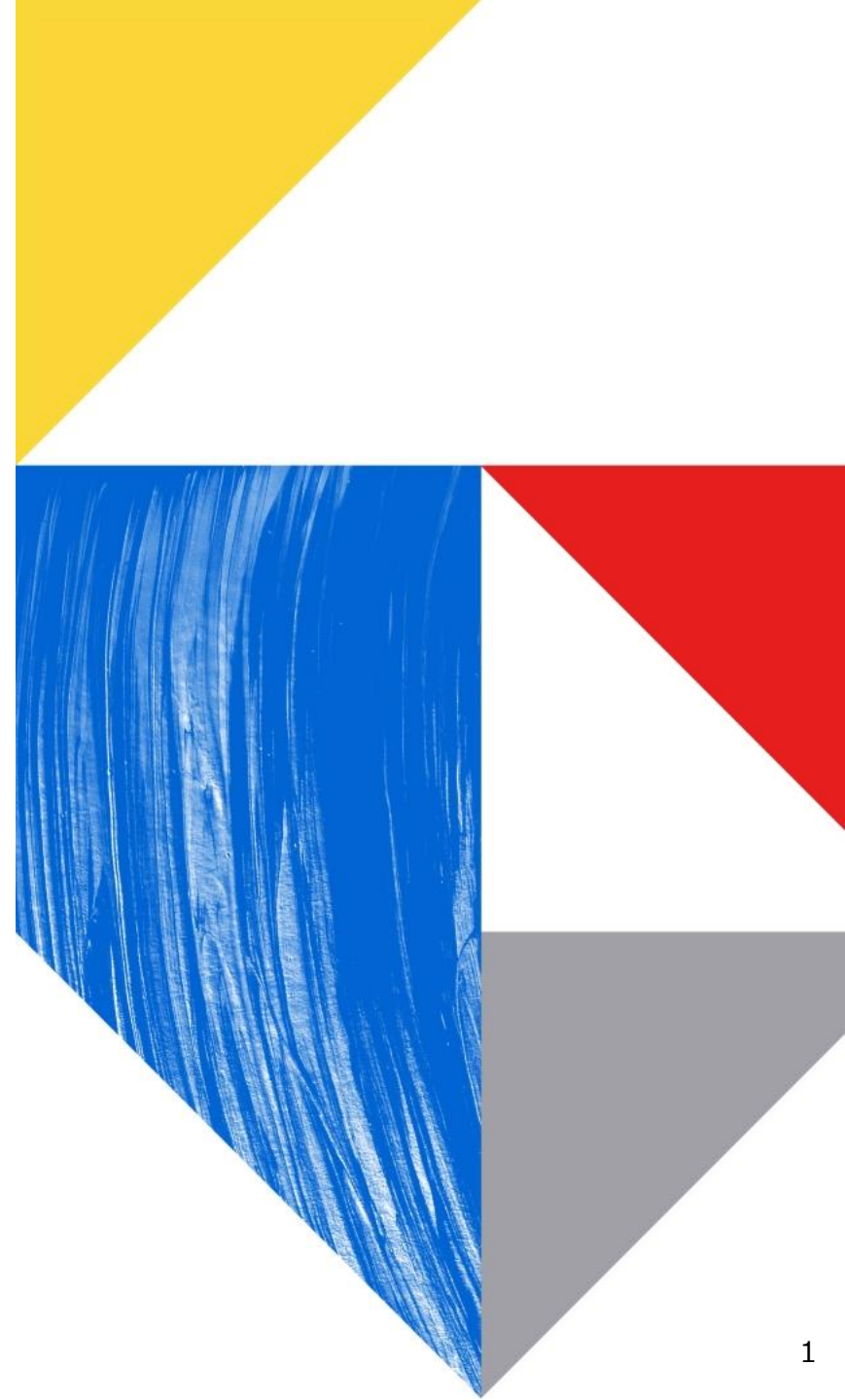


NFTについて



概要

NFT(Non-Fungible Token)は、イーサリアム上で構築できる代替不可能なトークンとして、ゲームや暗号資産などで利用されています。しかしビジネス用途ではイーサリアムの利用自体が難しい場合があります。そこで、同様のトークンをBC+上でも実現し、（これもNFTと呼ぶことにします）エンタープライズ領域でのNFTの活用を広くすすめていきたいと考えています。

本家イーサリアムのNFTはスマートコントラクトとして実装されています。ERC721と呼ばれる標準的なインターフェースが定義されており、NFTの開発者がそれに準拠してスマートコントラクトを作成することにより、アプリケーションからの利用を容易にしています。これと同様に、DNCWARE Blockchain+(BC+)においても、NFTの標準的なインターフェースを定義し、それに準拠する形でNFTをスマートコントラクトとして実装します。

本書では、下記のドキュメントについての概要や利用方法を説明していきます。

- ・NFTインターフェース定義： BC+におけるNFTの標準的なインターフェースを定義したものです。
- ・NFTサンプルコード： 上記インターフェース定義に準拠したNFTコントラクトのサンプルコードです。
- ・NFTサンプルコード利用例： NFTサンプルコードの利用例をノートブック形式で説明した文書です。

なお、コインについても、NFTと同様にインターフェース定義とサンプルコードがあります。

インタフェース定義

- nft-spec.html:NFTに関するインタフェース定義
- coin-spec.html:コイン(Fungible Token)に関するインタフェース定義

サンプルコード

- nft100.mjs:NFTコントラクトの最小限の実装を行ったサンプルコード
- nft200.mjs:上記に加え、mint/burnと転送履歴を実装したサンプルコード
- coin100.mjs:コインコントラクトの最小限の実装を行ったサンプルコード

利用例

- nft100-usage1.html:nft100の各機能の確認例
- nft100-usage2.html:nft100でコントラクトにapproveする例
- nft200-usage1.html:nft200の各機能の確認例
- coin100-usage1.html:coin100の各機能の確認例
- coin100-usage2.html:coin100でコントラクトにapproveする例

NFTインタフェース定義の見かた

HTML形式のファイルとして提供されます。ブラウザで開いて見ることができます。

BC+におけるNFTの標準的なインタフェースを定義しています。

章立てはたとえば下記のようになっています。

- ・用語の説明
- ・NFTコントラクトのプロパティ
- ・NFTコントラクトの動作
- ・NFTコントラクトの各機能の動作
 - ⇒ 1 5 の各機能の仕様
- ・onNFTReceivedの定義
- ・特記事項
- ・独自の機能拡張

DNCWARE Blockchain+におけるNFTインタフェース定義（

用語の説明

用語	説明
NFT	Non Fungible Tokenの略称です。 本格的には非代替性のトークンひとつひとつのことを指しますが、意味が抽象的に広がる場合もあります
トークン	ブロックチェーン上に記録されたデジタル情報です。デジタル資産としての価値を持つ場合があります。 代替性の有無によって種類が分けられ、非代替性の性質をもつトークンはNFTとよびます。
NFTコレクション	同じテーマやコンセプトに基づいた一連の非代替性トークンの集合です。 NFTコレクションに含まれるトークンは、その中でユニークなIDを持ちます。
NFTコントラクト	NFTコレクションを実装した、ブロックチェーン上のスマートコントラクトです。 NFTコントラクトをブロックチェーンにデプロイすることで、NFTコレクションが実装されます。

NFTコントラクトのプロパティ

NFTコントラクトのプロパティの設定は以下の通りとします。

プロパティ	設定値などの説明
name (オブジェクト名)	(任意)
description (説明)	(任意)
argtypes (引数型)	[func: "string", args: "any"]
mask (制限)	inaccessible 無効 (実行可)
	inaccessible as subcontract 無効 (サブコントラクトとして呼び出し可)
	inaccessible by delegation 無効 (delegationによる呼び出し可)
	nonrecording 無効 (トランザクションを記録する)
code	(インタフェース定義に準拠した任意のコード)
accessible_to (実行許可先リスト)	(NFTの流通範囲に合わせて設定)
disclosed_to (履歴開示先リスト)	(NFTの取引履歴等の開示範囲を設定)

なお、1 つのスマートコントラクトで 1 つの NFT コレクションを実装します。
つまり、NFT コレクションと NFT コントラクトの対応は 1 対 1 です。



NFTコントラクトの動作

NFTコントラクトの動作は以下の通りとします。

NFTコントラクトの呼び出しでは、funcに機能名を指定し、argsにその機能に関する引数を1
NFTコントラクトは、funcで指定された機能をargsで指定された引数に応じて実行し、戻り

NFTコントラクトの各機能の動作

NFTコントラクトの各機能の動作は以下の通りとします。

凡例：

func 機能名を表す文字列
args 機能に関する引数の形式
戻り値 NFTコントラクトが返す値

name

NFTコレクションの名称を返します。

func 'name'

～中略～

特記事項

- ・トークンのIDは英数字（上限100文字まで）で構成される文字列とします。
- ・トークンのIDはNFTコレクション内で一意であることとします。
- ・トークンのIDは人が識別しやすい文字列であることが望ましいです。
- ・トークンのオーナーは、ユーザまたはコントラクトのいずれかです。anonymous、ウォレット、システムコントラクト。
- ・NFTコントラクトは、管理者による変更不可・削除不可とすることが望ましいです。
- ・transferFrom/safeTransferFromの処理の一環としてdiscloseTo(from,to)を実行すること。

独自の機能拡張

以下に該当する場合、この文書で定められていない独自の機能をNFTコントラクトに実装することができます

- ・ NFTコントラクトの引数型に引数を追加しても良いです。
- ・ funcで指定される、NFTコントラクトの各機能を増やしても良いです。例えばトークンを生成するfunc
- ・ 戻り値がオブジェクト形の場合は、戻り値にプロパティを追加しても良いです。例えば、balanceOf()
- ・ 戻り値が「なし」の場合、どのような値を返しても良いです。
- ・ 各機能の引数に指定するIDとしてUnifiedName形式を受け付けるとも良いです。



NFTサンプルコードの見かた

NFTサンプルコードは、NFTコントラクト（スマートコントラクト）のコードのサンプルです。javascriptのモジュール形式のファイルとして提供され、任意のエディタ（メモ帳、VSCodeなど）で開いて見ることができます。

javascriptモジュールの形式(*.mjs)となっていますが、これは便宜上の形式であり、DNCWARE Blockchain+のスマートコントラクトとしてデプロイした場合のみ正しく動作するコードです。

なお、スマートコントラクトのコードとして使用するのには、exportしているfunctionのコード部分のみです。ファイルの最初の行と最後の行は使いません。（右図参照）

```
export default function nft100(func, args) {  
  
  'use strict';  
  var NFT_NAME = 'simple NFT implementation #100';  
  var NFT_SYMBOL = 'NFT100';  
  var TOTAL_SUPPLY = 100;  
  var COLLECTION_OWNER = 'u0000000000';  
  var BASE_URI = 'https://anywhere.com/';  
  
  var caller = getCallerId();  
  
  if (caller === 'anonymous') {  
    throw 'It cannot be accessed anonymously.';  
  }  
  
  if (!keyValueGet('initialized')) {  
  
    ~~~中略~~~  
  
  }  
  
  function requireInteger(i, message) {  
    require(typeof i === 'number' && isFinite(i) && Mat  
  }  
  
  function assert(condition, message) {  
    if (!condition) abortTransaction('ASSERT:' + (messa  
  }  
  
}
```

使わない

コード部分



スマートコントラクトの
コードとして使用する

使わない

NFTサンプルコード利用例の見かた

HTMLファイルとして提供され、ブラウザで開いて見ることができます。

右のようなノートブック形式となっており、説明つきでサンプルコードとその実行結果が示されています。

実行は、上から下へと進みます。

コードセルの中にコードそのものが示されています。コードセルの前にその説明、後ろにその実行結果が示されています。（右図参照）

NFTコントラクトの各インタフェースの動作確認

name

コードセル[9]の説明

NFTコレクションの名称を取得します。

コードセル[9] In [9]:

```
var resp = await rpc.call(users[5].wallet, nftid, {func: 'name'});
console.log('name:', resp);
```

```
name: {
  txno: 537825,
  txid: 'xnFVJsdBfhUvCq4tiUf8dLpFpjQDUHaJNwQDfp46ognq3',
  status: 'ok',
  value: 'sample NFT collection #123'
}
```

コードセル[9]の実行結果

symbol

コードセル[10]の説明

NFTコレクションの略号を取得します。

コードセル[10] In [10]:

```
var resp = await rpc.call(users[5].wallet, nftid, {func: 'symbol'});
console.log('symbol:', resp);
```

```
symbol: {
  txno: 537826,
  txid: 'xp2Y6TWMSg69hXFwXNZcztNyCRbNhZQC6L6ySjAXY2C8z',
  status: 'ok',
  value: 'SMP123'
}
```

コードセル[10]の実行結果

totalSupply

コードセル[11]の説明

NFTコレクションに含まれるトークンの総数を取得します。

コードセル[11] In [11]:

```
var resp = await rpc.call(users[5].wallet, nftid, {func: 'totalSupply'});
console.log('totalSupply:', resp);
```

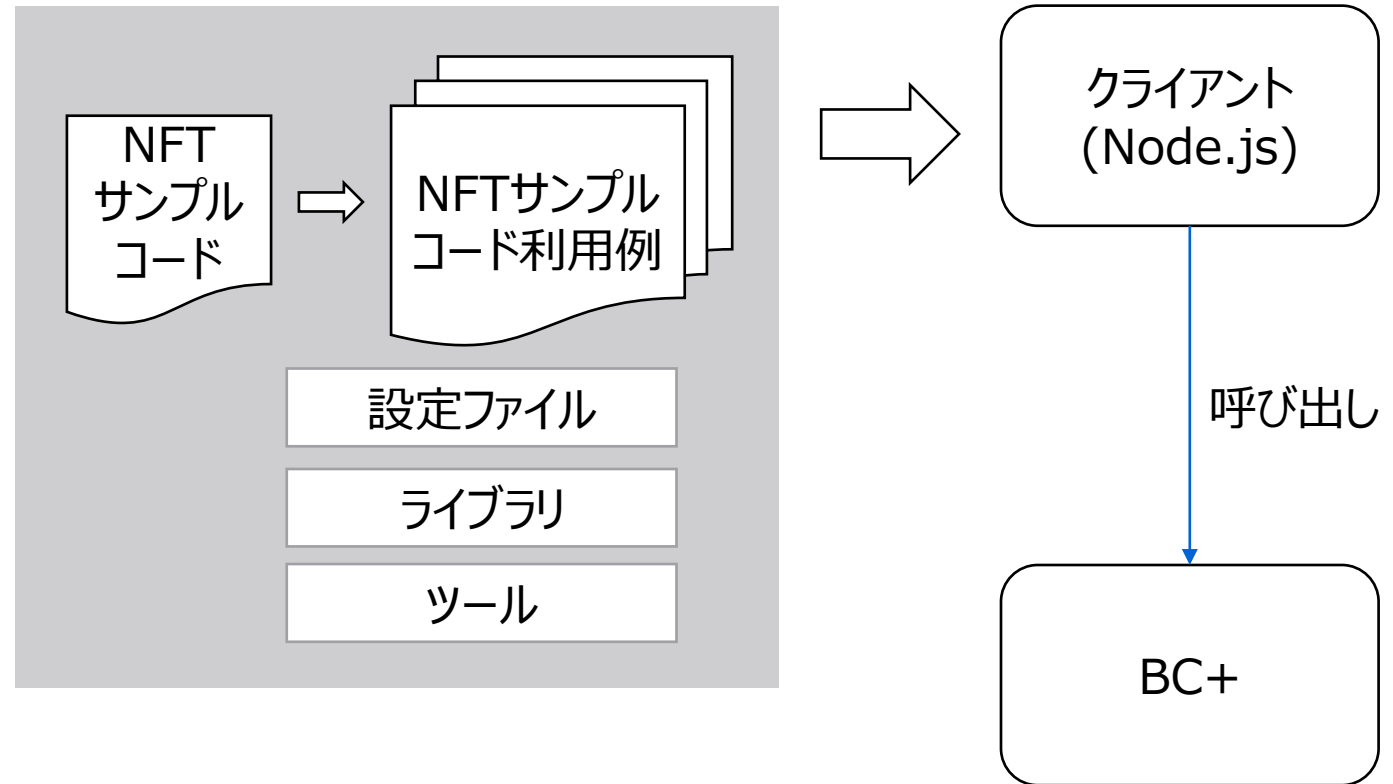
```
totalSupply: {
  txno: 537827,
  txid: 'xhG49hkUEj5uzKSdEsCr8D6nErcLaevsPW3SnpZ5JMC3LB',
  status: 'ok',
  value: 10
}
```

コードセル[11]の実行結果

NFTサンプルコード利用例の動作環境

NFTサンプルコード利用例は、他のサンプルコードと共通の動作環境を前提としています。その詳細については、サンプルコード全般の説明を参照してください。

本書では、これ以降、こちらの利用例を使った説明はしません。（プログラミングベースの方法であるため）
代わりに、本書では、より手軽に実施できる、管理画面のGUIを使った利用方法について説明していきます。



ブロックチェーンへの導入方法

ブロックチェーンへの導入方法 – 概要

ブロックチェーンへのNFTの導入方法の概要を説明します。
ここでは、自身で管理するドメインにNFTを導入する方法を説明します。
事前に、「NFTインタフェース定義」の「用語の説明」を理解しておいてください。

手順

- (1) 管理画面にドメイン管理者のウォレットでログインし、ブロックチェーンに接続する。
- (2) ドメイン内にNFT用のスマートコントラクトを作成する。
- (3) NFTコントラクトのプロパティを設定する。（アクセス権限を除く）
- (4) 「NFTサンプルコード」からコードをコピーし、カスタマイズする。
- (5) 動作の確認をする。
- (6) アクセス権限をNFTの流通範囲に合わせて設定する。
- (7) （必要に応じて）NFTコントラクトを書き換え不可に設定する。

以下、手順の(2)から順に説明していきます。

ドメイン内にNFT用のスマートコントラクトを作成する

c1createにより、NFTコントラクト用のスマートコントラクトを作成します。

The image shows a two-step process for creating an NFT contract. The first step is on the main interface where 'c1create' is searched for. The second step is a 'New Transaction' modal where specific details are entered.

Step 1: Search for c1create

The main interface shows a search bar with 'c1create' entered. Below it, the 'Smart Contract' section displays the 'c1create' contract details.

Configuration	
Name	
Description	system contract
Argtypes	type name description domain
Mask	
Accessible to	all
Disclosed to	
Domain	@default

Step 2: New Transaction Dialog

The 'New Transaction' dialog is shown with the following fields and values:

- Contract: c1create@default
- Parameter Name: type, name, description, domain
- Value: contract, NFT123, NFTサンプルコードの説明用, @NFTsample123
- Mode: write

Red arrows point from the Japanese labels to the corresponding fields in the dialog:

- contractを記入 (Enter contract)
- 名前を記入 (Enter name)
- 説明を記入 (Enter description)
- 作成先ドメインを記入 (Enter creation destination domain)

The 'Submit' button is circled in red. Below the dialog, a green box displays the resulting contract ID: "c011768653".

成功すると、作成したコントラクトのIDが表示される



NFTコントラクトのプロパティを設定する

作成したコントラクトの画面を開き、
「NFTインタフェース定義」の「NFTコントラクトのプロパティ」に合わせて設定します。（アクセス権限を除く）
ここで実際に設定変更が必要なのはArgtypesの項目だけです。

DNCWARE Blockchain+ V3 trial.dncware-blockchain.biz

WALLET: eS3UGjidwuxT9ew7oVan4wDsfEWmC4 Log Out

Smart Contract c011768653

Configuration

Name	Edit	NFT123
Description	Edit	NFTサンプルコードの説明用
Argtypes	Edit	
Code	Edit	View
Mask	Edit	
Max Steps	Edit	default
Accessible to	Edit	domainAdmin@NFTsample123
Disclosed to	Edit	
Domain		@NFTsample123

DNCWARE Blockchain+ V3

Edit Argtypes

NFT123@NFTsample123

Type	Parameter Name	
string	func	×
any	args	×

parameter nameAdd

SubmitCancel

Argtypes Edit func args

この通りに入力

「NFTサンプルコード」からコードをコピーし、カスタマイズする

NFTコントラクトのコードは独自に作成しても良いですが、
ここでは NFTサンプルコード(nft100.mjs)からコードをコピー & ペーストし、冒頭部分をカスタマイズします。
特にINITIAL_OWNERは必ず変更してください。

DNCWARE Blockchain+ V3 trial.dncware-blockchain.biz

WALLET: eS3UGjdWuxT9ew7oVan4wDsfEWmC4 Log Out

Smart Contract c011768653

Configuration

Name	Edit	NFT123
Description	Edit	NFTサンプルコードの説明用
Argtypes	Edit	func args
Code	Edit	
Mask	Edit	
Max Steps	Edit	default
Accessible to	Edit	domainAdmin@NFTsample123
Disclosed to	Edit	
Domain		@NFTsample123

DNCWARE Blockchain+ V3 Edit Code

xfU2Ws_nft100@NFTsample123

Argtypes func args

Code

```
1 'use strict';
2 var NFT_NAME = 'sample NFT collection #123';
3 var NFT_SYMBOL = 'SMP123';
4 var TOTAL_SUPPLY = 10;
5 var INITIAL_OWNER = 'u30398064';
6 var BASE_URI = 'https://anywhere.com/';
7
8 var caller = getCallerId();
9
10 if (caller === 'anonymous') {
11   throw 'It cannot be accessed anonymously.';
12 }
13
14 if (!keyValueSe
15   keyValueSe
16   for (var i
17     var to
18     keyVal
19     keyVal
20     keyVal
21     keyValueSe
22   }
23
24
25 switch (func) {
26   case 'name':
27     return NFT_NAME;
28 }
```

カスタマイズする

nft100.mjsから
本体部分をコピー

Submit Cancel

カスタマイズの詳細

NFTサンプルコード(nft100.mjs)の冒頭部分のカスタマイズの詳細について説明します。

```
var NFT_NAME = '*****';
```

NFTコレクションの名称を設定します。

```
var NFT_SYMBOL = '*****';
```

NFTコレクションの略称を設定します。

```
var TOTAL_SUPPLY = ***;
```

NFTコレクションに含まれるトークンの総数を設定します。（100以下の数を指定します）

```
var INITIAL_OWNER = '*****';
```

NFTコレクションの初期オーナーのIDを設定します。
例えば、ドメイン管理者（自分）のユーザIDに設定します。

```
var BASE_URI = '*****';
```

メタデータの場所を設定します。（URIが実際に存在するかをコード上ではチェックしません。）
BASE_URI+tokenIdの形でtokenURIが決まります。

動作の確認をする

例えば、コレクションオーナーのbalanceOfを確認します。

DNCWARE Blockchain+ V3 trial.dncware-blockchain.biz 🔍

WALLET: eS3UGjldwuxT9ew7oVan4wDsfEWmC4 Log Out

Smart Contract c011768653

Configuration

Name	Edit	NFT123
Description	Edit	NFTサンプルコードの説明用
Argtypes	Edit	func args
Code	Edit	View
Mask	Edit	
Max Steps	Edit	default
Accessible to	Edit	domainAdmin@NFTsample123
Disclosed to	Edit	
Domain		@NFTsample123

New Transaction

DNCWARE Blockchain+ V3 New Transaction

NFT123@NFTsample123

Parameter Name	Value
func	balanceOf
args	{ owner: "u73123870" }

Mode

write

Submit Cancel

成功すると、TOTAL_SUPPLYに設定した数が表示される

OK

アクセス権限をNFTの流通範囲に合わせて設定する

アクセス権限を設定します。これにより、NFTの流通が可能になります。

DNCWARE Blockchain+ V3

trial.dncware-blockchain.biz

WALLET: eS3UGjldwuxT9ew7oVan4wDsfEWmC4

Log Out

Smart Contract

c011768653

Configuration

Name	Edit	NFT123
Description	Edit	NFTサンプルコードの説明用
Argtypes	Edit	func args
Code	Edit	View
Mask	Edit	
Max Steps	Edit	default
Accessible to	Edit	DomainAdmin@NFTsample123
Disclosed to	Edit	
Domain		@NFTsample123

New Transaction

DNCWARE Blockchain+ V3

Edit "Accessible to"

NFT123@NFTsample123

Accessible to

☒ all

☐ following list

Submit

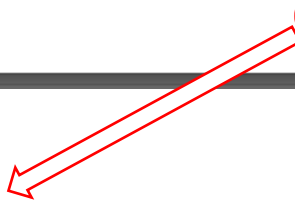
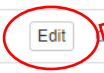
Cancel

Accessible to

Edit

all

流通範囲に制限を設けない場合には
allを設定する



NFTコントラクトを書き換え不可に設定する

もともとのイーサリアム上のNFTコントラクトは、デプロイ後のコードの書き換えが不可となっています。将来にわたって誰もコードを変更することができない性質は、非中央集権的な特徴の一つと考えられています。

DNCWARE Blockchain+でも、NFTコントラクトのコードの変更を不可に設定することができます。そして、ひとたび変更不可に設定すると、元に戻すことはできません。これにより、（必要に応じて）イーサリアムと同様の性質のNFTを作成できます。

変更不可に設定する方法について、ここでは2通りの方法を説明します。

（1）ドメインの管理権限(managed_by)を空欄にする。（管理者がいない状態にする）

コントラクトのプロパティの変更のためにはドメインの管理権が必要なため、管理権限をすべて削除すれば、当然コントラクトのコードは変更不可になります。また、管理権限の変更もその管理権が必要なため、だれも管理権を取得できなくなっています。そのため、将来にわたって、そのドメイン内のオブジェクトを変更できなくなります。

（2）c1freeze機能を使ってコントラクトを変更不可に設定する。（バージョン3.3から導入される機能）

管理権限とは関係なしに、オブジェクトごとにプロパティの変更を不可にする機能がc1freezeです。NFTコントラクトをc1freezeすることにより、NFTコントラクトのコードの変更を不可に設定することができます。この場合でも、将来にわたって、NFTコントラクトの全プロパティを変更できなくなります。

使用例

典型的には、NFTコントラクトはアプリケーションプログラムからAPIによって操作されます。それに近い形での説明は、別に「NFTサンプルコード利用例」で示しています。アプリ開発の実践としてはそちらを参照してください。

一方、ここではNFTコントラクトの理解を深める目的で、管理画面を使った利用例を簡単に説明します。

ownerOfの実施例

トークンのオーナーを問い合わせる例

token1のオーナーを問い合わせた結果、ユーザID:u73123870がオーナーであると返ってきました。

※なお、以降の実施例では、この管理画面にログインしているユーザが、トークンのオーナーであるu73123870と同一であることを暗黙のうちに仮定しています。

DNCWARE Blockchain+ V3

New Transaction

NFT123@NFTsample123

Parameter Name	Value
func	ownerOf
args	{ tokenId: "token1" }

Mode

write

Value

"u73123870"

OK

transferFromの実施例

トークンを転送する例

トークンの転送とは、すなわち、トークンの
オーナーを変更することです。

token1を、
ユーザID:u73123870から
ユーザID:u49489844へ
転送を要求したところ、成功しました。
(Valueの背景が緑は成功を意味します)

DNCWARE Blockchain+ V3

New Transaction

NFT123@NFTsample123

Parameter Name	Value
func	transferFrom
args	{ tokenId: "token1", from: "u73123870", to: "u49489844" }

Mode

write

Value

null

OK

ownerOfの実施例(転送後)

トークンの転送後、トークンのオーナーを問い合わせる例

token1のオーナーを問い合わせた結果、ユーザID:u49489844がオーナーであると返ってきました。

オーナーが転送先に変更されていることが分かります。

DNCWARE Blockchain+ V3

New Transaction

NFT123@NFTsample123

Parameter Name	Value
func	ownerOf
args	{ tokenId: "token1" }

Mode

write

Value

"u49489844"

OK

transferFromの実施例（転送後）

トークンの転送に失敗する例 1

（前回と同じ転送要求）
token1を、
ユーザID:u73123870から
ユーザID:u49489844へ
転送を要求したところ、失敗しました。
（Valueの背景が赤は失敗を意味します）

なぜなら、fromに指定したユーザと、トークンのオーナーが一致しないからです。

DNCWARE Blockchain+ V3

New Transaction

NFT123@NFTsample123

Parameter Name	Value
func	transferFrom
args	{ tokenId: "token1", from: "u73123870", to: "u49489844" }

Mode

write

Error

{ status: "thrown", value: "unexpected args.from" }

OK

transferFromの実施例（転送後）

トークンの転送に失敗する例 2

（前回と逆向きの転送要求）
token1を、
ユーザID:u49489844から
ユーザID:u73123870へ
転送を要求したところ、失敗しました。
（Valueの背景が赤は失敗を意味します）

なぜなら、管理画面にログインしたユーザ
（caller）と、トークンのオーナーが一致しない
からです。

※なお、ownerとcallerが一致しない場合でも、委任（approve等）があれば、転送できる場合もあります。詳細は、「インタフェース定義」のtransferFromの項を参照してください。

DNCWARE Blockchain+ V3

New Transaction

NFT123@NFTsample123

Parameter Name	Value
func	transferFrom
args	{ tokenId: "token1", from: "u49489844", to: "u73123870" }

Mode

write

Error

{ status: "thrown", value: "caller not authorized" }

OK

TOSHIBA