Projects Reference

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

Interfaces

`CreateProjectParams`

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

```
interface CreateProjectParams {
  name: string;
  companyName: string;
  contactEmail: string;
  starkKey: string;
}
```

Attributes

- `name`- project name
- `companyName` - company name that will be working on the project
- `contactEmail` - contact email
- `starkKey` - Stark Key, has to start with `0x`

`ProjectResponse`

Data structure returned by the createProject() method.

```
interface ProjectResponse {
  status: string;
  data: ProjectResponseData[] | ProjectResponseData | undefined;
}
```

Attributes

- `status` - registration status: success, failure
- `data` - an object or array of ProjectResponseData

`ProjectResponseData`

Data structure that contains `data` value of ProjectResponse.

```
interface ProjectResponseData {
  id: number;
  createdAt: string;
  updatedAt: string;
  name: string;
  companyName: string;
  contactEmail: string;
  collectionLimitExpiresAt?: string | null;
  collectionMonthlyLimit?: number;
  collectionRemaining?: number;
  publicId?: string;
  starkKey: string;
}
```

Attributes

- `id` - project id on the Myria network
- `createdAt` - when the project was created
- `updatedAt` - when the project was updated last time
- `name` - project name
- `companyName` - company name that will be working on the project
- `contactEmail` - contact email
- `collectionLimitExpiresAt` - expiration date of your current limit to create collections
- `collectionMonthlyLimit` - the max number of collections to create each month for a given project
- `collectionRemaining` - the number of collections remained to create this month
- `publicId` - public id of your project within the Myria network
- `starkKey` - Stark Key, has to start with `0x`

`UpdateProjectParams`

Data structure passed to updateProject() method, which contains required data to update a project.

```
interface UpdateProjectParams {
  id: number;
  name: string;
  companyName: string;
  contactEmail: string;
  starkKey: string
}
```

Attributes

- `id` - project id on the Myria network
- `name`- project name
- `companyName` - company name that will be working on the project
- `contactEmail` - contact email
- `starkKey` - Stark Key, has to start with `0x`

Methods

`createProject()`

Creates a new Myria project.

```
createProject(payload: CreateProjectParams): Promise<ProjectResponse |
undefined>;
```

Parameters

- CreateProjectParams object

Returns

Returns a project object.

Example

Typescript

```typescript
import { ProjectManager, CreateProjectParams, ProjectResponse, EnvTypes
} from "myria-core-sdk";

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

  const projectManager: ProjectManager = new ProjectManager(env);

  const params: CreateProjectParams = {
    name: "PROJECT_NAME",
    companyName: "COMPANY_NAME",
    contactEmail: "COMPANY_EMAIL",
    starkKey: "STARK_KEY",
  };

  const newProjectResult: ProjectResponse | undefined =
    await projectManager.createProject(params);
})();
```

Response

▸ ProjectResponse

`getProjectDetail()`

Returns details of the Myria project.

```typescript
getProjectDetail(id: string): Promise<ProjectResponse | undefined>;
```

Parameters

- `id` - project id

Returns

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

Example

Typescript

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

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

  const projectManager: ProjectManager = new ProjectManager(env);

  const projectDetailResponse: ProjectResponse | undefined =
    await projectManager.getProjectDetail("PROJECT_ID");
})();
```

Response

▸ ProjectResponse

`getProjectList()`

Returns a list of available Myria projects.

```typescript
getProjectList(): Promise<ProjectResponse | undefined>;
```

Parameters

No parameters.

Returns

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

Example

Typescript

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

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

  const projectManager: ProjectManager = new ProjectManager(env);

  const projectListResponse: ProjectResponse | undefined =
    await projectManager.getProjectList();
})();
```

Response

▸ ProjectResponse

`updateProject()`

Updates a Myria project by a given id.

```
updateProject(payload: UpdateProjectParams): Promise<ProjectResponse |
undefined>;
```

Parameters

- UpdateProjectParams object

Returns

Returns a project object.

Example

Typescript

```
import { ProjectManager, UpdateProjectParams, ProjectResponse, EnvTypes
} from "myria-core-sdk";

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

  const projectManager: ProjectManager = new ProjectManager(env);

  const params: UpdateProjectParams = {
    id: PROJECT_ID
    name: "PROJECT_NAME",
    companyName: "COMPANY_NAME",
    contactEmail: "COMPANY_EMAIL",
    starkKey: "STARK_KEY"
  };

  const updatedProjectResult: ProjectResponse | undefined =
    await projectManager.createProject(params);
})();
```

Response

▸ ProjectResponse