# Treetrunk.io 版税友好NFT智能合约标准

## 摘要

该解决方案在扩展ERC-721标准的智能合约框架中直接链接了NFT和版税，以防止链上核心结构操纵或规避版税享有者的合法收入。

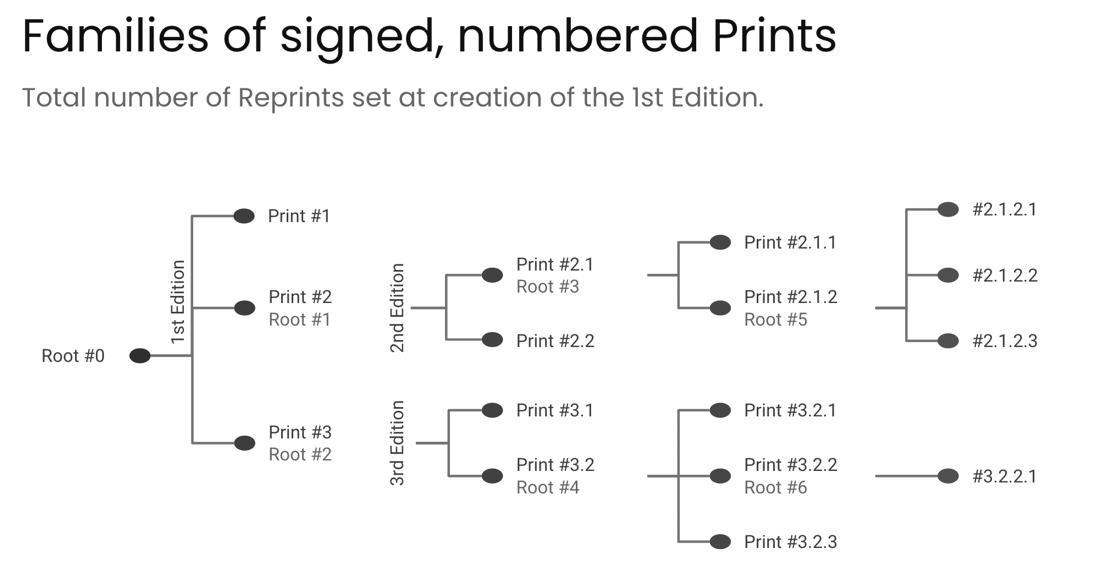
该提案建立在OpenZeppelin智能合约工具箱架构的基础上，并将其扩展到包括版税账户管理（CRUD）、版税余额和支付管理、简单的交易能力如发行/取消发行/购买等功能--以及追踪交易所交易的功能。版税管理功能支持分层树状结构（在此称为版税树），通过逻辑上连接原始NFT迂衍生艺术品，并以递归形式允许衍生品再创作。

## 动机

难以实装合约、管理模棱两可、存在大量欺诈事件等问题一直是NFT版税分配的阻碍。

上述情况对于结构性版税来说尤其如此，在这种情况下，一种或多种资产来自于原始资产，如一幅原画的印刷品，或一首歌曲被用于创作另一首歌曲，或发行权和报酬通过一系列关联公司管理。

在下面的例子中，创作原作的艺术家有资格从每一次印刷品的销售和转售中获得收益。



下图展示了利用上述阶梯版税机制的基本概念：



为了解决复杂的继承问题，本解决方案将N层树状结构的递归问题分解为N个独立的问题。这种结构能够最有效地实行底层向上的遍历问题。

这使得创作者和原作艺术品分销者能够从再创作过程中获取收入，除却原本内在价值外，NFT价值因持续现金流的存在而进一步提升。

## 合约概述

这一解决方案，引入了几个新概念，作为ERC721标准的扩展。

* 版税账户（Royalty Account,RA）

版税账户通过NFT特定tokenId与之相连，并将其他个人账户或版税账户作为子账户。版税账户通过账户标识符进行识别。

* 账户类型（Account Type）

账户类型用于指明RA子账户属于个人拥护还是另外一个RA。若另一个RA作为子账户，则分配的余额需按照规则再分配给下辖的子账户。

* 版税分成（Royalty Split）

每个子账户在销售与RA相关的NFT时获得的百分比。

* 版税余额（Royalty Balance）

与RA相关的版税余额

* 子账户版税余额(Sub Account Royalty Balance)

与每个RA子账户相关的版税余额。注意，只有个人账户才有可以用于支付的余额，当一个RA的子账户中有另一个RA时，其最终的子账户余额必须是0。

* 代币类型（Token Type）

代币类型以ETH或支持ERC 20/23/777的代币，如DAI。

* 资产ID（Asset ID）

该RA所属的tokenId

* 父级（Parent）
* 这表明哪个tokenId是RA所属的tokenId的直接父级。

## 数据结构

* 为了创建一个连接NFT和RA的互联数据结构，并进行搜索优化，需要对ERC721的全局数据结构进行以下补充：
* 为版税账户和相关的版税子账户添加数据结构，以建立版税账户与子账户的概念。
* 将raAccountId定义为tokenId的keccak256哈希值、实际所有者地址和当前区块编号block.blocknumber。
* 将一个tokenId映射到raAccountID，以便将RA的raAccountId与tokenId连接起来
* 将raAccountID映射到RoyaltyAccount，以便将账户标识符与实际账户联系起来。
* 继承关系映射图谱
* 支持的代币类型与它们的起源合同和最后验证的余额的映射（用于交易和版税支付）
* 一个带有struct的映射，用于在executePayment函数中进行注册付款，并在safeTransferFrom中进行验证。其底层原理是，一旦收到付款，并调用safeTransferFrom函数进行分配，将立刻从映射中删除
* 待售的NFT图谱映射

## 版税账户功能

版税账户RUD（读取-更新-删除）函数的定义和接口。因为RA是在造币功能中创建的，所以没有必要单独设立一个创建版税账户的功能。

### Mint支持版税的NFT

当一个NFT被铸造时，必须创建一个RA并与NFT和NFT所有者相关联，如果父账户存在，则与父级RA相关联。为此，本规范在一个新定义的mint函数中利用\_safemint函数，通过更改函数输入，实现上述规则。

列出待售 NFT 并删除列表

授权用户地址可以列出非交易所中介 NFT 购买的待售 NFT。

### 从买家到卖家的付款功能

为避免规避版税，买方始终直接向 NFT 合约而非卖方付款。卖家通过版税分配获得报酬，之后可以要求付款。

### 付款流程取决于付款媒介是 ETH 还是 ERC 20 代币：

如果是ERC 20代币，买方必须批准购买价格的 NFT 合同，支付所选支付代币（ERC20 合同地址）。对于 ERC20 支付代币，买方必须调用 NFT 合约中的 不直接涉及到ERC20的executePayment函数。对于非 ERC20 支付，买方必须向 NFT 合约发送协议令牌（ETH），并且需要发送编码为购买的 NFT uint256[] tokenId 数组的 msg.data。

### 重载的 NFT 传输功能，包括分配版税所需的交易数据

输入参数必须满足 NFT 的几个要求，以便在版税适当分配后进行转移。此外，还考虑了一次转移多个token的能力。

该合约中定义了：

* 输入参数验证
* 支付参数验证
* 分配版税
* 用支出更新 RA 所有权
* 转移 NFT 的所有权
* 转账成功后移除registeredPayment中的Payment条目

### 版税分配

分配版税的方法是将相互关联的 RA 的层次结构分解成层，然后每次处理一层，其中每个token与其父节点之间的关系遍历 RA 链，直到到达根节点和关联的 RA .

### safeTransferFrom 函数中的地址向 NFT 所有者支付版税

这是解决方案的最后一部分。

支付函数有两个版本——公共函数和内部函数。

Public 函数具有如下接口：

function royaltyPayOut (uint256 tokenId, address \_RAsubaccount, address payable \_payoutaccount, payable uint256 \_amount) public virtual nonReentrant returns (bool)

其中我们只需要 tokenId、RA 子账户地址、所有者 \_RAsubaccount 以及要支付的金额 \_amount。注意，因为资金处于”支付”状态吗，因此该函数具有nonReentrant保护。

### 发送付款

需要采取以下步骤：

* 根据RAaccount和subaccountPos找到RA子账户并提取余额
* 从子账户中提取token类型
* 根据token类型，发送payout支付（不超过可用余额）