# Mintbase Flutterchain Integration

If this is your first time interacting with Mintbase, we recommend you read the Mintbase documentation.

The next step is to download the library, set up, and initialize **NearBlockChainService**.

```
NearBlockChainService.defaultInstance();
```

Then, choose the network you want to work with. You can switch the network by using the **setBlockchainNetworkEnvironment** function. Just input the *NearNetworkType* enum.

```
NearBlockChainService().setBlockchainNetworkEnvironment(

        newUrl: widget.networkType == NearNetworkType.mainnet
            ? NearBlockChainNetworkUrls.listOfUrls.elementAt(1)
            : NearBlockChainNetworkUrls.listOfUrls.first,
      );
```

# Collection Interaction

Now you can create an NFT collection on Mintbase. You can do this with **deployNFTCollection**. For this operation, you need your account ID, public and private keys, symbol, name for the collection, and the owner's ID. You can also choose an icon, spec, base URI, reference, factory contract, or save the default options. To create a collection, your balance must be more than 3.7 NEAR. And name collection characters must be in lower case.

```
Future<BlockchainResponse> deployNFTCollection({
    required String accountId,
    required String publicKey,
    required String privateKey,
```

```
    required String name,
    required String ownerId,
    String? spec = "nft-1.0.0",
    String? icon =
        "
lEQVRYR8VXW4hVVRj+vrX3OYbFNBl2eSsoa8pkxMJmzpGGoIwMY0wwImuCXmKKkiikKayHQIpAwbdAsws
kBYqJSlANek7TUOPURGKCWdmFkJnsoZnhzNn/F2vcZ9gez2UfEVoP+2Gv///Xt771X4mUq6enpz2Kogck
3S2pk+R1ki736iT/kfQzyW/N7PNsNrtvcHDwTBrTbCbU1dV1cxiGGyWtA3BJM/l4f1rSLjPbPDQ0dKyRT
l0AS5YsubStre11AP0AwiYHi+SJmJGFCdmypG3T09MDIyMjk7Vs1ASQy+VuIrkbQEezG3vqJfXGTzHhnF
tlZttIuoTuUQC9hULheLW98wDkcrnbARwkeWWzw+P9ewCsB/AYgBkz6yW5lmRfUl/SOMmVhUJhJPn/HAD
+5gCKLRzubS0AMO59MTa828z2Oud2VF8gBtGdZGIOQPzmX6ehPWk4iqKOIAg+BnCr/y/pJUmnnHPv1WHw
6NTU1B0Vn5gDkM/ntwB4NiXtc2KS9kdR1B+G4aMA/pD0kaQHGwDwILcUi8UNs37jPz7UgiD4PoW318Qn6
QSA/STbJc2T9EkjAAB8dCwuFos/zgLI5/M7YydqlYBa8vvMbFcTAJ6FncVisY/5fP4KT10LSaYZyFQAAE
yHYXitB+BD6N1mVlvYTwsAZraeuVxuR3XMNjpM0mFJW0n+VUfub0mnSS6S5JxznQBeBpDMkBXV7R7AKEk
v1HRJGiLpvX0rgHYzGyB5S5yuZ/VJHioUCv25XO4Vn/0AHDCzD51z35DMViWnUf8EEwC8H6RZj5O8Py5M
3pF+D4JghZn9lFDeB+AtAF9U/vmw9CBJ3lt1yIQHYIks1gzEapJPSlodJ53xcrncmclkTiUOOwRgM8n9i
X8Pk/S+tqqKAWsZgJmddM6971kzs+f9W5P0VbOyZmZmZjyo5yStcc4djKJoE8kxkvNrAWjlCVZLugHAYe
fcpJmtIzkAIKii7ldJL5jZkTAMF0t6A8CNNeidaMkJAXjqNwLobvZWafYljbYahhcVAIDtrSaiiwpgNhG
lSMVjAI5J8m3Xm3FILapD8UKSd9XwiVriZ1Ox36lXjCRtMLPPwjC8s1QqHRgeHv6tu7v7IefcNYkQG8tk
MqPlcnmNrylm9q9z7lMAlzXyg7li5IXqlOOvADwN4EsAPoOdLpVKHdlsdm/SCUku8/UdwIo4NzxF8noAL
zYAcG45jlmobkjelnTc014xRPI+Sa8BWJ4w7lsyH8qVtUfSHpLv1ANwXkPiBWu0ZEeiKOoLgsAzMd+nXT
Nb7pz7geTsQBLfuIvkqwBWnu3I9ASAZSSfqQOgdkvmhaubUpKboij6IAiCpSSH/aAB4JEqw9+VSqXeTCZ
zm6Q/nXNXA/A94rxqAA2b0opwjbb8JADfzy8FcFWdW01K8kwtqFdZU7XlCRCpB5M0GQ9A+sGkYrDF0awe
jgsbzZLW/rfhtPpKTcbzM5J+uZDx/D8+0FUx/4DhyAAAAABJRU5ErkJggg==",
    String? baseUri = "https://arweave.net",
    String? reference,
    String? referenceHash,
    String? factoryContract,
  })
```

You can also check which collections you own with the **checkOwnerCollection** function, and where you're a minter with the **checkMinterCollection** function. For these functions, you need to input the owner ID.

```
Future<List<dynamic>> checkOwnerCollection({required String owner_id})
```

```
Future<List<dynamic>> checkMinterCollection({required String owner_id})
```

After creating a collection, you can add or delete minters using the **addDeleteMinters** function. You need to input your account ID, public and private keys, your NFT collection's full name, the account ID you want to interact with, and specify whether you want to add (set `isAdd` to `true`) or remove (`false`) the minter.

```
Future<dynamic> addDeleteMinters({
    required String accountId,
    required String publicKey,
    required String privateKey,
    required String nftCollectionContract,
    required String name,
    required bool isAdd,
  })
```

Another function is to transfer a collection. You can do this with the **transferNFTCollection** function. You need to input your account ID, public and private keys, your NFT collection's full name, the new owner's account ID, and whether you want to keep the old minters (set `true`) or not (`false`).

```
Future<dynamic> transferNFTCollection({
    required  String  accountId,
    required  String  publicKey,
    required String privateKey,
    required String nftCollectionContract,
    required String new_owner,
    required bool keep_old_minters,
  })
```

You can check the minters in a collection using the **getMinters** function. For this function, you need to input your account ID, public and private keys, and the NFT collection's full name.

```
  Future<List<dynamic>> getMinters({
    required String accountId,
    required String publicKey,
    required String privateKey,
    required String nftCollectionContract,
  })
```

## NFT Interaction

The first step is creating an NFT. You can do this with the **mintNFT** function. For this function, you need to input your account ID, public and private keys, your NFT collection's full name, owner ID, description, title, and media (this will be the main media for the NFT). You can also input the media type, animation, number to mint (by default 1), split between (Forever Royalties), split owners (Split Revenue), tags, extra (Custom), and for category, you must choose

the *CategoryNFT* enum, document, and base URL (by default `https://arweave.net/`).

```
Future<BlockchainResponse> mintNFT({
    required String accountId,
    required String publicKey,
    required String privateKey,
    required String nftCollectionContract,
    required String owner_id,
    required String description,
    required String title,
    required Uint8List media,
    String? media_type,
    Uint8List? animation,
    int num_to_mint = 1,
    Map<String, int>? split_between,
    Map<String, int>? split_owners,
    List<String>? tags,
    List<dynamic>? extra,
    CategoryNFT? category,
    Uint8List? document,
    String? baseURL = "https://arweave.net/",
  })
```

After creating an NFT, you can transfer it using the **transferNFT** function with the following parameters: account ID, public and private keys, your NFT collection's full name, and `tokenIds` (with two parameters: NFT ID and account ID).

```
Future<BlockchainResponse> transferNFT(
      {required String accountId,
      required String publicKey,
      required String privateKey,
      required String nftCollectionContract,
      required List<List<String>> tokenIds})
```

Next, you can multiply your NFT using the **multiplyNFT** function. For this function, you need the following parameters: account ID, public and private keys, NFT collection's full name, NFT name, and number to mint. Additional options include media and reference.

```
Future<BlockchainResponse> multiplyNFT({
    required String nameNFTCollection,
```

```
    required String nameNFT,
    required String accountId,
    required String publicKey,
    required String privateKey,
    required int numToMint,
    String? media,
    String? reference,
  })
```

You can also get information about your NFT using the **checkNFTInfo** function with the following parameters: owner ID.

```
Future<List<dynamic>> checkNFTInfo({required String owner_id})
```

## Marketplace

Before interacting with the marketplace, you need to make a deposit using the **listingActivate** function. For this function, you need the following parameters: account ID and public and private keys. You can also change the marketplace contract ID.

```
Future<BlockchainResponse> listingActivate({
      required String accountId,
      required String publicKey,
      required String privateKey,
      String? market_account_id,})
```

You can check the listing of an NFT using the **checkListingNFT** function with the following parameter: owner ID.

```
Future<List<dynamic>> checkListingNFT({required String ownerId})
```

## Simple List

You can simply list your NFT using the **simpleListNFT** function. For this, you need the following parameters: account ID, public and private keys, NFT collection's full name, token ID, and price.

```
Future<BlockchainResponse> simpleListNFT({
    required String nameNFTCollection,
    required String tokenId,
    required String price,
    required String accountId,
    required String publicKey,
    required String privateKey,
    String? market_account_id,
  })
```

You can also buy a simple list NFT using the **buySimpleListNFT** function with the following parameters: account ID, public and private keys, NFT collection's full name, and token ID. Additionally, you can add the following parameters: referrer ID and market account ID.

```
Future<BlockchainResponse> buySimpleListNFT({
    required String accountId,
    required String publicKey,
    required String nameNFTCollection,
    required String privateKey,
    required int tokenId,
    String? referrer_id,
    String? market_account_id,
  })
```

Before buying a simple list NFT, you can check the price using the **getPriceForBuySimpleListNFT** function. For this function, you need to input the NFT collection's full name and token ID.

```
 Future<String> getPriceForBuySimpleListNFT(
      {required String nftContractId, required int tokenId})
```

To remove the NFT from the marketplace, you can use the **unlistNFT** function with the following parameters: account ID, public and private keys, NFT collection's full name, and token ID. You can also change the market account ID.

```
Future<BlockchainResponse> unlistNFT(
      {required String accountId,
      required String publicKey,
      required String nameNFTCollection,
      required String privateKey,
      required int tokenId,
      String? market_account_id})
```

# Rolling Auction

For a rolling auction NFT, you can use the **rollingAuctionNFT** function. For this function, you need the following parameters: account ID, public and private keys, NFT collection's full name, token ID, and price. You can also change the market account ID.

```
Future<BlockchainResponse> rollingAuctionNft(
      {required String accountId,
      required String publicKey,
      required String nameNFTCollection,
      required String privateKey,
      required int tokenId,
      required String price,
      String? market_account_id})
```

To make offers, you can use **offersToRollingAuction** with the following parameters: account ID, public and private keys, NFT collection's full name, token ID, and price bid. You can also input a timeout in hours and change the market account ID.

```
  Future<bool> offersToRollingAuction({
    required String accountId,
    required String publicKey,
    required String nameNFTCollection,
    required String privateKey,
    required int tokenId,
    required String priceBid,
    int? timeoutInHours,
    String? market_account_id,
  })
```

You can check the starting price bid using the **getPriceForBuySimpleListNFT** function and the maximum bid at the moment using the **checkMaxPriceBidNFT** function. For this function, you need to input the NFT contract ID and token ID.

```
  Future<String> getPriceForBuySimpleListNFT(
      {required String nftContractId, required int tokenId})
```

```
Future<Map<String, String>> checkMaxPriceBidNFT(
      {required String nftContractId, required int tokenId})
```

To remove the NFT from the marketplace, you can use the **delistNFT** function with the following parameters: account ID, public and private keys, NFT collection's full name, and token ID. You can also change the market account ID.

```
Future<BlockchainResponse> delistNFT(
      {required String accountId,
      required String publicKey,
      required String nameNFTCollection,
      required String privateKey,
      required int tokenId,
      String? market_account_id})
```