

# Linkdrops

Linkdrops allow users to distribute assets and onboard people to Web3 apps through a simple web link.

They work by storing assets and linking [AccessKeys](#) to them. The [AccessKeys](#) are then distributed to users in the form of web links. These links take users to a website that automatically uses the keys to call the `claim` method in the `linkdrop` contract.


In order for a contract to be considered a Linkdrop-contract it has to follow the [NEP-452 standard](#). The [NEP-452](#) explains the minimum interface required to be implemented, as well as the expected functionality.

## Keypom

The simplest way to create Linkdrops is by using [Keypom](#). Keypom is a community project that allows you to create Linkdrops for NEAR tokens and NFTs in a few clicks.

## AccessKeys

In order to create any kind of drop, you need to first generate key pairs. You will need to create one key per drop.

- The linkdrop contract will store the public part of the key.
- You will give the private part of the key to the user you want to receive the drop.
-  Component
  - [Keypom WebApp](#)
  - [Keypom CLI](#)
  - [Keypom API](#)


```
const dropsNumber =
"2" ; const keysGeneratorUrl =
"https://keypom.sctuts.com/keypair/" ;
asyncFetch ( keysGeneratorUrl + dropsNumber +
"/rootEntropy" ) . then ( ( res )
=>
{ const keyPairs =
JSON . parse ( res . body ) ; const pubKeys =
[] ; const privKeys =
[] ;
keyPairs . forEach ( ( e )
=>
{ pubKeys . push ( e . pub ) ; privKeys . push ( e . priv ) ; } ) ;
const obj =
{ publicKey : pubKeys , privKeys : privKeys , } ;
State . update ( obj ) ; } ) ; * near-api-js * Keypom API
import
{
  KeyPair
}
from
'near-api-js' ;
const newKeyPair =
KeyPair . fromRandom ( 'ed25519' ) ; newKeyPair . public_key
= newKeyPair . publicKey . toString ( ) ; const state =
{ } ;
const dropsNumber =
"2" ; const keysGeneratorUrl =
"https://keypom.sctuts.com/keypair/" ;
fetch ( keysGeneratorUrl + dropsNumber +
"/rootEntropy" ) . then ( ( res )
=>
{ const keyPairs =
JSON . parse ( res . body ) ; const pubKeys =
[] ; const privKeys =
[] ;
keyPairs . forEach ( ( e )
```

```
{ pubKeys . push ( e . pub ); privKeys . push ( e . priv ); } ;  
state . publicKey  
= pubKeys ; state . privKeys  
= privKeys ; } ) ; * Near CLI * Keypom API
```

near generate-key Example response:

```
curl https://keypom.sctuts.com/keypairNUMBER_OF_DROPS/rootEntropy
```

To create a NEAR drop you will ask the contract to create a drop (`create_drop`), passing the public part of the keys you generated, and how much you want to drop on each key use (`deposit_per_use`).

-  Component
- WebApp
- CLI

## NFT Drops



## 2. Transferring the NFT

Having the Drop ID, you now need to transfer the NFT to the linkdrop contract, specifying to which drop you want to add it.

```
• ✎ Component
• WebApp
• CLI

const nftTokenId =
"1" ;

Near . call ( [ { contractName : nftContract , methodName :
'nft_transfer_call' , args :
{ receiver_id : keypomContract , token_id : nftTokenId , msg : dropld . toString ( ) } , deposit :
"1" , gas :
"3000000000000000" } ] ) ; import
{
Wallet
}
from
'./near-wallet' ;

const
KEYPOM_CONTRACT_ADDRESS
=
"v2.keypom.near" ; const
NFT_CONTRACT_ADDRESS
=
"nft.primitives.near" ; const
NFT_TOKEN_ID
=
"1" ; const
DROP_AMOUNT
=
"1000000000000000000000000000000" ;

const nftConnectedWallet =
new
Wallet ( {
createAccessKeyFor :
NFT_CONTRACT_ADDRESS
} ) ;

await wallet . callMethod ( { method :
"nft_transfer_call" , contractId :
NFT_CONTRACT_ADDRESS , args :
{ receiver_id : keypomContract , token_id : nftTokenId , msg : dropld . toString ( ) } , deposit :
1 , gas :
"1000000000000000000000000000000" , } ) ; TheWallet object comes from ouquickstart template near call nft.primitives.near nft_transfer_call '{"receiver_id": "v2.keypom.near", "token_id":, "msg":}' --
depositYocto 1 --gas 1000000000000000000000000000000 --accountId bob.near tip Thelinkdrop contract will validate that you are transferring the NFT to a drop that belongs to you
```

## FT Drops

The process to drop a Fungible Token is very similar to that of creating an [NFT drop](#) . You will first create the drop, and then fund it with FTs.

### 1.Creating a drop

To create a FT drop you will call thecreate\_drop method, now passing aftData argument, which will tell the linkdrop contract to wait for a certain amount of FT to be transferred.

The contract will then create a drop andreturn the numerical ID that identifies it.

```
• ✎ Component
• WebApp
• CLI

const keypomContract =
"v2.keypom.near" ; const ftContract =
"ft.primitives.near" ; const dropAmount =
"1000000000000000000000000000000" ;

Near . call ( [ { contractName : keypomContract , methodName :
"create_drop" , args :
```

```

{ public_keys : state . publicKeys , deposit_per_use : dropAmount , ftData :

{ contractId : ftContract , senderId : accountId , // This balance per use is balance of human readable FTs per use. amount :

"1" // Alternatively, you could use absoluteAmount, which is dependant on the decimals value of the FT // ex. if decimals of an ft = 8, then 1 FT token would be absoluteAmount =
100000000 } , } , deposit :

"2300000000000000000000000000" ,

// state.publicKeys.length * dropAmount + 3000000000000000000000000000, gas :

"10000000000000000" } , ] ) ; import

{

Wallet

}

from

'./near-wallet' ;

const

KEYPOM_CONTRACT_ADDRESS

=

"v2.keypom.near" ; const

FT_CONTRACT_ADDRESS

=

"ft.primitives.near" ; const

DROP_AMOUNT

=

"1000000000000000000000000000" ;

const wallet =

new

Wallet ( {

createAccessKeyFor :

KEYPOM_CONTRACT_ADDRESS

} ) ;

await wallet . callMethod ( { method :

"create_drop" , contractId :

KEYPOM_CONTRACT_ADDRESS , args :

{ public_keys : state . publicKeys , deposit_per_use :

DROP_AMOUNT , ftData :

{ contractId :

FT_CONTRACT_ADDRESS , senderId : accountId ,

// TODO How to get account id // This balance per use is balance of human readable FTs per use. amount :

"1" // Alternatively, you could use absoluteAmount, which is dependant on the decimals value of the FT // ex. if decimals of an ft = 8, then 1 FT token would be absoluteAmount =
100000000 } , } , deposit :

"2300000000000000000000000000"


// state.publicKeys.length * dropAmount + 3000000000000000000000000000, gas :

"10000000000000000" } , } ) ; TheWallet object comes from our quickstart template near call v2.keypom.near create_drop '{"public_keys": , "deposit_per_use": "1000000000000000000000000000",
"ftData": {"contractId": "ft.primitives.near", "senderId": "bob.near", "amount": "1"}}' --depositYocto 2300000000000000000000000000 --gas 1000000000000000 --accountId bob.near

```

## 2. Transferring FT

Having the Drop ID, you now need to transfer the fungible tokens to the linkdrop contract.

note To transfer FTs to an account, you need to first [register](#) the receiver account (e.g. the keypom contract) on the FT contract. \*  Component \*

WebApp \*

```
Near . call ( [ { contractName : ftContract , methodName :
```

```
"ft_transfer" , args :
```

```
{ receiver_id : keypomContract , amount :
```

```
"1" , } , deposit :
```

```
"1" , gas :
```

```
"3000000000000000" , } , ] ) ; import
```

```
{
```

```
Wallet
```

```
}
```

```
from
```

```
'./near-wallet' ;
```

```
const
```

```
"v2.keypom.near" ; const
```

[illegible]

## Building drop links

```
const
getLinks

=

()

=>

{ const links =

[] ;

// It assumes that private keys have been already stored in State by using State.init() and State.update() method state . privKeys . map ( ( e , i )

=>

{ const link = "https://app.mynearwallet.com"

+

"/linkdrop/v2.keypom.near/"

+ e ; links . push ( link ) ; } ) ;

return links ; } ; Example response [
'https://app.mynearwallet.com/linkdrop/v2.keypom.near/ed25519:2H32THYM8ob336yk81cZUxpidvKi34zLck6a97ypmCY8bbSAuEfrCTu9LWmWGiG9df2C6vkg2FGKGZzY9qE4aEcj' ,
'https://app.mynearwallet.com/linkdrop/v2.keypom.near/ed25519:3eoMcqKmmY9Q6qgBy3hZy65HisZ8NXQd9aGGYUGe6RRsmNpGJS5YN64MgZaBVVYJjhbFXhQ2ca3DRRBiKh1rYM48'
] note If you didn't save your linkdrop links before closing NEAR App, you can always find them on KeyPom app .
```

## Additional Resources

- ### 1. Linkdrop plus

2. allows to create a Simple Drop. Powered by [KeyPom](#)
3. [Keypom Drop Viewer](#)
4. shows drops created by current logged in user. [Edit this page](#) Last updated on Jan 30, 2024 by garikbesson Was this page helpful? Yes No

[Previous Non-Fungible Tokens \(NFT\)](#) [Next Price Oracles](#)