了使钱包和 dApp 可以支持各种数字物品，需要有与元数据相关的标准。我们的方法是根据物品的类型定义模板。以下几个是我们提出的候选模板，但这是一项协作练习，同时我们希望提供一个被社区所认可的模板仓库。所有的元数据都以 JSON 格式化，并且必须指定模板的类型。

4.1 类型：3d 游戏资产

```
{
// 必选
"name" - string; 标识代币所代表的资产
"description": string; 简述代币所代表的资产
"imageSmall": 指向图像资源大小为150 x 150的URI
"imageLarge": 指向图像资源大小为1024x1024的URI
"3drender": 指向渲染3d图像的js webgl的URI
"details": 渲染详细内容的键值对，例如{"strength": 5}
// 可选
"authenticityImage": 指向表示真伪性证书的mime类型图像资源的URI
}
```

4.2 类型：2d 游戏资产

```
{
// 必选
"name": string; 标识代币所代表的资产
"description": string; 简述代币所代表的资产
"imageSmall":指向图像资源大小为150 x 150的URI
"imageLarge":指向图像资源大小为1024x1024的URI
"details":渲染详细内容的键值对，例如{"strength": 5}
// 可选
"authenticityImage":指向表示真伪性证书的mime类型图像资源的URI
}
```

4.3 类型：门票

```
{
// 必选
"name": string; 标识该门票是用于哪个活动
"description": string; 简述该门票代表的活动内容
```

"location": string; 活动被举办的地理位置
"address": string; 活动被举办的地址
"date": Unix时间; 活动开始的日期
"time": Unix时间; 活动开始的时间
"imageSmall":指向图像资源大小为150 x 150的URI
"imageLarge":指向图像资源大小为1024x1024的URI
"details":渲染详细内容的键值对，例如{"openingAct": "Nickelback"}
// 可选
"authenticityImage":指向表示真伪性证书的mime类型图像资源的URI
"duration": string; 活动持续的时间
"row": string; 座位位置的行数
"seat": string; 座位的号码或者代表随意位置的GA
}

4.4 类型：艺术品

{
// 必选
"vame": string; 标识代币所代表的资产
"description": string; 简述代币所代表的资产
"creator": string; 艺术品的创作者
"imageSmall":指向图像资源大小为150 x 150的URI
"imageLarge":指向图像资源大小为1024x1024的URI
"date": unix时间; 艺术品被创作的时间
"details":渲染详细内容的键值对，例如 {"materials": ["oil", "wood"], "location": "Norton Simon Museum"}
// 可选
"authenticityImage":指向表示真伪性证书的mime类型图像资源的URI
}

4.5 类型：珠宝

{
//Required Fields
"name": string; 标识代币所代表的珠宝
"description":简述代币所代表的珠宝
"imageSmall":指向图像资源大小为150 x 150的URI
"imageLarge":指向图像资源大小为1024x1024的URI
"image360":指向图像资源为360图像的URI
"manufactureDate": Unix时间; 珠宝被生产的时间
"manufacturePlace": string; 珠宝所被生产的地点
"details":渲染详细内容的键值对，例如 {"caret": 1}
//Optional Fields
"authenticityImage":指向表示真伪性证书的mime类型图像资源的URI
}

5.定义

- 内存 (RAM) – 指 EOS 区块链上的永久存储空间。必须通过发送 EOS 在 EOS 区块链上进行购买，并且可以在不使用的时候进行出售。
- 账户 (Account) – EOS 账户最多 12 个字符，包含字母 a-z，数字 1-5 和英文句号 (如果使用账户服务)
- 符号 (Symbol) – 定义为最多仅由 7 个大写字母组成
- 同质化 (Fungible) – 指的是像 EOS 一样的可以进行互换的代币，如果代币是同质化的，转账的代币都是同等的。
- 半同质化(Semi-fungible) – 技术上唯一的代币，他们拥有唯一的标识，但是可能和其他代币共享相同或相似的数据。许多的实体物品都属于这一类别，因为他们除了唯一的序列号外，在其他方面无法区分。
- 非同质化 (Non-fungible) – 任何一个代币都是唯一的