Abstract

This proposal is an extension of TZIP-016 and describes a metadata schema and standards for contracts and tokens.

The document is broken into two main sections: 1) The metadata schema, and 2) standards and recommendations for how to apply the schema to different token types.

Many of the terms in this standard are derived from The Dublin Core, RCF 2413.

Motivation & Goals

This metadata standard aims to:

1. Simplify the creation of rich metadata for tokens and assets
2. Provide a commonly understood interface
3. Confirm to existing and emerging standards
4. Allow global and international scope
5. Be extensible
6. Provide interoperability among ecosystem members (contracts, indexers, wallets, libraries, etc)

This standard also aims to rich enough to describe a wide variety of asset and token types, from fungible tokens to semi-fungible tokens to nonfungibles.

Table of Contents

Standards and Recommendations

All fields are defined and described in the Schema Definition section of the document.

It is strongly advised -- but not required -- that all tokens follow the following standards and recommendations.

Base Token Metadata Standard

The base token metadata standard extends the metadata standard previously defined in the FA2 (TZIP-012) and FA1.2 (TZIP-007) token standards, which defined the following fields:

- name
- symbol
- decimals

`decimals` is the only required field. However, the TZIP-021 Base Token Metadata Standard further emphasizes that either `name` or `symbol` should be present.

Example:

- Base example: JSON

Fungible Token Recommendations

In addition to the Base Token Metadata Standard, the following fields are recommended for all fungible tokens:

- symbolPreference
- thumbnailUri

Example:

- Example FA token (TZ21): JSON

Semi-fungible and NFT Token Recommendations

In addition to the Base Token Metadata Standard, the following fields are recommended for all nonfungible tokens (NFT) and semi-fungible tokens that act as NFTs:

- artifactUri
- displayUri
- thumbnailUri
- description
- minter
- creators
- isBooleanAmount

Multimedia NFT Token Recommendations

In addition to the Semi-fungible and NFT Token Recommendations, the following fields are recommended for all Multimedia NFTs:

- formats
- tags

Example:

- CryptoTaco Digital Collectible: JSON

Schema Definition

A JSON-Schema specification is provided as an annex to this document.

The schema may be provided at the contract metadata level and/or at the token metadata level.

If provided at the token metadata level, content should be provided in accordance with the recommendations of the token standard used (see TZIP-12, TZIP-7).

If provided at the contract metadata level, to prevent pollution in the top level of the metadata, content should be provided as a nested object under the key `assets`.

The schema defines the following additional types:

`asset` (object)

Properties of the `asset` object are designed to live at the root level of the token metadata, or as an object array under the key `assets`. at the contract metadata level.

`description` (string)

General notes, abstracts, or summaries about the contents of an asset.

`minter` (string) ***[format: tzaddress]***

The tz address responsible for minting the asset.

`creators` (array)

The primary person, people, or organization(s) responsible for creating the intellectual content of the asset.

The field is an array with all elements of the type `string`. Each of the elements in the array must be unique.

`contributors` (array)

The person, people, or organization(s) that have made substantial creative contributions to the asset.

The field is an array with all elements of the type `string`. Each of the elements in the array must be unique.

`publishers` (array)

The person, people, or organization(s) primarily responsible for distributing or making the asset available to others in its present form.

The field is an array with all elements of the type `string`. Each of the elements in the array must be unique.

`date` (string) ***[format: date-time]***

A date associated with the creation or availability of the asset as defined in the JSON Schema Specification.

`blockLevel` (integer)

Chain block level associated with the creation or availability of the asset.

`type` (string)

A broad definition of the type of content of the asset.

`tags` **(array)**

A list of tags that describe the subject or content of the asset.

The field is an array with all elements of the type `string`. Each of the elements in the array must be unique.

`genres` **(array)**

A list of genres that describe the subject or content of the asset.

The field is an array with all elements of the type `string`. Each of the elements in the array must be unique.

`language` **(string)** ***[format: RFC 1776]***

The language of the intellectual content of the asset as defined in RFC 1776.

`identifier` **(string)**

A string or number used to uniquely identify the asset. Ex. URL, URN, UUID, ISBN, etc.

`rights` **(string)**

A statement about the asset rights.

`rightUri` **(string)** ***[format: uri-reference]***

A URI (as defined in the JSON Schema Specification) to a statement of rights.

`artifactUri` **(string)** ***[format: uri-reference]***

A URI (as defined in the JSON Schema Specification) to the asset.

`displayUri` **(string)** ***[format: uri-reference]***

A URI (as defined in the JSON Schema Specification) to an image of the asset.

Used for display purposes.

`thumbnailUri` **(string)** ***[format: uri-reference]***

A URI (as defined in the JSON Schema Specification) to an image of the asset for wallets and client applications to have a scaled down image to present to end-users.

Recommend maximum size of 350x350px.

`externalUri` **(string)** ***[format: uri-reference]***

A URI (as defined in the JSON Schema Specification) with additional information about the subject or content of the asset.

`isTransferable` **(boolean)** ***[default: true]***

All tokens will be transferable by default to allow end-users to send them to other end-users. However, this field exists to serve in special cases where owners will not be able to transfer the token.

`isBooleanAmount` **(boolean)** ***[default: false]***

Describes whether an account can have an amount of exactly 0 or 1. (The purpose of this field is for wallets to determine whether or not to display balance information and an amount field when transferring.)

`shouldPreferSymbol` **(boolean)** ***[default: false]***

Allows wallets to decide whether or not a symbol should be displayed in place of a name.

`formats` **(array)**

The object is an array with all elements of the type `format`.

`attributes` **(array)**

Custom attributes about the subject or content of the asset.

The object is an array with all elements of the type `attribute`.

Example:

```
"attributes": [
  {
    "name": "Base",
    "value": "Starfish"
  },
  {
    "name": "Eyes",
    "value": "Big"
  },
  {
    "name": "Mouth",
    "value": "Surprised"
  },
  {
    "name": "Level",
    "value": "5",
    "type": "integer"
  },
  {
    "name": "Stamina",
    "value": "1.4",
    "type": "number"
  },
  {
    "trait_type": "Stamina Increase",
```

```
      "value": "10",
      "type": "percentage"
    }
  ]
```

`assets` **(array)**

Facilitates the description of collections and other types of resources that contain multiple assets.

The object is an array with all elements of the type `asset`.

`format` (object)

Properties of the `format` object:

`uri` **(string)** ***[format: uri-reference]***

A URI (as defined in the JSON Schema Specification) to the asset represented in this format.

`hash` **(string)**

A checksum hash of the content of the asset in this format.

`mimeType` **(string)**

Media (MIME) type of the format.

See IANA Media Types

`fileSize` **(integer)**

Size in bytes of the content of the asset in this format.

`fileName` **(string)**

Filename for the asset in this format. For display purposes.

`duration` **(time)**

Time duration of the content of the asset in this format.

`dimensions` **(dimensions)**

Dimensions of the content of the asset in this format.

`dataRate` **(dataRate)**

Data rate which the content of the asset in this format was captured at.

Example

```
{
  "uri": "ipfs://...",
  "hash": "e9ed141df1cebfc89e466ce089eedd4f125aa7571501a0af871fab60597117b7",
  "mimeType": "audio/mpeg",
  "fileSize": 7134739,
  "fileName": "Track-1.mp3",
  "duration": "00:56:46",
  "dataRate": {
    "value": 320,
    "unit": "kbps"
  }
}
```

`attribute` (object)

Properties of the `attribute` object:

`name` (string, required)

Name of the attribute.

`value` (string, required)

Value of the attribute.

`type` (string)

Type of the value. To be used for display purposes.

Example:

```
{
  "name": "Stamina",
  "value": "1.4",
  "type": "number"
}
```

`dataRate` (object)

Properties of the `dataRate` object:

`value` (integer, required)

`unit` **(string, required)**

Example:

```
"dataRate": {
  "value": 192,
  "unit": "kbps"
}
```

`dimensions` (object)

Properties of the `dimensions` object:

`value` **(string, required)**

`unit` **(string, required)**

Example:

```
"dimensions": {
  "value": "512x512",
  "unit": "px"
}
```

Implementations

Copyright