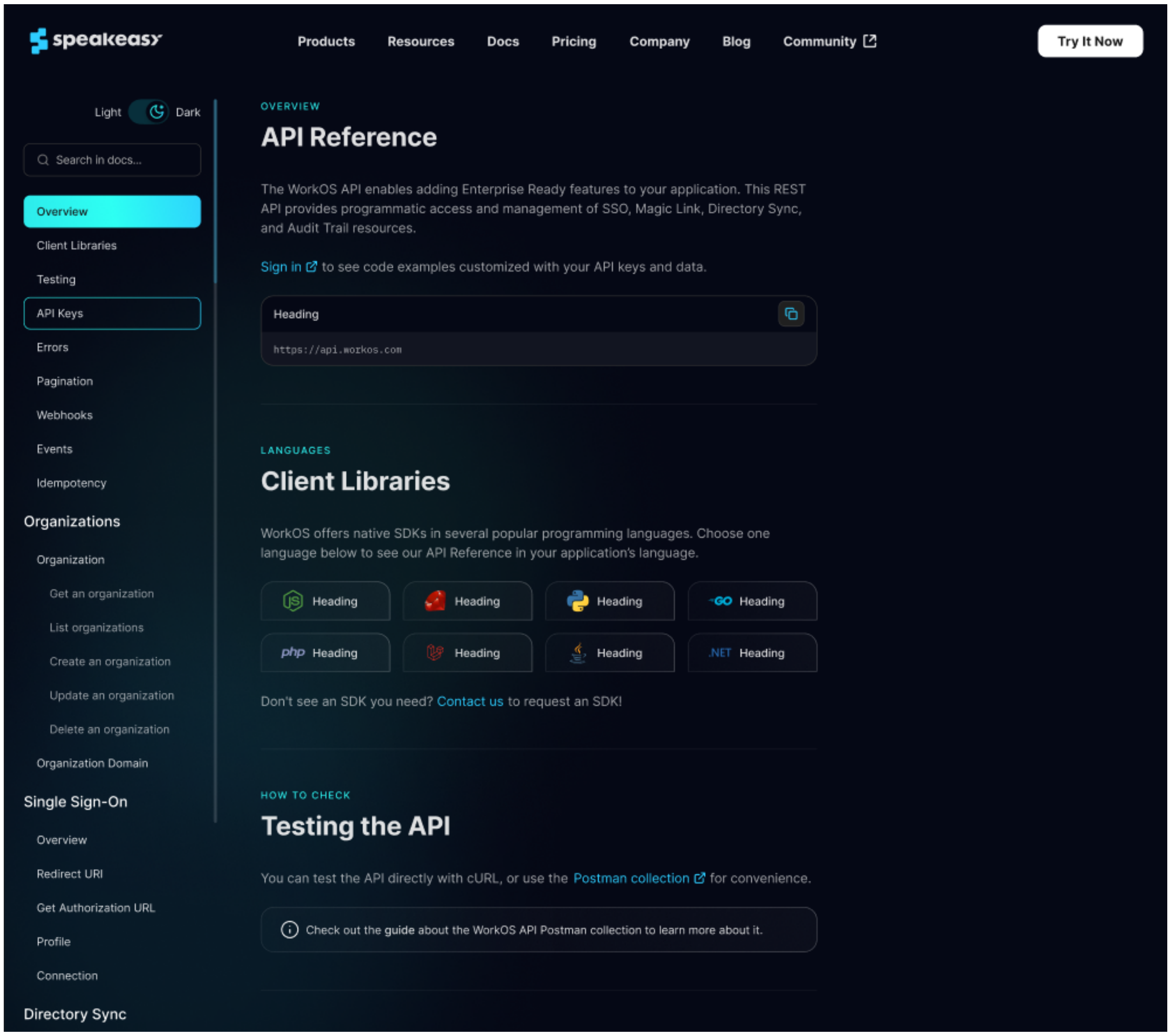
SDK Docs



OVERVIEW

API Reference

The WorkOS API enables adding Enterprise Ready features to your application. This REST API provides programmatic access and management of SSO, Magic Link, Directory Sync, and Audit Trail resources.

Sign in to see code examples customized with your API keys and data.

Heading

```
https://api.workos.com
```

LANGUAGES

Client Libraries

WorkOS offers native SDKs in several popular programming languages. Choose one language below to see our API Reference in your application's language.

Don't see an SDK you need? Contact us to request an SDK!

HOW TO CHECK

Testing the API

You can test the API directly with cURL, or use the Postman collection for convenience.

Check out the **guide** about the WorkOS API Postman collection to learn more about it.

SDK docs initial designs

Overview

Accelerate adoption and improve your API devX with an always up-to-date, SDK-first docsite.

Key Features

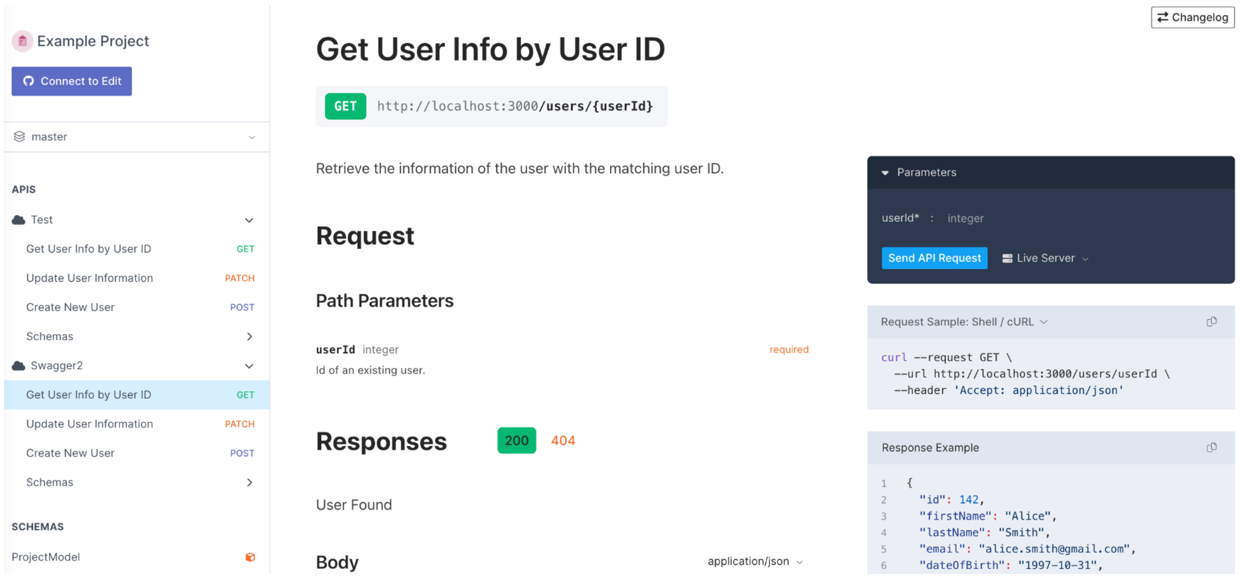
- **Always up to date**—docs pages are generated alongside the SDK itself, so your docsite always matches your SDKs
- **SDK-first**—rather than being just an API reference that has to be mapped to the particular SDK your user is integrating with, SDK-first docs simplify the integration process by matching the code users are writing one-to-one (more on this below)
- **Slick**—unlike other doc page generation tools (think Stoplight), our docs offering is polished and ready to present to the world out of the box (think WorkOS). We'll also include wow-factor features like AI search and personalized usage snippets (e.g. automatically including the user's API key in examples).
- **Other bonuses**—easily themeable with your own branding and supports OpenAPI 3.1 (unlike many OpenAPI docs tools)
- **AI-powered**—Speakeasy doesn't just *display* your docs—we actively improve them with AI. We use the full context of your spec plus the content of your website to improve descriptions, add examples, and even resolve validation errors and warnings.

Ways to Integrate

- **Hosted**—Speakeasy can generate and host a docsite for you as part of your existing SDK generation workflow. When your SDKs are updated, the docsite will update as well. The hosted URL can then be linked out from your primary docsite and/or your SDK's readme
- **Embeddable**—docs can also be generated as plain React in a standalone action. This enables you to generate docs in the same internal repo the rest of your docs already live in and host it however you want.
- **API reference only**—docs can also be generated without any SDKs. The resulting docs page would serve as a normal API reference.

What does "SDK-first" mean?

The docs you get from something like Swagger UI or the PAN Docusaurus plugin are documentation of your REST API itself. They talk about things in terms of HTTP calls. Notice in the screenshot below the references to HTTP methods (e.g. `GET`), path parameters, response codes, etc.



Get User Info by User ID

GET `http://localhost:3000/users/{userId}`

Retrieve the information of the user with the matching user ID.

Request

Path Parameters

`userId` integer required
Id of an existing user.

Responses 200 404

User Found

Body application/json

Parameters

userId* : integer

Send API Request | Live Server

Request Sample: Shell / cURL

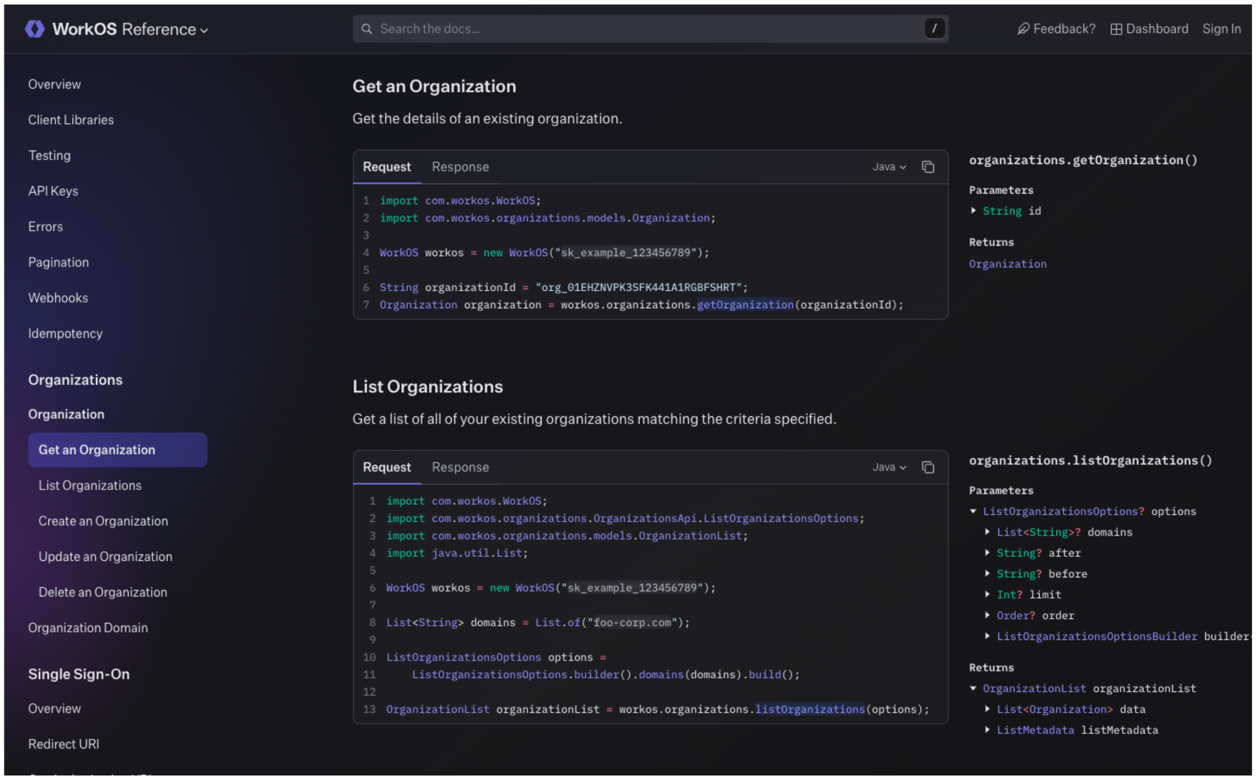
```
curl --request GET \
  --url http://localhost:3000/users/userId \
  --header 'Accept: application/json'
```

Response Example

```
{
  "id": 142,
  "firstName": "Alice",
  "lastName": "Smith",
  "email": "alice.smith@gmail.com",
  "dateOfBirth": "1997-10-31",
```

An example Stoplight API reference

SDK docs, in contrast, talk in terms of the specific language you are using. The fact that there is an API under the hood is abstracted away, simplifying the integration process by matching the code users are writing one-to-one. A side benefit of this is that the docs read closer to the product outcomes a user might want to achieve by not needing to cover so many HTTP-specific details. Notice in the screenshot below that everything is in terms of Java types, objects, and code. There's nothing that reveals that this is in fact calling out to a REST API under the hood. It reads like the documentation for a normal Java library.



Get an Organization

Get the details of an existing organization.

```java
import com.workos.WorkOS;
import com.workos.organizations.models.Organization;

WorkOS workos = new WorkOS("sk_example_123456789");

String organizationId = "org_01EHZNVPK3SFK441A1RGBFSHRT";
Organization organization = workos.organizations.getOrganization(organizationId);
```

```
organizations.getOrganization()

Parameters
▸ String id

Returns
Organization
```

List Organizations

Get a list of all of your existing organizations matching the criteria specified.

```java
import com.workos.WorkOS;
import com.workos.organizations.OrganizationsApi.ListOrganizationsOptions;
import com.workos.organizations.models.OrganizationList;
import java.util.List;

WorkOS workos = new WorkOS("sk_example_123456789");

List<String> domains = List.of("foo-corp.com");

ListOrganizationsOptions options =
    ListOrganizationsOptions.builder().domains(domains).build();

OrganizationList organizationList = workos.organizations.listOrganizations(options);
```

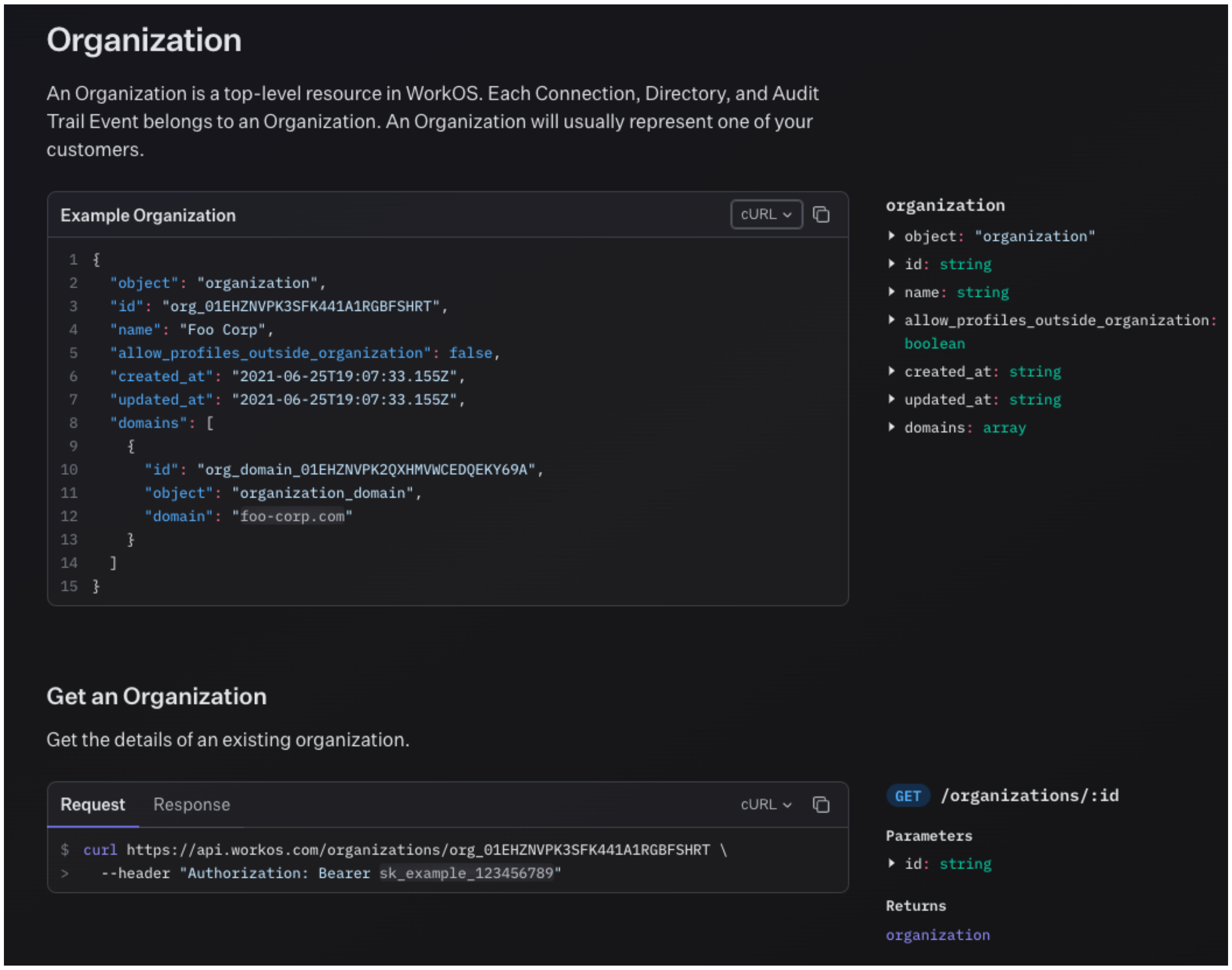
```
organizations.listOrganizations()

Parameters
▾ ListOrganizationsOptions? options
  ▸ List<String>? domains
  ▸ String? after
  ▸ String? before
  ▸ Int? limit
  ▸ Order? order
  ▸ ListOrganizationsOptionsBuilder builder

Returns
▾ OrganizationList organizationList
  ▸ List<Organization> data
  ▸ ListMetadata listMetadata
```

The WorkOS API / SDK reference docs

It's worth noting that SDK docs still serve as an API reference. See the below screenshot.


Organization

An Organization is a top-level resource in WorkOS. Each Connection, Directory, and Audit Trail Event belongs to an Organization. An Organization will usually represent one of your customers.

Example Organization

```
{
  "object": "organization",
  "id": "org_01EHZNVPK3SFK441A1RGBFSHRT",
  "name": "Foo Corp",
  "allow_profiles_outside_organization": false,
  "created_at": "2021-06-25T19:07:33.155Z",
  "updated_at": "2021-06-25T19:07:33.155Z",
  "domains": [
    {
      "id": "org_domain_01EHZNVPK2QXHMVWCEDQEKY69A",
      "object": "organization_domain",
      "domain": "foo-corp.com"
    }
  ]
}
```

```
organization
▸ object: "organization"
▸ id: string
▸ name: string
▸ allow_profiles_outside_organization:
  boolean
▸ created_at: string
▸ updated_at: string
▸ domains: array
```

Get an Organization

Get the details of an existing organization.

Request Response

```
$ curl https://api.workos.com/organizations/org_01EHZNVPK3SFK441A1RGBFSHRT \
>   --header "Authorization: Bearer sk_example_123456789"
```

```
GET /organizations/:id

Parameters
▸ id: string

Returns
organization
```

The WorkOS docs, with “cURL” selected as the language

AI Powered

Speakeasy offers powerful AI workflows that improve and maintain your OpenAPI spec for you. This ranges from simple things like suggesting better descriptions, examples, and operation IDs all the way up to actually solving validation errors and refactoring parts of your spec for you. This makes your spec better in the immediate term and reduces the maintenance burden as your API changes in the long term. This benefits your docs (and SDKs) directly, as the quality of both of those surfaces is determined by the quality of your OpenAPI spec.