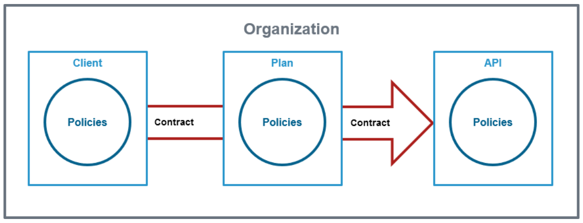
The Concepts of API Management

API Management uses a hierarchical data model that consists of five primary elements:

- **Organizations**
- **Plans**
- **APIs**
- **Clients**
- **Policies**

The main elements of API Management are **Clients**, **Plans** and **APIs**. All of them can contain **Policy** definitions. To be used, they need to be grouped by an **Organization** and related by a **Contract**.

Figure: Overview on the API Management Data Model



Element	Description
Organization	Almost everything in the API Management data model exists in the context of an organization: • An organization is a logical unit within API Management. This can be a company, department, etc. • An organization is a container of other elements: plans, APIs, and clients are defined per organization. • Every user must be associated with at least one organization to be able to manage elements in the application. • API Management implements role-based access control for users. You can give organization members different roles to restrict the actions he is able to perform and the elements he can manage within the organization. • Membership for each organization can be easily managed in the **Organization** tile. **Expert Advice** We recommend the following best practices regarding organizations: • Create organizations as fine-granular as possible, e.g. one organization for each logical group of APIs (purchase, order processing, billing). • Use a separate, dedicated organization for testing or development. • Do not test your API in an organization that holds productive data.
Client	A client is only required if you want to use a **private** API (refer to API Types: Public vs. Private). The client is the consumer of the API: • The client consumes managed APIs offered through API Management. • Each client can consume multiple APIs within API Management. The relation between client and API is defined via a contract and a plan. • As with an API or a plan, you can also add policies to a client. • When a client version is created, the system generates a unique API Key. All requests made to the API by a client through the gateway must include this API Key.

On this Page:

Related Pages:

- API Types: Public vs. Private
- APIs
- Clients
- Contracts
- Organizations
- Plans
- Policies
- Developer Portal

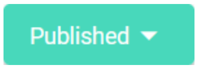
Related Documentation:

- Administration Guide
  - Managing Users
  - Managing Profiles

Plan	A plan is only required if you want to use a **private** API (refer to API Types: Public vs. Private). A plan is a set of policies that defines the level of service API Management provides for an API: • Plans enable users to define multiple different levels of service for their APIs. • Plans specify the contract between a client and an API. • It is common to define multiple plans with divergent configuration options for the same API. **Example:** An organization offers two plans for the same API: Plan A is more expensive than plan B, but it offers a higher level of API requests in a given (and configurable) period of time.
API	APIs in API Management represent real backend APIs (Application Programming Interfaces). An API is also known as a **service**, meaning anything that can be invoked remotely by some sort of client. API Management provides a way to turn unmanaged (raw) back-end APIs into **managed** APIs by attaching policies to them. Every managed API can be published as **public** API or **private** API or both (refer to API Types: Public vs. Private): • **Public** APIs are available to consumers without a key. Only policies defined on the API apply to public APIs. • **Private** APIs are only accessible for known consumers, called clients. Every client has an individual key to access the API. Policies defined on the client, the selected plan in the contract and the API apply. In API Management, users can create new APIs manually or easily import them from the PAS Administration.
Policy	Policies are at the lowest level of the data model, but they are the most important concept: A policy is a rule or a set of rules API Management uses to manage access to your APIs. • Policies are applied to all API requests and represent a unit of work applied at runtime to the request by API Management. • You can define a policy chain, a defined order in which the policies will be applied to API requests. **Expert Advice** We recommend the following best practices regarding policies: • Give a thought or two on where to add your policy, because policies can be added to clients, plans and APIs, which has impact on the policy chain. ○ On **API level**, you will typically use modification policies, such as **URL Rewriting** or **API Key**. ○ On **plan level**, you will typically use limiting policies, such as **Rate Limiting**. This way, each plan will allow for a different amount of requests. ○ On **client level**, you will typically apply authentication and authorization policies, such as **BASIC Authentication** or **Authorization**, or other security policies. • Testing APIs or verifying concepts with policies is much simpler with public APIs.
Contract	A contract is only required if you want to use a **private** API (refer to API Types: Public vs. Private). A contract relates a client to an API, using a plan.

Versioning

API Management supports versioning for APIs, plans and clients. All three elements share one behavior: They have to be determined to be available for use in the gateway.

- APIs must be 

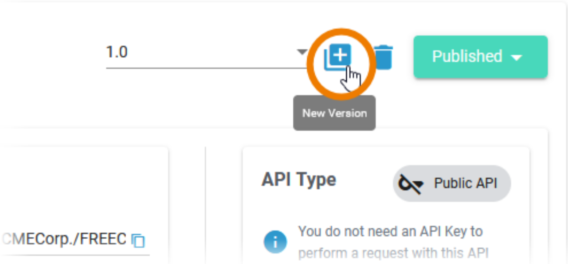
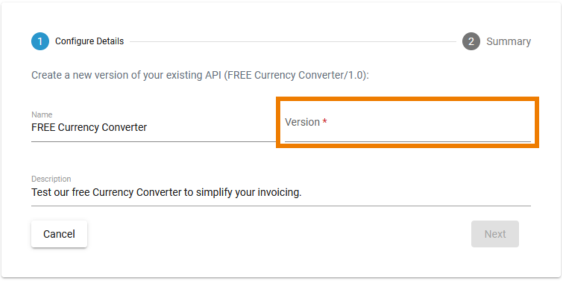
- Plans have to be Locked ▾
- Clients must be Registered ▾

> ⓘ While it is still possible to modify a published API and a registered client, a locked plan cannot be revised.

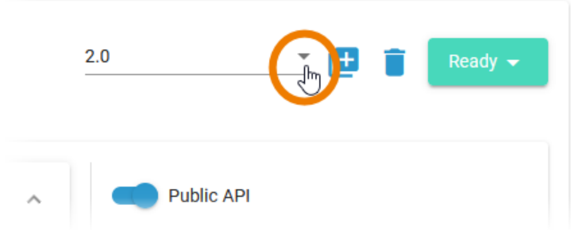
If modifications to the API Management configuration are necessary, you can do this by:

- Creating a new element.
- Modifying an existing element (only APIs and clients).
- Create a new version of an existing element.

Versioning allows you to modify the configuration of an existing element, but retain the previous version.

 1.0 New Version Published ▾ API Type Public API CMECorp./FREEC You do not need an API Key to perform a request with this API	To create a new version of an element, open its details page and click **New Version** ⊞ in the basic settings.
 Create a new version of your existing API (FREE Currency Converter/1.0): Name: FREE Currency Converter Version * Description: Test our free Currency Converter to simplify your invoicing.	In the version wizard, **Name** and **Description** cannot be changed. Use field **Version** to enter a new name oder version number. Click **Next** to access the summary. If you are satisfied with the changes, click **Create** to add the new version. It will be saved to the element. ⓘ You can enter nu

mbers and text in the **Version** field which allows the use of version numbers (e.g. 1.0, 2.1...) as well as version descr

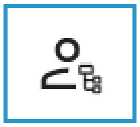
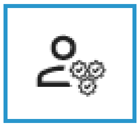
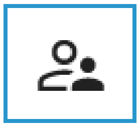
	iptions (e.g. Gold, Super etc.).
	Use the **Version** drop-down in the basic settings of the element to switch between the different versions.

Visibility

The visibility concept of API Management defines which user groups can find the APIs in the Developer Portal. The visibility resides on top of the permission system as another security layer. Visibilities are applicable to public APIs and plans for private APIs. The chosen visibility affects the content of the API Developer Portal from where API consumers can find the APIs. Relevant for the visibility is the identity management (IDM) group a user belongs to. The view in API Management itself is not affected by the chosen visibility.

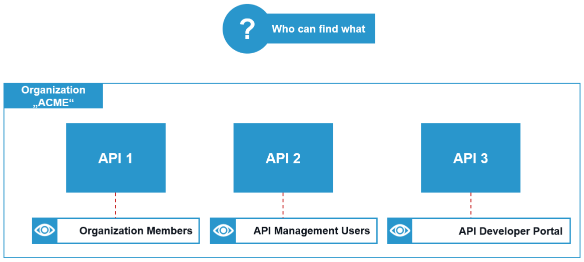
In API Management, a user can see all APIs for which he has explicit permissions (roles **Viewer** and **Editor**). The permissions are assigned in the corresponding organizations, refer to Administrating Organization Members > Applicable Roles.
In addition, a user can be assigned the profile **api_management_admin** in the user management (refer to Administration Guide) which makes him a "superadmin" who can basically see and do everything in API Management (refer to Administration for details).

You can choose between three different visibilities:

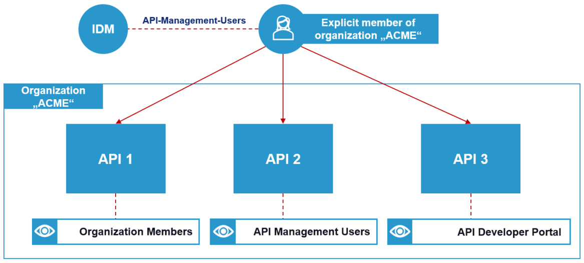
Icon in API Management	Visibility
	**Organization Members (default)**
	**API Management Users**
	**API Developer Portal**

Visibility Example

Three APIs have been created in organization ACME. Each API is assigned a different visibility.

 ? Who can find what Organization „ACME" API 1 API 2 API 3 Organization Members API Management Users API Developer Portal	- API 1: Organization Members - API 2: API Management Users - API 3: API Developer Portal Visitors

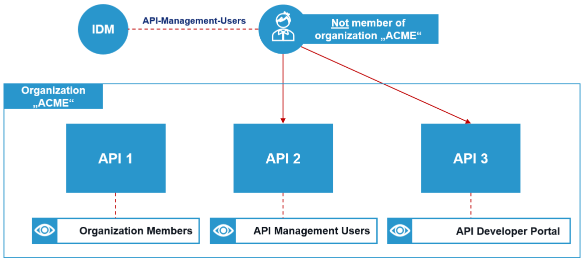
Organization Members (default)		

 IDM API-Management-Users Explicit member of organization „ACME“ Organization „ACME“ API 1 API 2 API 3 Organization Members API Management Users API Developer Portal		APIs are secured by default: If you create a new API, the default visibility setting is **Organization Members**. Only members

		of the organization the API has been created in are allowed to see it in the API Developer Portal.

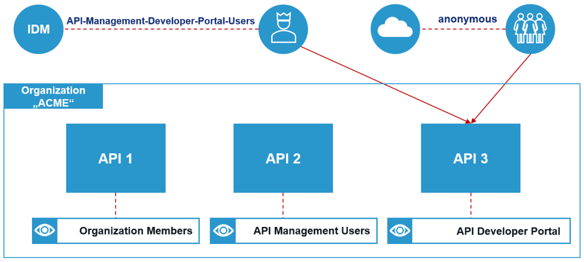
		Example: **API Management Users** who are explicit member of organization ACME can see API 1, API 2 and API 3.

API Management Users



		Select the visibility **API Management Users** to allow all API Management users to see your API in the API Developer Portal.

		Example: All **API Management Users** or **API Management Administrators** can see API 2 and API 3, they do not n

		eed to be member of organization ACME.
API Developer Portal		
 IDM API-Management-Developer-Portal-Users anonymous Organization „ACME“ API 1 API 2 API 3 Organization Members API Management Users API Developer Portal		Choose the visibility **API Developer Portal** to allow API Developer Portal use

rs and all (anonymous) portal visitors to see your API in the API Developer Portal.

Example: **API Management Develo**

		per Portal Users and anonymous portal visitors without a PAS login can see API 3 only.