Collections Reference

This document is a reference to the Collections module of the Myria Core SDK. The module contains information about Myria collections.

Interfaces

`CreateCollectionParams`

Data structure passed to createCollection() method, which contains required data to create a new collection.

```
interface CreateCollectionParams {
  name: string;
  collectionImageUrl?: string;
  description?: string;
  iconUrl?: string;
  contractAddress: string;
  ownerPublicKey: string;
  metadataApiUrl: string;
  starkKey: string;
  projectId: number;
}
```

Attributes

- `name` - collection name
- `collectionImageUrl` - url for main collection image
- `description` - collection description
- `iconUrl` - url for collection icon
- `contractAddress` - contract address used to withdraw assets to the Ethereum network
- `ownerPublicKey` - public key of the collection owner
- `metadataApiUrl` - api url that will store collection metadata
- `starkKey` - Stark Key, has to start with `0x`
- `projectId` - project id of the collection

`CreateCollectionResponse`

Data structure returned by the createCollection() method.

```
interface CreateCollectionResponse {
  id: number;
  createdAt: string;
  updatedAt: string;
  name: string;
  collectionImageUrl: string;
  description: string;
  iconUrl: string;
  contractAddress: string;
  ownerPublicKey: string;
  metadataApiUrl: string;
  starkKey: string;
  pubicId: string;
}
```

Attributes

- `id` - collection id
- `createdAt` - when the collection was created
- `updatedAt` - when the collection was updated last time
- `name`- collection name
- `collectionImageUrl` - url for main collection image
- `description` - collection description
- `iconUrl` - url for collection icon
- `contractAddress` - contract address used to withdraw assets to the Ethereum network
- `ownerPublicKey` - public key of the collection owner
- `metadataApiUrl` - api url that will store collection metadata
- `starkKey` - Stark Key, has to start with `0x`
- `pubicId` - public id of the collection

`GetCollectionParams`

Data structure passed to getCollectionList() method, which contains required parameters for querying collections.

```
interface GetCollectionParams {
  limit?: number;
  page?: number;
  isHot?: boolean;
}
```

Attributes

- `limit` - the max number of collections to return per response
- `page` - the page number of returned collections based on the defined `limit`
- `isHot` - whether the collection should be included in the hot section on the marketplace

`CollectionDetailsResponse`

Data structure that contains collection details response.

```
interface CollectionDetailsResponse {}
```

`CollectionDetailsResponseData`

Data structure that contains collection details response data.

```
interface CollectionDetailsResponseData {
  id: number;
  createdAt: string;
  updatedAt: string;
  name: string;
  collectionImageUrl: string;
  description: string;
  iconUrl: string;
  contractAddress: string;
  ownerPublicKey: string;
  metadataApiUrl: string;
  starkKey: string;
  publicId: string;
  metadataSchema: any[];
  project: ProjectResponseData;
}
```

Attributes

- `id` - collection id
- `createdAt` - when the collection was created
- `updatedAt` - when the collection was updated last time
- `name`- collection name
- `collectionImageUrl` - url for main collection image
- `description` - collection description
- `iconUrl` - url for collection icon
- `contractAddress` - contract address used to withdraw assets to the Ethereum network
- `ownerPublicKey` - public key (wallet address) of the collection owner
- `metadataApiUrl` - api url that will store collection metadata
- `starkKey` - Stark Key, has to start with `0x`
- `publicId` - public id of the collection
- `metadataSchema` - an array of MetaDataShema
- `project` - ProjectResponseData object

`CollectionListResponse`

Data structure returned by the getCollectionList() method.

```
interface CollectionListResponse {
  id: number;
  createdAt: string;
  updatedAt: string;
  name: string;
  collectionImageUrl: string;
  description: string;
  iconUrl: null;
  contractAddress: string;
  ownerPublicKey: string;
  metadataApiUrl: string;
  starkKey: string;
  publicId: string;
  isHot: boolean;
  metadataSchema: MetaDataSchema[];
  project: ProjectResponseData;
}
```

Attributes

- `id` - collection id
- `createdAt` - when the collection was created
- `updatedAt` - when the collection was updated last time
- `name`- collection name
- `collectionImageUrl` - url for main collection image
- `description` - collection description
- `iconUrl` - url for collection icon
- `contractAddress` - contract address used to withdraw assets to the Ethereum network
- `ownerPublicKey` - public key of the collection owner
- `metadataApiUrl` - api url that will store collection metadata
- `starkKey` - Stark Key, has to start with `0x`
- `publicId` - public id of the collection
- `isHot` - whether the collection should be included in the hot section on the marketplace
- `metadataSchema` - an array of MetaDataShema
- `project` - ProjectResponseData object

`MetaDataSchema`

Data structure that contains metadata schema each item in the collection will follow.

```
interface MetaDataSchema {
  god: string;
  name: string;
  type: number;
  attack: number;
  rarity: number;
  element: string;
  product: number;
  imageUrl: string;
  collectable: true;
  animationUrl: string;
  animationUrlMimeType: string;
}
```

`CollectionMetadataSchemaParams`

Data structure passed to createCollectionMetadataByAddress() method, which contains required data for creating a metadata schema field.

```
interface CollectionMetadataSchemaParams {
    name: string;
    type?: string;
    filterable?: boolean;
}
```

`CollectionMetadataResponse`

Data structure returned by the getCollectionMetadataByAddress() method.

```
interface CollectionMetadataResponse {
    name: string;
    type?: string;
    filterable?: boolean;
}
```

Attributes

- `name` - metadata schema field name
- `type` - metadata schema field type
- `filterable` - whether a field should be filterable in the marketplace

`CreateCollectionMetadataResponseType`

Data structure that contains an array of metadata schema fields.

```
interface CreateCollectionMetadataResponseType {
    metadata: any[];
}
```

Attributes

- `metadata` - metadata schema fields

Methods

`createCollection()`

Creates a collection.

Parameters

- CreateCollection object

Returns

Returns a collection object.

```
createCollection(payload: CreateCollectionParams):
Promise<CreateCollectionResponse | undefined>;
```

Returns

Typescript

```
import { CollectionManager, CreateCollectionParams,
CreateCollectionResponse, EnvTypes } from "myria-core-sdk";

(async (): Promise<void> => {
  // STAGING or PRODUCTION
  const env = EnvTypes.STAGING;

  const collectionManager: CollectionManager = new
CollectionManager(env);

  const params: CreateCollectionParams = {
    name: "COLLECTION_NAME",
    description: "COLLECTION_DESCRIPTION",
    contractAddress: "CONTRACT_ADDRESS",
    metadataApiUrl: "METADATA_API_URL",
    ownerPublicKey: "OWNER_PUBLIC_KEY",
    projectId: "PROJECT_ID",
    starkKey: "STARK_KEY",
  };

  const newCollectionResponse: CreateCollectionResponse | undefined =
    await collectionManager.createCollection(params);
})();
```

Response

▸ CreateCollectionResponse

`getCollectionList()`

Returns a list of collections.

```
getCollectionList(params?: GetCollectionParams):
Promise<APIResponseType<CommonPaginateDataTypes<CollectionListResponse[]>>
| undefined>;
```

Parameters

- GetCollectionParams object

Returns

Returns an object with response call status and the `data` object that contains a list of collections.

Returns

Typescript

```
import { CollectionManager, EnvTypes } from "myria-core-sdk";

(async (): Promise<void> => {
  // STAGING or PRODUCTION
  const env = EnvTypes.STAGING;

  const collectionManager: CollectionManager = new
CollectionManager(env);

  const collectionListResponse = await
collectionManager.getCollectionList();
})();
```

Response

▸ CollectionListResponse[]

`getCollectionById()`

Returns a collection by its id.

```
getCollectionById(id: number): Promise<CollectionDetailsResponse |
undefined>;
```

Parameters

- `id` - collection id

Returns

Returns an object with response call status and the `data` object that contains information about the project.

Returns

Typescript

```
import { CollectionManager, EnvTypes } from "myria-core-sdk";

(async (): Promise<void> => {
  // STAGING or PRODUCTION
  const env = EnvTypes.STAGING;

  const collectionManager: CollectionManager = new
CollectionManager(env);

  const collectionResponse =
    await collectionManager.getCollectionById(COLLECTION_ID);
})();
```

Response

▸ CollectionDetailsResponse

`getCollectionMetadataByAddress()`

Returns metadata of the collection.

```
getCollectionMetadataByAddress(contractAddress: string):
Promise<APIResponseType<CollectionMetadataResponse> | undefined>;
```

Parameters

- `contractAddress` - contract address used to withdraw assets to the Ethereum network

Returns

Returns an object with response call status and the `data` object that contains information about the collection's metadata.

Returns

Typescript

```
import { CollectionManager, EnvTypes } from "myria-core-sdk";

(async (): Promise<void> => {
  // STAGING or PRODUCTION
  const env = EnvTypes.STAGING;

  const collectionManager: CollectionManager = new
CollectionManager(env);

  const collectionResponse =
    await
collectionManager.getCollectionMetadataByAddress("CONTRACT_ADDRESS");
})();
```

Response

▸ CollectionMetadataResponse

`getCollectionByPublicId()`

Returns a collection by its public id.

```
getCollectionByPublicId(publicId: string):
Promise<APIResponseType<CollectionDetailsResponseData> | undefined>;
```

Parameters

- `publicId` - public id of the collection

Returns

Returns an object with response call status and the `data` object that contains information about the collection.

Returns

Typescript

```typescript
import { CollectionManager, EnvTypes } from "myria-core-sdk";

(async (): Promise<void> => {
  // STAGING or PRODUCTION
  const env = EnvTypes.STAGING;

  const collectionManager: CollectionManager = new
CollectionManager(env);

  const collectionResponse =
    await
collectionManager.getCollectionByPublicId(COLLECTION_PUBLIC_ID);
})();
```

Response

▸ CollectionDetailsResponseData