Guidelines

NFT data formats and storage

NyzoSy and Iyomisc

nyzo

Version 1.0

Introduction

NFT, data formats and storage

- Nyzo data field is small
- Specific tactics have to be applied not to spam the chain or complexify handling of data.

Guidelines only

- Nothing in this document is – nor will be – enforced at protocol level.
- Data formats and interpretation remains free – as in speech – and are application specific
- Having a common ground however avoid re-inventing the wheel and make inter-applications operations a reality.
- This document suggests a scaffold and practical examples for common uses.
- New specs can be added by anyone

Base document

- This document supposes prior knowledge of the Nytro protocol.
  Please refer to the main nytro_protocol.pdf and nft_protocol.pdf documents for basic concepts of the protocols.

Various data fields

The NFT protocol makes it possible to store data in several places:

Recipient as data

- With “NA:” and ”ND:” operations, the recipient address is handled as 32 bytes of data.

Comment as data

- With several operations, including “NA:” and ”ND:”, a small comment field is also available.

Generic rule

- No atomic data field larger than 32 bytes is to be stored on Nyzo chain.
- Large data is to be stored elsewhere - like IPFS – and only its pointer stored on-chain

Use of “comment” field

With “NA:” and ”ND:” operations, the recipient address is handled as 32 bytes of data and a comment is available:
The whole operation (nyzo native data or “memo” field) looks like
ND:nCLASS_NAME:UID:Comment

nCLASS_NAME is 11 chars max
UID is 8 chars max
With ND and “:”, this means there are at least 8 chars left for the optional “comment”

Comment as key

- NFT Data storage can be seen as a key-value store
- Comment is the key
- Recipient – as data – is the associated data
- If both user and admin define the same key, the application has to decide what that means (usually, admin value takes precedence)
- If both a class and an instance define the same key, the application has to decide what that means (usually, class value is meant as default value, and instance value takes precedence)
- If the same key is defined several times, then the latest usually is meant to replace the previous one.
- **This way, we have a layered key-value store with default value, access level, and full history.**

Default format “comment” field

We advised to use “comment” as key.

Leaving comment as empty is alike an empty default key.

The recommended meaning for this empty key is for the “Format” reference.
In an effort to normalize the keys to be used and allow for possible cross-operations between applications, ease up the job of explorers, marketplaces and such, we will provide a few scaffold for usual NFT items.

These will specify the minimal keys to provide for compliance.

You can then pick the closest format to your need, fill in the required keys and add the extra keys and data you need. These extra keys will not be visible on explorers for instance, but your NFT will.

To specify the format you use, we recommend to use the empty key and provide the format name as value.

The value should be one of the pre-defined formats we describe later on and its version.

More formats can be contributed by the community

NFT Instance ID is data!

In addition, the NFT instance id can be considered as data as well.

Say you are an artist issuing a limited series art.

Instead of using “DA” operation to store the NFT name and Number, you can directly embed them into the ID.

For instance:

- Issue a “MYART” class with 10 supply, and mint “MYART:01” to “MYART:10”
- Issue a “MYCOL” with infinite supply, and mint ”MYCOL:A1_1_10” to ”MYCOL:A1_10_10” then ”MYCOL:A2_1_50” to ”MYCOL:A2_50_50” for 2 separate arts, A1 of 10 items and A2 of 50.

  You could then use class data for format definition, and admin data to point to the ipfs storing the actual art (see further on).

IPFS Data storage and pointer

IPFS CID are “multihash” that encode more than the hash itself.
With CID Version 0, hash function and hash length are also encoded.
Default hash for IPFS is SHA2-256.
Since Nyzo can only store up to 32 bytes (256 bits), hash function and length are omitted in the value field, and default SHA2-256 is supposed.

We recommend using the comment field as key to properly define the binary hash alone, and allow for rebuilding of the IPFS CID.

The key for this is `_IPFS`

When using another hash than default, use the optional `IPFS_H` key, with Hash and hash length encoded in value under a binary format, as in the binary CID.

Since Nyzo can not store more than 32 bytes (256 bits) of data, max hash len is 256, but other hash functions like blake2 could be used later on.

Defining them in the optional `IPFS_H` key allows for easy rebuilding of IPFS CID from hash and key, and the other way around.

See https://cid.ipfs.io/ for decoding of IPFS CIDs.

We recommend to only use the default SHA2-256 hash unless required for a specific reason.

Json or several keys?

When using IPFS as storage, it’s usual to point to a Json file that provides a structured storage of various metadata, including another IPFS CID for raw data.

You can also choose to only use IPFS for raw data, and send several “data” operations for each of your keys.

Json

1 single data with IPFS hash
Immutable or IPFS hash changes
Allows for long values
Allows for long keys

Several “DA” operations

1 tx per field
Can change a single value with no IPFS op
Limited to 32 bytes max values
Limited to 8 chars keys

NFTs Formats

NyzoSy and Iyomisc

Version 1.0

Reminder

Guidelines only

- Nothing in this document is – nor will be – enforced at protocol level.
- Data formats and interpretation remains free – as in speech – and are application specific
- Having a common ground however avoid re-inventing the wheel and make inter-applications operations a reality.
- This document suggests a scaffold and practical examples for common uses.
- New specs can be added by anyone

WIP Reference

- We will try to document and add the most common structures first, but this is a work in progress.
- Fee free to contribute
- Remind you can extend an existing format by adding application specific keys to it, while not having to define a specific base format.

NFT Format: Media

Value for empty key

- Value is `Media:1` , 1 denoting the format version.
- This translates as `4d656469613a3100000000000000000000000000000000000000000000000000` or `id__84TCq6CyeA4000000000000000000000000000000whADs9PQ` recipient once encoded.

Exemple data payload

- `NA:nCLASS_NAME:UID`
- Recipient `id__84TCq6CyeA4000000000000000000000000000000whADs9PQ`

Usage

- NFT where a graphic, audio or video item is the main data
- Limited edition artwork
- Image, animated image, meme
- Song, Loop, Music, Jingle

- Base for in-game items, playing cards, collectibles, proof of ownership, event tickets...

This format is likely generic enough it can be used for almost everything we can think of.

NFT Format: Media

Required keys

These keys have to exist if you use this format

- _FORMAT: “Media:1” - For IPFS stored json only. Has to match the registered format, this is a safety check.
- AUTHOR: “” - String, the author name or safer, his address. Keep it short since it could be trimmed at display.
- NAME: “” - String, name of the instance or class, ex: art name
- THUMB: “” - String, an IPFS CID or hash pointing to an image thumbnail, 256x256 pixels.
- TYPE: One of: “IMAGE”, “AUDIO”, “VIDEO”, “DOCUMENT”
- EXT: “PNG” - String, extension of the media file
- DATA: “” - String, an IPFS CID or hash pointing to the media file
- SIZE: “NNNNxNNNN” - the size of the image or video in pixels. Empty or undefined for AUDIO and DOCUMENT type.
- DURATION: 60 – Integer, duration of the media in seconds. 0 or undefined for IMAGE and DOCUMENT type.

Optional keys

- DESC: “” - String, an optional description of the NFT
- COLL:”” - String, the collection this NFT belongs to.
- NUM : 1 – Int, id of that instance
- OF_TOTAL: 10 – Int, total number of this NFT
- IPFS_H: 0xnnnn - hash function and length for IPFS hashes, when using “NA:” or “ND:” operations. Same binary prefix as in the CID. Leave this key undefined for default and recommended SHA2-256.

NFT Format: Media - Example 1

Json on IPFS NFT Art

Check this NFT: nDEMO:A1_02_10

- One mint transaction
- One empty key admin data transaction, for the whole NFT class, stating “Media:1” format
- One `_IPFS` key for the instance, with default SHA2_256 hash:
  “CB6551DA5B4C1655AAB2FFD225F03586FDA79B8E86C7D1344B6C9D5DFDE82B43”

- Using the IPFS CID to NFT data Helper at https://tokens.nyzo.today/helpers/CID , this points us to the following IPFS CID: `Qmc2d5awuDbbmD73jt5U25LtsJYqUPdj6mrmXwaygemjFY`
  You can get its content from Pinata gateway for instance:
  https://gateway.pinata.cloud/ipfs/Qmc2d5awuDbbmD73jt5U25LtsJYqUPdj6mrmXwaygemjFY
- This shows the following Json payload:


```
{ “_FORMAT”:      "Media:1",
“AUTHOR”:   "id__85mgCI12L3WshWfHHM8_u0Qjx6WZ9ha0qBpRPV3GGgYy5ZWY2kzb",
“NAME”:     "nDEMO A1 NFT",
“THUMB”:    "QmNiJKEHbsyqJeCmUcnoxT4xXVdVKTcTVWVDbSYSgy9mnq",
“TYPE”:     "IMAGE",
“EXT”:      "PNG",
“DATA”:     "QmcEGf1xQ62p4TYqaeUdRPFKWsJjXKQzrNvCsPBn2rvdwg",
“SIZE”:     "460x460",
“DESC”:     "Second Picture NFT on Nyzo!",
“COLL”:     "Demo",
“NUM”:      2,
“OF_TOTAL”:10 }
```


- Fields are explicit, and we can again use the pinata gateway to see the “Thumb” from the IPFS CID: https://gateway.pinata.cloud/ipfs/QmNiJKEHbsyqJeCmUcnoxT4xXVdVKTcTVWVDbSYSgy9mnq

NFTs Creation Example Workflow

NyzoSy and Iyomisc

nyzo

Version 1.0

NFT Minting - Ex. Workflow 1/3

This is aimed at dapps and services minting Media NFTs.
Following this workflow should ensure a proper listing on future NFT graphic explorers and marketplaces.
Further tools will likely make it simpler.

Pre-requisites:
- Have a NFT class you own, to mint Instances from.
- Have the picture file to embed.
- Have some IPFS client and/or pinning service (ipfs-desktop and Pinata for instance, see resources later on in this document)

Workflow (1/3):
- “Upload” the picture file to IPFS, copy its CID (This will be “DATA”)
- Shrink the image to a 256x256 thumbnail, upload it to IPFS, copy its CID (This will be “THUMB”)
- Create a “your_nft_name.json” empty file, and paste the following template inside:

```
{
 “_FORMAT”:"Media:1",
 “AUTHOR”:  "id__",
 “NAME”:    "",
 “THUMB”:   "Qxxx",
 “TYPE”:    "IMAGE",
 “EXT”:     "PNG",
 “DATA”:    "Qxxx",
 “SIZE”:    "0x0",

 “DESC”:    "",
 “COLL”:    "",
 “NUM”:     1,
 “OF_TOTAL”:10
 }
```

NFT Minting - Ex. Workflow 2/3

Workflow (2/3)

- Edit “your_nft_name.json” empty file, by filling in the proper values:
  - Leave _FORMAT unchanged
  - AUTHOR should be the author Nyzo address (id__)
  - NAME is your NFT instance name
  - THUMB is the IPFS CID of the thumb (256x256 image). You can use “Qm” as well as “bafy” CIDs.
  - EXT is the filename extension of the data
  - DATA is the IPFS CID of the image itself (full size image). You can use “Qm” as well as “bafy” CIDs.
  - SIZE is the size – in pixels – of the full size image, width x height

  - DESC is optional
  - COLL is optional
    NUM and OF_TOTAL are optional as well.

- Save the Json file, make sure it's valid Json. Beware of quotes, commas aso.
- You can check by pasting the Json content on https://jsoneditoronline.org/

- “Upload” the Json to IPFS, copy its CID
- Use the “IPFS CID to NFT Data” helper at https://tokens.nyzo.today/helpers/CID to get the data that encodes this CID (“NFT IPFS Data”)

NFT Minting - Ex. Workflow 3/3

Workflow (3/3)

- Mint the NFT instance to the final recipient. You can not mint to yourself (This is a Nyzo layer-1 limit). If you want to mint to yourself and deliver later on, use a second address of yours.
  Use the “NFT Instance Mint” Helper at https://tokens.nyzo.today/helpers/nft_mint

- Set the format of the NFT class or NFT instance to “Media:1”.
  This is done with the NFT Admin data helper, empty comment and “Media:1” as NFT Admin data at https://tokens.nyzo.today/helpers/nft_admin_data .
  We advise to do this once for your class (just NFT class then, no instance name), and you don't have to do it for every instance you mint thereafter.

- Set the IPFS hash of your instance.
  This is done by sending another admin data transaction, with “_IPFS” as comment and the “NFT IPFS Data” you converted from the previous step.

- You're done!

Hopefully, integrated tools and services will make this faster and error proof, but there is nothing complicated once you did it once.

IPFS ressources

NyzoSy and Iyomisc

nyzo

NFT Format: Media - Example 1

Since IPFS is likely to be widely used with NFTs, here are a few useful resources:

- An IPFs Primer
  https://blog.infura.io/an-introduction-to-ipfs/
- IPFS Doc
  https://docs.ipfs.io
- A Closer look at IPFS
  https://medium.com/mvp-workshop/a-closer-look-to-the-inter-planetary-file-system-b3f3af31a3c7

- IPFS-Desktop – Easy to use desktop client, Windows Mac and Linux
  https://github.com/ipfs/ipfs-desktop
- Pinata: Free account for less than 1GB total pinned files
  https://pinata.cloud

- What's in a CID?
  https://codeclimbing.com/whats-in-a-cid-multi-multi-multi/
- Encode and decode IPFS CID:
  https://cid.ipfs.io/