Digital Value

Introduction

Digital Value propose a solution for a no-code platform to help to create Smart Contract from Templates. The template can build Smart Contracts to solve common problems normally found in the business. It's not a solution for all business case, just for the simple scenario found for several companies. The Template engine are muti-chain and can generate code for several blockchain.

Templates are used to solve common problem that exists in many organizations and using similar solutions. In this case we can create a template that get some data and quickly generate the code. This approach is not adequate for all use cases and intent to solve common use cases and reduce the time to market for a new project. Using this approach we can solve in hours a project that takes week on the traditional workflow.

Blockchain

The Digital Value templates can work with follow networks

- Cardano
- Ethereum
- Solana
- Polkadot
- Chainlink
- Veras

Components

- Template Compiler
- Template Engine
- Template Repository
- Contract Frontend
- Smart Contract Generator
- Contract Tester
- Contract Deploy

Template Flow

To create a new Smart contract the user should follow these steps:

1. Select a pre-existing Contract Template

2. Configure the variables for the Contract
3. Generate the Smart Contract
4. Validate the Contract at Test Network
5. Publish the Smart Contract

To execute the Contract, we need to create a form template to call the Contract following these steps:

- Create a form
- Publish Form
- Execute Form
- Call Smart Contract

Use Cases

1. NFT

A **non-fungible token** (**NFT**) is a non-interchangeable unit of data stored on a blockchain, a form of digital ledger. Types of NFT data units may be associated with digital files such as photos, videos, and audio. Because each token is uniquely identifiable, NFTs differ from blockchain cryptocurrencies, such as Bitcoin. NFT ledgers claim to provide a public certificate of authenticity or proof of ownership, but the legal rights conveyed by an NFT can be uncertain. NFTs do not restrict the sharing or copying of the underlying digital files and do not prevent the creation of NFTs with identical associated files. Specific token standards have been created to support various blockchain use-cases. Ethereum was the first blockchain to support NFTs with its ERC-721 standard and is currently the most widely used. Many other blockchains have added or plan to add support for NFTs with their growing popularity. ERC-721 provides core methods that allow tracking the owner of a unique identifier, as well as a permissioned way for the owner to transfer the asset to others. The ERC-1155 standard offers "semi-fungibility", as well as providing an analogue to ERC-721 functionality (meaning that an ERC-721 asset could be built using ERC-1155). Unlike ERC-721 where a unique ID represents a single asset, the unique ID of an ERC-1155 token represent a class of assets, and there is an additional quantity field to represent the amount of the class that a particular wallet has. The assets under the same class are interchangeable, and the user can transfer any amount of assets to others. Because Ethereum currently has high transaction fees (known as gas fees), layer 2 solutions for Ethereum have emerged which also supports NFTs. Cardano introduced native tokens that enable the creation of NFTs without smart contracts with its March 2021 update. Cardano NFT marketplaces include CNFT and Theos. The Solana blockchain also supports non-fungible tokens. Digital Value use the following standards to generate NFT:

- Ethereum – OpenZeppelin
- Cardano

Solana - With Metaplex we do not need to write our own contract. Metaplex has already deployed its own standard NFT contracts that any developer can interact with and build their own NFT collections on. It's like a smart-contract-as-a-service. Solana allows parallels transaction which makes it faster, but this makes the code more complex, so tools like Metaplex are extremally useful.

1. Finance
2. Gaming
3. Randomness
4. Insurance
5. Enterprise Systems
6. Supply Chain
7. Utilities
8. Authorization and Identity
9. Government
10. Sustainability
11. Off-Chian Computing

Smart Contracts Standard

OpenZeppelin ERC-721

On Ethereum, to create an NFT what we'd do is create our own custom OpenZeppelin ERC-721

Solana Metaplex

Metaplex (https://www.metaplex.com/) is the fastest and cheapest NFT ecosystem for martketplaces, games, art & collectibles. You can own your own NFT storefront and pay only for network and storage costs with no middleman. The cost to using this solution in January 13th, 2022

Task	SOL	Us$
Minting	0.01	1.47
Auction	0.03	4.40
Total	5.87	0.04

Megaplex's non-fungible-token standard is a part of the Solana Program Library (SPL) and can be characterized as a unique token with a fixed supply of 1 and 0 decimals. We extended the basic definition of an NFT on Solana to include additional metadata such as URI as defined in ERC-721 on Ethereum.

Below are the types of NFTs that can be created using the Metaplex protocol.

- **Master Edition** - A master edition token, when minted, represents both a non-fungible token on Solana and metadata that allows creators to control the provenance of prints created from the master edition. Rights to create prints are tokenized itself, and the owner of the master edition can distribute tokens that allow users to create prints from master editions. Additionally, the creator can set the max supply of the master edition just like a regular mint on Solana, with the main difference being that each print is a numbered edition created from it. A notable and desirable effect of master editions is that as prints are sold, the artwork will still remain visible in the artist's wallet as a master edition, while the prints appear in the purchaser's wallets.
- **Print** - A print represents a copy of an NFT, and is created from a Master Edition. Each print has an edition number associated with it. Usually, prints are created as a part of an auction that has happened on Metaplex, but they could also be created by the creator manually. For limited auctions, each print number is awarded based on the bid placement. Prints can be created during an Open Edition or Limited Edition auction.
- **Normal NFT -** A normal NFT (like a Master Edition) when minted represents a non-fungible token on Solana and metadata but lacks rights to print. An example of a normal NFT would be an artwork that is a one-of-a-kind that, once sold, is no longer within the artist's own wallet, but is in the purchaser's wallet.

Metaplex currently supports four types of auctions that are all derived from English auctions. Basic parameters include:

- Auction start time
- Auction end time
- Reservation price

Additionally, Metaplex includes a novel concept of the participation NFT. Each bidding participant can be rewarded a unique NFT for participating in the auction. The creator of an auction also can configure a minimal price that should be charged for redemption, with the option to set it as "free".

- **Single Item** - This type of auction can be used to sell normal NFTs and re-sell Prints, as well as the sale of Master Edition themselves (and the associated printing rights) if the artist so wishes. While this last behavior is not exposed in the current UI, it does exist in the protocol.
- **Open Edition** - An open edition auction requires the offering of a Master Edition NFT that specifically has no set supply. The auction will only create Prints of this item for bidders: each bidder is guaranteed to get a print, as there are no true "winners" of this auction type. An open edition auction can either have a set fixed price (equivalent to a Buy Now sale), can be set to the bid price (Pay what you want), or can be free (Make any bid to get it for free).
- **Limited Edition -** For a limited-edition auction, a Master Edition NFT (of limited or unlimited supply) may be provided to the auction with several copies as the set number of winning places. For each prize place, a Print will be minted in order of prize place and awarded to the winning bidder of that place. For example, the first-place winner will win Print #1; the second-place winner

Print #2; and so on. It is required for limited supply NFTs that there is at least as much supply remaining as there are desired winners in the auction.

- **Tiered Auction -** A tiered auction can contain a mix of the other three auction types as winning placements. For instance, the first-place winner could win a Print of Limited Edition NFT A, while the second-place winner could win a Normal NFT, etc. Additionally, all participants who did not win any place could get a Participation NFT Print from a Master Edition (if the Master Edition had no supply limit).

Metaplex can seamlessly create on-chain artist splits that remove the awkwardness out of collaboration. Tag each collaborator, set custom percentages, and you're off to the races. Each NFT can also be minted with configurable royalty payments that are then sent automatically back to the original creators whenever an artwork is resold on a Metaplex marketplace in the future.

Metaplex's off-chain component allows creators to launch a custom storefront, similar to Shopify or WordPress. This open-source project provides a graphical interface to the on-chain Metaplex program, for creators, buyers, and curators of NFTs. The design and layout of storefronts can be customized to suit the needs of the entity creating it, either as a permanent storefront or an auction hub for a specific auction or collection.

All identification on the Storefront is based on wallet addresses. Creators and store admins sign through their wallets, and users place bids from connected wallets. Custom storefronts allow creators to create unique experiences per auction. Additionally, the Metaplex Foundation is working on multiple partnerships that will enable building immersive storefronts using VR/AR.

A collection is an NFT. It has the same data layout on-chain as any other NFT. An NFT is linked to a collection in a belongs_to style where the NFT has a reference back to the collection. This is implemented through the addition of a new collection field in the Token Metadata struct.

```
pub struct Metadata {
    pub key: Key,
    pub update_authority: Pubkey,
    pub mint: Pubkey,
    pub data: Data,
    // Immutable, once flipped, all sales of this metadata are considered secondary.
    pub primary_sale_happened: bool,
    // Whether or not the data struct is mutable, default is not
    pub is_mutable: bool,
    /// nonce for easy calculation of editions, if present
    pub edition_nonce: Option<u8>,
    /// Since we cannot easily change Metadata, we add the new DataV2 fields here at the end.
    /// Collection
    pub collection: Option<Collection>,
    ...
}

#[derive(BorshSerialize, BorshDeserialize, PartialEq, Debug, Clone)]
pub struct Collection {
  pub verified: bool, // Whether or not the collection is verified
  pub key: Pubkey,    // The SPL token mint account of the collection NFT
}
```

As token usage has evolved on Solana, it has become clear that there are more types of tokens than simply "fungible" and "non-fungible" tokens. Token Standards are defined by a Rust enum:

```
pub enum TokenStandard {
    NonFungible,    // This is a master edition
    FungibleAsset, // A token with metadata that can also have attributes, sometimes called Semi Fungible
    Fungible,      // A token with simple metadata
    NonFungibleEdition, // This is a limited edition
}
```

Basic Single Item Auction End To End

Now that we've gone over the contracts, let's run through an example of how the contracts interact to create an NFT and sell it. I personally find these stories the most informative way to learn a new ecosystem.

1. Minting an NFT

On the Solana network an NFT is represented as a Token with only 1 in circulation and further minting is disabled, but that's not very useful. A token contains very little data about itself. In fact it doesn't even know its own name. Solana tokens are a primitive construct that we build on top of. That's where Metaplex comes in. As we mentioned on our Terminology Page Metaplex is the standard way to add metadata to tokens. This metadata allows the tokens to secure Images, Audio, Video and anything else you can dream up. In order to create an NFT using metaplex we will follow these steps:

1. Optional: Pay for the Upload
2. Creating a Token Mint
3. Optional: Creating A Token Associated Account - This is a special type of account that allows you to receive a token or tokens made by a mint.
4. Creating A Token Metadata Account
5. Upload the Files
6. Mint one Token
7. Create Master Edition

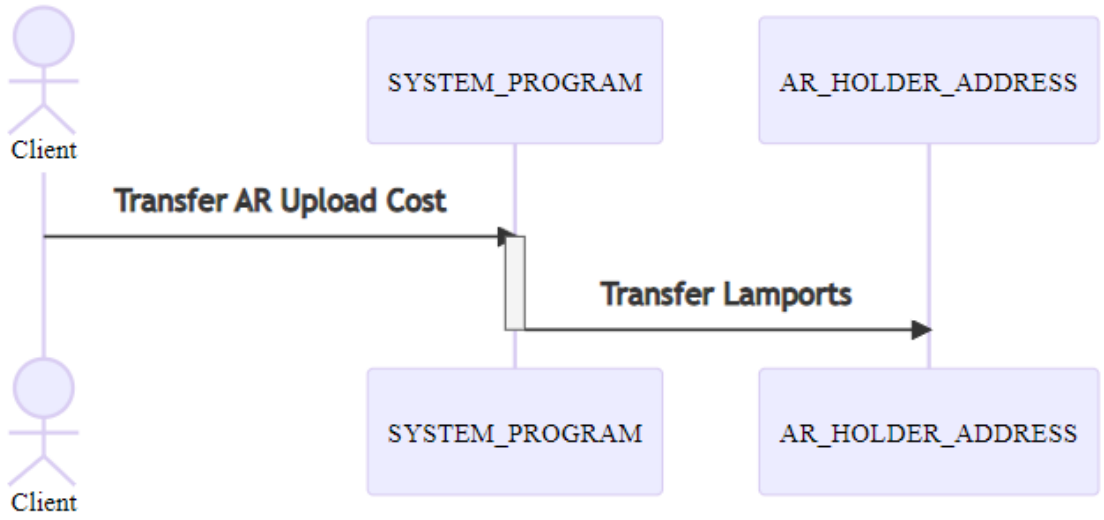
We are breaking these steps down to make it easier to follow. Steps 1-4 happen as one Transaction on the Solana network, and Steps 6-7 are also just one transaction. Lets dive into each step.

Pay For the Upload

Ironically, the actual file upload is one of the last things to happen. Nothing is free, and uploading NFT content is no exception, so we currently need a way to pay for the file(s) to be store on Arweave. To do this we need to transfer lamports(tiny fractions of a SOL

token) to a specific wallet address. This wallet address is connected to a Web2 web application that handles the file upload and the payment to Arweave. This step is only needed in the frontend, as you can specify the URI without going through Arweave

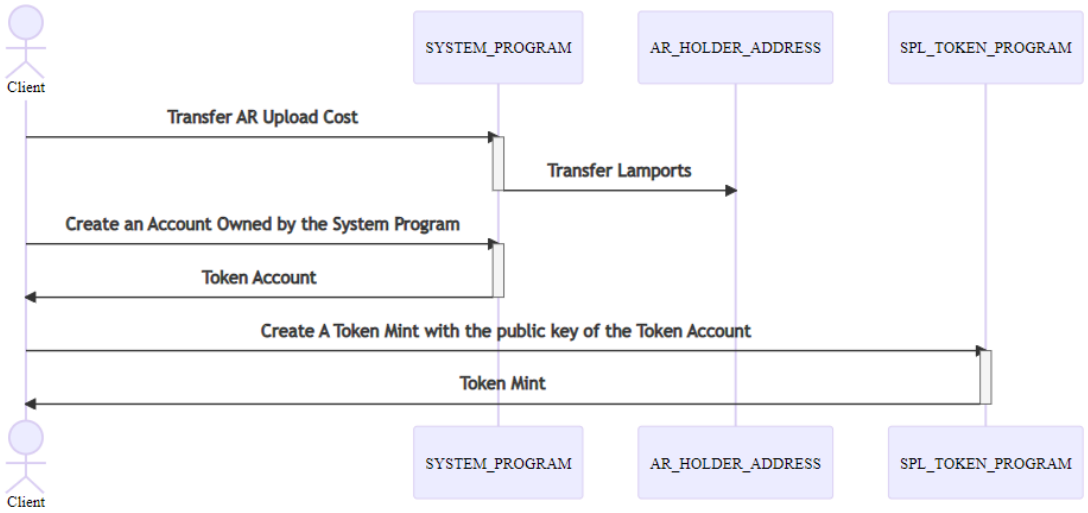
Here is a visual representation of what happened



Creating the Token Mint

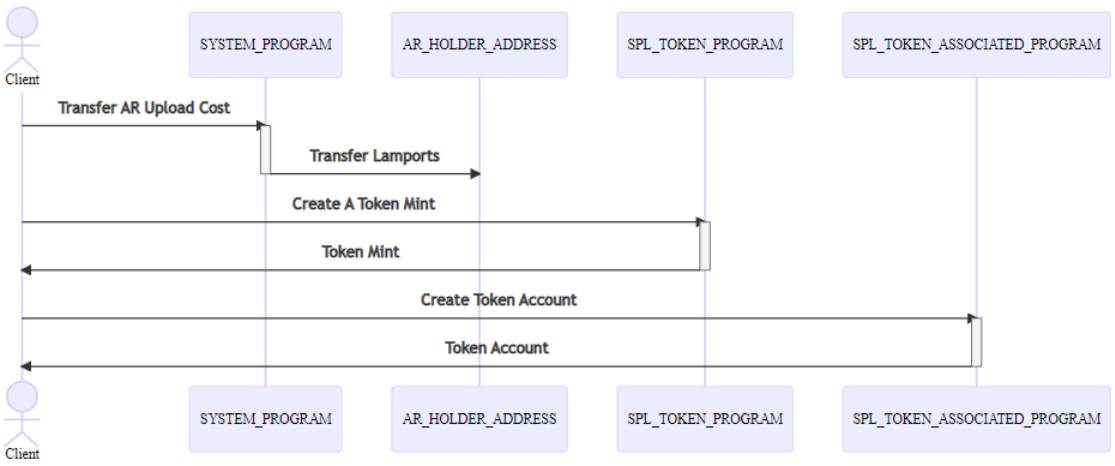
A token mint is how you make tokens, a mint that has a supply of 1 only allows you to make one token. When we make the mint, we are not actually making the token, but a container that can make tokens. In Solana, accounts are like containers for data, like a file. To create a mint you need to creat an account in Solana that "holds" the mint.

Adding to our diagram we see the Mint created.



Creating A Token Associated Account

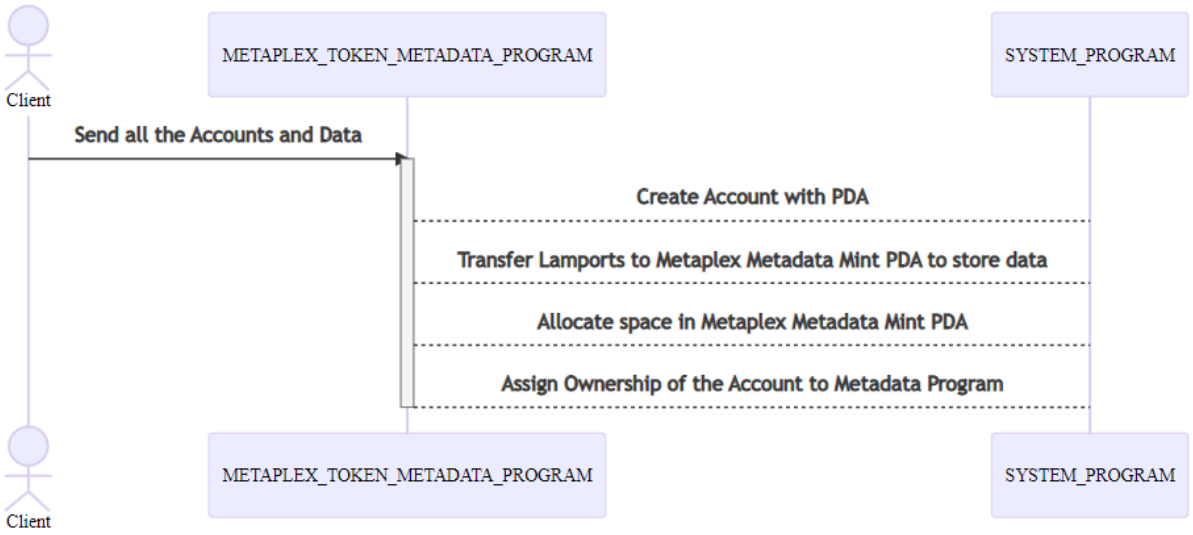
This step is necessary only when your wallet will need to hold the Token after the first MasterEditionor Prints become minted. On the Metaplex frontend, currently this is needed when you are manually uploading and creating an NFT. Without this Account your wallet could not receive the Token.



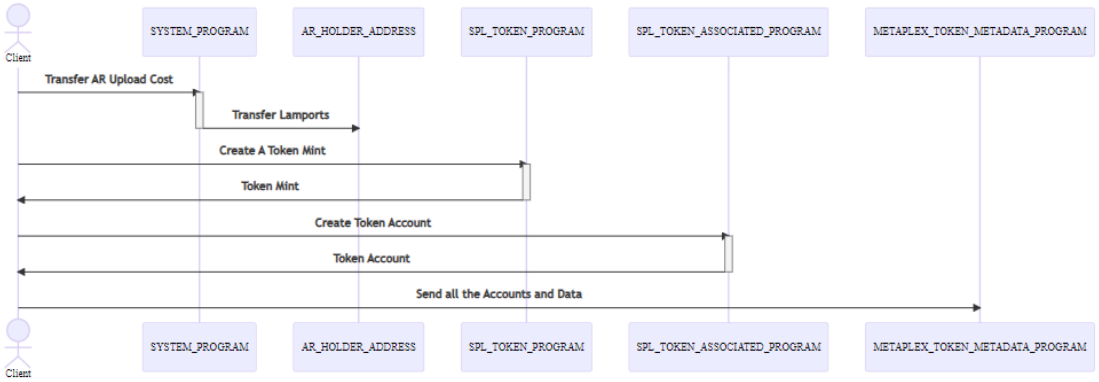
Creating A Token Metadata Account

As we said above the steps we have gone through thus far are represented as one transaction on the Solana network. These are called instructions, and this is the last set of instructions before we hit the network. This step is the backbone of Metaplex. This allows us to store additional information with a Token. There are alot of variables needed to execute these set of instructions, lets go through them. In the frontend, the uris are blank and then updated after the actual upload to ARWEAVE succeeds.

1. The Data - this is a blob of data that conforms to the Token Metadata Standard.
2. Your Public Key
3. A Metadata program Derived account Address - Read more about PDAs here. This is a uninitialized address that the account will be stored at. We do this so we can deterministically find this address again in the future.
4. The mint account public key



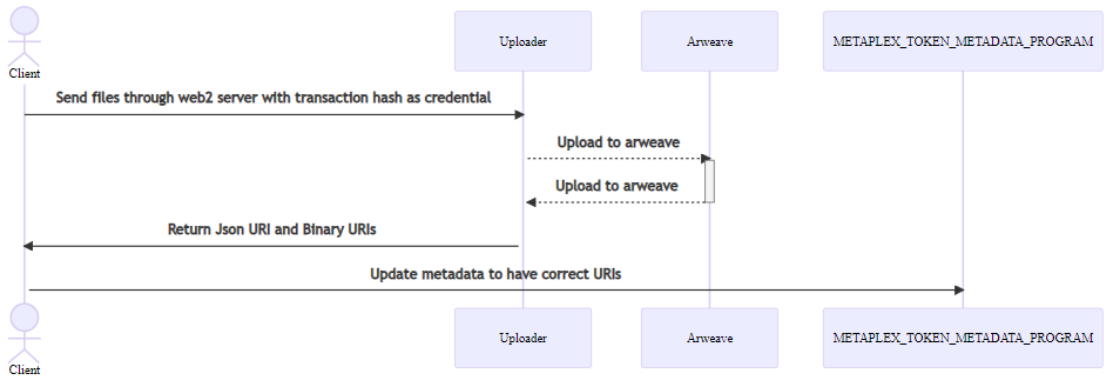
This adds to our ever-growing diagram to complete Transaction 1.



All these instructions are bundled up into one transaction and sent over RPC to the Solana network. At the end of this transaction, you have completed steps 1-4 and the transaction id that you get back from the network will help you in the next step.

Upload the Files

If you are using the frontend, after the above transaction succeeds you will get a Transaction ID. This ID will then be used as a credential to the Web2 uploading system. This system checks the transaction id, mint key and files for validity and size. It then uploads them to arweave and finishes paying arweave out of the SOL you transferred to the special upload wallet.



If you are not using the frontend then this step may not be necessary for you.

Mint One Token

Finally, the mint! We will now begin building the second transaction starting with the `MintTo` instruction. MintTo needs to be passed some data to function, it needs the following:

1. The Mint address - this is the public key of the account we made to store the mint.
2. The destination address - this is that PDA(Program Derived Address) that we used to store the Token MetaData, and it is now the address or public key of the Token Metadata Account.
3. Your public key - since you are the Minting Authority or Owner of the mint, you have the ability to mint.
4. An amount, in this case 1

We will start fresh with our transaction diagram.

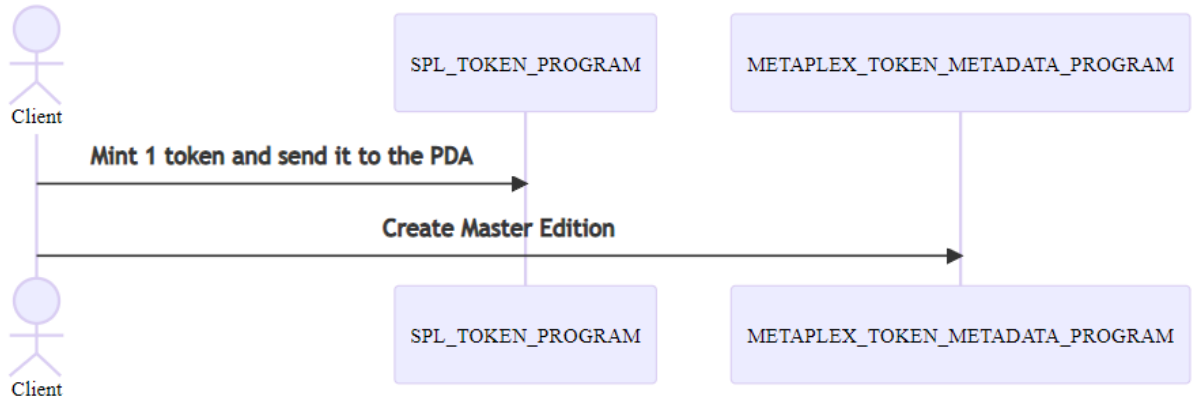
Create Master Edition

In Metaplex, you can make an NFT that is a true one-of-a-kind token, but you can also use the `MasterEdition` construct to create `Prints` of the master edition, just like a painting. You now want to label this token account as a MasterEdition NFT that has a limited supply of 10 possible Limited Edition Prints. Cool! Remember that the point of Metadata is to label mints - not just NFTs. To do this you call the `create_master_edition` endpoint on the Token Metadata Program. By doing this, minting authority is taken away from you, and it creates a new Master Edition PDA that contains information about how large a supply you want to have.

Lets do a deep dive on this Program.

When you want to mint Editions now, you'll need to present a token account containing the token from this Master Edition mint as proof of ownership and authority to do so. Therefore, we will later hand this token over to the Auction Manager, so that it can do the same to print off Editions for winners!

Let's look at our transaction diagram now.



The above instructions will get bundled up into one instruction and sent to the Solana network. Once successful your token account has a bonafide NFT Master Edition in it, we can run an auction where we auction off Limited Edition prints! Let's say we want to auction off three such prints.

2. The Auction

5. Next, we create a token vault using the `init_vault` endpoint of the token vault contract. We'll store our master edition token in it by adding it to the vault using the `add_token_to_inactive_vault` endpoint. This will create a safety deposit box in the vault that contains the the token.
6. Then we will call the `activate_vault` command which **Activates** the vault, locking everything inside.
7. We now **Combine** the vault using `combine_vault`, which is to say, we "open it," so the current authority could, if they wanted, withdraw the tokens inside it. The Auction Manager can only work with vaults in this state, which is why we have to go through the **Activation** phase to get here even though it seems a little nonsensical. See the in depth guide for more color on why these different states exist.
8. Next up, we create the auction, and we say its resource is this vault. The auction has not yet been started, but it has the right resource (the vault). We do this via the `create_auction` command on the Auction contract.
9. Now that we have an auction and a vault, we can go and call the `init_auction_manager` endpoint on the Metaplex contract with both of these accounts among a few others to create an AuctionManager, which ties them both together. Note that `init_auction_manager` takes a special struct called AuctionManagerSettings that allows one to specify how many winners there are and what winners get which items from which safety deposit box. At this point, we can't yet start the auction. The AuctionManager is in an invalidated state and we need to validate it. We do this by validating that the safety deposit boxes we provided to it in the vault have actually what we said are in them when we provided the AuctionManager with it's settings struct.
10. Before we begin validation, we call `set_authority` on both the vault and auction to change its authority to the auction manager, so that it has control over both of those structs. This is a requirement for the validation phase and the rest of the contract lifecycle. **Now you no longer have control over your items.**
11. We call the `validate_safety_deposit_box` endpoint on the Metaplex contract with the one safety deposit box in the vault, and the logic in this endpoint checks that there are exactly 3 printing tokens from the right mint in this box, matching the 3 printing tokens we promised it would have in our AuctionManagerSettings. Once we do this, and because this is the only safety deposit box in the vault, the AuctionManager is now validated.

12. We now call `start_auction` on the Metaplex contract, which, because the AuctionManager has authority over the Auction, calls `start_auction` on the Auction contract, and the auction begins!
13. Users can go and call `place_bid` on the Auction contract to place bids. When they do this, tokens of the `token_mint` type used by the auction are taken from the account they provide, tied to their main wallet, and stored in bidder pot accounts in the auction contract.
14. In order to update a bid, a user must first cancel the original bid, and then place a new bid.
15. Once the auction is over, a user can refund their bid if they did not win by calling `cancel_bid` again. Winners of the auction cannot cancel their bids.
16. The winner of a bid creates a mint with decimals 0, a token account with 1 token in it, and calls the `redeem_printing_v2_bid` endpoint on the Metaplex contract, all in a single transaction. This token is now *officially* a Limited Edition of the "Bob's Cool NFT" Master Edition NFT!
17. You, the auctioneer, visit /#/auction/id/billing and hit the settle button. This first iterates over all three bidders and for each wallet used, calls `claim_bid` on the Metaplex contract, which proxy-calls a `claim_bid` on the Auction contract, telling it to dump the winner's payment into an escrow account called `accept_payment` on the AuctionManager struct. It has the same token type as the auction. Once all payments have been collected, the front end then calls the `empty_payment_account` endpoint one time (since you are the only creator on the Metadata being sold) and the funds in this escrow are paid out to a token account provided of the same type owned by you.