Assets and Metadata Structuring¶

AssetMantle provides NFT creators the option of off-chain metadata storage.

Creators are recommended to use a decentralized data storage solution like InterPlanetary File System (IPFS).

This documentation will go over how creators can structure their NFT projects and how they can use storage solutions like NFT.Storage, Web3.Storage, and Pinata to upload to IPFS.

Storage Mechanisms¶

NFT.Storage¶

NFT.Storage is a service that lets you upload off-chain NFT data for free, with the goal to store all non-fungible tokens as a public good. The user's uploaded content will be permanently stored in Filecoin's decentralized storage network and made available over IPFS via its unique content ID so it can easily persist across different versions of THE website or mobile application (if any).

Creators are able to upload a virtually unlimited amount of NFT data. However, there currently exists a limit of 31 GB per individual upload.

Web3.Storage¶

Web3.Storage provides a simple way for developers to integrate Filecoin's decentralized storage system into their apps and websites without having any technical knowledge of how it all works.

Just upload the content once using the platform's client libraries, then pin it on different nodes around the world! The data is protected by 3 redundant copies that can be cryptographically verified.

Pinata Upload¶

Coming soon.

How to Structure the Collection?¶

Ensure that every asset has a corresponding.json file as shown below:

Note: To link the respective assets (images/videos) with respective metadata, it is very important to have the folder structure in the following format for assets' representation and minting. Project Folder: - images - 1.jpg - 2.jpg - 3.jpg - metadata - 1.json - 2.json - 3.json type - String, bool, number, URI, height; NOTE: All files are required to be numbered sequentially and the metadata file names must also match these. NOTE: If using NFT.storage, remove the .json extension from metadata files or switch to the branch nft.storage NOTE: Images can be JPG, PNG, SVG, GIF. We are adding support for other formats soon.

How to Structure the Metadata?¶

// 1.json { "properties": [{ "name": "Block", "type": "String", "value": "Flash" }, { "name": "Eyes", "type": "String", "value": "black" }, { "name": "edition", "type": "number", "value": 1 }, { "name":"rare", "type": "bool", "value": true },], "description": "This is the sample metadata file", "image": "ipfs://baadasdalasdas/images/1.png", "name": "Sample Metadata" }

Reserved Properties¶

The following properties are reserved at the protocol level so creators are requested to avoid using the following properties as the traits to avoid conflict of interest.

Authentication; Burn; Expiry; ExchangeRate; Lock; MaintainedProperties; MakerOwnableSplit; NubID; Permissions; Supply; TakerID;

NFT Collection Upload¶

Once the assets and metadata files are ready, the next step is to upload the same at some storage mechanism. It is recommended to use a decentralized storage solution such as IPFS.

This documentation highlights four pathways of uploading NFTs and their metadata to IPFS; via NFT.Storage (with UI and script) and Web3.Storage (with UI and script).

For NFT collections with a large number of assets, using a script is ideal.

Previous Contact AssetMantle Next Using NFT.Storage (without Script)