Developer Portal User's Guide

Version 11.0

October 2023

WEBMETHODS

This document applies to webMethods Developer Portal 11.0 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: DPO-UG-110-20240708

Table of Contents

1 Overview

Why do Organizations Expose APIs?

Organizations often lack the resources to support mobile Bring Your Own Device (BYOD), supply chain, or eCommerce initiatives. By opening a set of APIs to external developers, organizations can reduce costs, expand the reach of their products or services, and create new channels of revenue in the following ways:

- Mobile application developers can create mashups and apps that satisfy a particular user niche and are optimized for specific mobile device types and platforms.
- Enterprise application developers can leverage APIs to simplify integration with suppliers and B2B partners.
- The involvement of external developers fosters innovation and collaboration throughout the development community. In return, the resulting developed applications offer the organization additional potential revenue as those applications reach new markets or customers in new ways.

Why do APIs Need to be Managed?

The APIs that an organization exposes contain core assets the organization would want to protect. As with the services they support, these APIs have a life cycle, need to be managed and governed, and require mediation and security at run time.

From an API provider's perspective, an API management tool is needed that enables the provider to do the following:

- Maintain an inventory of APIs and their associated resources.
- Publish, secure, and retire APIs according to defined service level agreements.
- Onboard API developers and give those developers the ability to publish APIs on behalf of the organization.
- Onboard API consumers who use the published APIs in their own applications.
- Provide tiered access to APIs, for example according to authorization level.
- Track key performance indicators (KPIs) to help monitor and interpret API use.

From an API consumer’s perspective, an API management tool should provide the ability to:

- Browse a catalog of APIs and obtain details and code samples for a specific API.
- Sign-up and request and manage access tokens to download an API and its associated resources and documentation.
- Test the functionality of an API.
- Collaborate with other API consumers by way of forums or integration with social media.

About webMethods Developer Portal

webMethods Developer Portal is a web-based, self-service portal that enables an organization to securely expose APIs to external developers, partners, and other consumers for use in building their own applications on their desired platforms.

Developer Portal provides the following features:

- **Branding and customization**. Administrators can customize their portal's logo, colors, and fonts to match their organization's corporate identity. Administrators can further customize their portal by modifying pages, incorporating widgets, and changing the appearance and organization of APIs, adding custom pages, components, and labels.
- **Support for three types of APIs** . Developer Portal supports traditional SOAP-based APIs, REST-based APIs, and OData APIs. This support enables organizations to leverage their current investments in different types of APIs.
- **Quick, secured provisioning of access tokens** . Approval workflows simplify the provisioning of applications. These workflows enable the API provider to individually approve access token requests that developers submit from Developer Portal. API key, OAuth2, and JWT credentials are supported as part of this feature.
- **Easy discovery and testing of APIs** . Full text search capabilities help developers quickly find APIs of interest. API descriptions and additional documentation, usage examples, and information about policies enforced at the API level provide more details to help developers decide whether to adopt a particular API. From there, developers can use the provided code samples and expected error and return codes to try out APIs they are interested in, directly from within Developer Portal, to see first-hand how the API works. APIs can be grouped and grouped based on various filters in the gallery for easier discovery. For example, APIs in a large catalog can be grouped by business domain, free versus paid, or public versus B2B partner. APIs can also be flagged based on maturity level (for example, beta versus production or release).
- **Quick, secure onboarding of new users** . Easy to configure approval workflows in webMethods Developer Portal graphical user interface to define how the user onboarding should take place, with or without confirmations.
- **Platform to collaborate**. Developer Portal provides a collaborative community environment where API consumers can rate APIs and contribute to open discussions with other developers.
- **Support for custom assets**. In addition to APIs and packages, you can publish assets of your choice to Developer Portal. This helps you publish all required references for your consumers.
- **Built-in usage analytics** . Developer Portal provides the Dashboard feature that the Developer Portal Administrator, API Providers, and API Consumers can access based on their roles to view Key Performance Indicators (KPIs) based on page views and API views by users, track total number of logins, the success and failure of logins, user registrations, and user audit log, study the API's invocations per user and its performance during runtime, study the API invocation trends by response time, success and failure rates, and track the total API requests over a period of time, requests over time per API, and API request log. This information helps

you understand how the APIs are being used, which in turn can help identify ways to improve users' portal web experience and increase API adoption.

- **Support for Localization**. Developer Portal supports localizing API information and description.

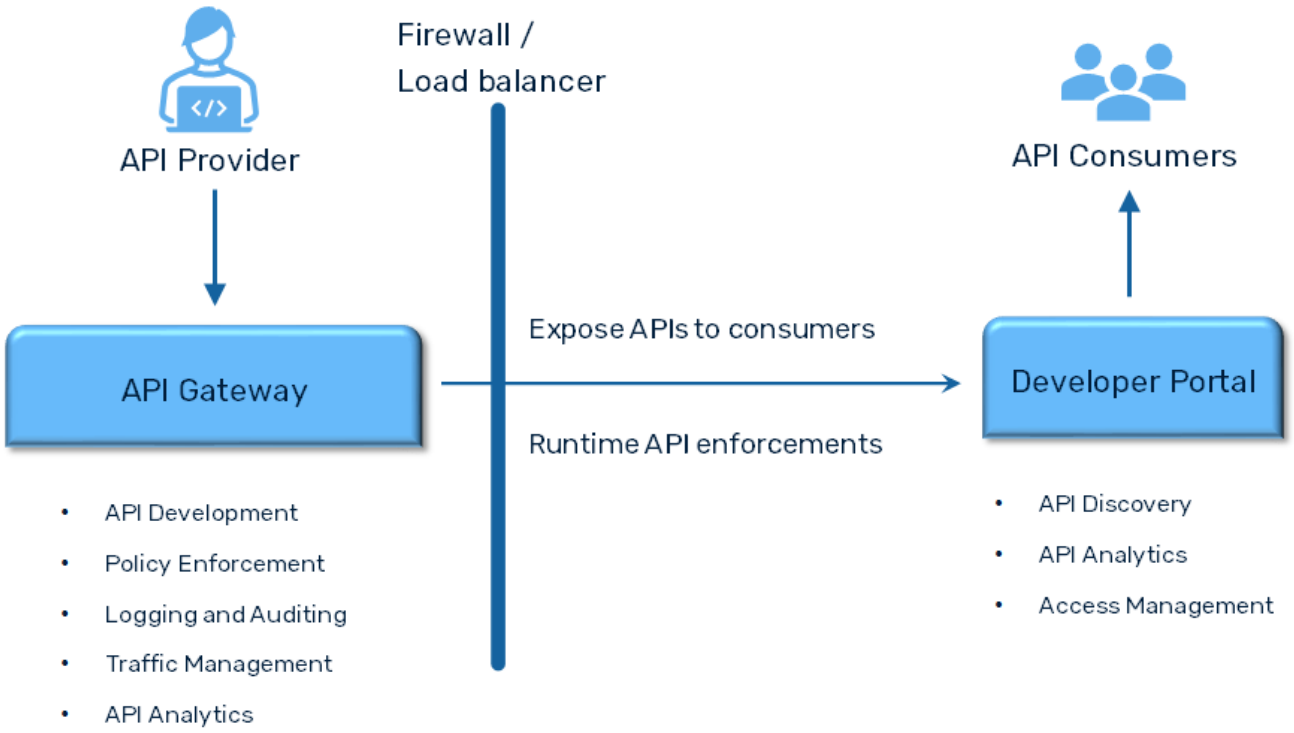
The webMethods API management suite products include the following:

- **webMethods Developer Portal**. In Developer Portal, API consumers browse the catalog of APIs that a provider has published. Consumers can sign up and request an access token to test the API.

  Users can view the API usage analytics data in the Developer Portal dashboard based on their privileges.

- **webMethods API Gateway**. API Gateway enables an organization to securely expose APIs to external developers, partners, and other consumers for use in building their own applications on their desired platforms. It provides a dedicated, web-based user interface to perform all the administration and API related tasks from the API creation, policy definition and activation, creation of applications, and API consumption. API Gateway gives you rich dashboard capabilities for API Analytics. APIs created in API Gateway can also be published to Developer Portal for external facing developers' consumption. API Gateway supports REST APIs, SOAP APIs, and WebSocket APIs, provides protection from malicious attacks, provides a complete run-time governance of APIs, and information about gateway-specific events and API-specific events.

The following diagram illustrates the communication flow between API Gateway and Developer Portal.



2 Administration

Overview

This section explains the options available to configure the Developer Portal settings. They include:

- SMTP configuration settings
- Password policy configuration
- User account and account security settings
- Email notification templates
- Events notification to external systems using webhooks
- Developer Portal URL configuration
- License management

How do I configure SMTP settings to send emails?

Developer Portal sends notifications over emails to its users as part of various functionalities such as user registration, application request, and so on. To enable Developer Portal to send email notifications, you have to register your SMTP server and set the sender's email address.

This use case starts when you want to configure SMTP settings and ends when you have completed the configuration.

To configure SMTP settings

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **SMTP**.
3. Provide the following details:

Field	Description
Host name	Host name or IP address of the SMTP server.
Port	Port number used by the SMTP server.
Sender address	Email address that must appear as sender address for all emails. This must be a valid email address.

4. Turn **Use SSL** on to enable SSL.
5. If you enable the SSL mode, perform the following:
   - Select a value from the **SSL mode** field that specifies the method to use for a secured connection.

Options available are:

- **STARTTLS**. Transforms a connection that was initially untrusted into an encrypted connection without requiring a specific port to secure communication.
- **SSL**. Establishes a trusted connection with a dedicated port.

- In the **Connection timeout** field, provide the duration, in milliseconds, after which the attempt to connect to the SMTP server is cancelled.

6. Turn **Use authentication** on to use authentication to the SMTP server. Provide the username and password used for authentication in the corresponding fields.
7. Click **Save**.

   Your changes are saved.

How do I configure password policy?

Password policy determines the conditions to be imposed on passwords specified by users.

This use case starts when you want to configure a password policy and ends when you have completed the configuration.

To configure password policy:

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Password policy** from the left pane.
3. In the **General** tab, provide the required values in the following fields:

   If the password specified by a user does not satisfy the requirements specified in this section, the password will not be accepted.

Fields	Description
Minimum length	Select the minimum length of the password.
Maximum length	Select the maximum length of the password.
Minimum number of lowercase letters	Select the minimum number of lowercase characters that must be provided.
Allow special characters	Select whether special characters are allowed.
Minimum number of special characters	Select the minimum number of special characters that must be provided.
Special characters	Provide the special characters that are allowed.
Allow uppercase letters	Select whether uppercase characters are allowed.

Fields	Description
Minimum number of uppercase letters	Select the minimum number of uppercase characters that must be provided.
Allow numbers	Select whether numbers are allowed.
Minimum number of numbers	Select the minimum number of digits that must be provided.
Allow commonly used password	Select whether commonly used passwords can be provided.
Common password (s)	Provide the list of common passwords that must not be allowed.
Allow sequential characters	Select whether sequential characters are allowed.
Minimum sequential characters	Select the minimum number of sequential characters that must be provided.
Allow repetitive characters	Select whether redundant characters are allowed.
Minimum repetitive characters	Select the minimum number of repetitive characters that must be provided.
Allow context-related password	Select whether context-related passwords are allowed.
Minimum context-related characters	Select the minimum number of context-related characters that must be provided.

4. In the **Advanced** tab, enable the following based on your requirements:

Field	Description
Force change before first login	Turn on to enforce the password change during their first sign in.
Force change after reset	Turn on to enforce the password change when user reset the password and new password was shared to user over email.
Force different password	Turn on to enforce user for a different password if the user provides a password that was already in use.
Activate reset confirmation	Turn on to send a confirmation email for password reset. If turned on, a link to reset password is sent. Else, the reset password is sent.
Activate password expiry	Turn on to specify the number of days after which a password expires.

Field	Description
	In the **Password lifetime (in days)** field, specify the number of days a password is valid.

5. Click **Save**.

   Your configurations are saved.

The set of password rules enabled here enhances the user account security by mandating users to employ strong passwords and use them properly.

Security Settings

You can configure the following security settings:

- Account lockout settings. For more information, see "How do I configure user account lockout settings?" on page 16.
- Multi-factor authentication settings. For more information, see "How do I configure multi-factor authentication settings?" on page 17.
- Advanced settings. For more information, see "How do I configure advanced security settings?" on page 18.

For information about hardening Developer Portal security using TLS and ciphers, see the **Setting up a Secure Transmission Protocol** section in the *webMethods Developer Portal Security Best Practices Guide*.

How do I configure user account lockout settings?

This use case explains the steps to configure the number of times that a user can attempt to sign in with an incorrect password.

Multiple attempts of users providing passwords to access their accounts would also mean that the user does not have the required authentication. Hence, to ensure safety, you can configure the number of attempts that a user can make to provide their password to log on to Developer Portal.

This use case starts when you want to configure account lockout settings and ends when you have completed the configuration.

To configure account lockout settings

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Security**.
3. In the **Account lockout** tab, enable **Lock users after failed login attempts** to specify whether a user account must be temporarily locked when there is certain number of failed logins.

4. Provide the following details:

Field	Description
Attempt limit	Number of failed log in attempts that are allowed before user account is locked.
Lockout duration (in seconds)	Number of seconds for which the user account remains locked. If the **Lock users after failed login attempts** setting is enabled and if there are too many failed sign in attempts, then the user account is locked for the specified number of seconds.
Lock counter duration (in seconds)	Number of seconds after which users can try to sign in after a failed sign in attempts is reset. Users must retry to sign in to Developer Portal after the duration specified in this field. For example, if you have entered 1800 in this field, users can retry to sign in after 30 minutes of their failed attempt.

5. Click **Save**.

   Your configurations are saved.

Next steps:

- The user accounts get locked if they exceed the number of attempts that you have configured.

How do I configure multi-factor authentication settings?

Multi-factor authentication enforces users to pass through an extra step, in addition to password entry, to sign in to their account. The additional step involves the entry of an OTP received over the registered email of users.

This use case starts when you want to configure multi-factor authentication and ends when you completed the configuration.

To configure multi-factor authentication settings

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Security**.
3. In the **Multi-factor authentication** tab, enable **Use multi-factor authentication** to specify whether multi-factor authentication is required.
4. Provide the values:

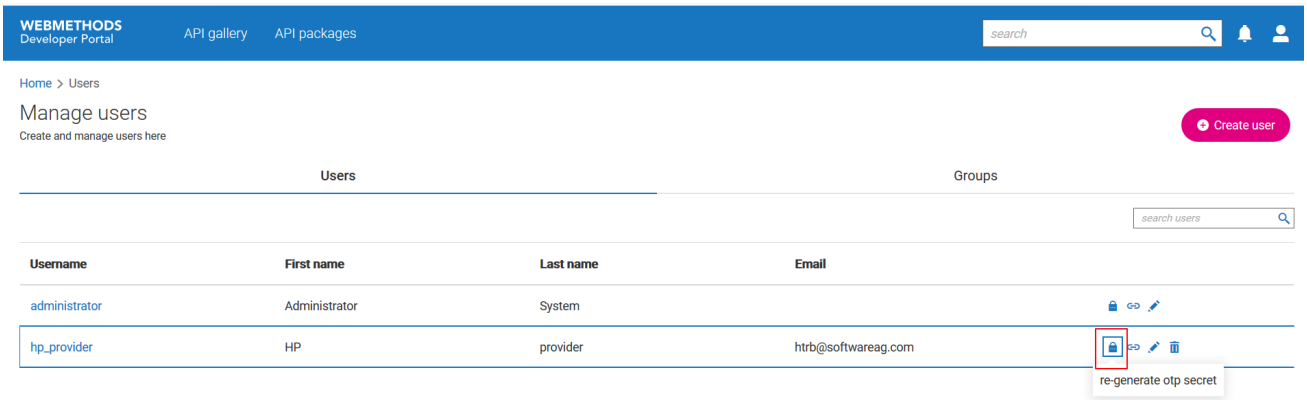
Field	Description
Clock skew intervals	Value based on which the validity of an OTP is calculated. An OTP is valid for the previous and current interval based on the specified value. For example, if you provide `1` in this field, the generated OTP will be valid for the 30 seconds prior to receipt of the OTP and 30 seconds from the receipt of the OTP.
Excluded users	List of user login names, separate by commas, for whom the multi-factor authentication is not required. For example, `administrator`.

5. Click **Save**.

   Your changes are saved.

Next steps:

- An OTP is sent to the user who tries to sign in through their registered email address and they can provide the OTP to sign in to the application. This step ensures that only the authenticated users have access to the application.
- Administrators can send an OTP secret token to users by clicking the generate OTP secret token icon from the **Manage users** page.



   If multi-factor authentication is enabled, the secret token is sent to the email of users who sign up to the application. If there are existing who onboarded when the multi-factor authentication was not enabled, you can send them the OTP secret token generator by clicking the generate OTP secret token icon.

How do I configure advanced security settings?

This use case explains the steps to enable logs related to user authentication transactions, generate user statistics, and configure OTP validity. You can view the logged events using the **Events log** screen or the using the **Events** REST API. For information on viewing from the **Events log** screen, see "How do I view audit log events?" on page 34; and for information on the **Events** REST API, see "Viewing Developer Portal events" on page 258.

This use case starts when you want to specify advanced security settings and ends when you have completed the configuration.

To configure advanced security settings

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Security** from the left pane and click **Advanced**.
3. From the following list, enable the list of data that you want to log:

Field	Description
Log authentication	Enables the logging of user authentication events.
Log changes to configuration	Enables the logging of changes made to configurations.
Log changes to licenses/privileges	Enables the logging of changes made to licenses and privileges.
Log changes to users/user groups	Enables the logging of changes made to users and user group details.
Generate user statistics	Generate the user statistics data.
User statistics in backup	Includes user statistics data in the backup files. This data is included if you select the **User** module when you create the backup. For information on creating backup, see "How do I take a backup?" on page 43.
Use OTPs	Enables OTP generation as a part of user multi-factor authentication.

4. *Optional*. If you have enabled the **Use OTPs** setting, provide the number of seconds that an OTP must be valid in the **OTP Lifetime** field.
5. Click **Save**.

   Your changes are saved.

The logs enabled using the settings are used to monitor activities related to user accounts.

How do I configure user session settings?

You can configure the user session settings from the **Users** page of the **Administration** section.

This use case begins when you want to configure user session settings and ends when you have saved the configuration.

To configure user session settings

1. Click the menu options icon  from the title bar and click **Administration**.
2. Select **Users**.
3. Turn **Email address required** on to specify that the entry of users' email address is mandatory during sign-up and sign-in.
4. Turn **Validate email address** on to specify that the email address is entered by users must be validated.

   The email address validation is performed by sending an email to the registered email address.
5. Specify required values in the following fields:

Field	Description
Maximum length of login name	Provide the maximum number of characters allowed for user login name.
Maximum image size (in bytes)	Provide the maximum size of user image that can be uploaded.
Initial session duration (in minutes)	Provide the duration, in minutes, of the initial session of users.
Maximum session duration (in minutes)	Provide the maximum duration of user sessions.

6. Select the **Default group name** from the list.

   New users are assigned to the selected group by default.

How do I configure email notification templates?

Developer Portal sends email notifications to users on various events such as email verification during sign up, OTP email to users, or application approval notification and so on. The default email templates are used for such notifications. You can edit these notification messages if required. For the list of email notification templates, see "List of email notification templates available in Developer Portal" on page 21.

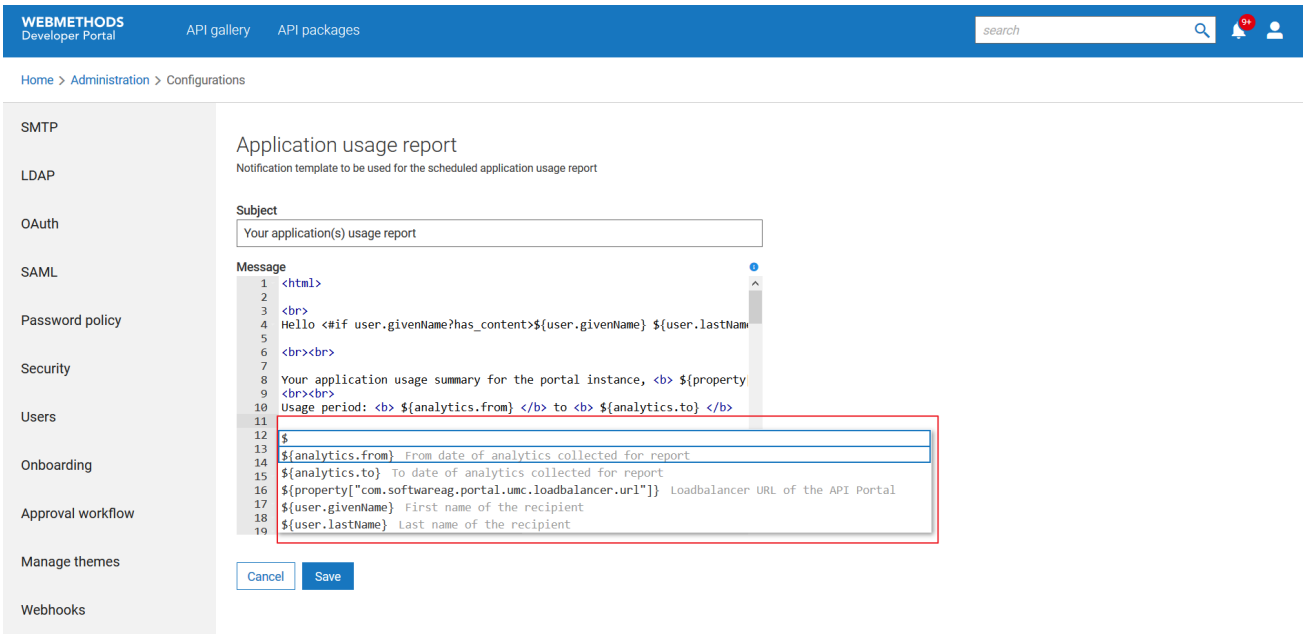
This use case starts when you want to edit an email template and ends when you have completed the edit.

To configure email notification templates

1. Click the menu options icon  from the title bar and click **Administration**.

2. Click **Email templates** from the left pane.
3. Click the edit icon  next to a template.
4. Make the required changes in the subject and body of the email notification.
5. Use predefined variables to formulate messages in a meaningful way.

   To view the available variables, type `$`. The list of available variables appears.



   Click the required option to insert it in the email template.

6. Click **Save**.

   Your changes are saved.

List of email notification templates available in Developer Portal

Here is the list of email notification templates available in Developer Portal and their intended recipients:

Name	Description	Intended recipients
Application usage report	Sent to users as per their configuration in the **Usage preference settings** section. The frequency can be `Daily`, `Weekly` or `Monthly`.	Users that have configured to receive the report from the **Usage preference settings** section.
API republish	Sent when an API is republished.	API followers.
API unpublish	Sent when an API is unpublished.	API followers.

Name	Description	Intended recipients
Application access share	Sent when an application is shared.	Application owner and users with whom the application is shared.
Application publish	Sent when an application is published.	Application owner and users with whom the application is shared.
Application access unshare	Sent when an application is unshared.	Application owner and users with whom the application is shared.
Application scope decrease	Sent an application scope is decreased by removing an API from the application.	Application owner and users with whom the application is shared.
Application scope increase	Sent when an application scope is increased by adding an API to the application.	Application owner and users with whom the application is shared.
Approval pending	Sent when there is a new approval that is pending for review.	Users that are configured as approvers.
Approval result	Sent to notify on approval stage result	Users that made the request and the approvers. In case of User onboarding approval process, the signed up user is the requester and in Application or Subscription creation approvals, the user who initiates the application/subscription creation is the requester.
Comment created	Sent when there is a new comment posted to a topic stream.	API and Package followers.
Email verification	Sent for email verification.	Users that are signing up to Developer Portal.
License expired	Sent to notify about Developer Portal license expiry.	Administrators.
License expires soon	Sent to notify about Developer Portal license expiry.	Administrators.
OTP request	Sent when a user requests for an OTP.	User that has requested for an OTP.
OTP secret change	Sent during the OTP secret changes.	User that has requested for an OTP secret change.
Package republish	Sent when a package is republished.	Package followers.

Name	Description	Intended recipients
Package unpublish	Sent when a package is unpublished.	Package followers.
Password changed	Sent to notify the password change.	User that has requested for a password change.
Password reset	Sent to notify the password reset.	User that has requested for a password reset.
Password reset requested	Sent to notify the password reset request.	User that has requested for a password reset.
Program registration	Sent to confirm the registration for an API program.	User that has registered for an API program.
Program unregistration	Sent to confirm the unregistration from an API program.	User that has unregistered from an API program.
SMTP validation email	Sent to validate SMTP connection.	Recipient provided during SMTP configuration.
Ticket created	Sent to confirm a ticket creation.	User that has created a ticket in an API program.
Ticket status change	Sent to notify the change in a ticket status.	User that has created the ticket.
Ticket type change	Sent to notify the change in a ticket type.	User that has created the ticket.
Topic created	Sent when is a new topic is posted for an API.	API and package followers.
Welcome note email	Sent when a new user is onboarded.	User that has newly onboarded.

How do I configure webhooks to notify events to an external system?

Webhooks are used by applications to provide real-time information to other applications.

You can create webhooks in Developer Portal to notify the specified events to an external application URL. For example, an API is published, or an application is shared. For the list of events that you can notify, see "List of events" on page 24.

This use case starts when you want to configure a webhook and ends when you have configured one.

To configure webhooks

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Webhooks** from the left pane.
3. Click **Create**.
4. Provide the **URL** of the destination system to which the notification has to be sent.
5. From the **Type** list, select the type of destination that you want to send notification:
   - **System**. To notify an external system endpoint.
   - **Provider**. To notify a provider endpoint.
6. Select one of the following from the **Security** field:
   - **Basic**. Select this option if the destination requires basic authentication, and provide the corresponding user name and password.
   - **None**. Select this option if the destination requires no authentication.
7. Select the required event type.

   You can select more than one event type.
8. Click **Save**.

   Your changes are saved.

The webhook is added. Notifications for the selected events are triggered and sent to the specified endpoint.

List of events

The following table lists the events for which you can create webbooks:

Events	Description
API published	Triggered when an API is published to the system.
API republished	Triggered when an API is republished to the system.
API unpublished	Triggered when an API is unpublished from the system.
Application granted	Triggered when application access is granted to a user.
Application publish	Triggered when an application is published.
Application request approval	Triggered when an application request is approved.

Events	Description
Application request approval pending	Triggered when an application request is pending for approval.
Application request rejected	Triggered when an application request is rejected.
Application revoked	Triggered when application access is revoked for a user.
Application scope change	Triggered when APIs are added or removed from an application.
Application unpublish	Triggered when an application is unpublished.
Comment create	Triggered when a new comment is posted for a topic.
Comment delete	Triggered when a comment is deleted.
Comment update	Triggered when a comment is updated.
Community create	Triggered when a community is created.
Community delete	Triggered when a community is deleted.
Community membership change	Triggered when a community membership is changed when a new member is added or when an existing member is removed.
Community scope change	Triggered when a community scope is modified.
Cron execution	Triggered when a cron execution is complete.
Email notification	Triggered when an email notification is sent.
External verification	Triggered when the external approval option is enabled as a part of user or application onboarding strategy.
Flag topic or comment	Triggered when a topic or comment is flagged.
Gateway application creation	Triggered when a request to create an application is made to the gateway.
Gateway application scope decrease	Triggered when a request to decrease the scope of an application is made to the gateway.
Gateway application scope increase	Triggered when a request to increase the scope of an application is made to the gateway.
Gateway application update	Triggered when a request to update an application is made to the gateway.
Invoke Try API option	Triggered when an API is invoked from the Try API page.
Package publish	Triggered when a package is published.

Events	Description
Package republish	Triggered when a package is republished.
Package unpublish	Triggered when a package is unpublished.
Plan publish	Triggered when a plan is published.
Plan republish	Triggered when a plan is republished.
Portal application creation	Triggered when a request is made to create an application.
Portal application deletion	Triggered when a request is made to delete an application.
Portal application scope decrease	Triggered when a request is made to decrease the scope of an application.
Portal application scope increase	Triggered when a request is made to increase the scope of an application.
Portal application update	Triggered when a request is made to update an application.
Program access granted	Triggered when an API program is assigned to another user.
Program access revoked	Triggered when the access to an API program is revoked.
Program created	Triggered when an API program is created.
Program deleted	Triggered when an API program is deleted.
Program registered	Triggered when an API program is registered.
Program unregistered	Triggered when an API program is unregistered.
Program updated	Triggered when an API program is updated.
Provider publish	Triggered when a provider is published.
Provider republish	Triggered when a provider is republished.
Provider scope change	Triggered when adding or removing APIs from a provider.
Provider unpublish	Triggered when a provider is unpublished.
System backup event	Triggered when a backup file is created.
Team delete	Triggered when a team is deleted.
Team expansion	Triggered when new members are added to a team.
Team shrink	Triggered when a user is removed from the team.
Ticket created	Triggered when a ticket is created in an API program.
Ticket status changed	Triggered when a ticket status is changed.

Events	Description
Ticket type changed	Triggered when a ticket type is changed.
Ticket updated	Triggered when a ticket is updated.
Topic create	Triggered when a new topic is posted in a stream.
Topic delete	Triggered when a topic is deleted from a stream.
Topic update	Triggered when a topic is updated in a stream.
User invited	Triggered when a user is invited to sign up.
User request delete	Triggered when a user request is to be deleted.
User request retry	Triggered when an existing application request is retried.
User signup	Triggered when a user signs up.

How do I configure webhooks to notify user sign-up and application requests to an external approval system?

You can use webhooks to send user sign-up requests, application requests to any external approval system.

This use case starts when you want to configure an external approval onboarding strategy and ends when you have completed the configuration.

To configure webhooks

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Webhooks** from the left pane and click **Create**.
3. Provide the external approval system endpoint **URL** in the field.
4. Select **System** from the **Type** list.
5. Provide the required **Security** preference.

   Available options are:

   - **Basic**. Indicates the basic credentials are required. Provide your user name and password.
   - **None**. Indicates that no authentication is required.
6. Select **External account verification initiated** from the **Event type** list.
7. Click **Save**.

   Your changes are saved.

The webhook is added. Notifications for the selected events are triggered and sent to the specified endpoint.

Next steps - Configuring external approval for user sign-up requests:

- When there is a user sign-up request, the user details are sent to the configured URL.

  Sample request

```
{
  "created" : "2022-02-09T13:37+0000",
  "documentType" : "EVENTS",
  "parameters" : {
    "details" : {
      "email" : "johnsmith@gmail.com",
    },
    "source" : "ExternalVerificationExecutor",
    "link_id" : "8688ae09-243b-4589-bea1-48cb53d9e702"
  },
  "type" : "EXTERNAL_VERIFICATION_EVENT"
}
```

  The **parameters.details** section in the payload has the newly signed up user information. Approvers check this data to approve or reject a sign-up request. By default, the **Sign up** page displays the **Email** and **Password** fields. If you add other fields, the data entered by users is included in the **parameters.details** section of the payload.

- The external system processes the new user sign-up request, and approves or rejects the request. The external system must specify the **link_id** value received through the payload and approve or reject the requests using the following REST resources:

  - PUT `/rest/v1/approvals/request/external/link_id/approve`. Approves the specified external approval request.
  - PUT `/rest/v1/approvals/request/external/link_id/approve?comments=`comments. Approves the specified external approval request with comments.
  - PUT `/rest/v1/approvals/request/external/link_id/reject`. Rejects the specified external approval request.
  - PUT `/rest/v1/approvals/request/external/link_id/reject?comments=`comments. Rejects the specified external approval request with comments.

Next steps - Configuring external approval for application creation or application scope increase requests:

- When there is an application creation request, the application details are sent to the configured URL.

  Sample request

```
{
  "created" : "2022-02-10T06:15+0000",
  "documentType" : "EVENTS",
  "parameters" : {
    "details" : {
```

```
      "user_request_id" : "25fb94d9-6d30-4515-aa14-65177484946b",
      "user" : "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
      "application_id" : "c02532b9-fd99-488f-98bb-b9c85fd081db"
    },
    "source" : "ExternalVerificationExecutor",
    "link_id" : "ff4e6597-1a86-4997-832a-38544d66632f"
  },
  "type" : "EXTERNAL_VERIFICATION_EVENT"
}
```

As a system administrator, you can get additional details about the request using the following REST call with required request Id:

```
GET  /rest/v1/requests/requestId
```

The **parameters.details.user_request_id** section in the payload has the request Id. The sample response of the payload with additional details:

```
{
    "owner": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
    "id": "25fb94d9-6d30-4515-aa14-65177484946b",
    "documentType": "USER_REQUEST",
    "type": "APPLICATION_CREATION_REQUEST",
    "status": "APPROVAL_PENDING",
    "context": {
        "apis": [
            "934d1e01-4dca-11ec-43ed-8eb69da50747"
        ],
        "custom": {},
        "name": "MyApp",
        "description": null,
        "redirect_uris": [
            "https://host.com/rest/v1/oauth/callback"
        ],
        "subscription": "false",
        "externalrefkey": "656ef5dc-53dd-4b33-8407-42da59781382",
        "user": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
        "tenant": "default",
        "provider_ref": "13b6dac1-9ed7-11eb-5403-2692ff6904c7"
    },
    "application": "c02532b9-fd99-488f-98bb-b9c85fd081db",
    "state": {
        "4cf77aca-d721-40b1-bc27-0cb0b7bd0a6d": "APPROVAL_PENDING"
    }
}
```

- View the details of the API associated with an application by making a REST call to the following endpoint with the required API Id:

  ```
  GET  /rest/v1/apis/id
  ```

- View the details of the user who created an application by making a REST call to the following endpoint with the required user Id:

  ```
  GET  /rest/v1/users/id
  ```

- View the details of an audit event by making a REST call to the following endpoint with the required event Id:

```
GET  /rest/v1/events/id
```

- The external system processes the application request, and approves or rejects the request. The external system must specify the **link_id** value received through the payload and approve or reject the requests using the following REST resources:
  - PUT `/rest/v1/approvals/request/external/link_id/approve`. Approves the specified external approval request.
  - PUT `/rest/v1/approvals/request/external/link_id/approve?comments=`comments. Approves the specified external approval request with comments.
  - PUT `/rest/v1/approvals/request/external/link_id/reject`. Rejects the specified external approval request.
  - PUT `/rest/v1/approvals/request/external/link_id/reject?comments=`comments. Rejects the specified external approval request with comments.

Next steps - Configuring external approval for package subscription requests

- When there is a package subscription request, the subscription details are sent to the configured URL.

  Sample request

```
{
  "created" : "2022-02-10T06:15+0000",
  "documentType" : "EVENTS",
  "parameters" : {
    "details" : {
      "user_request_id" : "7022738e-8367-48bf-a60a-6015409a129a",
      "user" : "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
      "application_id" : "45f6073c-4e5c-49d2-80cb-13218f228014"
    },
    "source" : "ExternalVerificationExecutor",
    "link_id" : "ff4e6597-1a86-4997-832a-38544d66632f"
  },
  "type" : "EXTERNAL_VERIFICATION_EVENT"
}
```

  As a system administrator, you can get additional details about the request using the following REST call with required request Id:

```
GET  /rest/v1/requests/requestId
```

  The **parameters.details.user_request_id** section in the payload has the request Id. The sample response of the payload with additional details:

```
{
    "owner": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
    "id": "7022738e-8367-48bf-a60a-6015409a129a",
    "documentType": "USER_REQUEST",
    "type": "SUBSCRIPTION_CREATION_REQUEST",
    "status": "APPROVAL_PENDING",
    "context": {
        "package": "dc43a88a-7de5-4d88-960d-912f8e82b44d",
        "custom": {},
```

```
        "name": "PackageSubscription",
        "description": null,
        "redirect_uris": [
            "http://hostname/portal/rest/v1/oauth/callback"
        ],
        "subscription": "true",
        "plan": "4b780d62-dc26-44cb-81d3-02e86204c3db",
        "user": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
        "tenant": "default",
        "provider_ref": "01719942-6ed5-4f6f-a000-1f4807e8b711"
    },
    "application": "45f6073c-4e5c-49d2-80cb-13218f228014",
    "state": {
        "4956cf28-b8c3-4a4f-969d-1a8baa1754ef": "APPROVAL_PENDING"
    }
}
```

- View the details of the subscribed package by making a REST call to the following endpoint with the required package Id:

  ```
  GET  /rest/v1/packages/id
  ```

- View the details of the subscribed plan by making a REST call to the following endpoint with the required plan Id:

  ```
  GET  /rest/v1/plans/id
  ```

- View the details of an audit event by making a REST call to the following endpoint with the required event Id:

  ```
  GET  /rest/v1/events/id
  ```

- View the details of the user who created a subscription by making a REST call to the following endpoint with the required user Id:

  ```
  GET  /rest/v1/users/id
  ```

- The external system processes the subscription request, and approves or rejects the request. The external system must specify the **link_id** value received through the payload and approve or reject the requests using the following REST resources:

  - PUT `/rest/v1/approvals/request/external/link_id/approve`. Approves the specified external approval request.
  - PUT `/rest/v1/approvals/request/external/link_id/approve?comments=`comments. Approves the specified external approval request with comments.
  - PUT `/rest/v1/approvals/request/external/link_id/reject`. Rejects the specified external approval request.
  - PUT `/rest/v1/approvals/request/external/link_id/reject?comments=`comments. Rejects the specified external approval request with comments.

How do I specify the Developer Portal URL for reference from external systems?

The **Load balancer URL** field allows you to provide the Developer Portal URL. The URL provided in this field is used in the emails sent from Developer Portal. Users who receive those emails can click the link to access Developer Portal.

The Load Balancer URL should be configured to receive email notifications with proper resolvable URL.

This use case starts when you want to modify the Developer Portal URL and ends when you have saved the configuration.

To specify the Developer Portal URL

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **General** from the left pane.
3. In the **Load balancer URL** field, specify the Developer Portal URL with the corresponding system name or IP address.
4. Click **Save**.

   Your configurations are saved.

The specified URL is provided to external systems for directing from anywhere to Developer Portal.

How do I specify Cross-Origin Resource Sharing (CORS) URLs?

The **CORS origin** setting allows you to provide the required Cross-Origin Resource Sharing (CORS) origin URLs. Developer Portal allows secure cross-domain requests and data transfers with the specified URLs.

This setting is mandatory when you need to access the Developer Portal resources from a different host or server.

For example, when you build an UI application for Developer Portal with Developer Portal as a headless server using the REST endpoints exposed by Developer Portal, you can deploy the UI application in a server or port that is different from that of Developer Portal.

Consider the following scenarios for this example:

- **Scenario 1**. UI and back-end hosted on same server but different ports.
  - **UI application** - *http://localhost:7777*
  - **Developer Portal** - *http://localhost:18101/portal*
- **Scenario 2**. UI and back-end hosted in different servers.

- **UI application** - *http://dev-portal-ui:7777*
- **Developer Portal** - *http://dev-portal-server:18101/portal*

In this example, the UI application URL has to be configured as a CORS origin in Developer Portal, that acts as the back-end server.

To specify CORS origin URLs

1. Click the menu options icon from the title bar and click **Administration**.
2. Click **General** from the left pane.
3. In the **CORS origins** field, specify the required URLs separated with commas.
4. Click **Save**.

   Your configurations are saved.

Developer Portal enables resource-sharing with the specified origins.

How do I update the Developer Portal license?

This use case explains the steps to upload a license file if you are using an outdated one.

This use case starts when you want to update your existing license and end when you have successfully uploaded the same.

To update the license file

1. Click the menu options icon from the title bar and click **Administration**.
2. Click **Licenses** from the left pane.
3. Click **Browse file** and select the required license file.
4. Click **Save**.

   Your license is updated.

How do I configure the default group and community for a new user?

This use case starts when you want to specify the group and community that must be assigned to a new user by default and ends when you have completed the configuration.

To specify the default group and community

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Onboarding strategy**.
3. Select the default community from the **Default community for onboarding users** list.

   You can select more than one community.
4. Click **Save**.

   Your changes are saved.
5. Click **Users**.
6. Select the default group from the **Default group name** from the list.
7. Click **Save**.

   Your changes are saved. New users are assigned to the default team and community.

How do I view audit log events?

The **Events log** screen lists:

- **Audit events**. Audit log events related to various assets such as APIs, applications, packages, plans, communities, and providers. It also includes topic and comment related events.
- **User events**. Events that Developer Portal creates based on your settings on the **Advanced** tab of the **Security** screen. For the list of log events that you can configure, see "How do I configure advanced security settings?" on page 18.

To view audit events

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Events log** from the left pane.

   The logged events list appears. You can filter the required events by typing the required event name in the text box.

   You can also select the type of events to view, and period for which the events must be displayed.

Configuring Default Language

As an administrator, you can select a default language for Developer Portal. When a guest user accesses Developer Portal, the application appears in the default language.

Users can sign in and set a language that they prefer to view; and the application appears in the language specified by users whenever they sign in.

As an administrator, you can also provide an option to select language by adding a language switcher icon. When added, this icon appears in the landing page. For more information, see "How do I add the language switcher icon?" on page 93.

Note:
If you do not select any default language in user profile, then the application appears in the default language selected by administrator. You can specify your preferred language or continue to view in the language configured by administrator.

To configure default language:

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **General**.
3. Select the required language from the **Default system language** field.
4. Click **Save**.

Configuring Personal Profile Settings

Users can specify their preferred language for Developer Portal.

When launched, the application appears in the default language configured by the administrator. Once users sign in to the application, the UI switches to the language that is configured using the steps provided in this section.

To configure personal profile settings:

1. Click the menu options icon  from the title bar and click your user name or email address to edit your profile.
2. Select **Account settings**.
3. Select the required language from the **Default system language** field.
4. Click **Save**.

   The selected language is saved.

Configuring Payment Gateway

Developer Portal supports direct monetization of APIs by allowing you to configure your payment gateway account.

As an administrator, you can enable the payment gateway feature in Developer Portal and add your account details. This, in turn, enables API consumers to provide the details of their credit card from which they would make the payment for plan subscriptions that they use.

Developer Portal supports Stripe payment gateway.

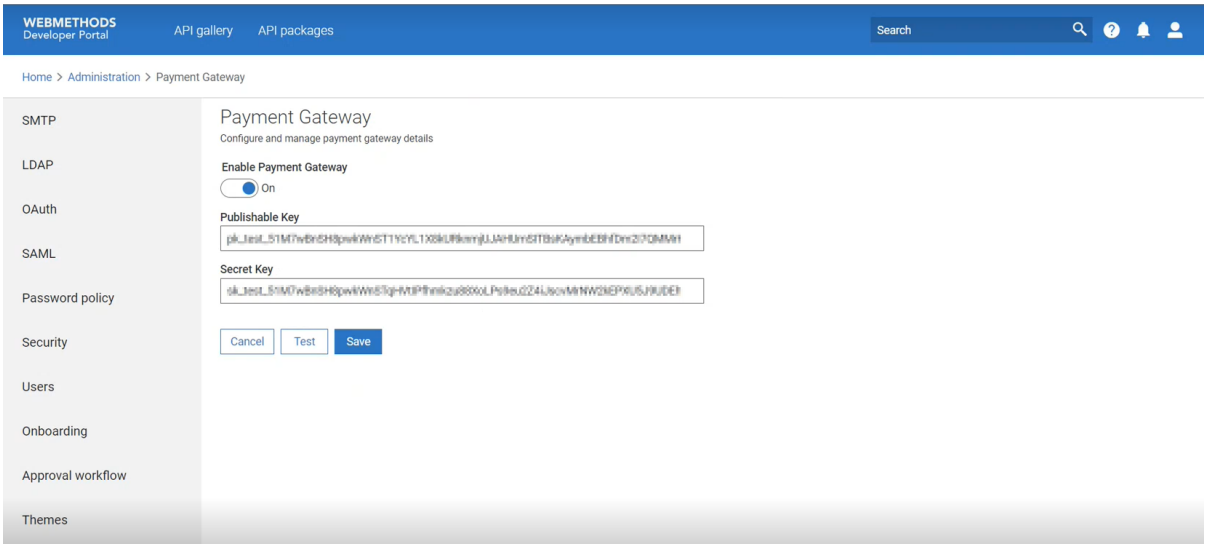
This section provides details about how to configure the payment gateway to collect credit card information using the Stripe payment gateway.

Pre-requisites

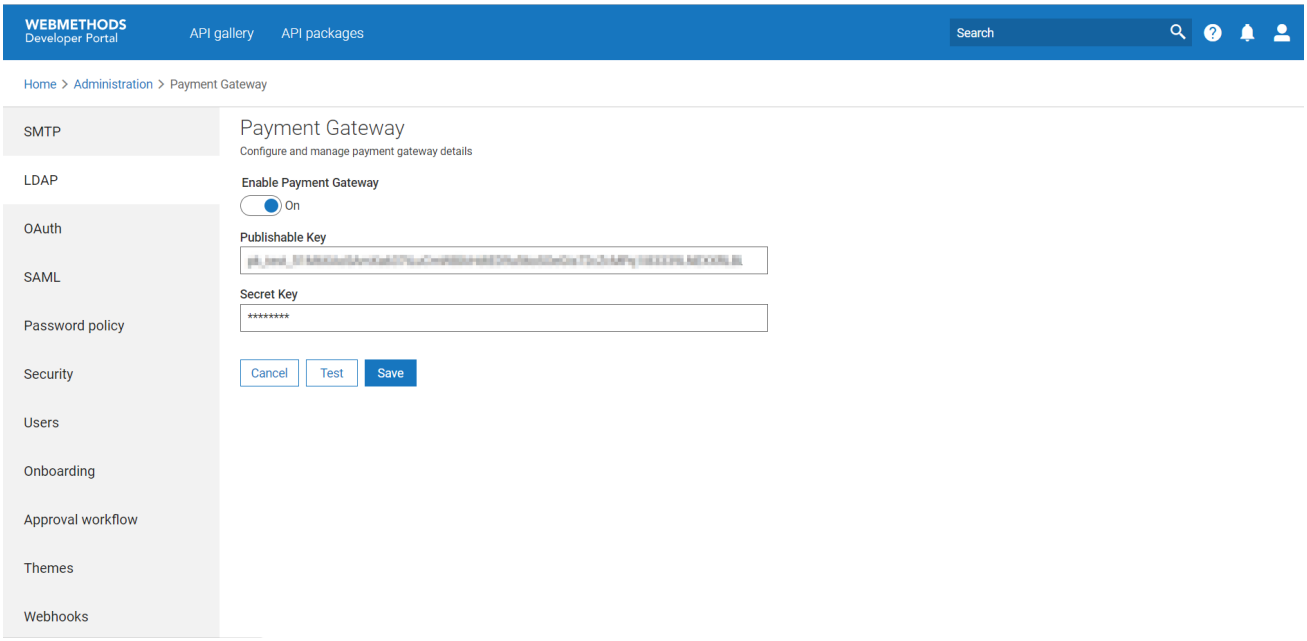
- Ensure you have an active Stripe account.
- Note down the following values from your Stripe account:
  - Publishable key
  - Secret key

  For information about retrieving the above values, see https://stripe.com/docs/keys.

1. Click the menu options icon from the title bar and click **Administration**.
2. Click **Payment gateway**.
3. Turn the **Enable payment gateway** slider on.
4. Provide the **Publishable key** and **Secret key** in respective fields.



5. [*Optional*] Click **Test** to validate the provided values.
6. Click **Save**.



System validates the provided values. The payment gateway feature is enabled.

Next steps

Once you enable Payment gateway integration, API consumers must provide their payment details for subscribing to any plan. Consumers can add their credit card details from the **Payment details** section from their user profile page.

Adding Payment Details (for consumers)

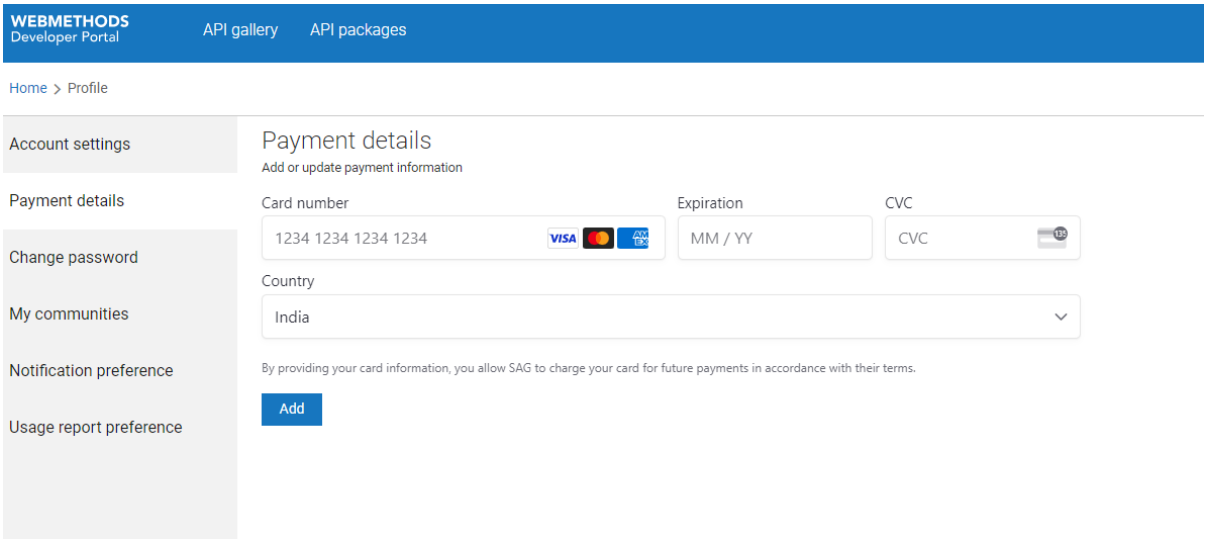
You, as an API administrator, can collect the details of API consumers' credit cards for receiving payments for the plan subscriptions that they use.

API consumers can provide the details of their credit card that they want to use for payments in Developer Portal by following the given steps.

Pre-requisites

- Ensure that an valid payment gateway account is configured.

1. Click the menu options icon from the title bar and click your user name or email address to edit your profile.
2. Select **Payment details**.

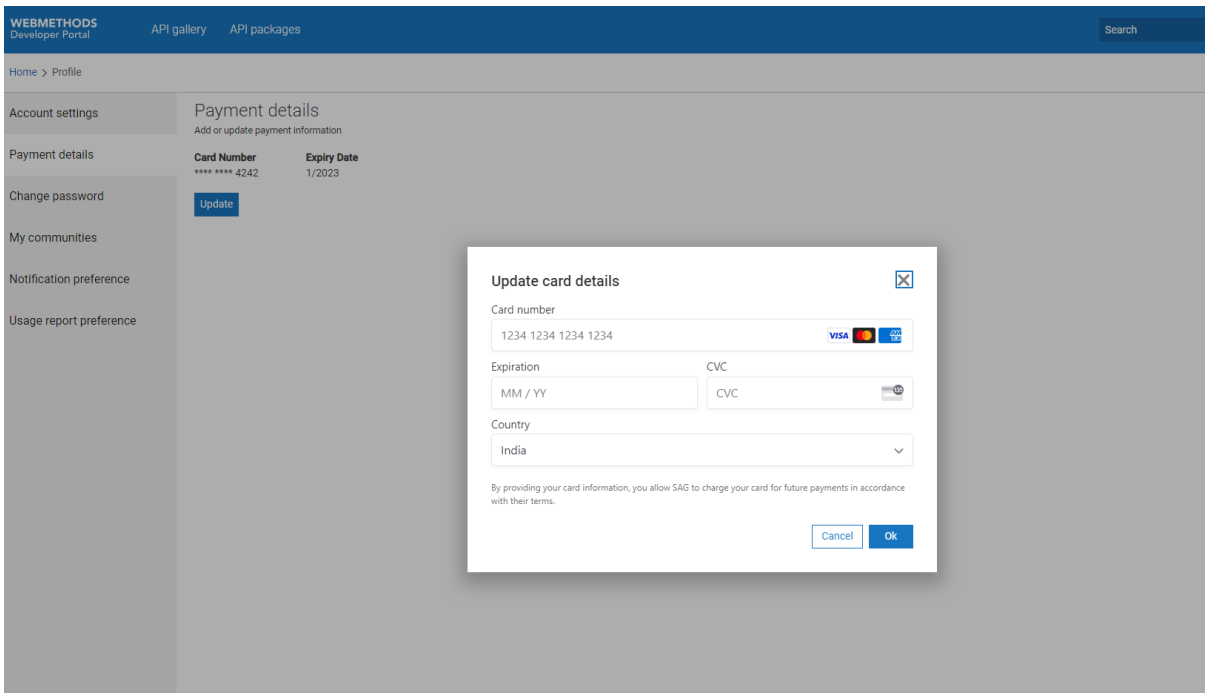


3. Provide your card details.
4. Click **Add**.

   The provided card details are saved.

Next steps

[*Optional*] Click **Update** to modify your card details, if required.



Externalizing configurations

Developer Portal can be configured on startup using the **dpowrapper.conf** and the **application_dev.yml** configuration files. These files are used to manage and provision configurations from a centralized location. The externalized configuration can be specified either within a single file providing all the necessary configurations or multiple files for individual configurations.

External Developer Portal configurations

Data management settings

- `-Dportal.server.config.max-results`. Specifies the maximum number of records that must be included in the backup file.

  When you create a backup, the system saves 1000 records of each core asset in the backup file by default. You can increase this limit If the number of records that you want to backup exceed *1000*. Core assets include APIs, Packages, access tokens of APIs, Applications, Communities, Providers, Custom assets and asset types.

Ignite configurations

The following parameters are used to specify the nodes in a cluster:

- `-Dportal.server.cache.distributed.cluster.peers.0`
- `-Dportal.server.cache.distributed.cluster.peers.1`
- `-Dportal.server.cache.distributed.cluster.peers.2`

Keystore configurations

The following parameters are used to specify Keystore configuration:

- `-Dserver.ssl.key-store`. Location of the keystore file.
- `-Dserver.ssl.key-alias`. Alias of the keystore file.
- `-Dserver.ssl.key-password`. Key password of the keystore file.
- `-Dserver.ssl.key-store-password`. Password of the keystore file.
- `-Dserver.ssl.key-store-type`. Type of the key store. Accepted values are JKS and PKCS12. The default value is *JKS*.

HTTPS port configuration

The following parameters are used to specify HTTPS port configuration:

- `-Dserver.ssl.enabled`. Enables or disables HTTPs port.
- `-Dportal.server.config.redirect-http-to-https`. Redirects incoming requests to a HTTPS port.

Community based asset visibility configuration

The following parameter is used to specify asset visibility configuration:

- `-Dportal.server.config.user-search-community-restricted`. Allows or restricts the consumers to view assets based on their communities.

API ratelimit configuration

The following parameters are used to specify API ratelimit configuration:

- `Dportal.server.config.enable-rate-limiting`. Enables or disables rate limit for APIs.
- `Dportal.server.config.rate-per-minute`. Specifies the number of invocations that you want to allow per minute for the given APIs.
- `Dportal.server.config.rate-limited-apis`. Specifies the list of APIs, separated by commas, that you want to configure rate limit.

Configuring HTTP Port

`portal.server.config.http-port: 8080`. Specifies the HTTP port in which Developer Portal runs.

User or Application onboarding approval process

The following parameters are used to specify the setting related to user or application onboarding approval:

- `portal.server.config.external-approval-expiry:`. Specifies the number of days after which the external approval request expires. The default value provided for this setting is *15d*.
- `portal.server.config.email-verification-expiry`. Specifies the number of days after which the e-mail verification link expires. The default value provided for this setting is *2d*.
- `portal.server.config.external-user-invitation-expiry`. Specifies the number of days after which a sign-up invitation expires. The default value provided for this setting is *7d*.
- `portal.server.config.enable-ldap-onboarding`. Specifies whether the user onboarding approval flow is enabled for the LDAP users. The default value provided for this setting is *false*.

General configurations

The following parameters are:

- `portal.server.config.max-zip-size:`. Specifies the maximum size of the zip file that can be uploaded to Developer Portal. The default value provided for this setting is *209715200*.

  The 200 in the default value indicates 200 MB.

- `portal.server.config.image-types:` Specifies the types of image files supported in Developer Portal. For example, *image/jpeg*, *image/png*, *image/gif*, *image/svg+xml*.
- `portal.server.config.doc-types:`. Specifies the types of documents that are supported in Developer Portal. The default values provided in this setting are *text/plain*, *application/msword*, *application/vnd.openxmlformats-officedocument.wordprocessingml.document*, *application/pdf*,

application/rtf, *application/json*, *application/zip*, *application/xml*, *text/yaml*, *text/x-yaml*, *application/x-yaml*.

- `portal.server.config.encryption-key:`. Specifies the key using which application credentials will be decrypted. This is applicable for owned or shared users. The default value provided for this setting is *yap-portAl$* .
- `portal.server.config.page-size:`. Specifies the page size that can be returned in a any single GET response. The default value provided for this setting is *12*.
- `portal.server.config.headers.content-security-policy:`. Specifies the resources that can be allowed to load in the Developer Portal pages. For example, if you want to allow resources from `okta.com` to be loaded in Developer Portal, then provide the `*.okta.com` value for this setting.

  If you do not provide any value in this setting, no resources are allowed to load.
- `portal.server.intranet-enabled.create-api`. Specifies whether users can create APIs using an intranet URL or not. The values applicable for the setting are:
  - `true`. Users can create APIs by providing an intranet URL.
  - `false`. Users cannot create APIs by providing an intranet URL.

Search configuration

The following parameters are used to specify search configurations:

- `portal.server.config.keyword-search.objects:`. Specifies the list of objects that can be searched in the header navigation. Only the objects specified in this list can be searched using the text box given in the UI header. The default values provided for this setting are *API*, *PACKAGE*, *CUSTOM_ASSET*.
- `portal.server.config.advanced-search.objects:`. Species the list of objects that can searched in the advanced search. The default values provided in this setting are *API*, *COMPONENT*, *ENDPOINT*, *METHOD*, *ODATA_OPERATION*, *ODATA_STRUCTURE*, *PACKAGE*, *RESOURCE*, *RESTMETHOD*, *CUSTOM_ASSET*.

Locale configuration

The following parameters are used to specify locale configuration settings:

- `portal.server.config.default-locale:`. Specifies the default language. The default value provided for this setting is *en_US*.
- `portal.server.config.supported-locales:`. Specifies the list of supported languages. The default values available for this setting are *en_US*, *ar_SA*, *zh_CN*, *nl_NL*, *pl_PL*, *pt_PT*,*it_IT*, *ru_RU*, *ko_KR*, *es_ES*, *fr_FR*,*de_DE*, *ja_JP*.

Logging configuration

The following parameters are used to specify settings to configure events logging in Developer Portal:

- `logging.pattern.file:`. Specifies the syntax of log entries saved in log files. The default value specified for this setting is

  ```
  %d|[%thread]|[%X{tenant:-default}]| %-5level| %logger{36} - %msg%n
  ```

  Sample log entry saved in default format:

  ```
  2024-03-16 17:39:11,265|[WrapperSimpleAppMain]|[default]| INFO |
  c.s.p.umc.inject.ServiceInitializer - Starting services...
  ```

- `logging.pattern.console:`. Specifies the syntax of console messages. The default value specified for this setting is

  ```
  %d|[%thread]|[%X{tenant:-default}]| %-5level| %logger{36} - %msg%n
  ```

- `logging.file.name:`. Specifies the name of the log file that will be created during start up of Developer Portal. The default value provided for this setting is `dev-portal.log`.
- `logging.logback.rollingpolicy.max-file-size:`. Specifies the maximum size of log file. When a log file reaches the specified size, then the current log file is renamed and the further logs are saved in a new file. You can provide the format for renaming old log files using the *logging.logback.rollingpolicy.file-name-pattern* setting. The default value provided for this setting is `10 MB`.
- `logging.logback.rollingpolicy.file-name-pattern:`. Specifies the syntax for renaming the log files that have reached the maximum size. The default value provided for this setting is *dev-portal-%d{yyyy-MM-dd}.%i.log*.

Community

The following parameters are used to specify settings related to the Developer Portal communities:

- `portal.server.config.communities.public-community-id:`. Specifies the Id for the public community. The default value specified for this setting is *3bdf8005-5685-3ef5-b132-de4681963fb6*.
- `portal.server.config.communities.public-community-name:`. Specifies the name of the public community. The default value specified for this setting is *Public Community* .

License management configuration

The following parameter is used to configure the license file expiry:

- `portal.server.config.license-expiry-alert:`. Specifies the number of days before the license expiry that the expiry alert has to be displayed. The default value provided for this setting is *5*.

Points to remember

Some point that you must remember when setting your configurations in **dpo_wrapper.conf** file.

- The **dpo_wrapper.config** file is located in the `SAGInstallDir/profiles/CTP/` folder of your installation folder.
- Format to provide the settings is as follows:

```
wrapper.java.additional.Settingincremental_number=Value of the setting
```

The incremental number must be a positive integer that increments with each setting. You can specify numbers of your choice. As a recommended practice, find the biggest number in the configuration file and start incrementing the numbers for the next entries as and when you add them to the file. For example, if you see 2008 as the biggest value, then start assigning 2009, 2010 and so on to further entries.

Sample

```
wrapper.java.additional.2013=-Dserver.ssl.enabled=false
```

Data Backup and Restore

Data is an integral part of Developer Portal and contains all the information about your APIs, packages, and your customized themes. Hence, it is essential to manage data efficiently. To protect any accidental loss of data, you must take regular backups of your data and save in a fail-proof repository.

Developer Portal provides you an option through the UI to create backup of your data and restore any backed up data.

The following sections explain the backup and restore processes.

How do I take a backup?

This use case starts when you want to create a backup of your data and ends when you have successfully created the backup.

You can configure the required backup limit. For more information, see "Setting Backup Limit" on page 44.

Before you begin:

Ensure that you have the **API Administrator** privilege.

To take backup:

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Backup and restore** from the left pane.
3. Select **Backup**.
4. Select the **Modules** that you want to be included in the backup.

   Available options are:

   - **Collaboration**. Collaboration groups, posts and discussions related data, and comments or posts related to APIs.

- **Core**. Includes the meta data information of APIs and packages, access tokens of APIs, Applications, Communities, Providers, Custom assets and asset types.
- **Theme**. Includes the UI customization templates.
- **User**. Includes details of users, user groups, and user privileges.
- **Analytics**. Includes Page views, User registrations, API lifecycle events, run-time analytics data for APIs, analytics metrics, and analytics events.

5. Click **Backup**.

   A dialog box appears to allow you save the backup file.

6. Click **Save**.

   The backup data, in zipped folder format, is saved to the default downloads location of your browser.

   Note:
   Ensure that your browser setting allows pop-ups.

Next steps:

- To view the list of assets saved in the backup, open the `core-result.pdf` file from the **Core** folder inside the backup folder.

Setting Backup Limit

When you create a backup, the system saves 1000 records of each core asset in the backup file by default. You can increase this limit If the number of records that you want to backup exceed 1000.

Core assets include APIs, Packages, access tokens of APIs, Applications, Communities, Providers, Custom assets and asset types.

To set backup limit

1. Open the `dpo_wrapper.conf` file from the location, `SAGInstallDir/profiles/CTP/configurations/`.
2. Add the following entry and specify the required limit.

   ```
   wrapper.java.additional.3007=-Dportal.server.config.max-results=maximum number of
    records
   ```

   Sample

   ```
   wrapper.java.additional.3007=-Dportal.server.config.max-results=50000
   ```

3. Click **Save**.

   The configuration is saved.

How do I restore data from a backup file?

This use case starts when you have a backup file to be restored and ends when you have successfully restored the backup data in your environment.

Important:
When you restore data in an environment that already has **User** data, it is not overwritten by the restored data. The restored data is integrated with the existing data. Data of other types in the target environment are overwritten with the data from the backup file.

Before you begin:

Ensure that you have the **API Administrator** privilege.

To restore from a backup:

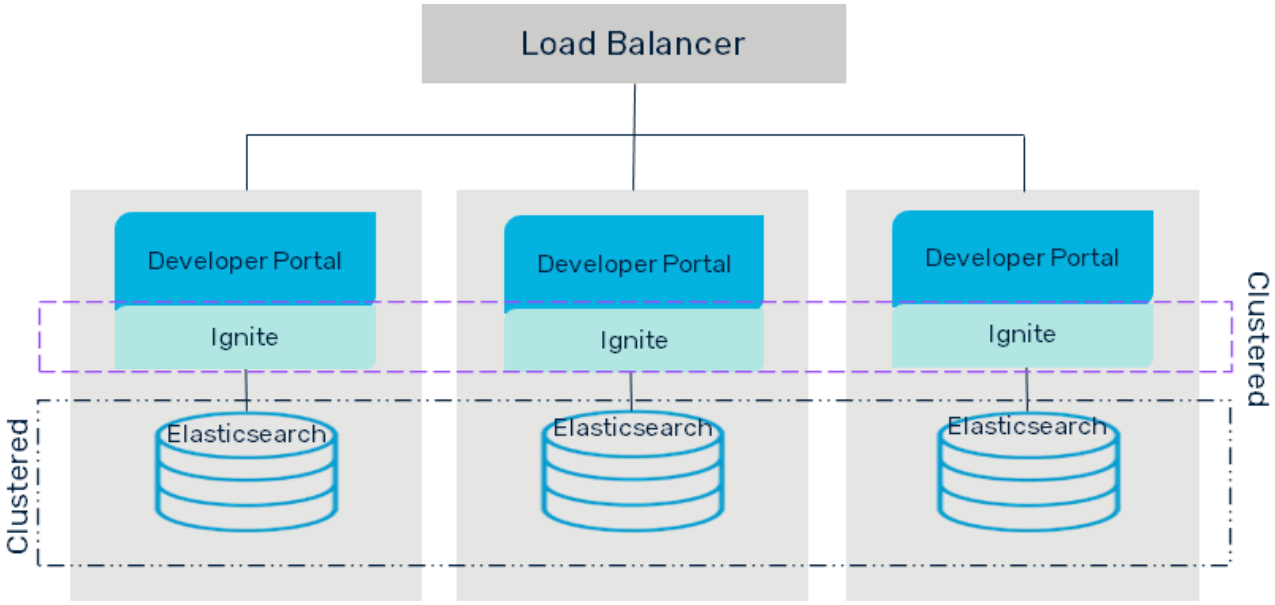
1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **Backup and restore** from the left pane.
3. Select **Restore**.
4. Click **Browse file** and select the backup file that has to be restored.
5. Select the **Modules** that must be restored.
6. Click **Restore**.

   The selected data is restored in your environment.

   Note:
   When you restore the **Core** module, you must publish the Developer Portal destination from your API Gateway instance. This is to ensure that your data is seamlessly captured in the Analytics dashboard.

High availability Configuration

Clustering the Developer Portal instances enable you to achieve high availability of the application.



In a cluster, each node must contain each of a Developer Portal instance, Ignite instance, and API Data Store. The load balancer distributes requests across the cluster nodes. The synchronization of the nodes is performed by clustering the Ignite instances and API Data Stores across the nodes.

Configuring High Availability

This procedure describes in detail the setting up of the High Availability (HA) setup for Developer Portal.

Before you begin:

- Ensure that there are a minimum three nodes available with Developer Portal installation and Ignite in them.
- Ensure that the required Load balancer is downloaded.
- Open appropriate ports on each machine so that they can access all components on the other machines. Else, the cluster is not formed.
- Ensure that the API Data Store and Developer Portal in all nodes are stopped.

Configuring API Data Store

Developer Portal installation includes the API Data Store that is used to store data. For clustering the API Data Stores of the instances, you must perform standard Elasticsearch clustering. For more information, see https://www.elastic.co/guide/en/elasticsearch/reference/8.2/high-availability.html.

You must perform the following steps in all nodes of a cluster.

To configure the API Data Store

1. Open the `elasticsearch.yml` file from the location, *`SAGInstallDir`*`/InternalDataStore/config/`.

   The following configuration is a sample of how the configuration appears initially.

   ```
   cluster.name: SAG_EventDataStore
   node.name: portalnode1
   path.logs: C:\SoftwareAG_DPO\InternalDataStore/logs
   network.host: 0.0.0.0
   http.port: 9240
   discovery.seed_hosts: ["localhost:9340"]
   transport.port: 9340
   path.repo: ['C:\SoftwareAG\InternalDataStore/archives']
   cluster.initial_master_nodes: "portalnode1"
   xpack.ml.enabled: false
   xpack.security.enabled: false
   ```

2. Provide the name of the cluster in the **cluster.name** property.

   Nodes with same cluster names form a cluster. That is, if there are three nodes in the cluster, the value in the **cluster.name** property must be same across all three nodes. In other words, Elasticsearch forms a cluster with nodes that have the same **cluster.name**.

   For example,

   ```
   cluster.name: Dev_portal_cluster
   ```

3. Provide the names of all participating nodes in the **discovery.seed_hosts** property in the following format.

   ```
   host_name:port_name
   ```

   For example,

   ```
   discovery.seed_hosts: ["node1:9340","node2:9340","node3":"9340"]
   ```

   Note:
   The value of the port must be same as specified in the **transport.port** property.

4. Provide the names of the master-eligible nodes in the initial cluster in the **cluster.initial_master_nodes** property.

   You must provide the **node.name** value in the **cluster.initial_master_nodes** property.

   This parameter enables cluster bootstrapping when you start a cluster for the first time. This property must be defined on every master-eligible node in the cluster. This setting helps prevent split-brain, the existence of two masters in a single cluster. For example,

   ```
   cluster.initial_master_nodes:["portalnode1"]
   ```

5. Provide the repository in which you want to save the API Data Store backup snapshots in the **path.repo** property.

   It is important to have a location that is accessible to all the nodes. The location could be network file system, S3, or Azure in the clustered setup.

Sample configuration:

```
cluster.name: Dev_portal_cluster
node.name: portalnode1
path.logs: /FS/fslocal/DPO1015/InternalDataStore/logs
network.host: 0.0.0.0
http.port: 9240
discovery.seed_hosts:["portal_server1:9340","portal_server2:9340","portal_server3":"9340"]
transport.port: 9340
path.repo: ['/FS/fslocal/DPO1015/InternalDataStore/archives']
cluster.initial_master_nodes: ["portalnode1","portalnode2","portalnode3"]
xpack.ml.enabled: false
xpack.security.enabled: false
xpack.security.transport.ssl.enabled: false
xpack.security.http.ssl.enabled: false
action.destructive_requires_name: false
```

6. Save the file.

   The specified API Data Store nodes are clustered.

Configuring Ignite

Developer Portal uses the embedded Ignite-core library to setup a cluster, without having to start any external runtime.

In Ignite terminology:

- a single Developer Portal with embedded ignite is an Ignite server node,
- together they form an Ignite cluster,
- and it is a stateless cluster as Developer Portal does not require persistence for its distributed caches.

Ignite cluster discovery

Each Ignite server node has to open a discovery port. Through this port, the nodes discover each other and form the cluster, and they detect the nodes that are added or removed from the cluster.

Each node in the cluster must be configured with a list of initial host names. These nodes will contact each other via the discovery port and form a cluster. Later more nodes can be added to the cluster, and even if their host names are not part of the initial host name list they can join the cluster by contacting one of the initial hosts, and then their host names will be communicated around the cluster.

Pre-requisites

Ensure that you have opened up the following ports:

- **47500 - 47509**. These ports enable the discovery of Ignite nodes.
- **47100 - 47200**. These ports enable communication between Ignite nodes. Ensure that these ports are accessible across other nodes.

If you have blocked the access of the port 47100 by any another application in the system, you need to open up the complete series. You can just open up the 47100 port. For more info refer https://ignite.apache.org/docs/latest/clustering/network-configuration.

- It is recommended that you do not open the following local ports when you system is behind firewall:
  - **10800**. Enables communication between the thin client and the Ignite cluster.
  - **11211**. Enables communication between the JDBC client and the Ignite database.
  - **47400**. Enables node discovery via the UDP port.

Perform the following steps in all nodes of a cluster.

To configure Ignite

1. Open the `dpo_wrapper.conf` file from the location, *SAGInstallDir*`/DeveloperPortal/configuration`.
2. Add the following entries:

```
wrapper.java.additional.310=-Dportal.server.cache.distributed.enabled=true
wrapper.java.additional.311=-Dportal.server.cache.distributed.cluster.peers.0=<devPortal1_hostname>:47500..47509
wrapper.java.additional.312=-Dportal.server.cache.distributed.cluster.peers.1=<devPortal2_hostname>:47500..47509
wrapper.java.additional.313=-Dportal.server.cache.distributed.cluster.peers.2=<devPortal3_hostname>:47500..47509
```

Note:
The values 311, 312, and 313 mentioned in the code block need not be the same. They are sample values only. You can specify numbers of your choice. As a recommended practice, find the biggest number in the `dpo_wrapper.conf` file and start incrementing the numbers for the next entries as and when you add them to the file. For example, if you see 2008 as the biggest value, then start assigning 2009, 2010 and so on to further entries.

3. Click **Save**.

   The configuration is saved.

4. Start all Developer Portal nodes one after the other.

Configuring Developer Portal

You must specify the load balancer URL in the Developer Portal instances of all nodes of the cluster.

1. Sign in to Developer Portal.
2. Click the menu options icon  from the title bar, and click **Administration**.

3. Click **General** from the left pane.

4. In the **Load balancer URL** field, provide the URL of the external Load balancer.

   For example, `http://<ext_lb_hostname>/portal`.

5. Click **Save**.

   The configuration is saved.

Configuring Load Balancer

Load balancer distributes the incoming requests to the nodes of a cluster with the aim of making their overall processing more efficient. Load balancer optimizes the response time and avoid unevenly overloading some nodes while other nodes are idle.

You can download and use a load balancer of your choice. You must also configure sticky session in the Load balancer for UI upstreaming.

A sample Nginx load balancer configuration file /etc/nginx/nginx.conf is as follows:

```
#******* @subdomain@Nginx Config ********
events {
    worker_connections  1024;
}

http {


upstream @subdomain@_ui_upstream {
  ip_hash;
  server @node1@:18101;
  server @node2@:18101;
  server @node3@:18101;
}

server {
  listen       80;
  listen       [::]:80;
  server_name  @subdomain@;

  access_log  /var/log/nginx/access.log;

  # DESIGN time should work only in https.
  location /portal {
    proxy_pass              http://@subdomain@_ui_upstream/portal/;
    proxy_set_header        Host $host;
    proxy_set_header        X-Real-IP $remote_addr;
    real_ip_header          X-Real-IP;
    proxy_http_version      1.1;
    proxy_set_header        Connection "";
  }

  location /portal/ {
    proxy_pass              http://@subdomain@_ui_upstream/portal/;
    proxy_set_header        Host $host;
```

```
    proxy_set_header        X-Real-IP $remote_addr;
    real_ip_header          X-Real-IP;
    proxy_http_version      1.1;
    proxy_set_header        Connection "";
  }
}

}
```

The nodes in the cluster are synchronized.

Post-configuration

Perform the following steps after you have completed the high availability configuration:

1. Start the Elasticsearch nodes one after the other.

2. Ensure that the started node is up and running before starting the next node.

3. Open your browser and enter the following URL in the address bar, and press **Enter** to verify the cluster configuration

   ```
   http://<node_name>:<port_number>/_cluster/health?pretty=true
   ```

   For example,

   ```
   http://ES-node1:<9393>/_cluster/health?pretty=true
   ```

   Sample response

   ```
   {
     "cluster_name" : "Dev_portal_cluster",
     "status" : "green",
     "timed_out" : false,
     "number_of_nodes" : 3,
     "number_of_data_nodes" : 3,
     "active_primary_shards" : 18,
     "active_shards" : 36,
     "relocating_shards" : 0,
     "initializing_shards" : 0,
     "unassigned_shards" : 0,
     "delayed_unassigned_shards" : 0,
     "number_of_pending_tasks" : 0,
     "number_of_in_flight_fetch" : 0,
     "task_max_waiting_in_queue_millis" : 0,
     "active_shards_percent_as_number" : 100.0
   }
   ```

Developer Portal Ports Configuration

Ports Configuration

Developer Portal uses the HTTPS port with the value you provide during installation. The default HTTPS port value is `18102`.

By default, Developer Portal uses its own self-signed keystore, which is not suitable for the production environment. Therefore, Software AG recommends that you use a self-signed keystore certificate or one from an authorized certificate authority.

Configuring Keystore

Once you have imported a valid keystore certificate, you must configure the keystore.

To configure the keystore

1. Open the **dpo_wrapper.conf** file from the following location, *SAGIntallDir*/DeveloperPortal/configuration.
2. Provide the required values in the following:
   - **wrapper.java.additional.2009=-Dserver.ssl.key-store**. Location of the keystore file.
   - **wrapper.java.additional.2010=-Dserver.ssl.key-alias**. Alias of the keystore file.
   - **wrapper.java.additional.2011=-Dserver.ssl.key-password**. Key password of the keystore file.
   - **wrapper.java.additional.2012=-Dserver.ssl.key-store-password**. Password of the keystore file.
   - **wrapper.java.additional.2013=-Dserver.ssl.key-store-type**. Type of the key store. Accepted values are JKS and PKCS12. The default value is *JKS*.

     Note:
     The values 2009 to 2012 in the variables need not be the same. You can increment these values if a value is already seen in the **dpo_wrapper.conf** file.

     Sample values

     ```
     wrapper.java.additional.2009=-Dserver.ssl.key-store=<keystore file path>
              wrapper.java.additional.2010=-Dserver.ssl.key-alias=<alias of the
     keystore>
             wrapper.java.additional.2011=-Dserver.ssl.key-password=<key password>

            wrapper.java.additional.2012=-Dserver.ssl.key-store-password=<key store
      password>
     ```
3. Save and close the configuration file.
4. Restart the server for the configuration changes to take effect.

Disabling HTTPS port

By default, uses HTTPS port. However, you can modify this behavior based on your requirement.

To disable the HTTPS port

1. Open the **dpo_wrapper.conf** file from the following location, *SAGIntallDir*/DeveloperPortal/configuration.
2. Set the following property to `false`.

```
wrapper.java.additional.2013=-Dserver.ssl.enabled=false
```

3. Save and close the configuration file.
4. Restart the server for the configuration changes to take effect.

Directing requests to HTTPS port

By default, HTTP and HTTPS will work independently on its respective ports. However, if required, you can redirect all requests of the HTTP port to the HTTPS port.

To direct requests to the HTTPS port

1. Open the **dpo_wrapper.conf** file from the following location, *SAGIntallDir*/DeveloperPortal/configuration.
2. Set the following property to `true`.

```
wrapper.java.additional.2014=-Dportal.server.config.redirect-http-to-https=true
```

3. Save and close the configuration file.
4. Restart the server for the configuration changes to take effect.

Configuring Developer Portal Communication with secured API Gateway

When the HTTP ports of an API Gateway instance are configured with a self-signed certificate, you must import the certificate in Developer Portal for a seamless communication with API Gateway.

To secure Developer Portal communication with API Gateway

1. Run the following command to add the self-signed certificated used in API Gateway to the Developer Portal truststore in the following location: `SAGInstallDir/jvm/jvm/lib/security/cacerts`:

```
keytool -import -trustcacerts -alias alias_name -file certificate_path -keystore
cacerts
```

   Sample command

```
keytool -import -trustcacerts -alias mdecert -file C:\apigw\mdeCert.cer -keystore
 cacerts
```

Note:
If the SSL exception message appear after you import the self-signed certificate to the mentioned path, then ensure that the **disable-host-name-verifier** setting in the `application-dev.yml` file is set to *true*. This setting allows you to turn off the host name verification during Developer Portal communication with other systems.

Securing API Data Store

You can secure API Data Store (a simple Elasticsearch instance), one of the components in the Developer Portal setup, to communicate securely over HTTPS.

This section explains the steps required to secure API Data Store.

To secure the API Data Store:

1. Navigate to the folder where the Developer Portal instance is installed.
2. Create a folder in the following path `tmp/es`.
3. Create a instance YAML file in the `/es` folder with the name, `vi instance.yml`.
4. Provide the following details in the `vi instance.yml` file.

```
instances:
  - name: 'daeapiportal08rh'
    dns: [ 'daeapiportal08rh.eur.ad.sag' ]
```

5. Open DOS command prompt from the location, `tmp/es` and run the following command:

```
bin/elasticsearch-certutil cert - --ca elastic-stack-ca.p12 --pem --in
~/Install/es/instance.yml --out ~/Install/es/certs.zip
```

6. Navigate to the config folder of the elasticsearch and run the following command to unzip the downloaded certificate:

```
unzip certs.zip -d ./certs
```

7. Create a folder inside `tmp/es` folder and copy the unzipped certificates to it.
8. Open the `elasticsearch.yml` file in editing mode.
9. Add the following lines to the file and remove the duplicate entries:

```
node.name: daeapiportal08rh.eur.ad.sag
network.host: daeapiportal08rh.eur.ad.sag
xpack.ml.enabled: false
xpack.security.enabled: true
xpack.security.http.ssl.enabled: true
xpack.security.transport.ssl.enabled: true
xpack.security.http.ssl.key: daeyapcent01/daeyapcent01.key
xpack.security.http.ssl.certificate: daeyapcent01/daeyapcent01.crt
xpack.security.http.ssl.certificate_authorities: daeyapcent01/daeyapcent01.crt
```

```
xpack.security.transport.ssl.key: daeyapcent01/daeyapcent01.key
xpack.security.transport.ssl.certificate: daeyapcent01/daeyapcent01.crt
xpack.security.transport.ssl.certificate_authorities: daeyapcent01/daeyapcent01.crt
```

Replace `network.host` with the required host name.

10. Run the following command to start API Data Store:

```
 ./startup.sh
```

11. Run the following command once the API Data Store is started

```
./elasticsearch-setup-passwords auto -u "https://daeapiportal08rh.eur.ad.sag:9240"
```

12. Secure the password for future use.
13. Access the `_cat/nodes` API via HTTPS.

```
curl --cacert ~/Install/es/certs/ca/ca.crt -u elastic
'https://daeapiportal08rh.eur.ad.sag:9240/_cat/nodes?v'
```

14. Open the file, `SAGInstallation/profiles/CTP/configuration/dpo_wrapper.conf`, add the following entry and save it:

```
wrapper.java.additional.2000=-Dspring.elasticsearch.uris=https://daeapiportal08rh.eur.ad.sag:9240
wrapper.java.additional.2006=-Dspring.elasticsearch.username=elastic
wrapper.java.additional.2007=-Dspring.elasticsearch.password=m8XnkjtqoyOLVgnU9qU5
```

15. Add the certificate to the following folder, `SAGInstallation/jvm/jvm/lib/security/cacerts`.
16. Restart the CTP server from the location, `SAGInstallation/profiles/CTP/bin`.

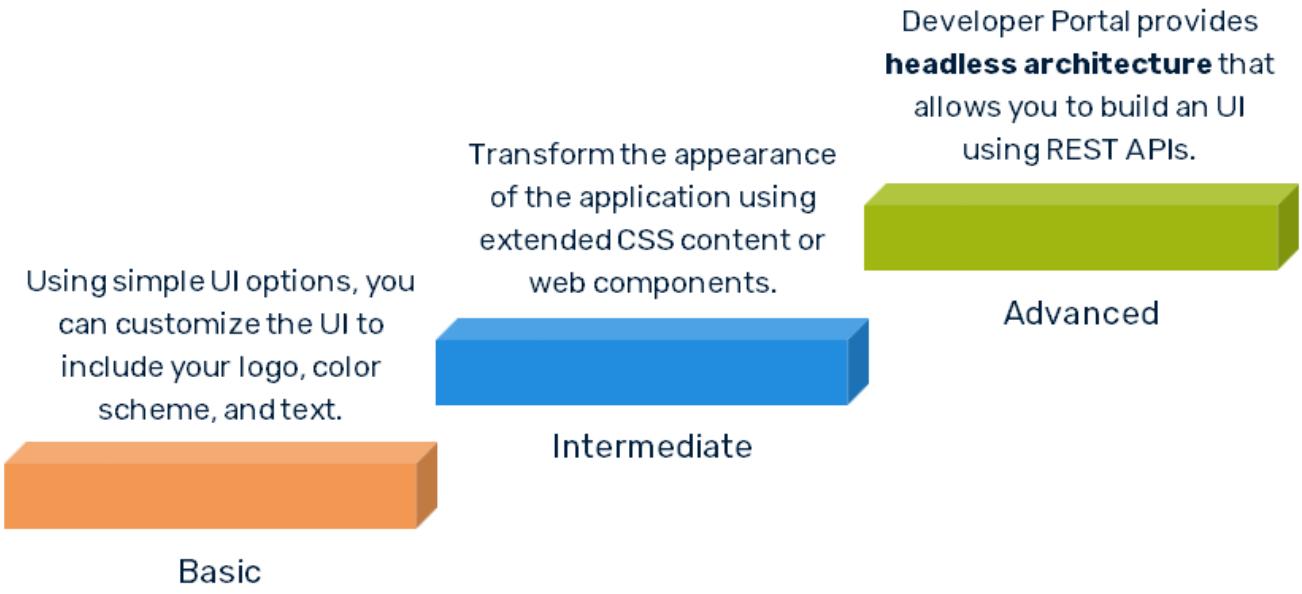
3 Customization

Overview

The customization feature provides you options to customize the Developer Portal UI to suit your needs. This feature allows you to completely customize the portal to establish your brand among your consumers. As administrators and API providers, who set up their portal to offer APIs, you would want your portal to be unique and recreate your brand by tailoring the portal the way you want. Customization helps you to achieve this.

Customization could involve changing logos, organizing available blocks or other UI components, based on your priority and marketing strategy. Developer Portal comes with a WYSWYG UI that allows you to preview your customization and proceed accordingly.

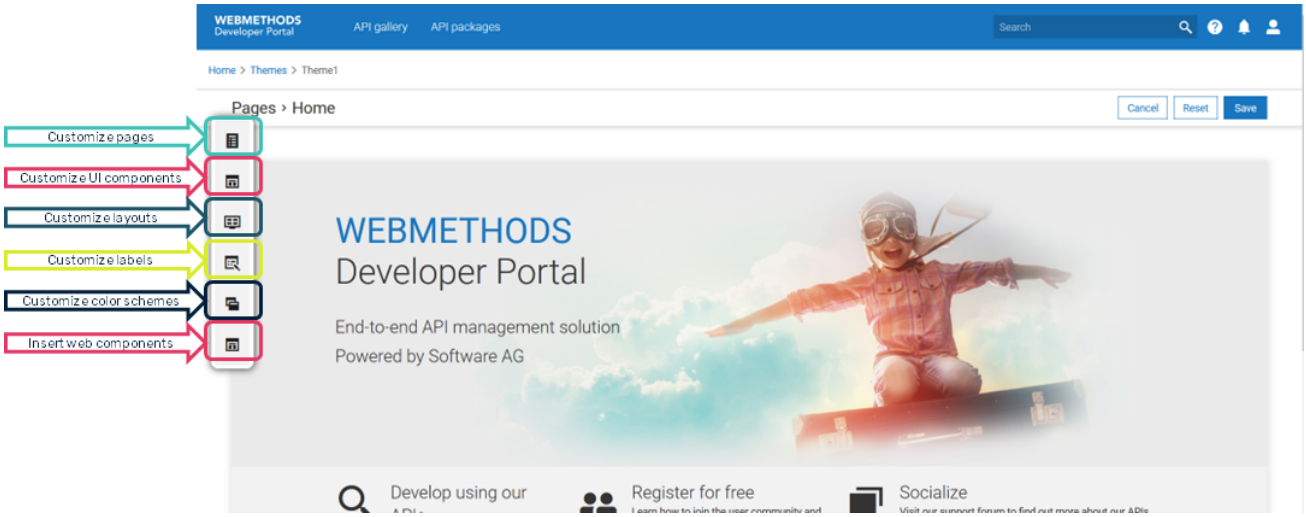
Developer Portal offers different levels of options to perform customization. They include:



This section describes the feature and detailed procedures for customizing Developer Portal.

Customization based on themes

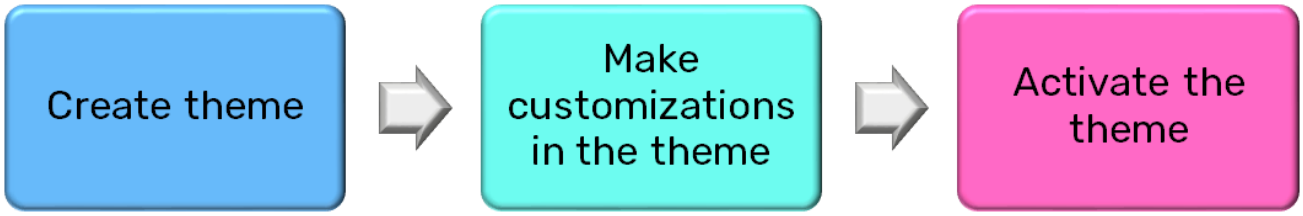
Developer Portal allows you to perform the customization through themes. The **Themes** feature under the **Administration** section provides you the options that you can use to specify your customization.



You can create themes, perform required customization, and activate the theme to apply the customization. You can create any number of themes and customize each of them differently. However, you can have only one theme activated for a portal at a time.

Managing Themes

Theme contains all information about the UI appearance and they determine the look and feel of the UI. So, you should create a theme, specify your changes and apply it to customize the UI. The high level steps are:



You can have more than one theme and apply them according to your requirements. However, you can have one theme activated at a time.

How do I create a theme for customizing the Developer Portal UI?

You can create a theme from scratch or clone one from an existing theme.

This use case starts when you want to create a theme for customizing your portal UI and ends when you have successfully created the theme.

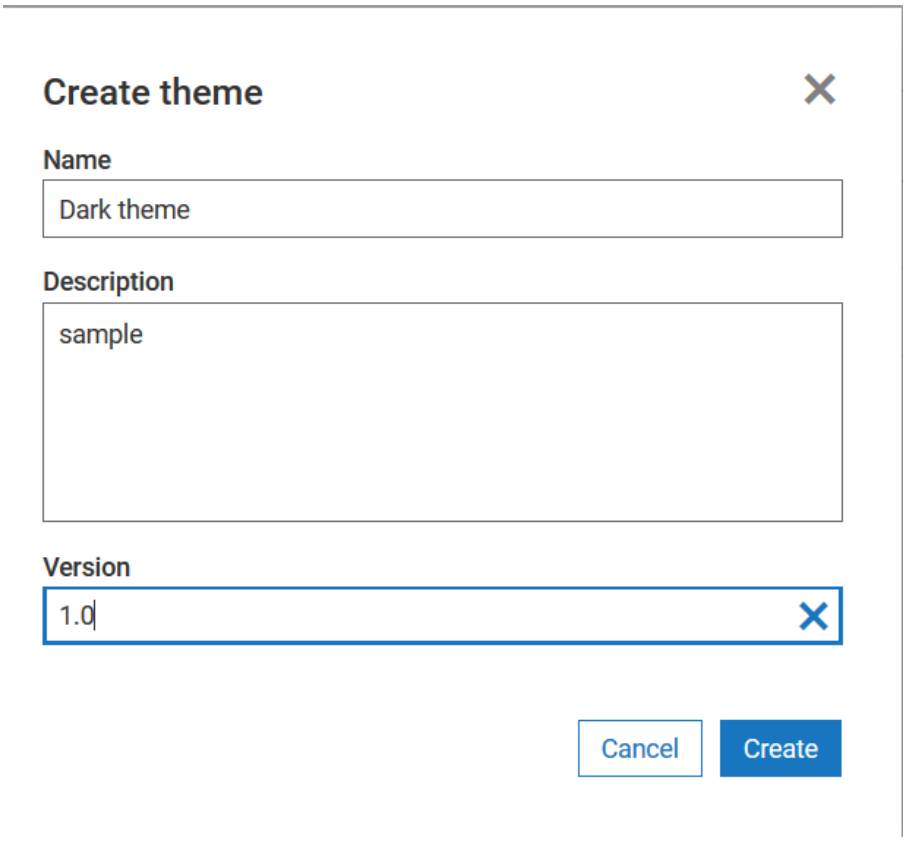
In this example, you create a theme named *Dark theme*.

Before you begin:

Ensure you have the **API Administrator** privilege.

To create a theme

1. Click the menu options icon from the title bar.
2. Select **Administration**, and click **Manage themes**.
3. Click **Create theme**.
4. Provide *Dark theme* in the **Name** field.
5. Provide *1.0* in the **Version** field.



6. Click **Create**.

   The new theme appears in the **Manage themes** page.

Alternative steps:

- In *Step 3*, you can click the clone icon next to a theme to clone the theme, provide a theme name and click **Create**.

Next steps:

- Click the edit icon next to the theme to edit it.

- Click the customize icon  next to the theme to customize it.
- Click the activate icon ✔ next to the theme to apply the theme or You can customize and activate the theme to see the changes on UI.

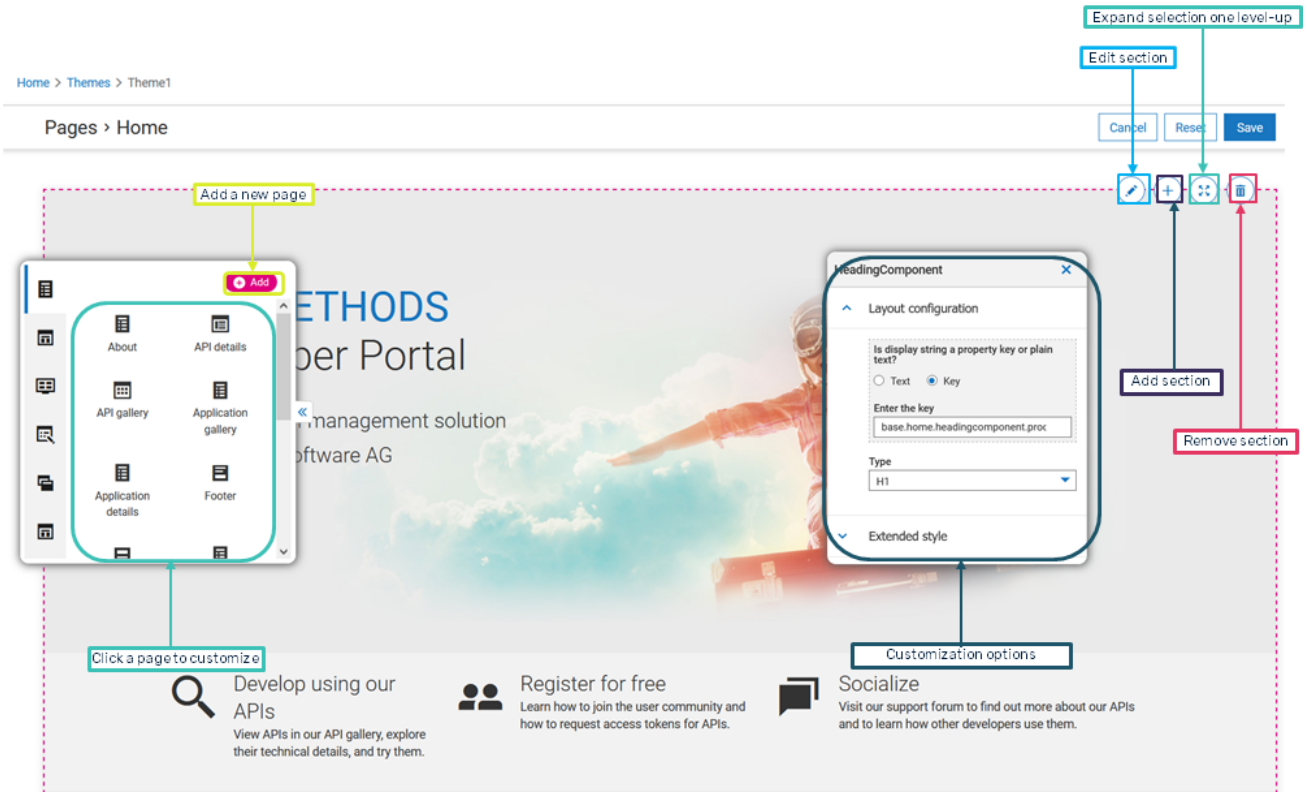
Customize pages

Developer Portal pages include multiple sections. Sections include different objects like headings, text, images, links, tags, and icons.

The **Pages** section provides you options to select a page and customize its sections or add a new page. From the **Pages** section, you can:

- Customize the existing sections of a page. For information about customizing a section, see "How do I customize a block on a page?" on page 62.
- Add a new section. For information about customizing a section, see "How do I add a new block and component?" on page 69.
- Remove a section. For information about removing a section, see "How do I remove a block from a page?" on page 74.
- Move a section up or down, or left or right. For more information, see "How do I move blocks in a page?" on page 72.
- Add a new page. For information about adding a new page, see "How do I add a page?" on page 75.

The screen indicates the options available for customizing the Developer Portal pages.



How do I customize a block on a page?

You can divide a page into multiple blocks and customize the blocks individually to transform the page.

This use case begins when you want to customize a block and ends when you have completed customization.

In this example, the first image in the **Welcome** page of the theme *Theme1* is modified.

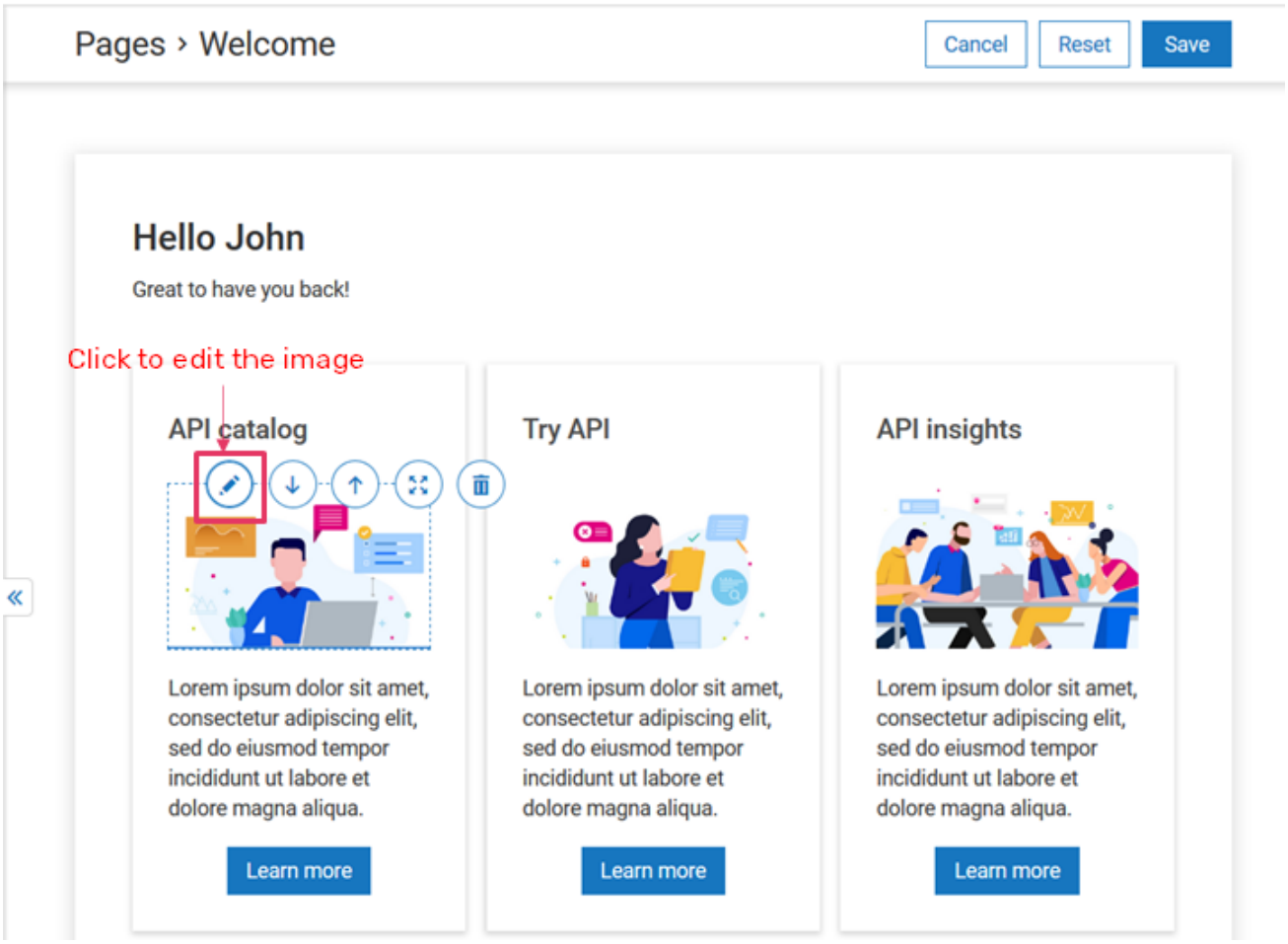
Before you start

Ensure that you have created a theme or have a theme to customize.

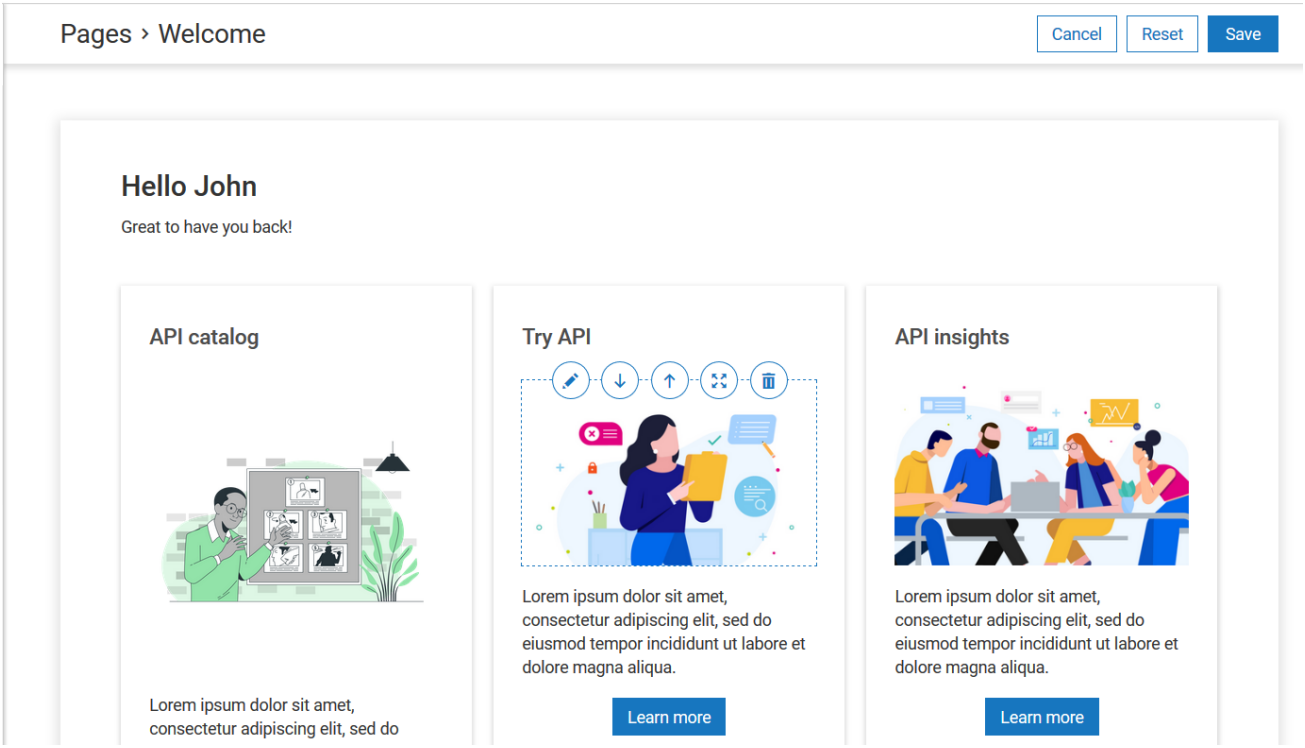
To customize a block

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Pages** and select **Welcome**.

   The **Welcome** page appears with the corresponding editing options for each of the blocks.
3. Move your mouse pointer over the first image on the **Welcome** page and click the edit icon  .



4. Click **Browse** and select the required image.



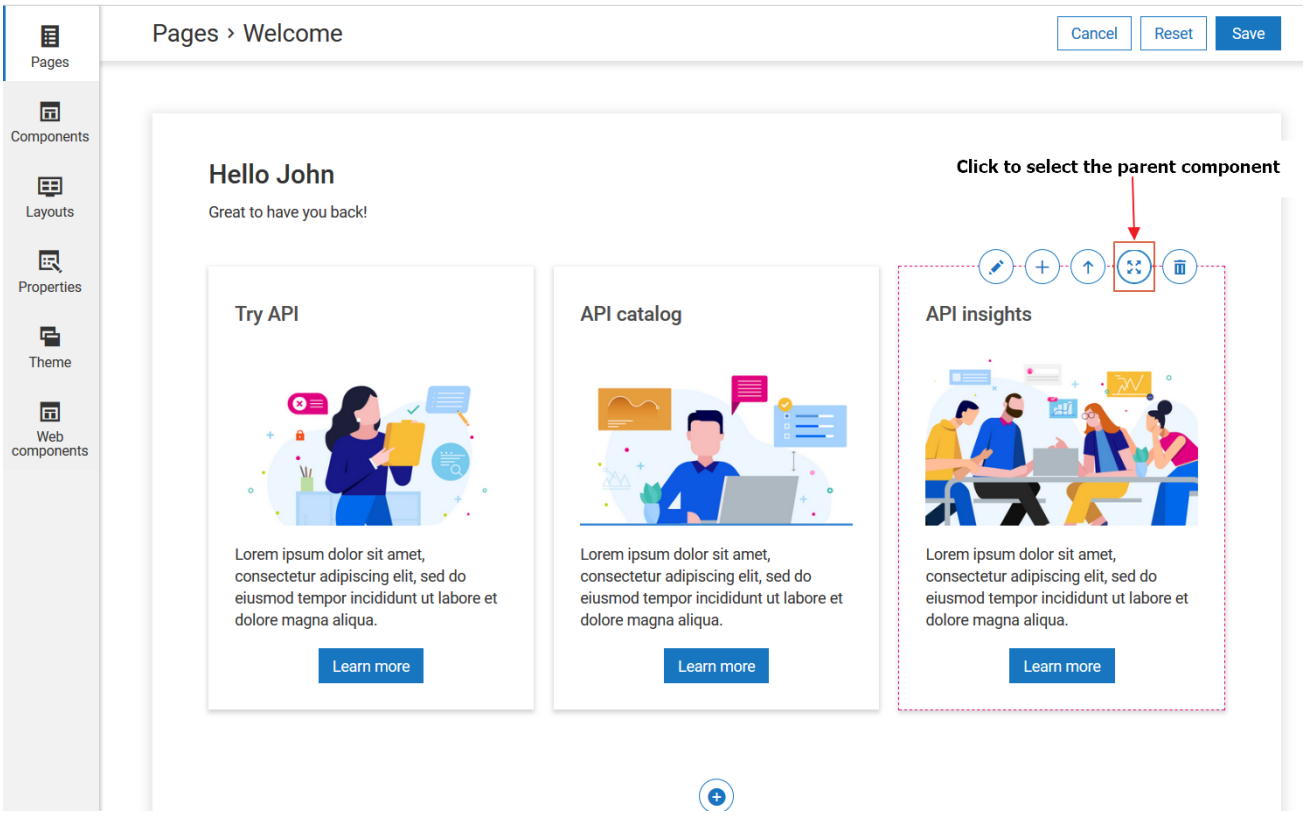
5. Click **Save**.

   Your changes are saved.

Alternative steps:

1. In *Step 3*, move your mouse pointer over a block for customization options. The options to customize the block appear in the **Layout configuration** section of the right pane. For information on the customization fields, see "Customization fields" on page 65.

2. When the focus is set on a component, click  of the component to shift focus to the parent component.



When you select a parent component for editing, press Esc to clear the selection, before selecting any other component.

Next steps:

- Click the activate icon ✓ next to theme in the **Manage themes** screen to activate the changes.

Customization fields

This section lists the fields that are used to customize the blocks on a page.

Layout configuration options for page alignment

Field	Description
Direction	Specifies the direction of text in a block. Available options are: ■ **Row**. Aligns components in a horizontal fashion like a row. ■ **Column**. Aligns components in a vertical fashion like a column.
Main axis	Specifies the option to determine how the components in page must be distributed between and around the main axis of the page.

Field	Description
	Available options are: ■ **Start**. Aligns components from the start of the block. ■ **Center**. Aligns components towards the center of the block. ■ **End**. Aligns components towards the bottom portion of the block. ■ **Space around**. Distributes components evenly. Components have a half-size space on either end. Aligns the paragraphs in a block allowing space around each paragraph. ■ **Space between**. Distribute components evenly The first item is flush with the start, the last is flush with the end **Tip:** You can try the options, check results, and select
Cross axis	Specifies the option to determine how the components in page must be distributed between and around the cross axis of the page. Available options are: ■ **Stretch**. Stretches and aligns components to fill the block. ■ **Center**. Aligns components towards the center of the block. ■ **Start**. Aligns components from the start of the block. ■ **End**. Aligns components towards the bottom portion of the block.
Minimum height	Specifies the minimum height of a component. It prevents the used value of the height property from becoming smaller than the value specified value.
Width	Specifies the width of the page.
Flex basis	Specifies the initial minimum width of the page.

Layout configuration options for headings

Field	Description
Is display string a property key or plain text?	Specifies the type of text displayed as heading. Select one of these options: ■ **Text**. To display plain text and provide the text to be displayed. ■ **Key**. To display localization text and provide the required key.

Field	Description
Type	Specifies the heading level. The value ranges from H1 to H6.

Layout configuration options for images

Field	Description
Has context?	Specifies if you insert an image using context. If you enable this, provide the JSON representation of the image that has to be inserted in the **Source** field. For example `{ 'icon' : '...' } ${icon}` The value of icon is retreived from the JSON provided in the field.
Browse	Allows you to select an image from your device.
Alt text	Provide the alt text for the image.
Height (in pixels)	Provide the height of the image in pixels.
Width (in pixels)	Provide the width of the image in pixels.

Layout configuration options for buttons and icons

Field	Description
Icon button	Specifies whether the button is an icon.
Icons customization options	
Size	Specifies the size of the icon. Available options are small, medium, large, and extra small. **Note:** You can try these options, check the preview, and select the required type.
Icon	Allows you to select an icon from the list of default icons such as **Save**, **Close** and so on.
Tooltip	Provide the tooltip that must appear for the icon.
Button customization options	

Field	Description
Button type	Specifies the style of button. Available options are Primary, Secondary, Tertiary, and Emphasis. **Note:** You can try these options, check the preview, and select the required type.
Size	Specifies the size of the icon. Available options are small, medium, and large. **Note:** You can try these options, check the preview, and select the required type.
Is display string a property key or plain text?	Specifies the type of text displayed on the button. Select one of these options: ■ **Text**. To display plain text and provide the text to be displayed. ■ **Key**. To display a localization text and provide the required key.

Layout configuration options for tags

Field	Description
Value	Provide the text, as comma separated values, that must appear on tags.

Layout configuration options for links

Field	Description
Type	Specifies the type of link. Available options are Internal and External.
Target link	Provide the required link.
Is display string a property key or plain text?	Specifies the type of text displayed as link. Select one of these options: ■ **Text**. To display plain text and provide the text to be displayed. ■ **Key**. To display a UI label and provide the required key.

Layout configuration options for text editor and HTML enabled

Field	Description
Configure the value	Provide the custom text or HTML code.

Layout configuration options for web components

Field	Description
Name	Select the name of the web component.
Element name	Select the required element name.

How do I add a new block and component?

Page is made of multiple blocks. You can add new blocks or customize the available blocks as per your requirements.

This use case starts when you want to add a block and ends when you have added it.

In this example, you add a new *heading* component to the **Home** page of the theme *Theme1*.

Before you start

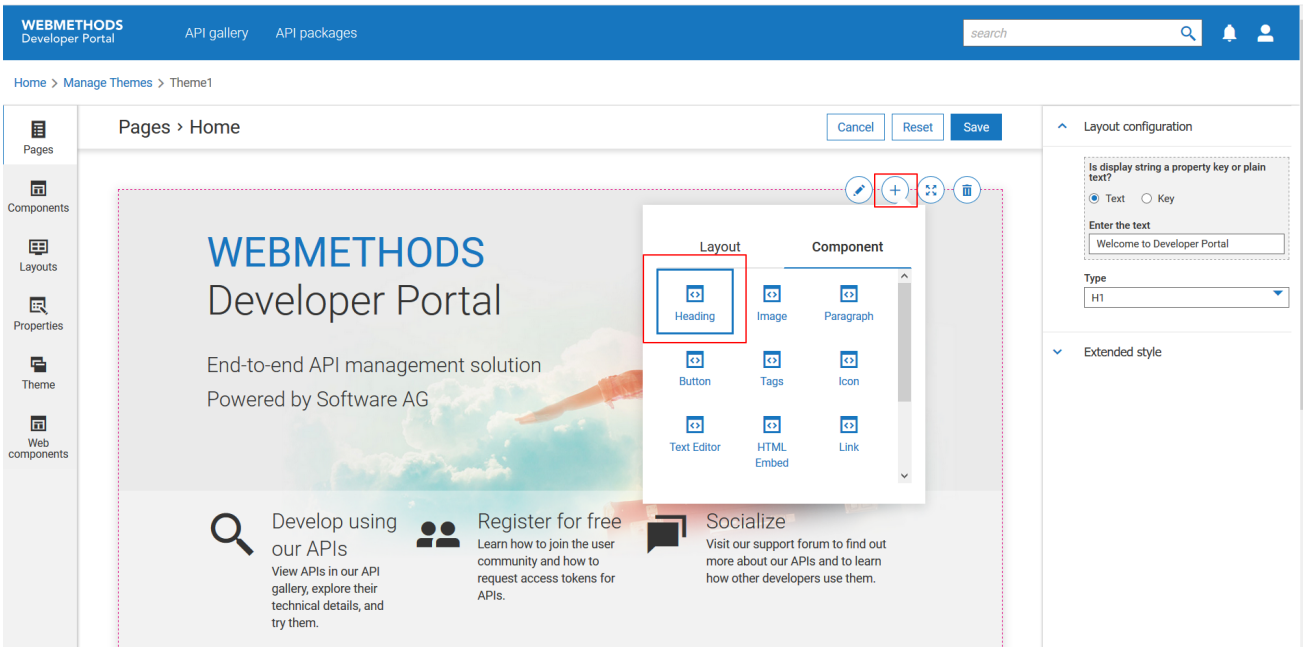
Ensure that you have created a theme or have a theme to customize.

To add a block and components

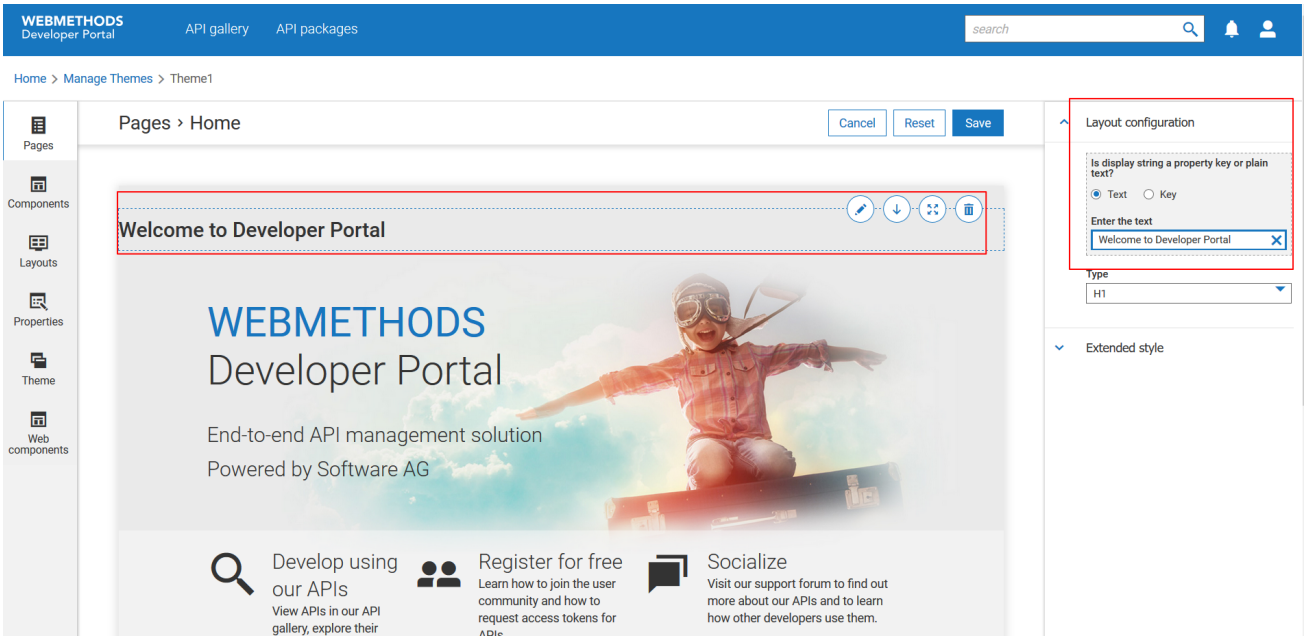
1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Pages** and select **Home**.

   The **Home** page appears with the corresponding editing options for each of the blocks.

3. Move your mouse pointer top-right corner of the page and click the add icon  .
4. Select *Heading* from the **Component** section.



5. Move your mouse pointer over the new block and click the edit icon .
6. Provide the heading text in the **Layout configuration** section.



7. Click **Save**.

   Your changes are saved.

Alternative steps:

You can select one of the following layouts or components for your new block from the **Layout** and **Components** sections respectively:

- Layouts

Layout	Description
Horizontal 1	Block with one horizontal panel.
Vertical 1	Block with one vertical panel.
Horizontal 2	Block with two horizontal panels.
Horizontal 3	Block with three horizontal panels.
Composite 1	Block with one horizontal and two vertical panels.
Composite 2	Block with three horizontal and one vertical panels.
Composite 3	Block with three horizontal panels.

Note:
This table lists only the default layouts. This section will also include the custom layouts to allow you select them. You can create custom layouts from the **Layouts** section.

- Components

Component	Description
Heading	Inserts plain text or a UI label as a heading.
Image	Inserts a default image. You can replace with the required image from the **Layout Configuration** section.
Paragraph	Inserts a paragraph that you can use to provide plain text. This is suitable to provide paragraphs that are more than one sentence.
Button	Inserts a default button.
Tags	Inserts a tag.
Icon	Inserts a default icon. You can replace with the required image from the **Layout Configuration** section.
Text editor	Inserts a text editor. This is suitable to provide paragraphs that are more than one sentence.
HTML embed	Inserts a HTML-embedded text block.

Component	Description
Link	Inserts a link. You can insert links that users can click to navigate to other web pages.
Web component	Inserts a registered web component.

- You can add multiple components in a block. You can preview them as you customize and proceed accordingly. For information on customizing the newly added blocks, see "How do I customize a block on a page?" on page 62.

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

How do I move blocks in a page?

You can move blocks within a page and cannot move across pages.

This use case starts when you want to move a block and ends when you have saved your changes.

In this example, you move down the three headings above the images in the **Welcome** page of the theme *Theme1*.

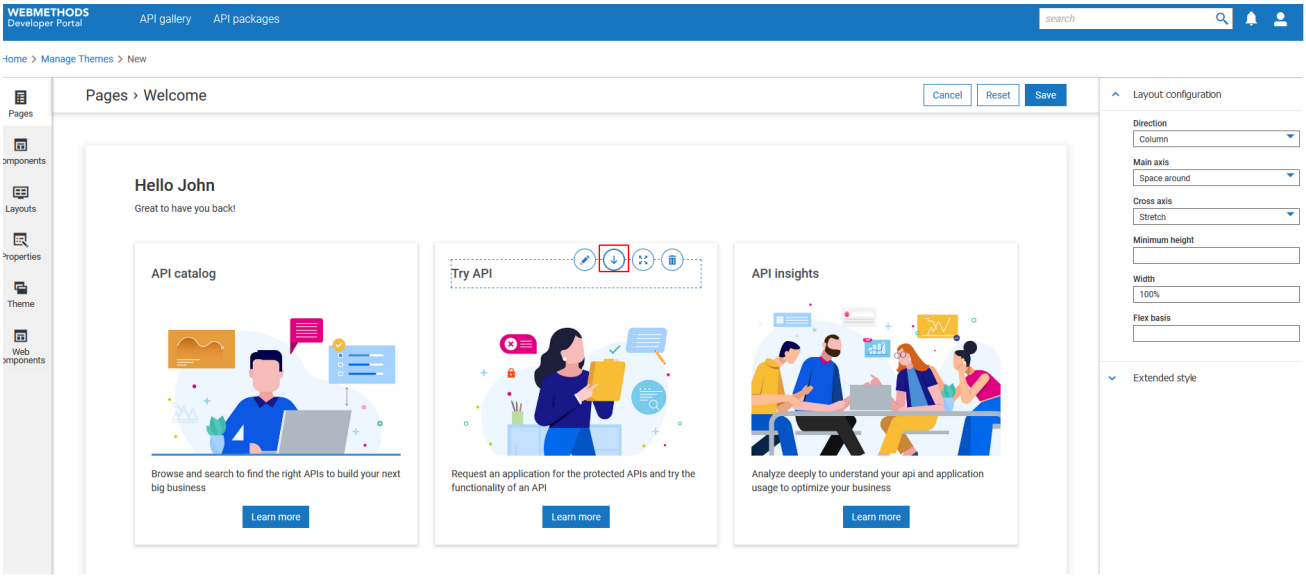
Before you start

Ensure that you have created a theme or have a theme to customize.

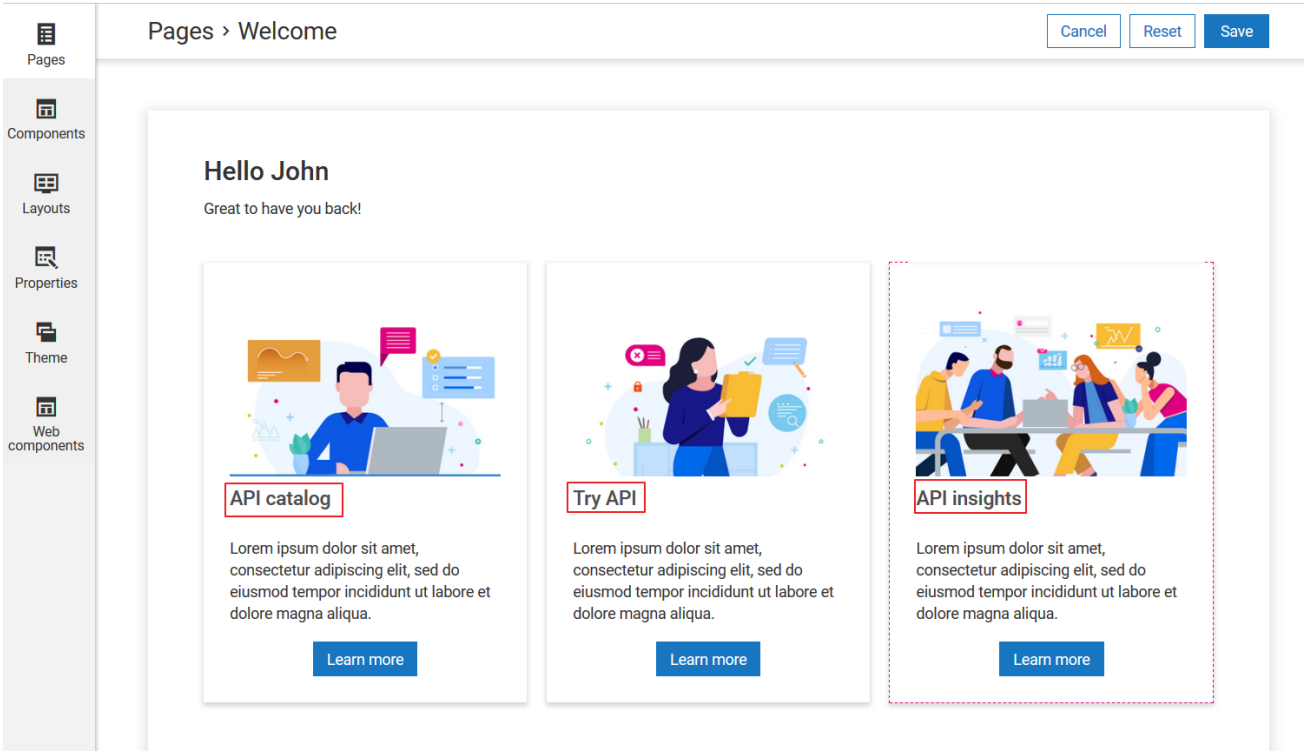
To move a block:

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Pages** and select **Welcome**.

   The **Welcome** page appears with the corresponding editing options for each of the blocks.
3. Move your mouse pointer over the image heading and click  .



4. Repeat for the other two headings.



5. Click **Save**.

   Your changes are saved.

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

How do I remove a block from a page?

This use case starts when you want to remove a block and ends when you have removed.

In this example, you remove the breadcrumbs section from the API gallery page of the theme, *Theme1*

Before you start

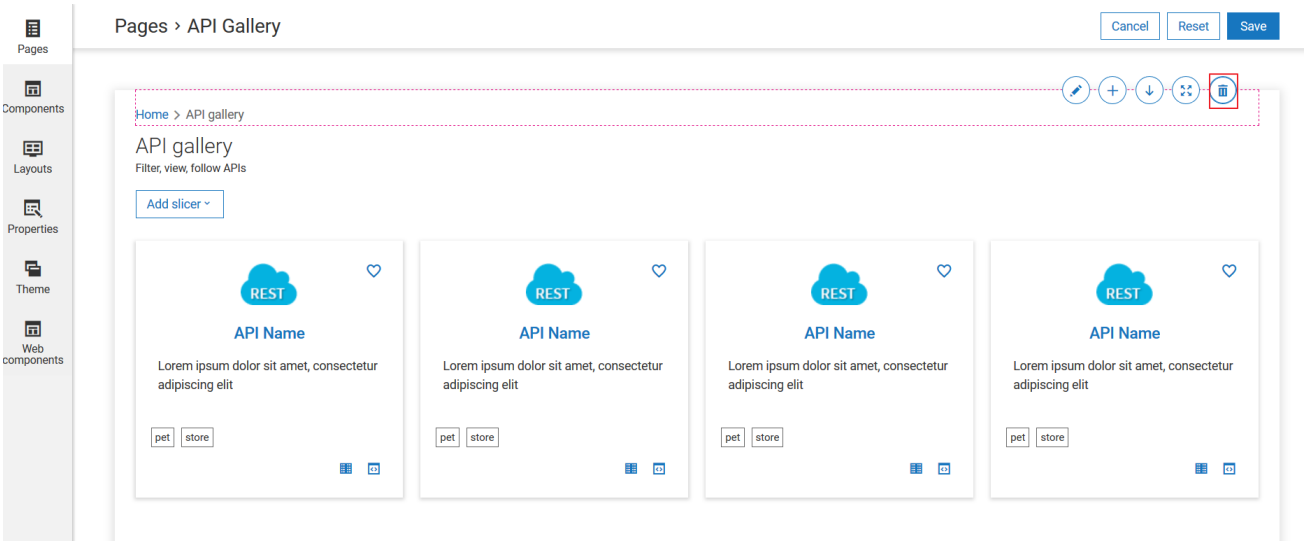
Ensure that you have created a theme or have a theme to customize.

To remove a block

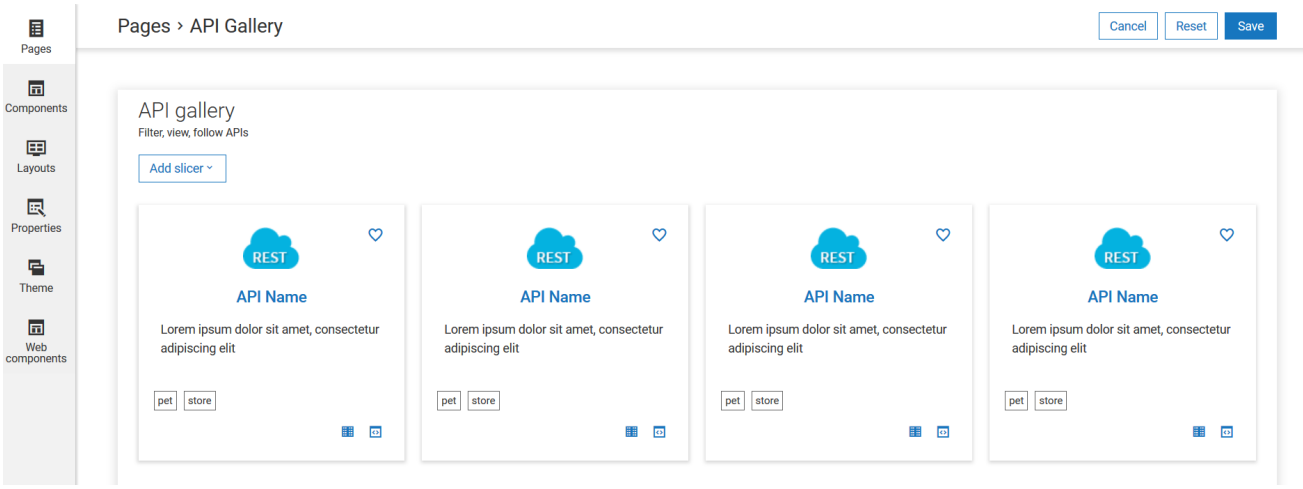
1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Pages** and select **API gallery**.

   The **API gallery** page appears with the corresponding editing options for each of the sections.

3. Move your mouse pointer over the breadcrumbs section and click  .



The block is deleted from the page.



4. Click **Save**.

   Your changes are saved.

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

How do I add a page?

You can create links to access the page from any other page or top navigation bar.

This use case starts when you want to add a page and ends when you have added the page.

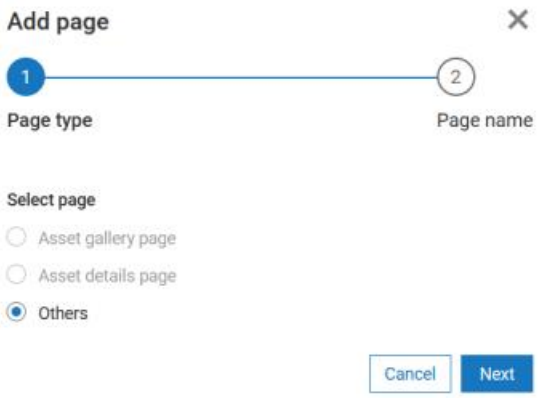
In this example, you add a new page *Page1*, in the theme, *Theme1*.

Before you start

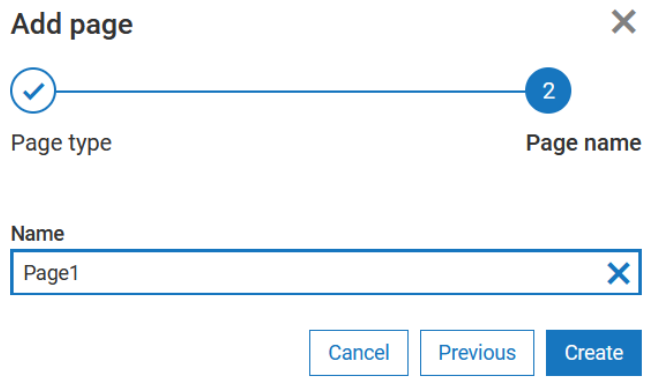
Ensure that you have created a theme or have a theme to customize.

To add a page

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Pages** and click **Add**.
3. Select **Others** from the **Add page** page and click **Next**.



4. Provide the page name, *Page1* in the **Name** field.



   The page is added.

5. Click **Save**.

   Your changes are saved.

Next steps:

- Add new blocks to the page and customize them.
- Assign or restrict the access of page to certain users. For information about assigning access to a custom page, see "How do I assign access to a custom page?" on page 77.
- Click the activate icon ✓ next to theme in the **Manage themes** screen to activate the changes.

Alternative steps:

- In *Step 3*, select one of the following options:
  - **Asset gallery page**. Adds a custom asset gallery page. This option is enabled only when you have added a new custom asset type.

- **Asset details page**. Adds a custom asset details page. This option is enabled only when you have added assets for a new custom asset type.
- **Others**. Adds a new page. Use this option to add any page other than the asset gallery or asset details pages.

How do I assign access to a custom page?

When you add a custom page, all users can access the page. You can assign or remove permissions, to certain users, for a custom page by selecting the required privilege, community, groups, or individual users. For information on adding a custom page, see "How do I add a page?" on page 75.

You can assign permission to:

Permission assigned to	Description
Community	All members of the community. That is, the users and the groups that are the part of the community will have permission to access the page.
Group	Users of the group will have permission to access the page.
Privilege	Users with the selected privilege will have permission to access the page.
Users	Specified users will have permission to access the page.

Users who are not assigned permission for a custom page cannot access the page. However, the users with the *Administrators* privilege can access custom pages irrespective of the permissions specified at the page level.

Note:
When you migrate a custom page from Developer Portal 10.11, it is accessible as it was in the 10.11 version. You can assign permissions to the required users based on your requirements.

This use case starts when you have to allow only certain users to access a custom page and ends when you have restricted the access.

In this example, you allow access to the page *Page1*, only to the users who are part of the *Developers* community.

Before you start

Ensure that you have:

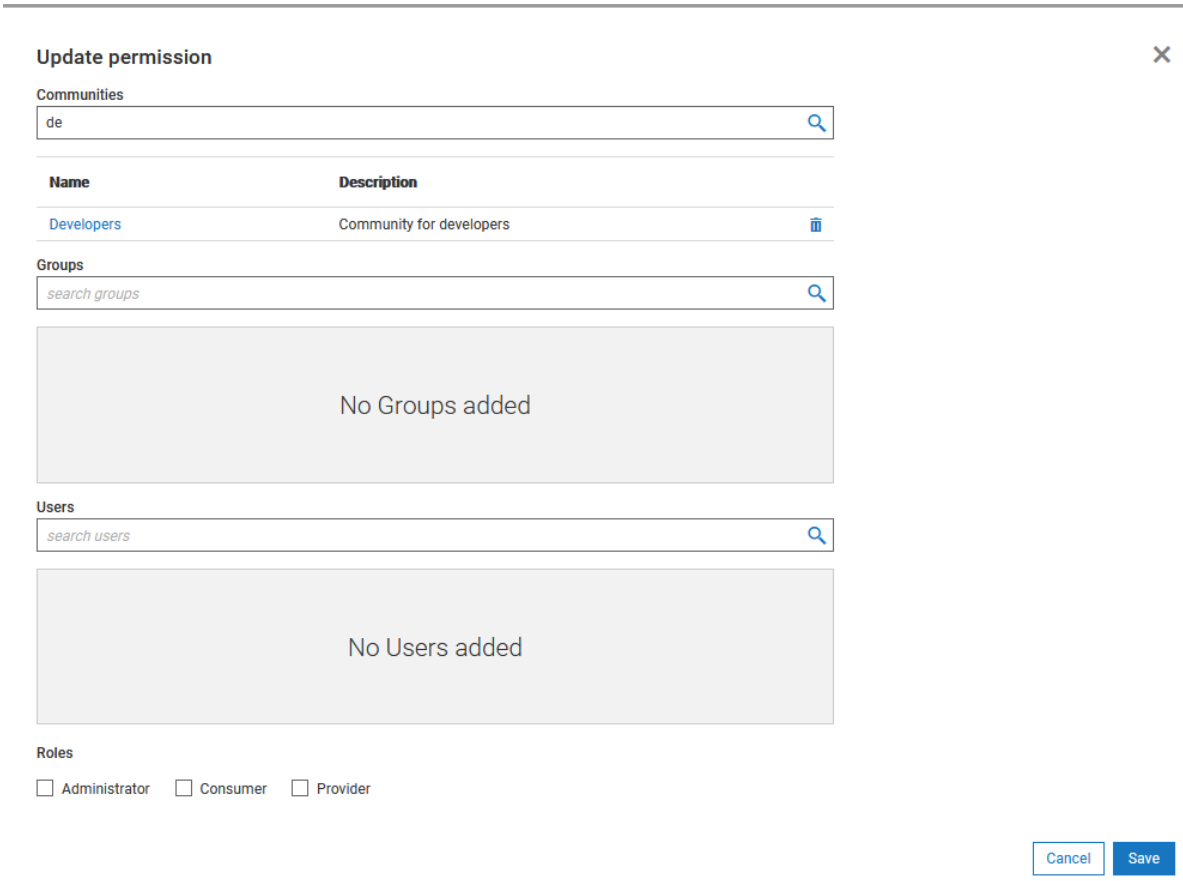
- Created a custom page and added links to access the same.

To assign access to a custom page

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.

2. Select **Pages** and click *Page1*.
3. Click the assign permission icon.

   The **Update permission** page appears.



4. Select *Developers* from the **Communities** field.
5. Click **Save**.

   The assigned permission is saved.
6. Click **Save**.

   The changes are saved.

Alternative steps:

- Select the required groups, users, or select the privilege, in *Step 4*, to who you want to assign access to the custom page.

Next steps:

- Modify the permissions of a custom page anytime by clicking the .

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

Customize UI Components

The **Components** section is used to customize the default UI components of Developer Portal. They include:

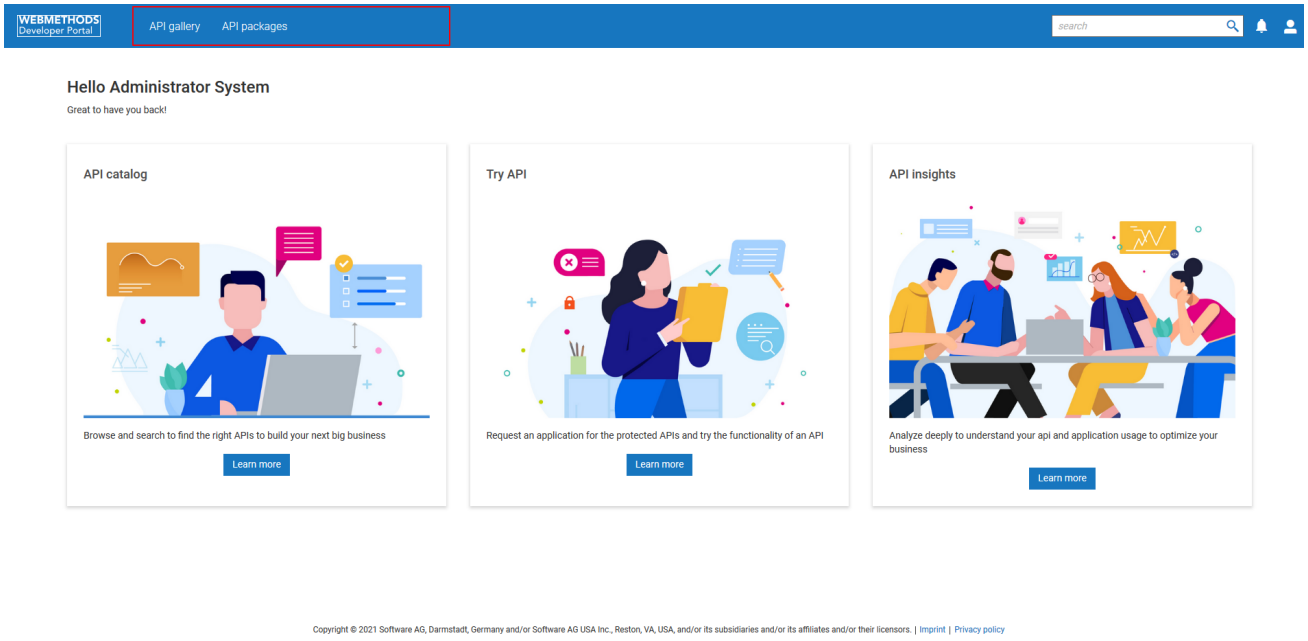
- Top navigation bar. For more information, see "How do I customize the top navigation bar?" on page 82.
- Registration form. For more information, see "How do I customize the Sign up page?" on page 83.
- API default box. For more information, see "How do I customize the API grid displayed in API gallery?" on page 85.
- Package default box. For more information, see "How do I customize the package grid?" on page 88.
- API details side bar. For more information, see "How do I customize the API details pane?" on page 89.
- Plan default box. For more information, see "How do I customize the API grid displayed in API gallery?" on page 85.

Before you begin the following use cases:

Ensure that you have created a theme or have a theme to customize.

How do I add a new item to the top navigation bar?

The top navigation bar of Developer Portal includes links to the API gallery and API packages pages by default.



You can customize this section to:

- Add a new item to the top navigation bar
- Edit the details of an existing items
- Modify the order of items
- Remove an item from the top navigation bar

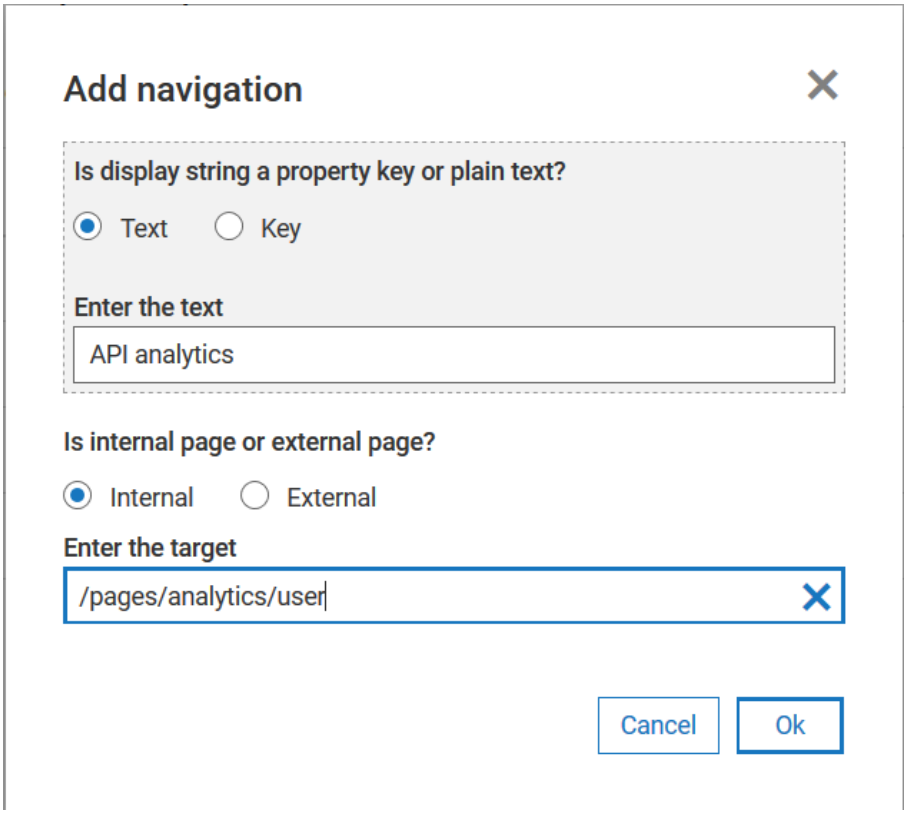
This use case starts when you want to add a new item to the top navigation bar and ends when you have added.

In this example, you add the link to Dashboard page in the top navigation page for the theme, *Theme1*.

To add a new item in the top navigation bar

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Components** and click **Top navigation**.
3. Click **Add navigation**.
4. Select **Text** from the **Is display string a property key or plain text?** field and provide API analytics.
5. Select **Internal** and provide the page link in the **Enter the target** like given below:

```
analytics/user
```



6. Click **Ok**.
7. Click **Save**.

   Your changes are saved and the new item appears in the top navigation bar.

Alternative steps:

- In step 4, select one of the following options based on the type of text to be displayed for the link from the **Is display string a property key or plain text?** field:
  - **Text**. To display plain text, and provide the text to be displayed.
  - **Key**. To display an UI label as a, and provide the required key.
- Provide an internal or external link from the **Enter the target** field. The internal link must be given in the following format:

  Internal pages

  ```
  /page name
  ```

  For example,

  ```
  /apis
  ```

  For custom pages, provide link in the following format:

  ```
  /pages/{page_name}
  ```

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

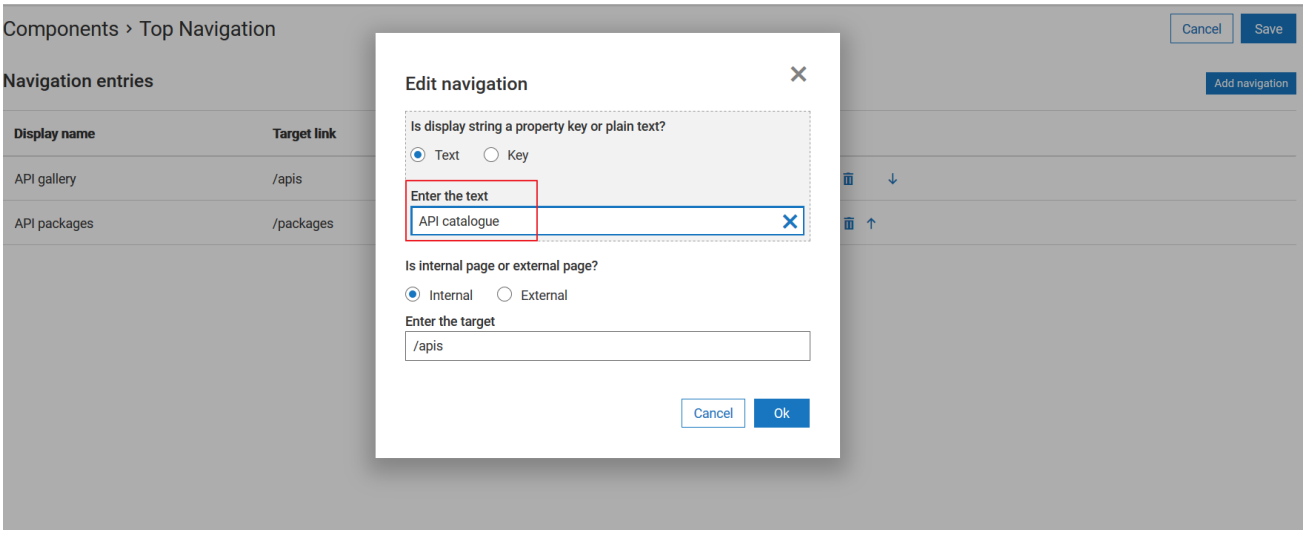
How do I customize the top navigation bar?

This use case starts when you want to customize the top navigation bar and ends when you have completed the customization.

In this example, you modify the *API gallery* as *API catalogue* for the theme, *Theme1*.

To customize the top navigation bar

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Components** and click **Top navigation**.
3. Click the edit icon next to **API gallery**.
4. Select **Text** and provide API catalogue in the **Enter the text** field.



5. Click **Ok**.
6. Click **Save**.

   Your changes are saved.

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

How do I customize the Sign up page?

The **Sign up** page includes the following fields, by default:

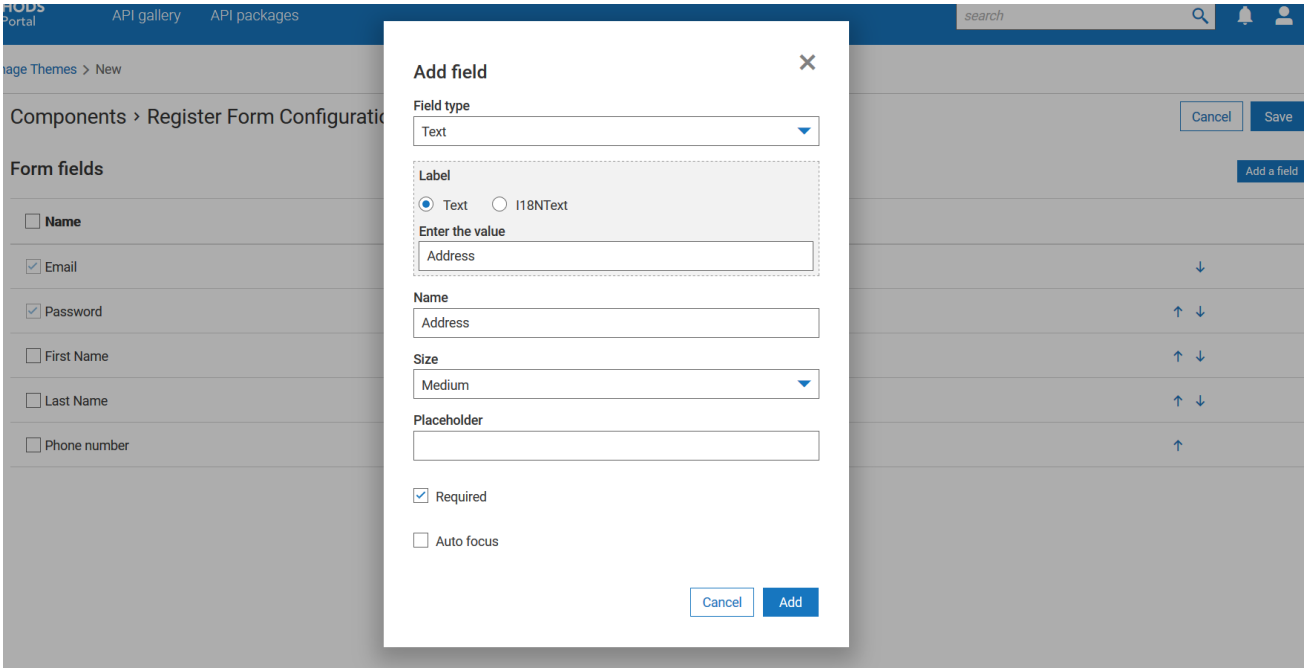
- Email
- Password

You can add new fields in the registration form to get more details from your users. You cannot edit or remove the existing default fields. However, you can move them up or down to change their order of appearance in the registration form.

In this example, you add the field, *Address* in the **Sign up** page for the theme, *Theme1*.

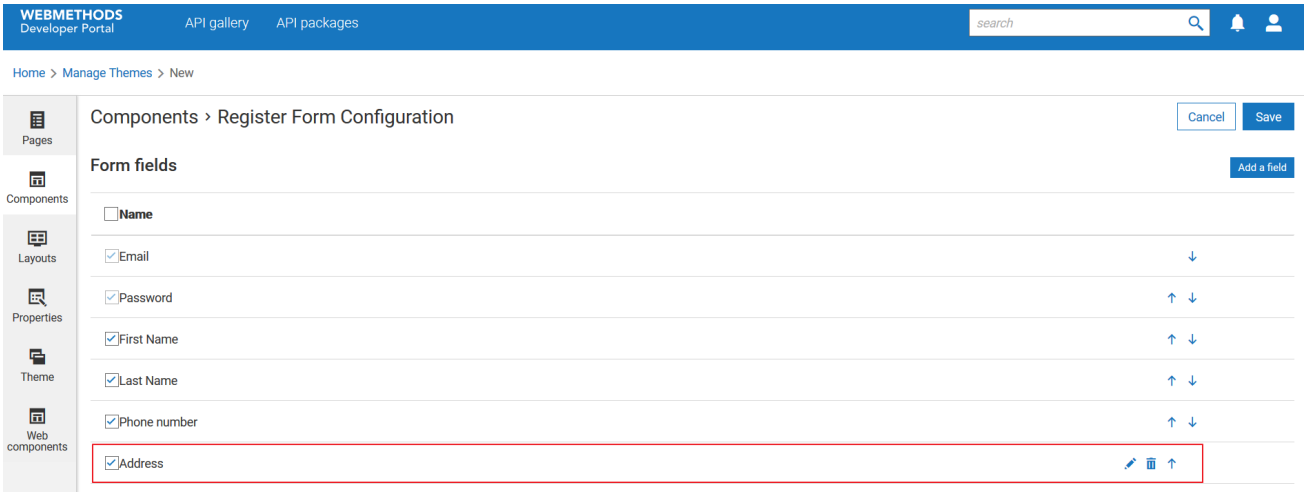
To customize the Sign up page

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Components** from the left pane and select the **Registration form**.
3. Click **Add a field**.
4. Select *Text* from the **Field Type** list.
5. Select *Text* from the **Label** section.
6. Provide *Address* in the **Name** field.
7. Select Medium from the **Size** list.
8. Select the **Required** check box to indicate that the field is mandatory for user registration.



9. Click **Add**.

   The new field appears.



10. Click **Save**.

    Your changes are saved.

Alternative steps:

1. In *Step 4*, add any of the following types of fields in the **Sign up** page. Select the required option from the **Field Type** list:

   - **Password**. To add a field that allows users provide their password.
   - **Select**. To add a field with options in a drop-down list.

- **Checkbox**. To add a field with options as check boxes.
- **Radio**. To add a field with options as radio buttons.

2. In *Step 5*, select the type of label to be displayed for the field from the **Label** section:
   - **Text**, if the value entered in the field must appear as they are.
   - **i18N**, if the value is in other languages. You can use English or any language that is supported by i18N standards.
3. Select the **Auto focus** check box to display the cursor in the new field by default.
4. *Optional*. If you have selected `Select`, `Checkbox`, `Radio` in the **Field type** drop-down list, click **Add new option** and provide the possible options.
5. Select the edit icon next to the field that you want to edit.

   You cannot edit the default fields.
6. Use the move up icon and the move down buttons next to a field to modify its position.
7. Click the delete icon next to a field to removed it.

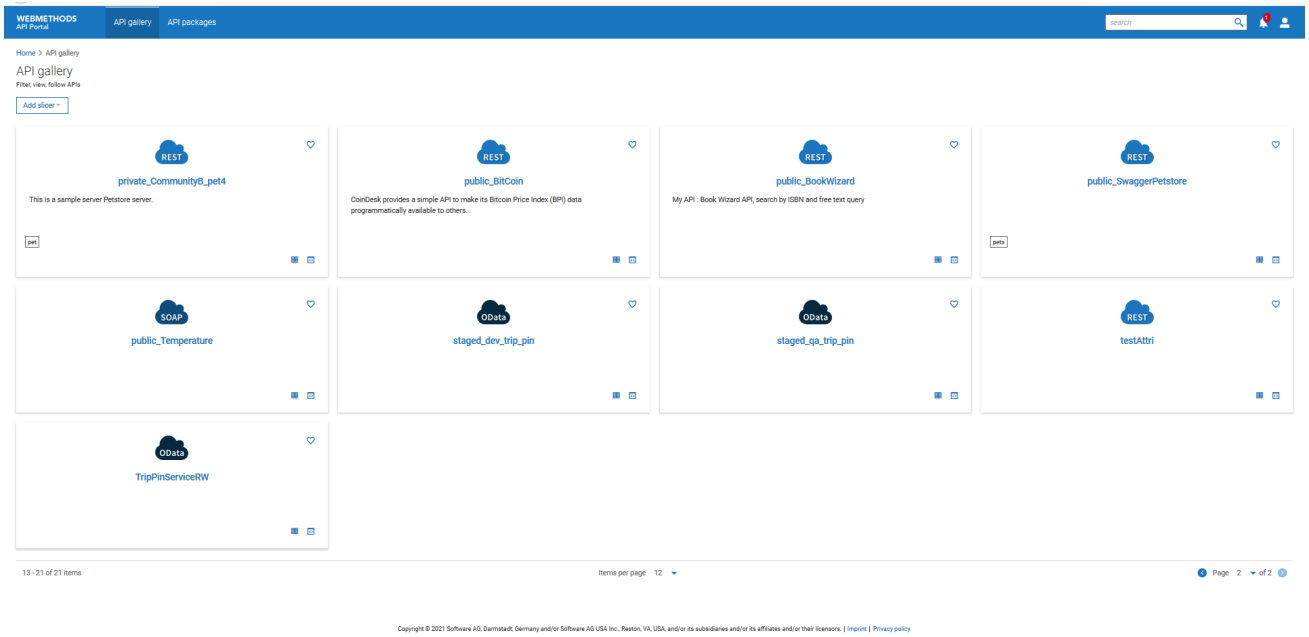
   You cannot remove default fields.
8. Select or clear the check box next to a field name to include or remove it respectively from the sign up page.

Next steps:

- Click the activate icon next to theme in the **Manage themes** screen to activate the changes.

How do I customize the API grid displayed in API gallery?

APIs, along with their details, are displayed as grids in API gallery. Each grid has an API and its details. You can customize the layout and details of the API grid to suit your requirements.

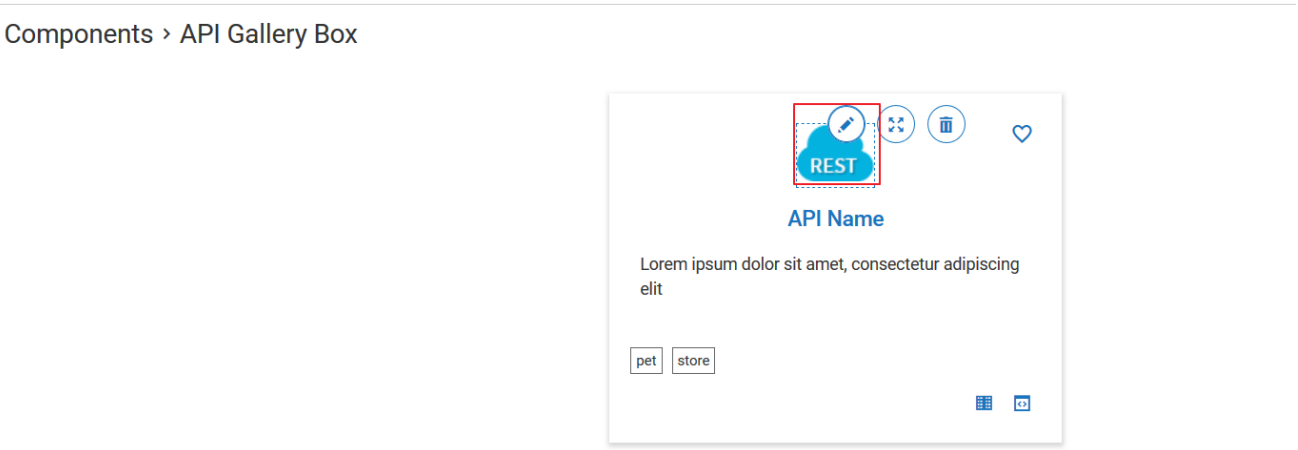


This use case starts when you want to customize API grid and ends when you have saved the changes.

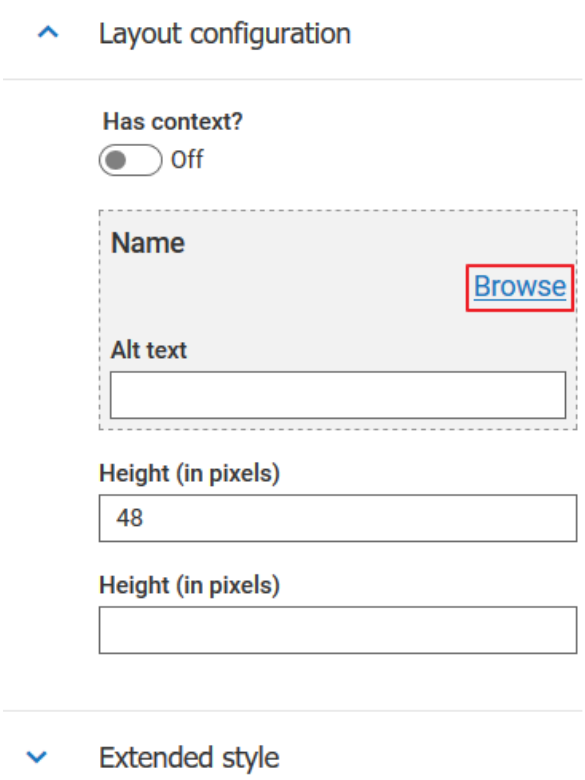
In this example, you edit the image displayed in the API grid for the theme, *Theme1*

To customize API grid

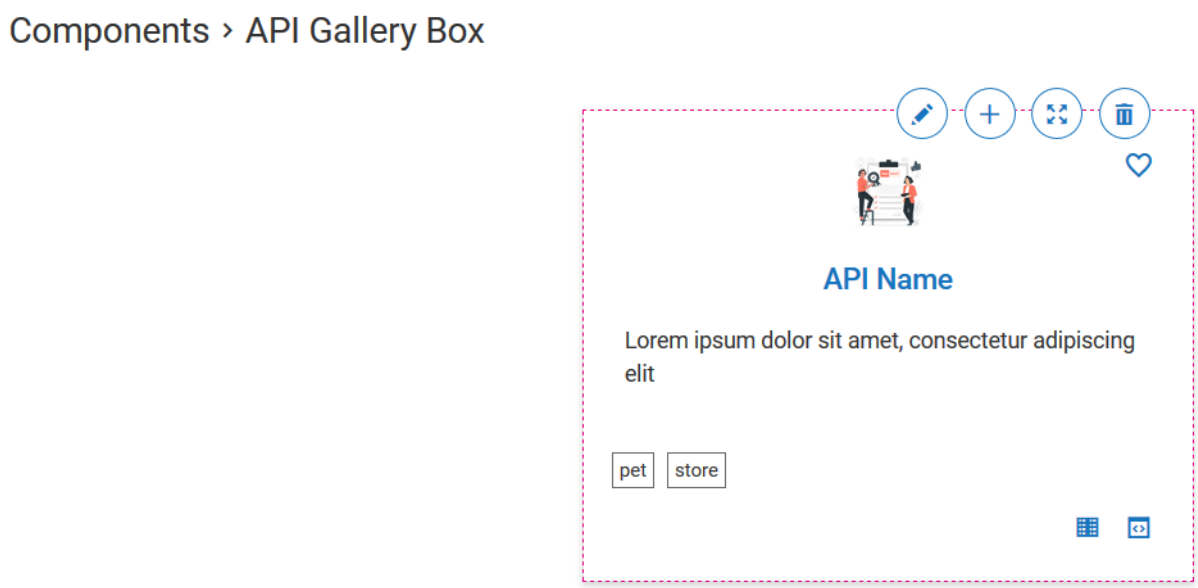
1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Components** from the left pane and select the **API default box**.



3. Move your mouse pointer over the image icon and click the edit icon  .



4. Click **Browse** and select the required image.



5. Click **Save**.

   Your changes are saved.

Alternative steps:

- Perform any of the following:
  - Add a new block to the grid. For information on adding a block, see "How do I add a new block and component?" on page 69.
  - Customize a block on the grid. For information on customizing a block, see "How do I customize a block on a page?" on page 62.
  - Modify the order of blocks on the grid. For information on moving the blocks, see "How do I move blocks in a page?" on page 72.
  - Remove a block from the grid. For information on removing a block, see "How do I remove a block from a page?" on page 74.

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

How do I customize the package grid?

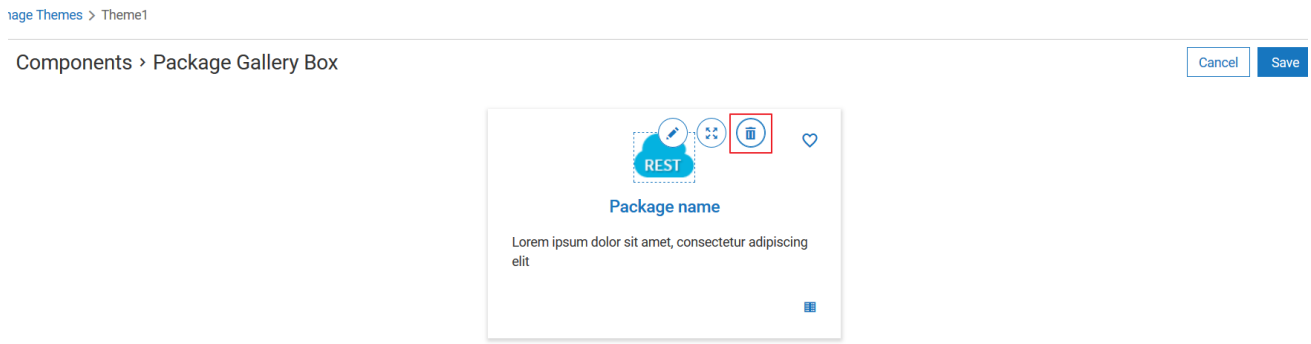
API packages, along with their details, are displayed as grids in the API packages page. Each grid has a package and its details. You can customize the layout and details of the grid to suit your requirements.

This use case starts when you want to customize API package grid and ends when you have saved the changes.

In this example, you remove the image component in the package grid for the theme, *Theme1*

To customize API package grid

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Components** from the left pane and select the **Package default box**.



3. Move your mouse pointer over the image and click the delete icon  .

4. Click **Save**.

   Your changes are saved.

Alternative steps:

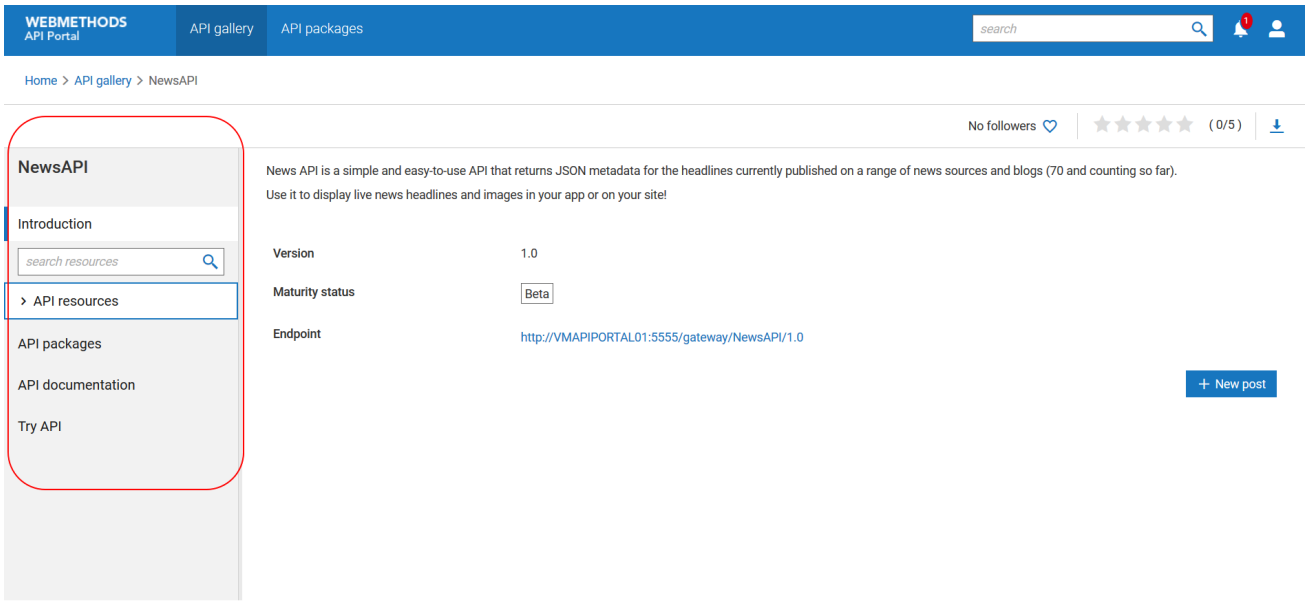
- Perform any of the following:
  - Add a new block to the grid. For information on adding a block, see "How do I add a new block and component?" on page 69.
  - Customize a block on the grid. For information on customizing a block, see "How do I customize a block on a page?" on page 62.
  - Modify the order of blocks on the grid. For information on moving the blocks, see "How do I move blocks in a page?" on page 72.
  - Remove a block from the grid. For information on removing a block, see "How do I remove a block from a page?" on page 74.

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

How do I customize the API details pane?

You can customize the entries displayed in the left pane of the **API details** screen to suit your requirements.

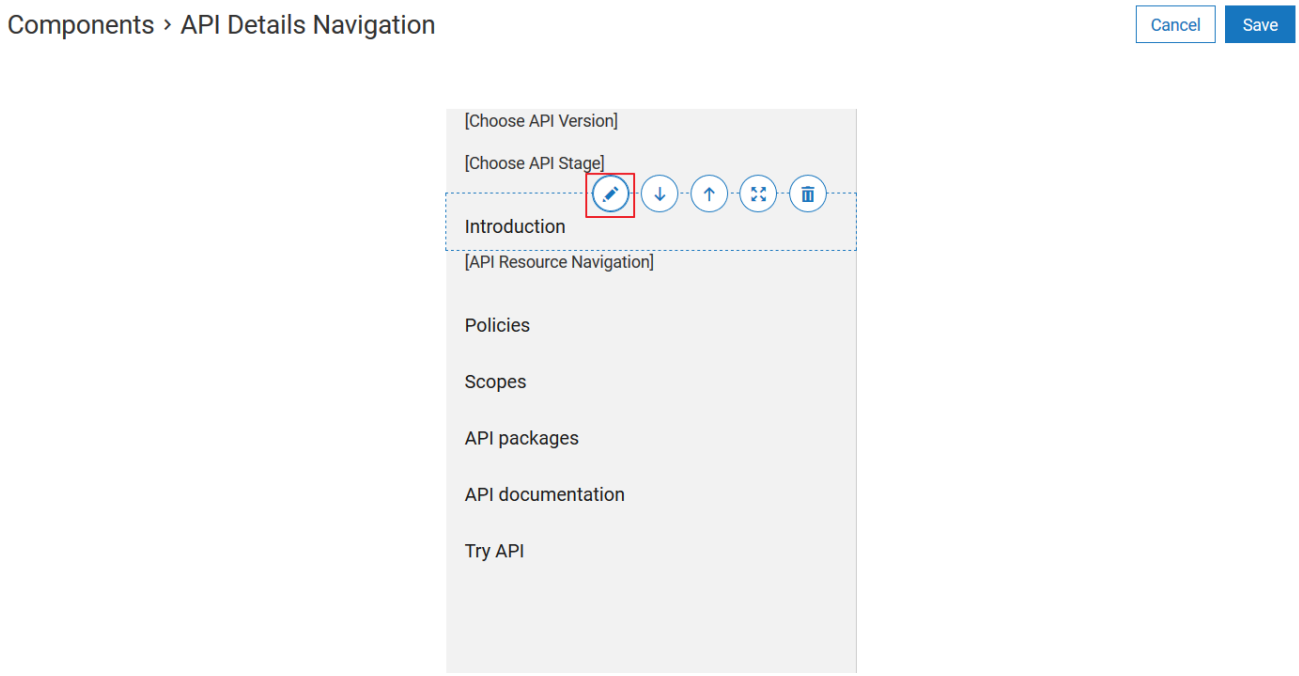


This use case starts when you want to customize the API details pane and ends when you have saved your customization.

In this example, you change the text *Introduction* to *API Overview* for the theme, *Theme1*.

To customize the API details pane

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Components** from the left pane and select the **API details side bar**.
3. Move your mouse pointer over the component and click the edit icon  .



4. From the **Layout configuration** section, select **Text**.
5. Provide *API Overview* in the **Enter the key** field.

6. Click **Save**.

Your changes are saved.

Alternative steps:

- Perform any of the following:
  - Add a new block to the grid. For information on adding a block, see "How do I add a new block and component?" on page 69.
  - Customize a block on the grid. For information on customizing a block, see "How do I customize a block on a page?" on page 62.
  - Modify the order of blocks on the grid. For information on moving the blocks, see "How do I move blocks in a page?" on page 72.
  - Remove a block from the grid. For information on removing a block, see "How do I remove a block from a page?" on page 74.
- You can also add the following components that are exclusive to the API details pane:

Component	Description
Global search	Inserts search box that allows users to search for API resources.
Stage chooser	Inserts option that allows users to choose the required API stage.
Version chooser	Inserts option that allows users to choose the required API version.
Comment streams	Inserts the comments stream section.
API Nav Item	Inserts link to the configured page. Click  next to the component and configure the required page in the **Layout configuration** section.
Download SDK	Inserts the **Download SDK** button that allows users to download client SDK for an API resource..

Next steps:

- Click the activate icon ✓ next to theme in the **Manage themes** screen to activate the changes.

How do I customize the plans grid?

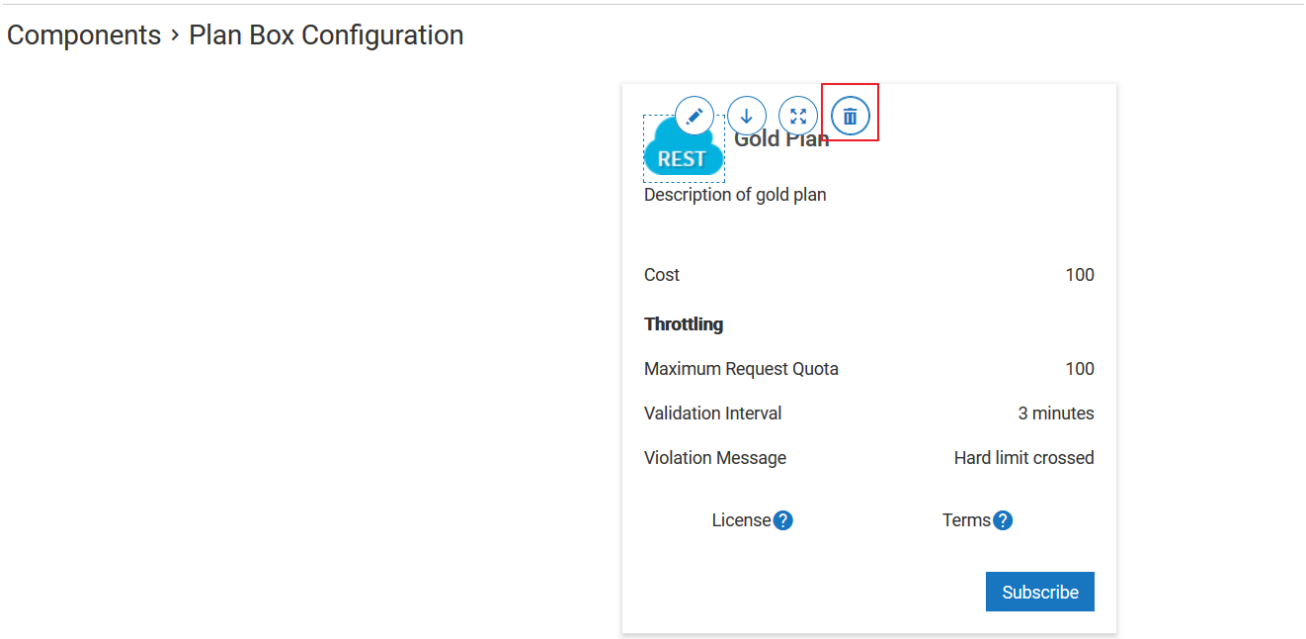
API subscription plans, along with their details, are displayed as grids in the API packages details page. Each grid has a plan and its details. You can customize the layout and details of the grid to suit your requirements.

This use case starts when you want to customize the plans grid and ends when you have saved your customization.

In this example, you remove the image component in the plans grid for the theme, *Theme1*

To customize the API plan grid

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Components** from the left pane and select the **Plan default box**.



3. Move your mouse pointer over the image and click the delete icon  .
4. Click **Save**.

   Your changes are saved.

Alternative steps:

- Perform any of the following:
  - Add a new block to the grid. For information on adding a block, see "How do I add a new block and component?" on page 69.
  - Customize a block on the grid. For information on customizing a block, see "How do I customize a block on a page?" on page 62.
  - Modify the order of blocks on the grid. For information on moving the blocks, see "How do I move blocks in a page?" on page 72.
  - Remove a block from the grid. For information on removing a block, see "How do I remove a block from a page?" on page 74.

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

How do I add the language switcher icon?

The language switcher icon allows users to select their preferred language to view Developer Portal. By default, the application appears in the language configured by the administrator. For information about configuring default language, see "Configuring Default Language" on page 34.

In this example, you add the language switcher icon to the header section through customization.

Before you start

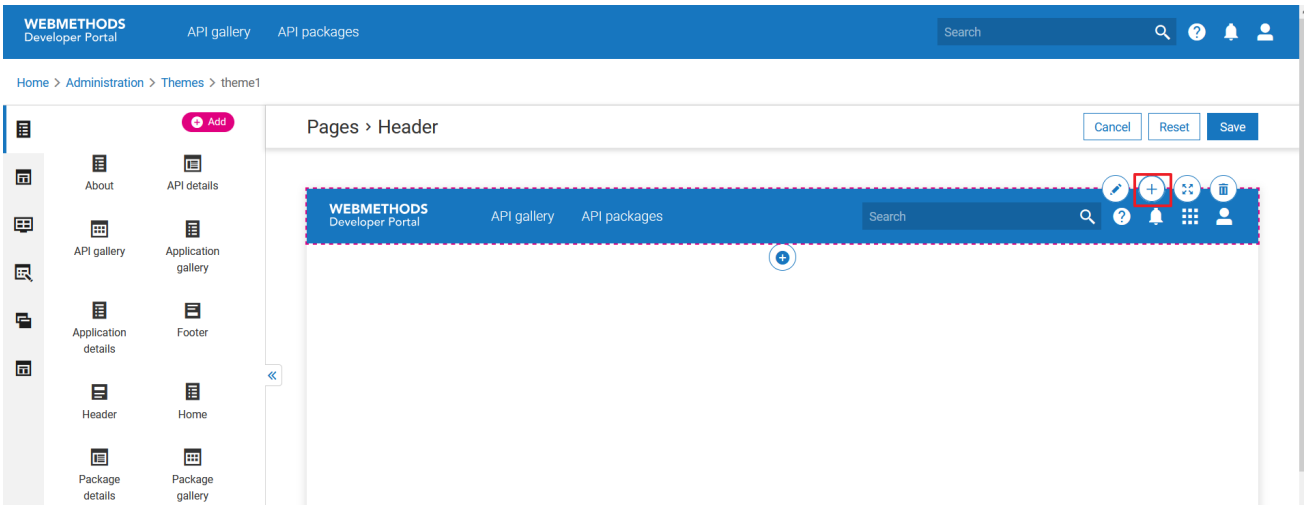
Ensure that you have created a theme or have a theme to customize.

To add the language switcher icon

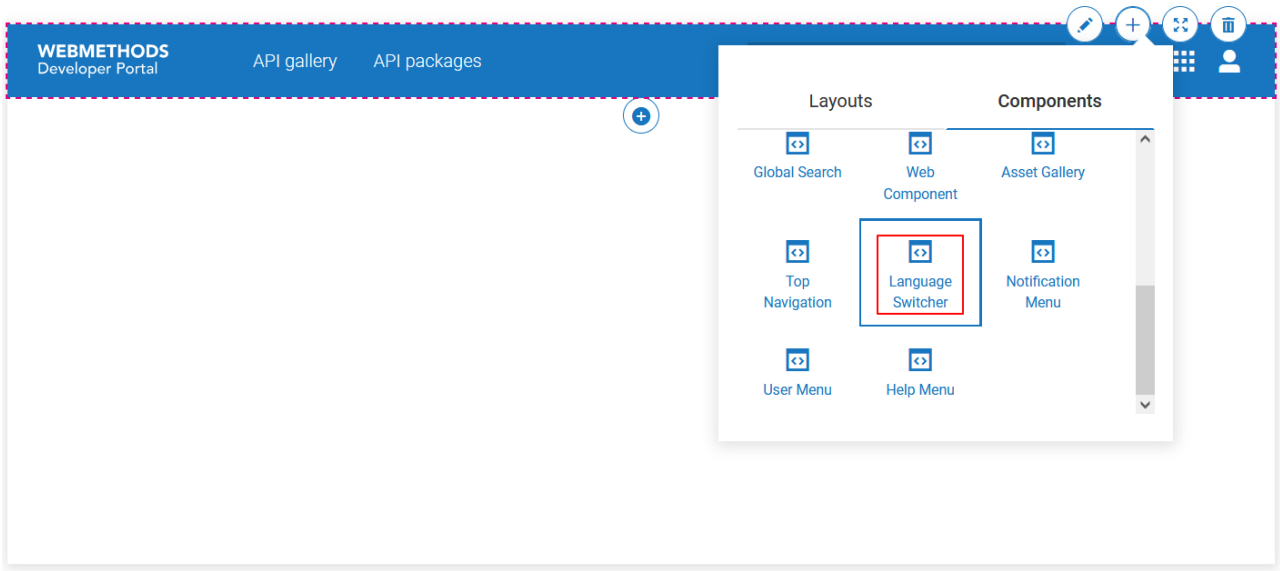
1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Pages** and select **Header**.

   The **Header** section appears with the existing components.

3. Move your mouse pointer top-right corner of the page and click the add icon  .

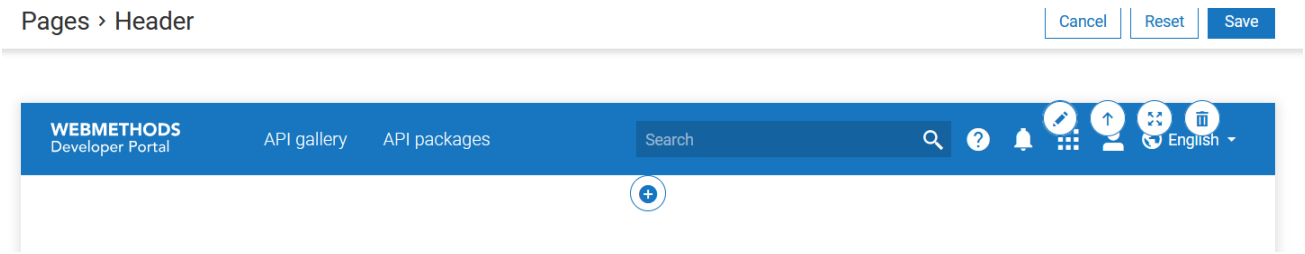


4. Select the **Components** tab and select **Language switcher**.

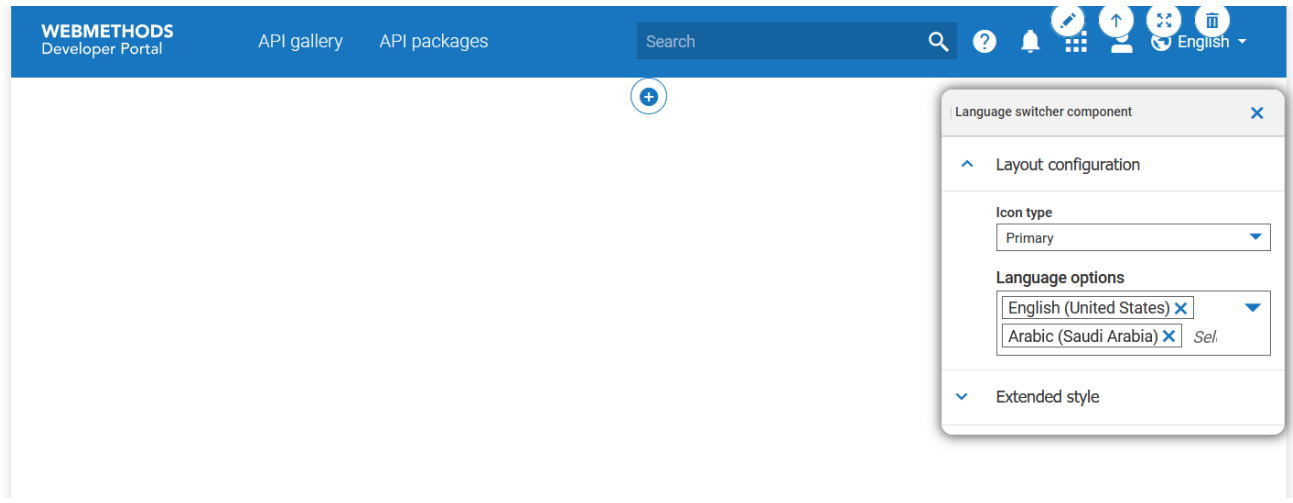


The icon is added to the **Header** section.

5. Click the locale switcher icon to view editing options.

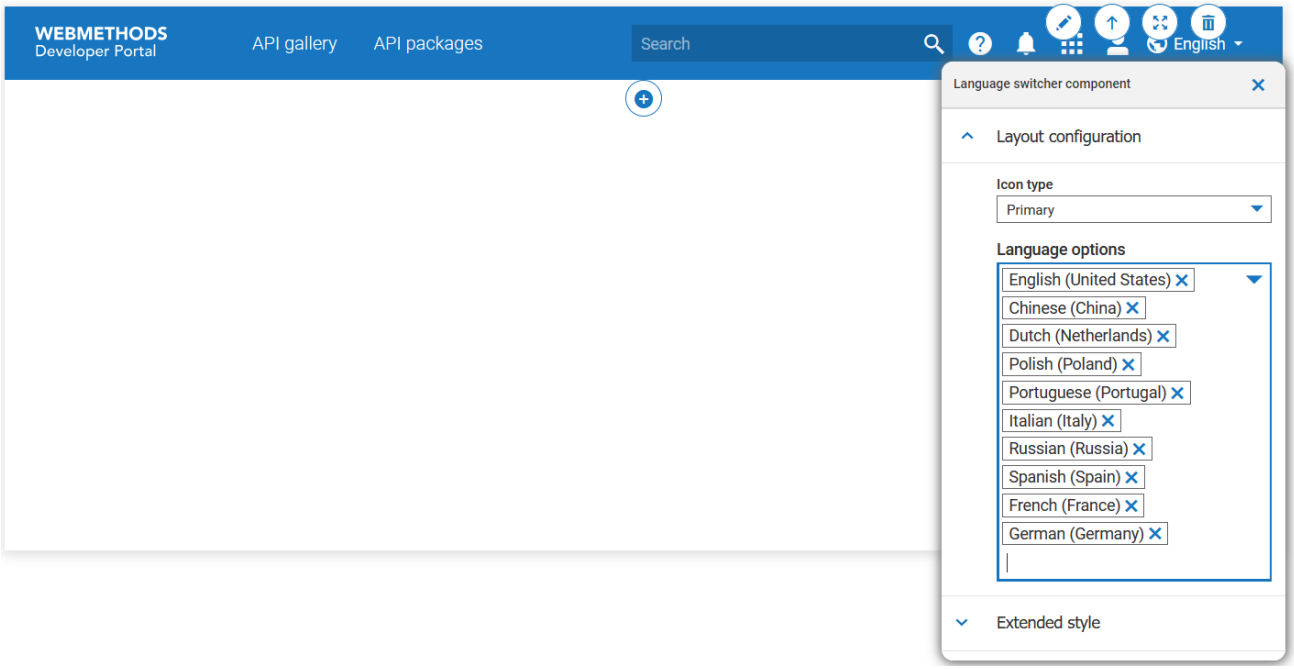


6. Click the edit icon  . The options that allow you to edit the icon appear.



7. Click anywhere in the **Language options** field to view the list of supported languages.

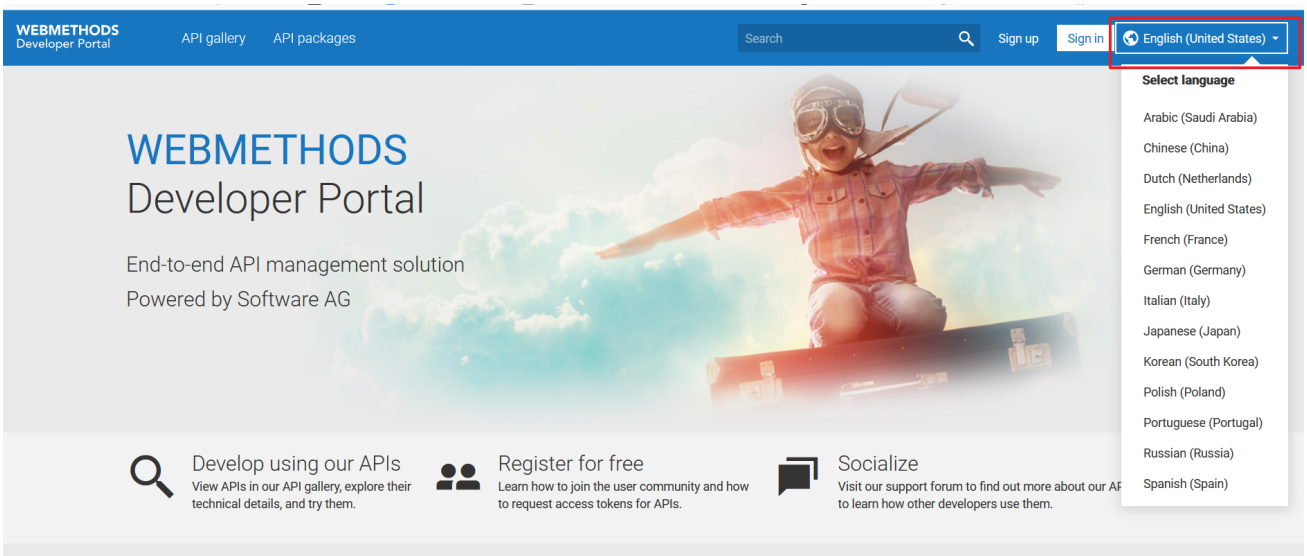
8. Click a language to add to the language switcher.



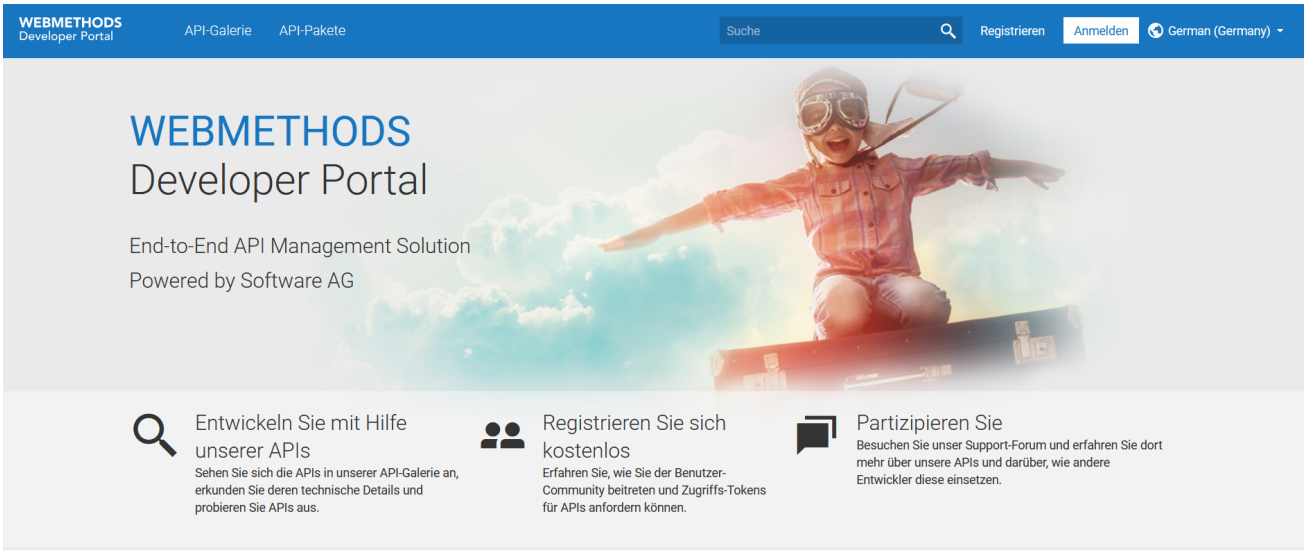
Click **Select all** to add all languages.

9. Click **Save**.

The language switcher is added to the **Header** section of the application.



You can now use this icon to select a language from the list. For example, select **German (Germany)** to view the application in *German*.



Next steps:

- If your customized theme is not activated, click the activate icon ✓ next to theme in the **Manage themes** screen to activate the changes.

Customize Labels

The **Properties** section is used to add custom labels on the Developer Portal UI such as:

- Screen names
- Field names
- Button names
- Message texts

In addition, you can also view the labels available in the UI based on the following categories:

- **Administration**. Includes the labels that are accessed by users who have the **API Administrator** privilege.
- **Provider**. Includes the labels that are accessed by users who have the **API Provider** privilege.
- **Consumer**. Includes the labels that are accessed by users who have the **API Consumer** privilege.
- **Base**. Includes the labels that appear in screens that do not require signing in to the application. For example, **API gallery** screen.

How do I add new UI labels?

You can add new labels and use them in place of existing labels or use them in the new blocks that you add to your theme.

This use case starts when you want to add a new UI label and ends when you have added labels.

In this example, you add a new label, *Request pending* for the theme, *Theme1*.

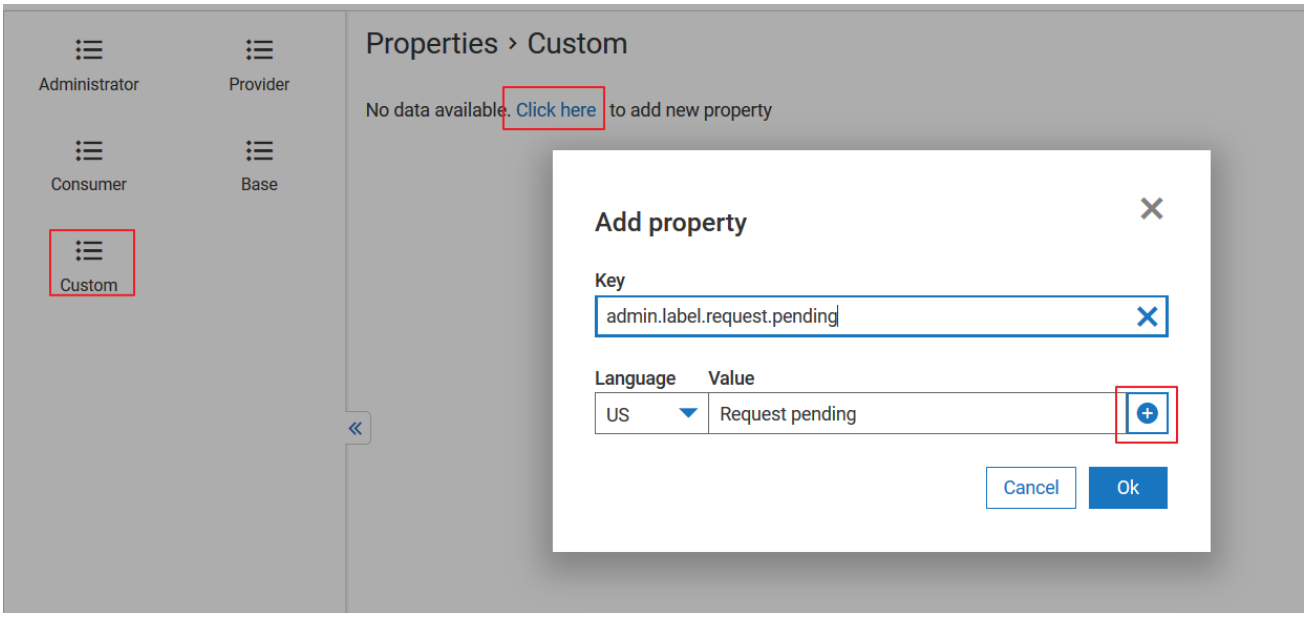
Before you start

Ensure that you have created a theme or have a theme to customize.

To add new labels in the UI

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Properties** and select **Custom**.
3. Click the **Click here** link.

   The **Add property** screen appears.



4. Provide *admin.label.request.pending* in the **Key** field.
5. Select *US* from the **Language** list.
6. Provide *Request pending* in the **Value** field.
7. Click the add icon  .

   The label is added.
8. Click **Save**.

   Your labels are saved and appear in the **Properties** screen.

Alternative steps:

- Repeat steps 3 to 7 to add more labels.

Next steps:

- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

Customize Color Schemes

The **Themes** section is used to customize the color schemes used in general and in different blocks of a page.

You can customize the colors and font size, in general, in the UI.

You can customize the color scheme used in:

- Screen headers
- Buttons
- Left navigation bar

How do I customize the color scheme used in a screen?

This use case starts when you want to customize the color scheme used in UI.

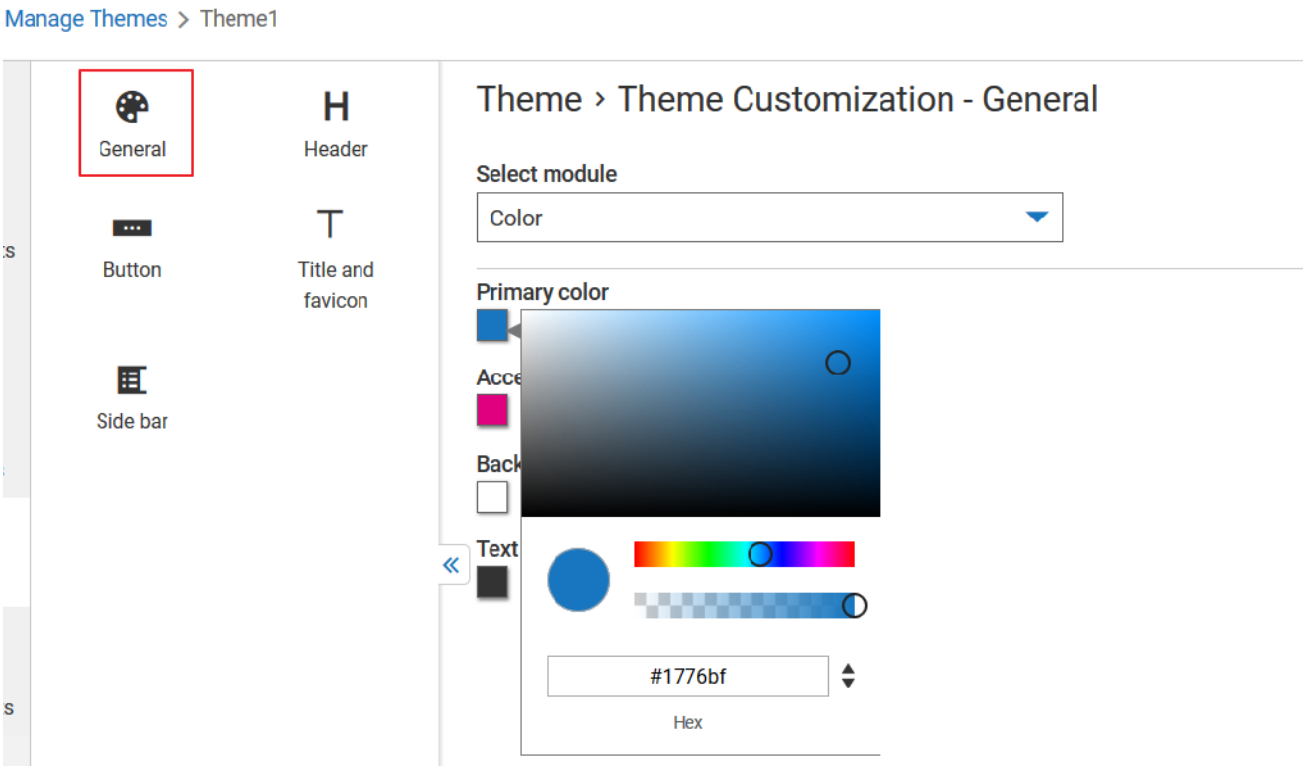
In this example, you modify the primary color of the application in general for the theme, *Theme1*

Before you start

Ensure that you have created a theme or have a theme to customize.

To customize the color schemes

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Theme** from the left pane and select **General**.
3. Select *Color* from the **Select module** list.
4. Select a color using the color slider.



5. Click **Save**.

   Your changes are saved.

Alternative steps:

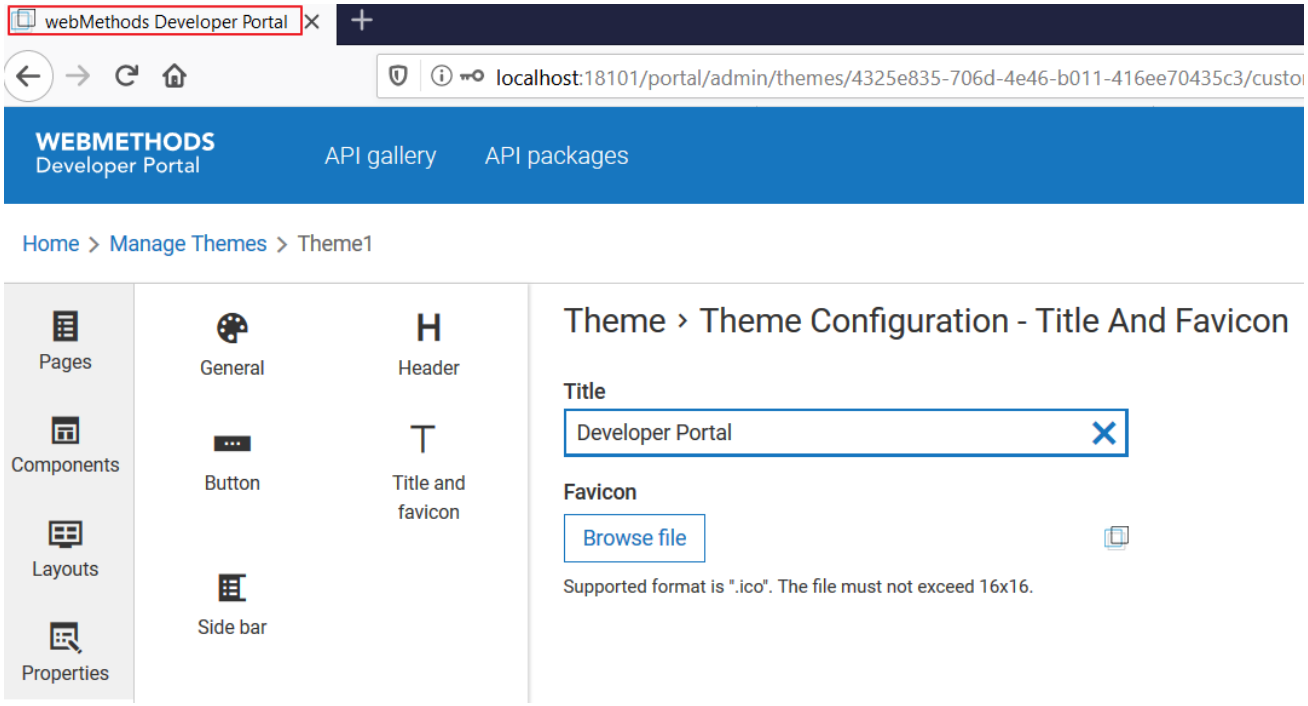
- Customize the color scheme of any of the following sections:
  - **General**. To customize the color scheme used in UI, in general.
  - **Header**. To customize the color scheme used in screen headers.
  - **Button**. To customize the color scheme used for buttons.
  - **Side bar**. To customize the color scheme used in the left navigation pane.
- Select *Font* from the **Select module** list to customize font size.

Next steps:

- Click the activate icon ✓ next to theme in the **Manage themes** screen to activate the changes.

How do I customize the text and icon displayed in browser header?

This use case starts when you want to customize the text and icon (thumbnail) displayed on browser header.



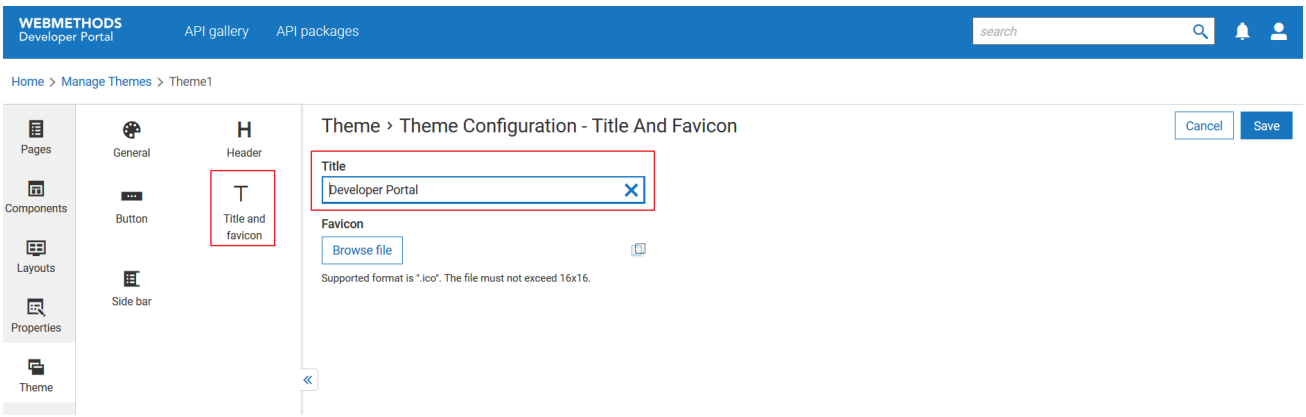
In this example, you modify the text in browser header as *Developer portal* for the theme, *Theme1*.

Before you start

Ensure that you have created a theme or have a theme to customize.

To customize the browser header

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Theme** and select **Title and Fav Icon**.
3. Provide *Developer Portal* in the **Title** field.



4. Click **Save**.

Your changes are saved.

Alternative steps:

- In *Step 3*, click **Browse** and select the icon to appear in the browser header.

Next steps:

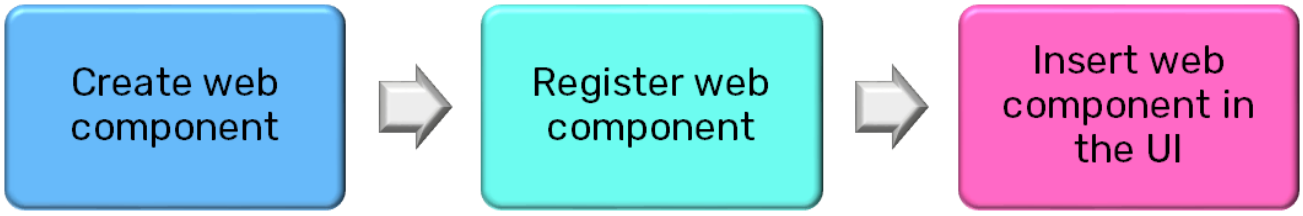
- Click the activate icon ✔ next to theme in the **Manage themes** screen to activate the changes.

Customization using Web components

Web components are a set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages.

You can use web components to add customized components and widgets to your portal.

The high level workflow of using web components for UI customization is as follows:



You can create web components using TypeScript or JavaScript. If you create your component in TypeScript, you must first compile it to JavaScript for registering it with Developer Portal.

The web component can be created by extending the following example:

```
export abstract class AbstractPortalElement extends HTMLElement {
    private context: ContextModel;

    abstract render(): void | Promise<any>;

    setContext(context: ContextModel) {
        this.context = context;
        this.render();
    }

    protected getData(): any {
        if (this.context && this.context.getData) {
            return this.context.getData();
        }
    }

    protected navigate(path: string): void {
        if (this.context && this.context.navigate) {
            this.context.navigate(path);
        }
    }

    protected getLocaleString(key: string): string {
        if (this.context && this.context.getLocaleString) {
            return this.context.getLocaleString(key);
```

```
        }
        return key;
    }
}
```

The methods in the above class definition include the following:

Method	Description
`setContext(context: ContextModel) { this.context = context; this.render(); }`	Developer Portal invokes this method to access the context data.
`protected getData(): any { if (this.context && this.context.getData) { return this.context.getData(); } }`	This methods returns the context data.
`protected navigate(path: string): void { if (this.context && this.context.navigate) { this.context.navigate(path); } }`	This method is used to navigate to the given page.
`protected getLocaleString(key: string): string { if (this.context && this.context.getLocaleString) { return this.context.getLocaleString(key); } return key; }`	This method is used to retrieve the i18N key for a given value.
`export class ContextModel { getData: () => any; navigate: (path: string) => void; getLocaleString: (key: string) => string; }`	This method registers custom-elements with component definitions.

Web components considerations

Remember the following points when creating web components:

- Use unique element names.
- The JavaScript file uploaded for a web component must be independent of other files. There should not be any dependency between the uploaded files.
- Element name should have an hyphen (-) based on custom element specification. For example, *api-gallery-item*.

How do I register a web component?

This use case starts when you want to register a web component and ends when you have completed the registration.

Before you begin

- Ensure that the JavaScript file to be registered is ready.

To register a web component

1. From the **Manage themes** page, click the customize icon  next to *Theme1*.
2. Select **Web components**.
3. Click **Create web components**.
4. Provide the **Name** and **Description**.
5. Click **Browse file** and select the web component in the JavaScript format.
6. Click **Save**.

   The new web component is added and can be included in the required block or page.

Sample Web component files

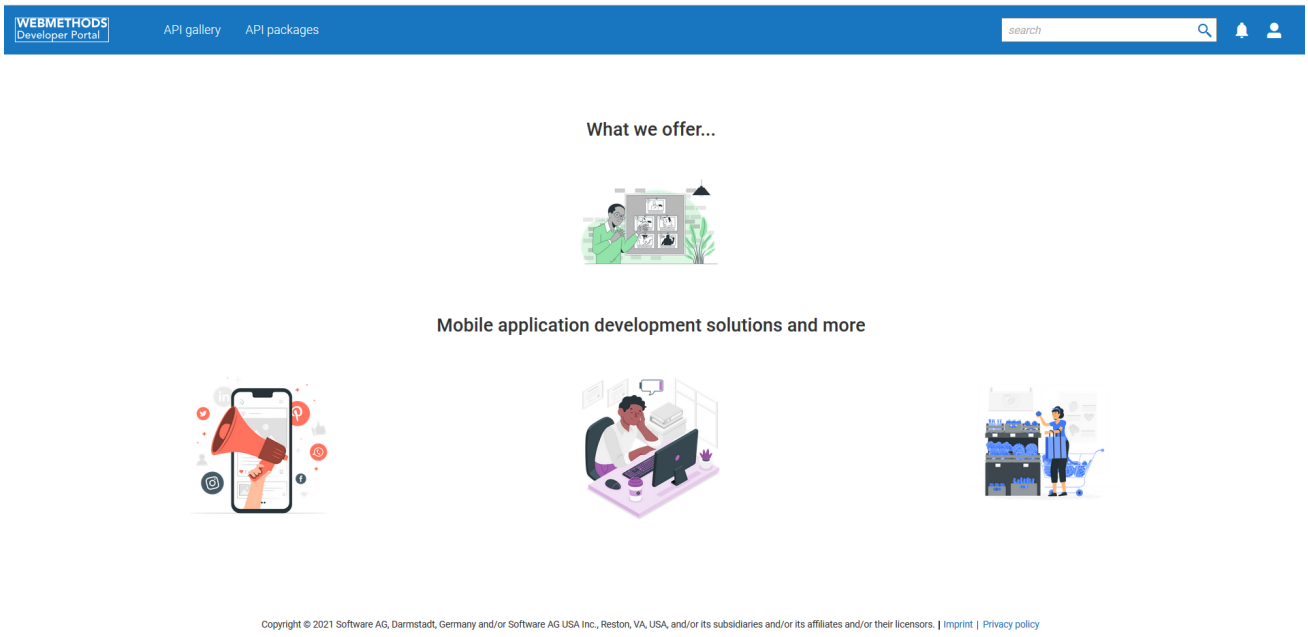
The following sample files are available *SAGInstallDir*`\developers\web-components\src\components`:

File name	Description
api-gallery	Retrieves the list of APIs and displays in the API gallery screen.
api-gallery-item	Allows to customize the API grid displayed in the API gallery screen.

Customization example

This use case is about an example scenario where a page is customized for a portal that exposes APIs for mobile applications.

For example, to customize the **Welcome** page like seen below.

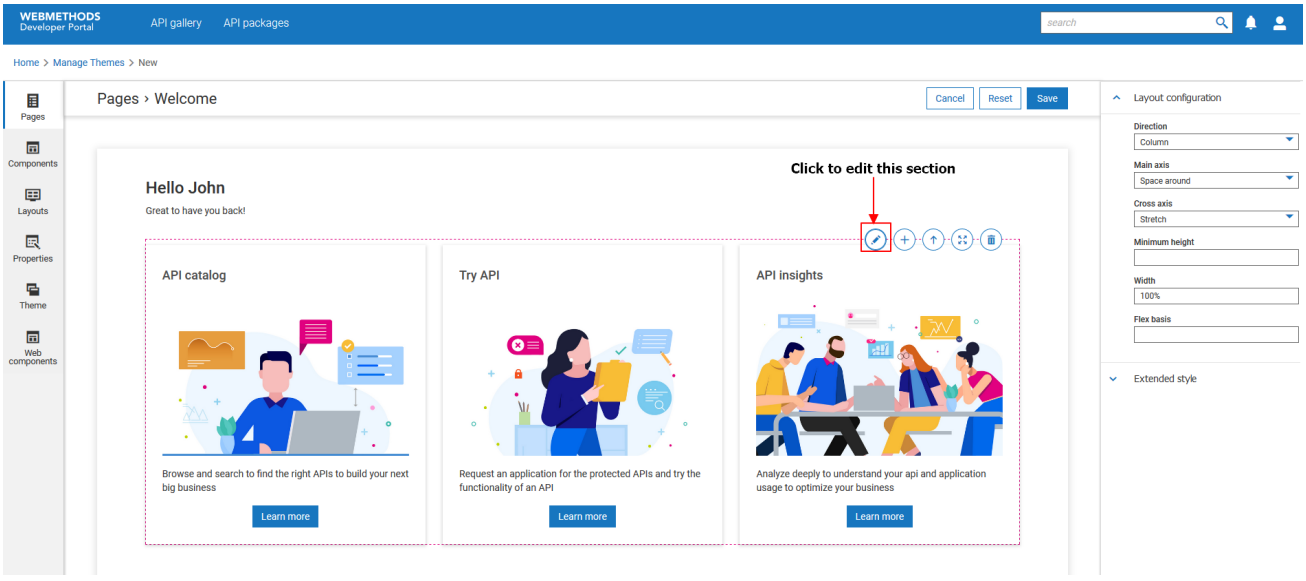


1. Create a theme. For information on creating themes, see "How do I create a theme for customizing the Developer Portal UI?" on page 59.

2. From the **Manage themes** page, click the edit icon next to the theme you created.

3. Select **Pages** from the left pane and select the page you want to customize.

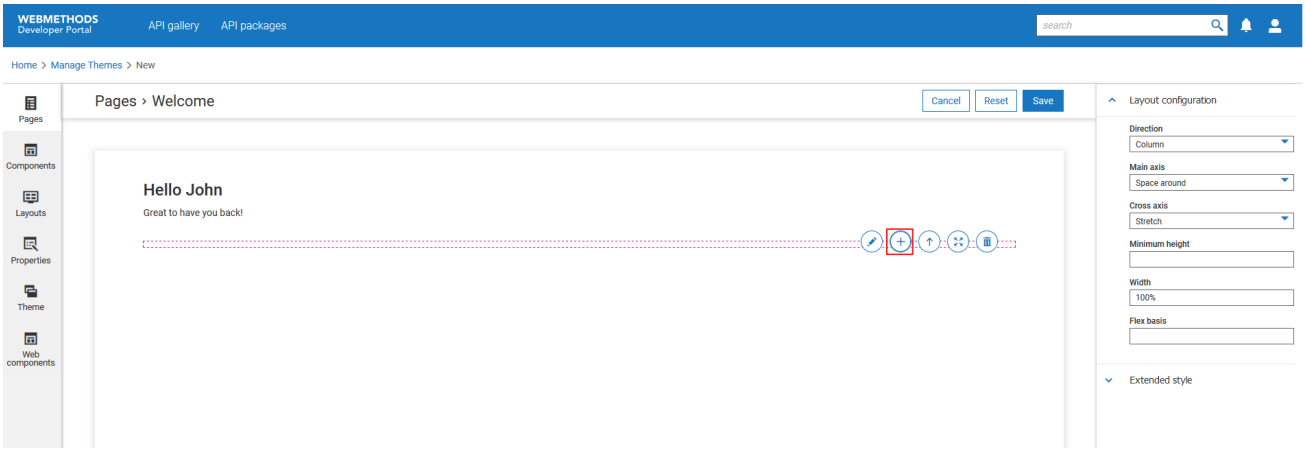
   The selected page appears with the corresponding editing options for each of the blocks. For example, let us customize the **Welcome** page.



   For example, let us remove the three blocks in the page and add four panels with images and headings.

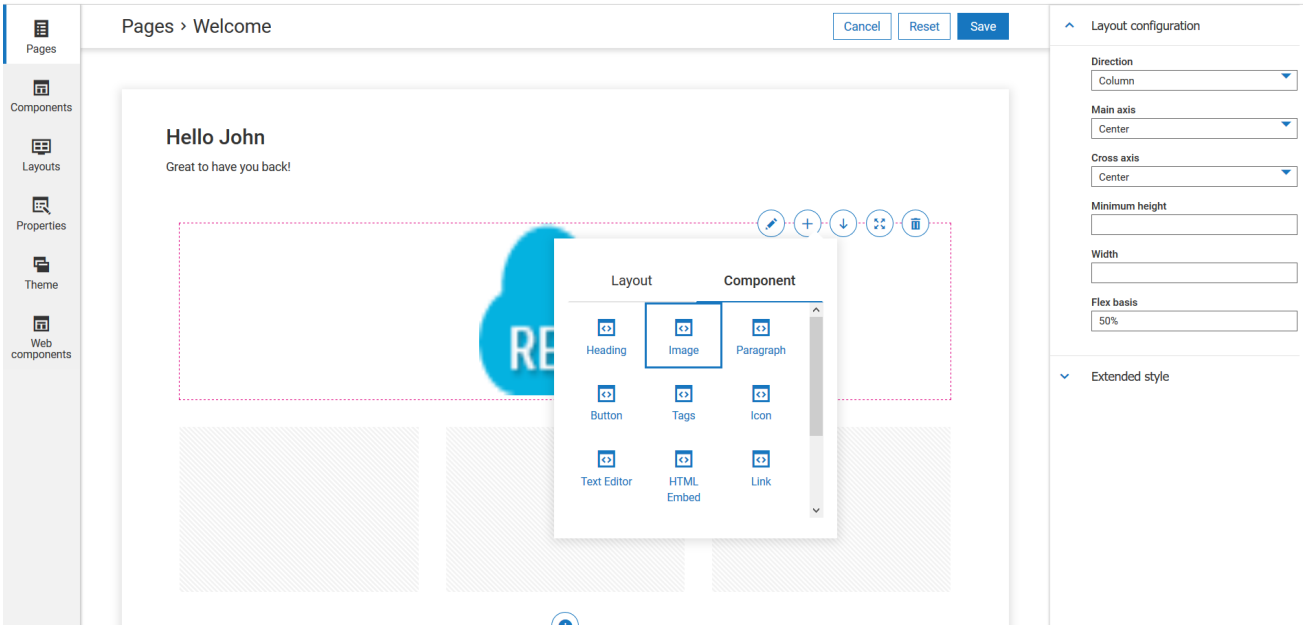
4. Click the delete icon next to the blocks to remove them.

5. Click the add icon to add new blocks.

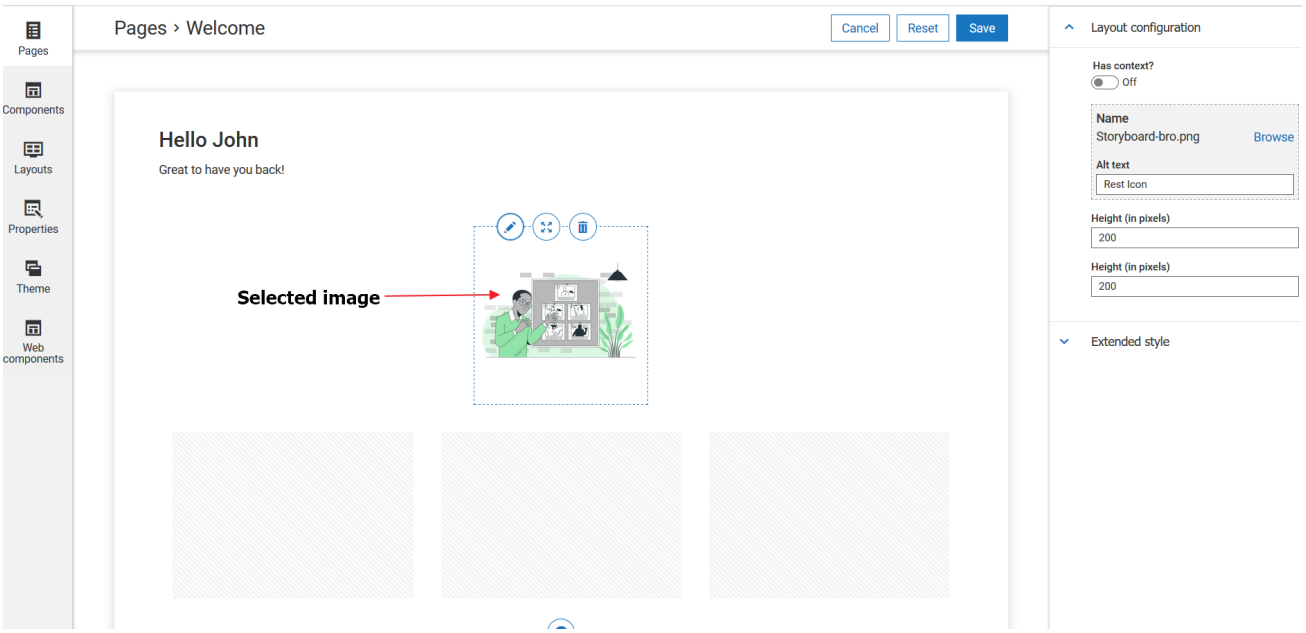


6. Select **Composite 3**. The new block with one horizontal and three vertical panels appear.

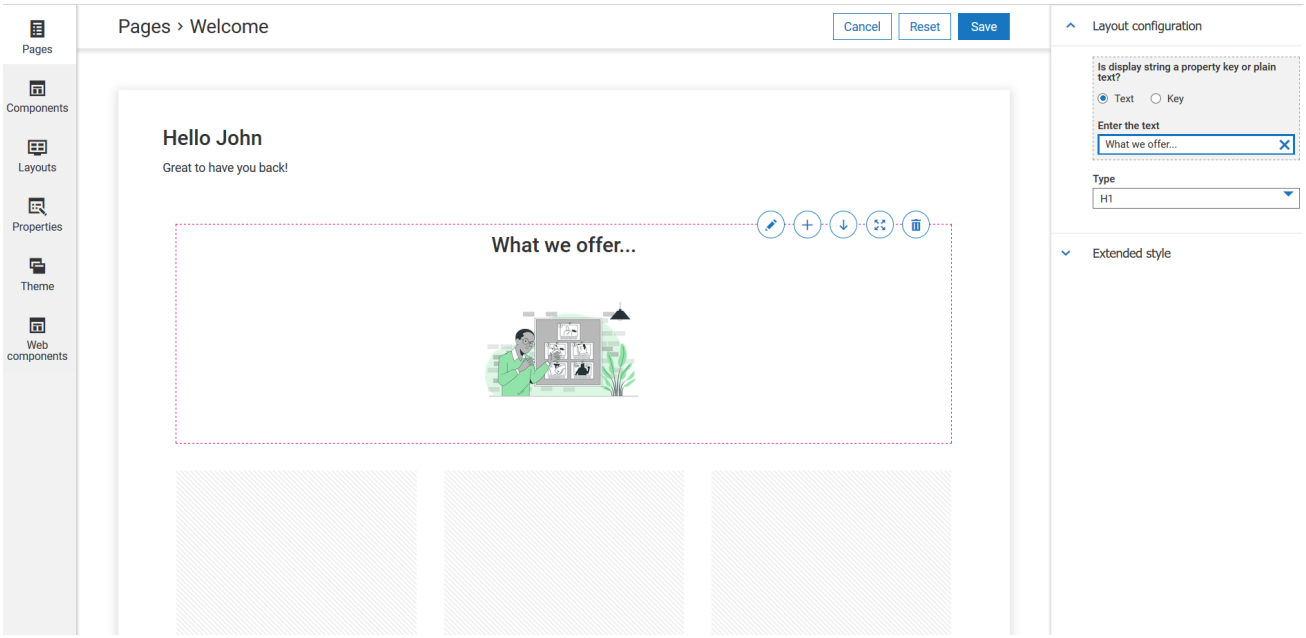
7. Click **Component** and select **Image**.



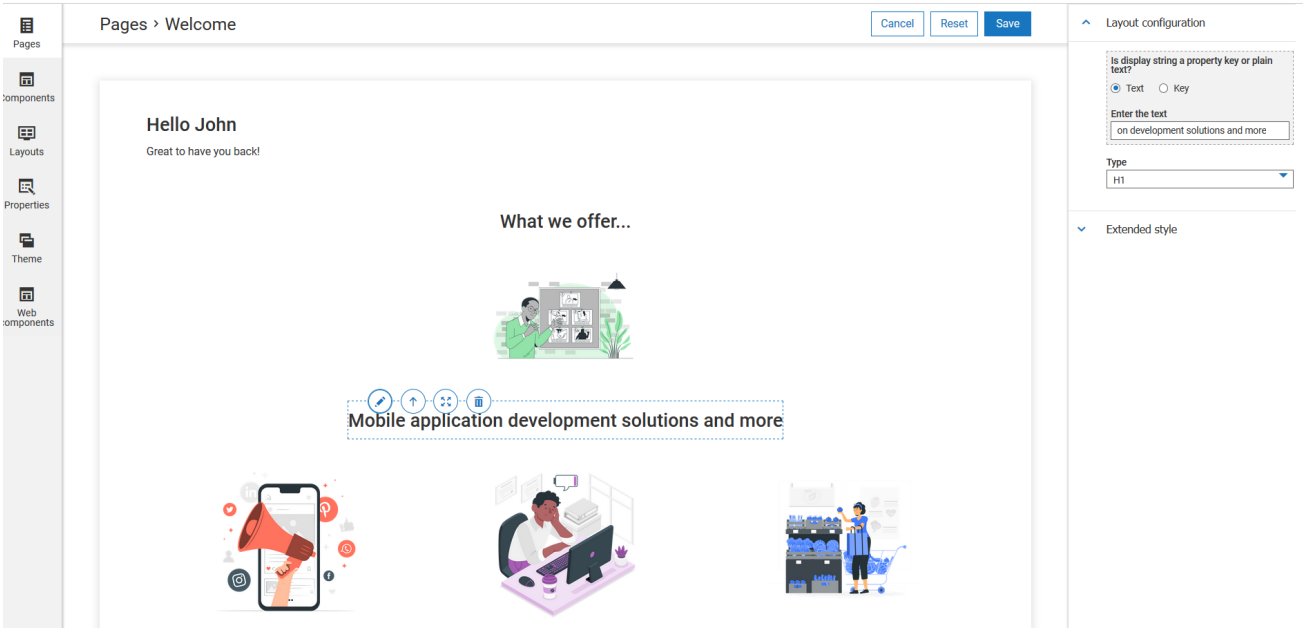
8. Click the edit icon next to the image, click **Browse** from the editing options, and select the required image.



9. Similarly, add a heading component and edit the component to give a required heading.



10. Repeat the steps to add images and headings.



11. Click **Save** to save your changes and activate the theme to see the customized **Welcome** page.

4 Users

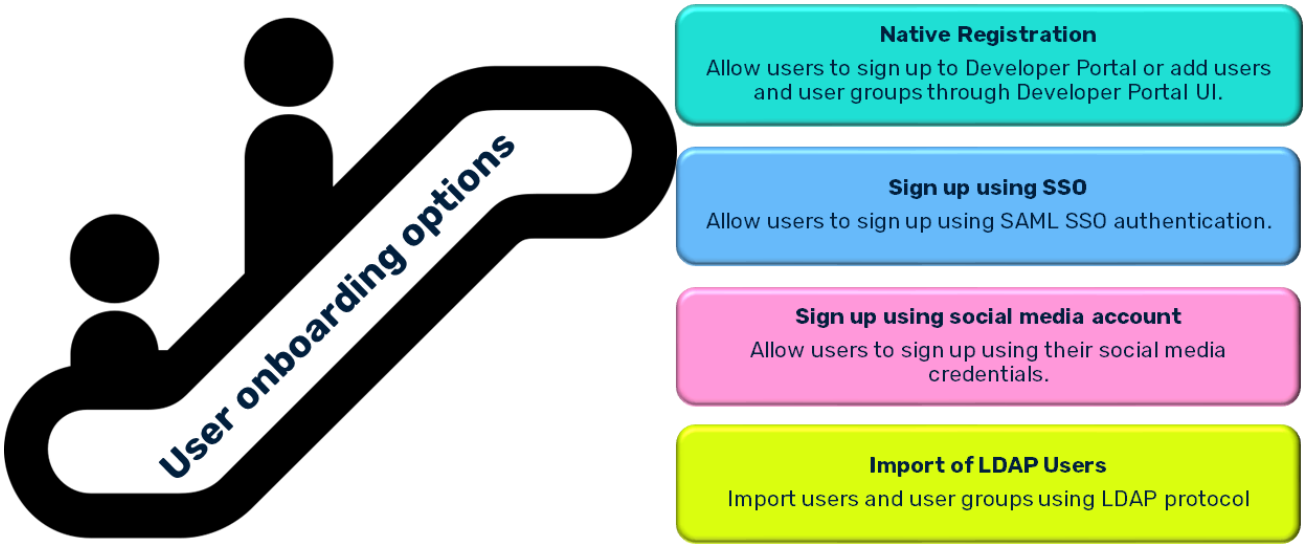
Overview

Developer Portal provides you with options to onboard users and manage their accounts. You can:

- Onboard users and user groups
- Configure an approval strategy to process new user sign-up requests
- Manage user privileges
- Configure advanced security settings to protect user accounts

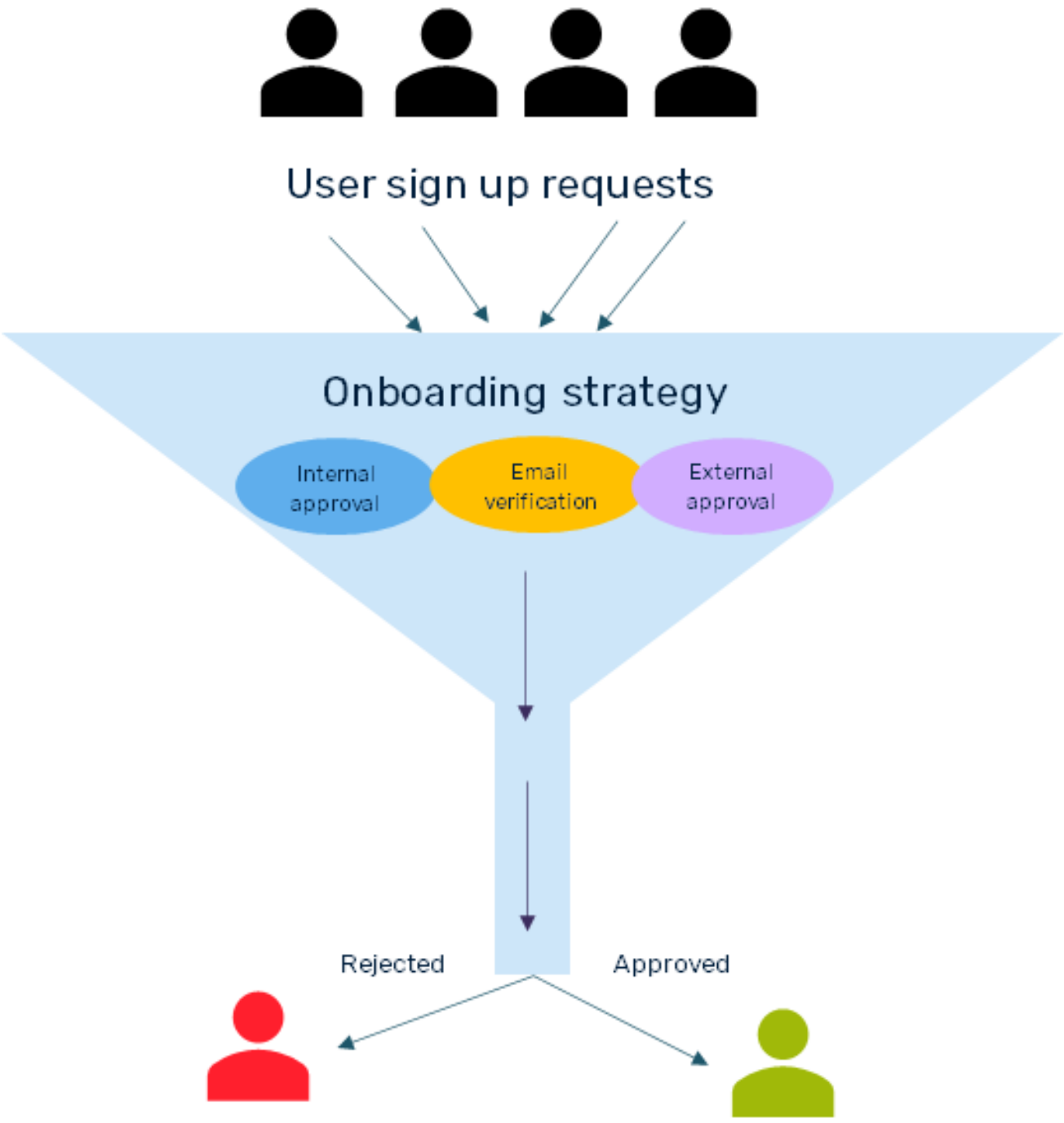
User onboarding

Developer Portal offers you the following options to onboard users into your portal:



Configure approval strategy

You can configure the strategy for approving user sign-up requests.



It is not mandatory to specify an onboarding strategy. If you do not configure an onboarding strategy, users who sign up are directly onboarded.

For information on configuring user onboarding strategies, see "Onboarding Strategy" on page 119.

Manage user privileges

You can assign one of the following privileges to users and user groups:

- **Administrator**. Has access to all modules and they can administer Developer Portal.
- **Provider**. Can manage assets such as APIs, packages, applications, and communities, and view API economy and usage dashboards.
- **Partner**. Can manage APIs and communities. Can publish APIs to private communities, and assign owners (users or teams) to APIs.
- **Consumer**. Can use API gallery, try APIs, and view API usage analytics.

When you assign a privilege to a user group, it will be applicable to all users in the group. For information on assigning or modifying user privileges, see "How do I assign privileges to a user?" on page 118.

Advanced user account security

You can make the user accounts more secure by enabling:

- **Multi factor authentication**. For information on multi-factor authentication, see "How do I configure multi-factor authentication settings?" on page 17
- **Account lockout settings**. For information on account lockout configuration, see "How do I configure user account lockout settings?" on page 16.
- **Password policy**. For information on password policy configuration, see "How do I configure password policy?" on page 14.

Native Registration

Native registration process allows:

- New users to sign up from the landing page of the application.
- Administrators to invite users to sign up or add new users.

Developer Portal provides the following options for native registration:

- **Sign up** page. New users can provide basic details such as their email address and password, and sign up for Developer Portal using the **Sign up** page accessed from the landing page. The sign-up request is forwarded for approval based on the onboarding strategy. You can configure an onboarding process for the incoming sign-up requests by specifying the required strategies from the **Onboarding** screen. For information about onboarding strategies, see "Onboarding Strategy" on page 119.

  You can disable this mode of registering new users if you want to onboard users using other modes like SSO and LDAP. For information about disabling this feature, see "Disabling user registration from the Sign up page" on page 119.

- **Manage users** section. You can invite users to sign up to the portal or add users and user groups from the **Manage users** page of the **Adminstration** section.
  - For information on inviting users, see "Inviting users to sign up" on page 117.

- For information on adding users, see "How do I add a user?" on page 113.
- For information on adding user groups, see "How do I add a user group?" on page 115.

- **Manage programs** section or the API program details page. You can invite participants to sign up for an API program (Hackathon or Beta program). This option is also available for API providers and partners who own or organize API programs. For information about inviting users for an API programs, see "Inviting participants for an API program" on page 283.

How do I add a user?

You can onboard users to Developer Portal by adding their details from the **Manage users** page.

In this example, you add a user, *user1*, include the user to the **API consumer** group and assign the **Consumer** privilege.

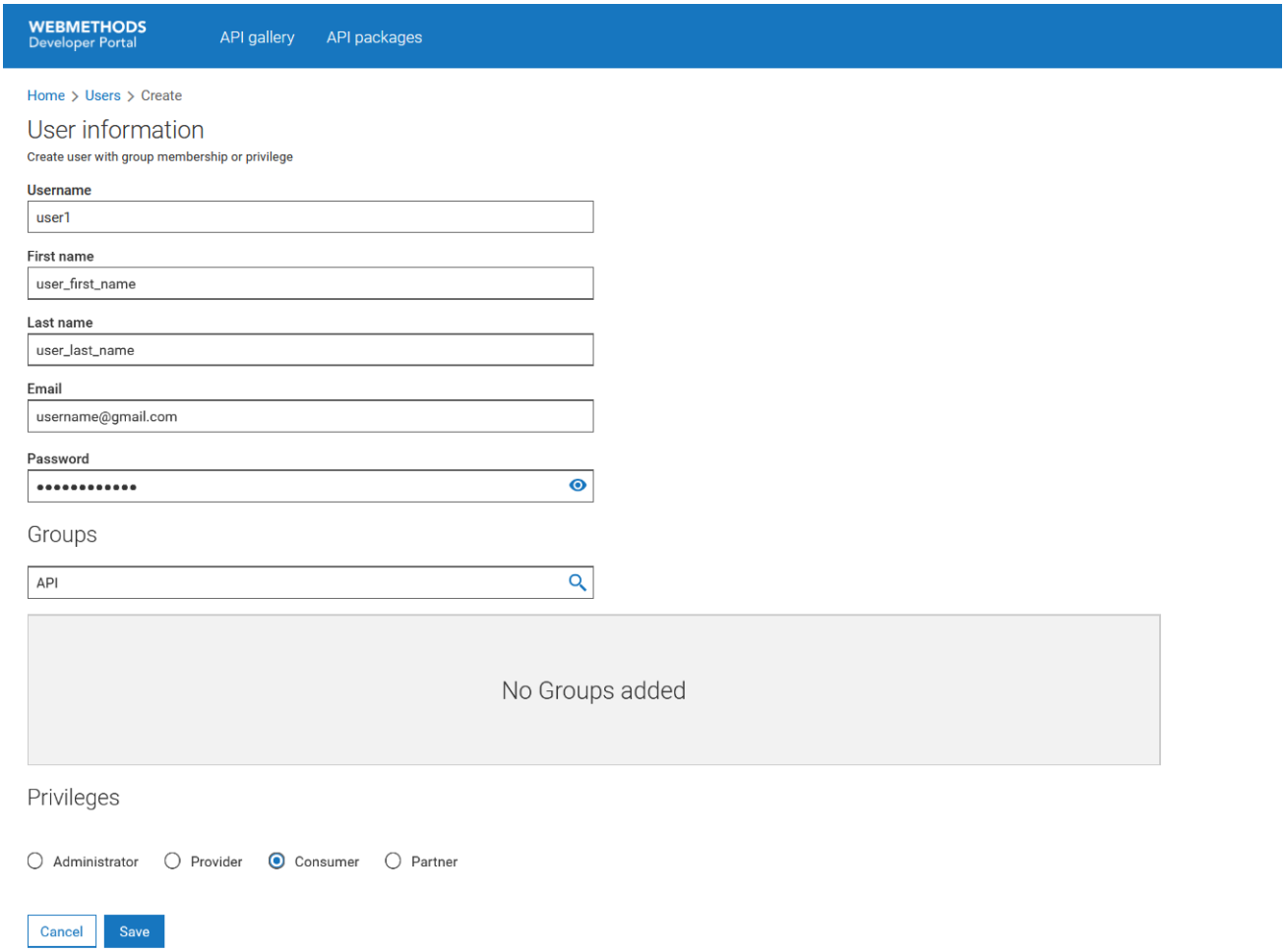
Before you begin

Ensure you have the **API Administrator** privilege.

To add a user

1. Click the menu options icon  from the title bar and click **Manage users**.
2. Click **Create user**.
3. Provide *user1* in the **Username** field.

   This is the user name that the user must provide during sign in.
4. Provide *user_first_name* in the **First name** field.
5. Provide *user_last_name* in the **Last name** field.
6. Provide *user@email.com* in the **Email** field.
7. Provide the **Password** that must be used to sign in.
8. Select the **API Consumer** group.
9. Select the **API Consumer** privilege.



10. Click **Save**.

    The new user appears in the **Manage users** screen.

Alternative steps:

1. In *Step 8*, you can add more than one group.

   You can also modify the list of groups later.

2. In *Step 9*, you can select one of the following privileges based on the role of the user in your organization:

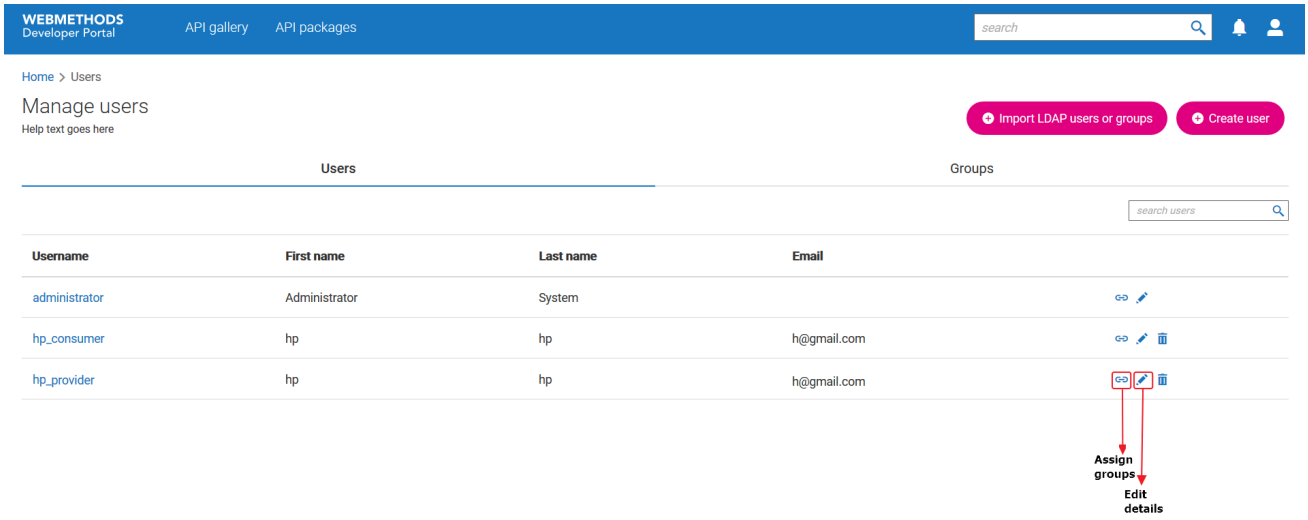
   - Administrator
   - Provider
   - Consumer
   - Partner

   Note:

In addition to the privilege that you assign to users, the users will have the privileges of the selected groups assigned to them. If you select more than one group, then the highest privilege among the groups added is applied to the user. For example, if you select API provider and API consumer groups for a user, then the user will have the API provider privilege.

Next steps:

- If you are a user:
  - You can sign in by providing their user name and password.
  - You must change your password when you sign in for the first time. The password you provide by the users must abide by the password policy. For information about configuring the password policy, see "How do I configure password policy?" on page 14.
- If you are an administrator:
  - You can click the edit icon next to a user to edit the user details.
  - You can click the assign icon next to a user to assign the user to the required groups.



How do I add a user group?

This use case starts when you want to add a user group and end when you have added one.

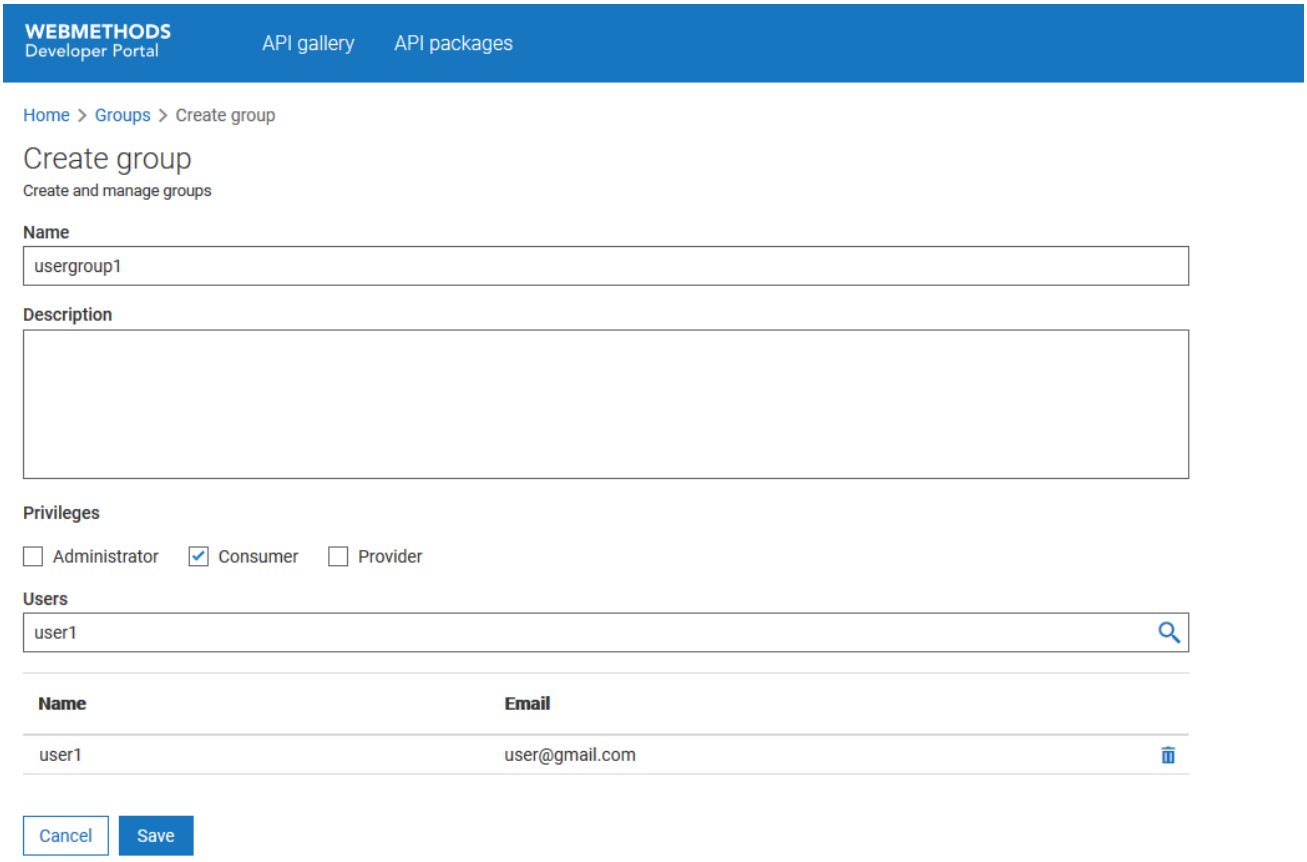
In this example, you add a user group, *usergroup1*, assign the **Consumer** privilege, and include the user **user1** to the group.

Before you begin

Ensure you have the **API Administrator** privilege.

To add a user group

1. Click the menu options icon from the title bar and click **Manage users**.
2. Click **Groups**.
3. Click **Create group**.
4. Provide *usergroup1* in the **Name** field.
5. Select the **Consumer** privilege.
6. Select *user1* from the **Users** list.



7. Click **Save**.

   The group is added.

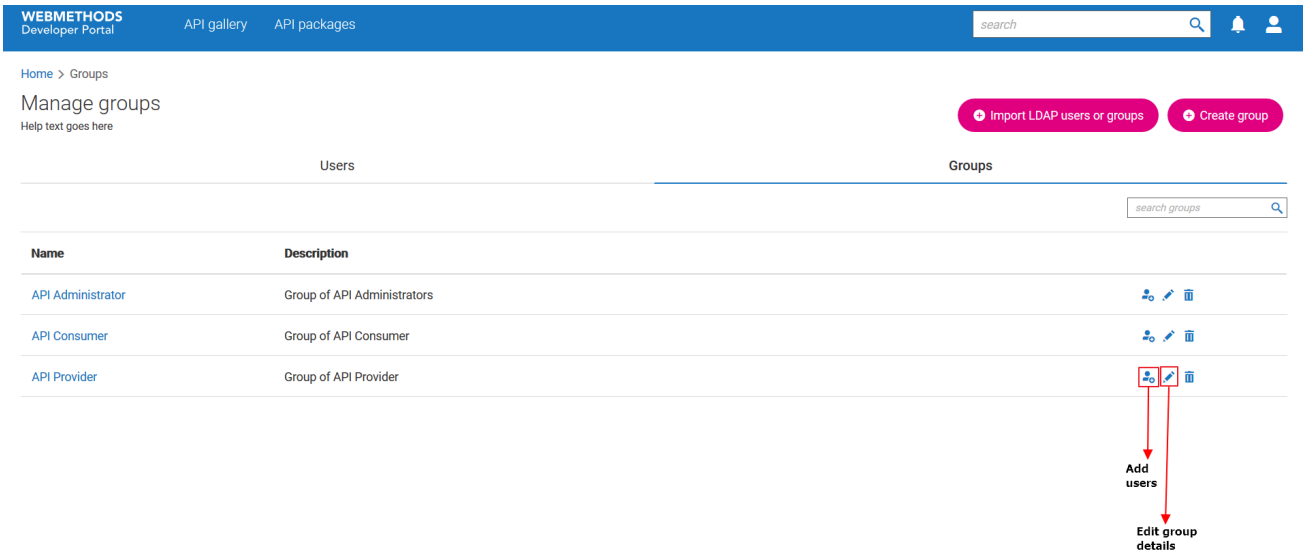
Alternative steps:

- You can add more than one user and you can also modify the list of users later.

- You can select more than one privilege for the group. If you select more than one privilege, then the highest privilege will be applied to the group. For example, if you select API provider and API consumer privileges for a group, then the group will have the API provider privilege.

Next steps:

- The new group appears in the **Groups** tab of the **Manage users** screen.
- You can assign groups as approvers for approving user or application.
- You can assign groups to the communities to allow users of the group to access the community's assets.
- Click the edit icon next to a group to edit the user details.
- Click the assign icon next to a group to assign the required users.



Inviting users to sign up

As an administrator, you can onboard users by inviting them by e-mails. Users who receive the invitation can click the link in the mail to sign up to Developer Portal.

To invite users

1. Click the menu options icon from the title bar and click **Manage users**.
2. Click **Invite user**.
3. Provide the e-mail addresses of the required users.

When you invite multiple users using one invite, provide an e-mail address, and provide a comma or press **Enter**. Repeat this step till you provide all e-mail addresses.

4. Select the communities, and privileges that must be applied.

   The selected communities and privileges are applied to the newly invited users when they are onboarded.

5. Click **Invite**.

   An invite mail is sent to the e-mail addresses you provided.

Next steps:

- Users who received the invite mail can click the link provided in the mail and follow the given instructions to sign up to the portal. During sign up, users can provide the password using which they can sign in to the portal.
- Users are included to the group that you have selected in the **Default group name** field in the **Administration** > **Users** page. If you have selected any group other than `API Consumer`, then the invited participant will also be added as an **API Consumer** group in addition to the selected group.
- Users invited by administrators using this feature do not undergo the onboarding approval process, if any. In addition, if you have enabled **Email verification** as a part of your onboarding process, it is ignored for users who sign up using the mail invite. For information about onboarding strategy and email verification, see "Onboarding Strategy" on page 119.

How do I assign privileges to a user?

Users can perform tasks based on their privileges. You, as an administrator, can assign privileges to users when you create them. For users who are onboarded using any other method, you can edit users or user groups and assign required privileges.

When you create users from the **Add user** page, you can assign the required privileges. However, you must edit the details of users who sign up through native registration or SAML SSO to assign required privileges to them.

This use case starts when you assign or modify user privileges and ends when you have successfully made the changes.

Before you begin:

Ensure you have the **API Administrator** privilege.

To assign privileges

1. Click the menu options icon  from the title bar and click **Administration**.

   The list of users appears.

2. Click the edit icon next to the required user.
3. Assign or modify the privileges to the user.

   You cannot modify the user privileges assigned through the groups.
4. Click **Save**.

   Your changes are saved.

Next steps:

Users can perform any transactions that require the assigned privilege.

Disabling user registration from the Sign up page

You can disable signing up of new users from the **Sign up** page, if you want the user registration only through other modes such as SSO and LDAP registration.

To disable user registration from the Sign up page

1. Click the menu options icon from the title bar and click **Administration**.
2. Click **General**.
3. Turn the **Enable User Registration** slider off.

   This is turned on by default.
4. Click **Save**.

   User signing up from the **Sign up** page is disabled.

Onboarding Strategy

The onboarding strategy is used to specify the process to approve or reject:

- User sign-up requests
- Application or subscriptions requests

You can specify any one or all of the following steps as a part of onboarding strategy:

- **Internal approval**. Approvers receive a notification when there is a request for a user, application, or subscription registration. They can view the pending approval requests, review them, and approve or reject them. You can configure the required registration approval workflow. For information on configuring user registration approval workflow, see "How do I configure an approval workflow to process an internal approval onboarding strategy?" on page 121.

- **External approval**. You can configure an external system to verify and approve or reject the requests. You can notify the required external approving system by creating a webhook. For information on configuring user sign-up notifications to your external approving system, see "How do I configure webhooks to notify events to an external system?" on page 23.
- **Email verification**. This is applicable only for user registration. An email is sent to the email address provided during sign-up. Users can click the link sent over the email to get verified. Usually, this step is combined with one of the above two.

How do I configure onboarding strategy to process user sign-up requests?

Onboarding strategy determines the process that user sign-up requests must undergo and it is optional. If you do not configure an onboarding strategy, then users' sign-up requests are automatically approved.

This use case starts when you want to configure onboarding process for user registration requests and ends when you have completed the configuration.

Before you begin:

Ensure that you:

- Configure an approval workflow. For information on configuring user registration approval workflow, see "How do I configure an approval workflow to process an internal approval onboarding strategy?" on page 121.
- **API Administrator** privilege.

To configure user onboarding strategy

1. Click the menu options icon  from the title bar and click **Administration**.
2. Select **Onboarding**.
3. From the **User onboarding** section, enable any or all of the required strategies:
   - **Internal approval**. Turn on and select the required approval workflow **Select a flow**. For information on configuring user registration approval workflow, see "How do I configure an approval workflow to process an internal approval onboarding strategy?" on page 121.
   - **External approval**. Turn on to enable external approval. You can notify the required external approving system by creating a webhook. For information on configuring user sign-up notifications to your external approving system, see "How do I configure webhooks to notify user sign-up and application requests to an external approval system?" on page 27.
   - **Email verification**. An email is sent to the email address provided during sign-up. Users must follow the steps given in the mail to get onboarded.
4. Use the arrow keys next to these strategies to change their order.

   The strategies are followed by the order they appear.

5. Click **Save**.

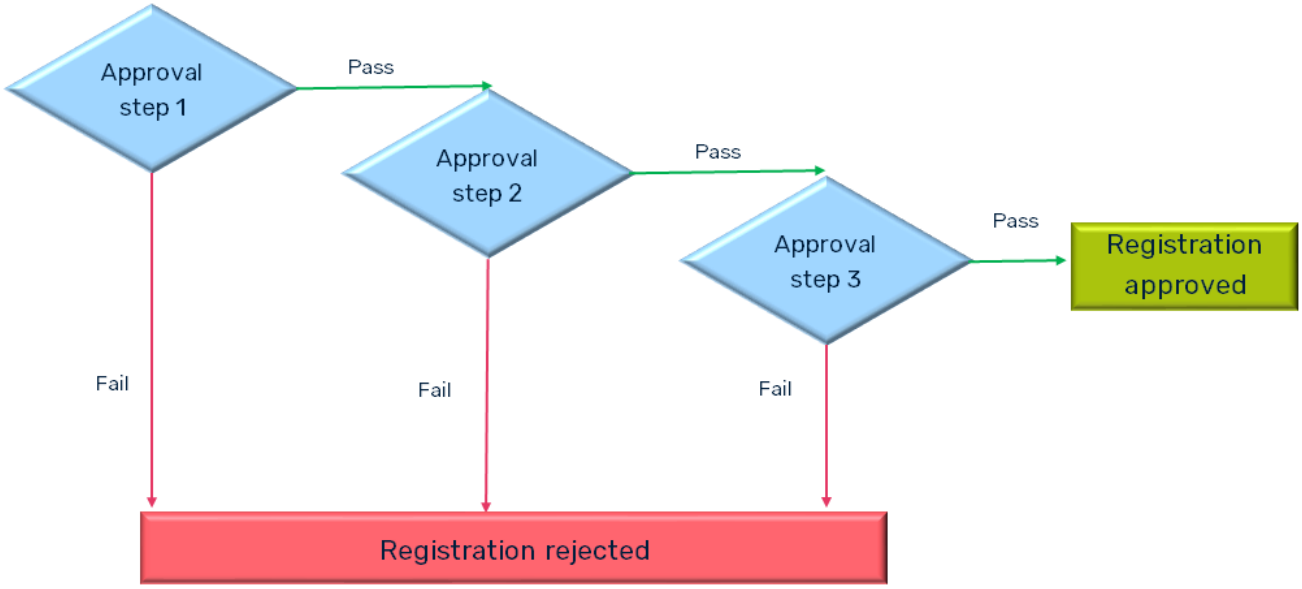
   The onboarding strategy is saved.

Next steps:

User sign-up requests are processed based on the onboarding strategy.

How do I configure an approval workflow to process an internal approval onboarding strategy?

Within a workflow, you can specify multiple approval steps. An application is successfully registered when the request passes through the steps configured in the approval workflow. You can also modify the sequence of approval steps based on your requirement.



This use case starts when you want to configure workflow with one or more approval steps with the required approvers to approve a user or application registration request.

In this example, you create a workflow, *workflow1* with *user1* as first level approver, and anyone from *ApproverGroup1* as second level approvers.

Before you begin:

Ensure that you have:

- List of users and user groups that you want to specify as approvers.
- **API Administrator** privilege.

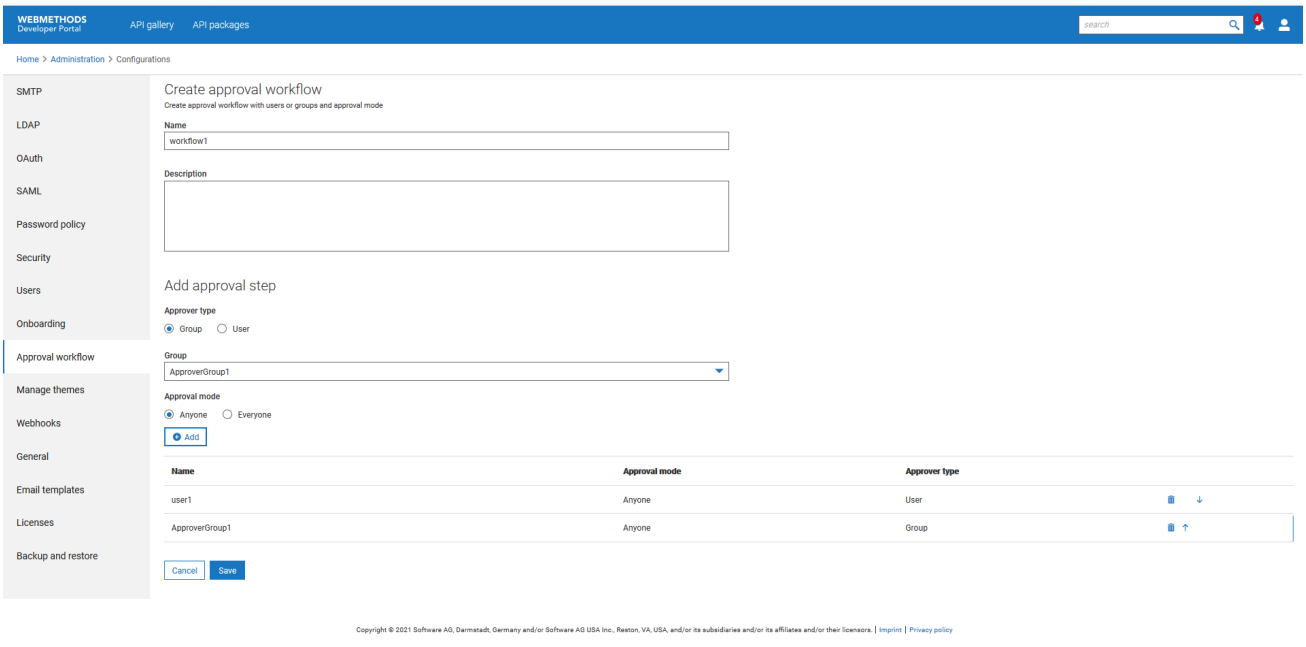
> **To configure an approval workflow to process an internal approval onboarding strategy**

1. Click the menu options icon from the title bar and click **Administration**.

2. Select **Approval workflow**.
3. Click **Create approval workflow**.
4. Provide *workflow1* in the **Name** field.
5. Select *User* from the **Approver type** field.
6. Select *user1* from the **User** list.
7. Click **Add**.
8. Select *Group* from the **Approver type** field.
9. Select *usergroup1* from the **Group** list.

> **Note:**
> This fields lists only the user groups that have the **API Administrator** or **API Provider** privilege.

10. Select *Anyone* from the **Approval mode** field.
11. Click **Add**.



12. Click **Save**.

    The approval workflow is created.

Alternative steps:

1. Use the move up  and move down  icons next to approval steps to change their sequence.

2. To specify that everyone from a group must approve the registration, select the required group and select *Everyone* from the **Approval mode** field. If you select *Everyone*, and if anyone in the user group rejects a request, then the request is rejected.

   When you reject a request, a notification e-mail is sent to the requestor as per the **Approval result** e-mail notification template. If you have specified more than one approver, and if one of the approver rejects a request then the notification e-mail is sent to the requestor and the other approvers. For more information on the e-mail notification template, see "List of email notification templates available in Developer Portal" on page 21.

Next steps:

- Assign the workflow to internal approval onboarding strategy.

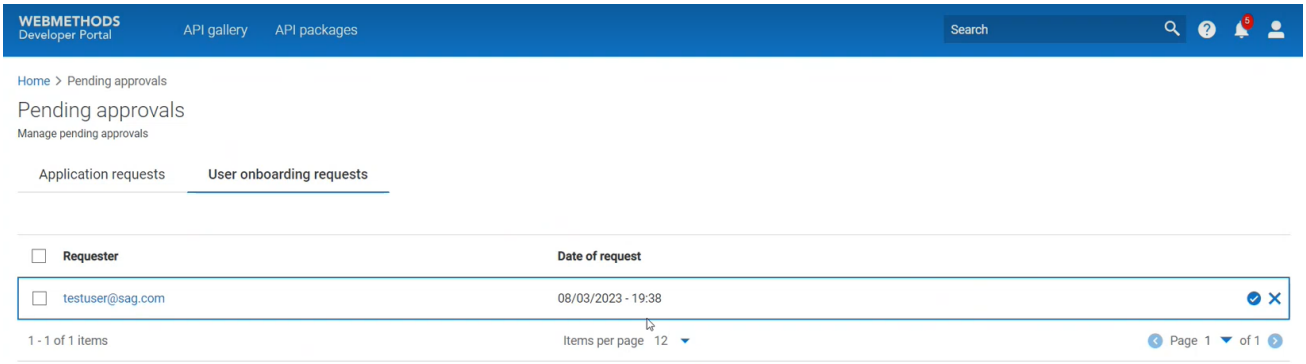
Approving User Onboarding Requests

If you have configured an approval workflow for onboarding new users, then approvers can view the pending requests from the **Pending approval** screen, and approve or reject them. For information about configuring an approval workflow, see "How do I configure an approval workflow to process an internal approval onboarding strategy?" on page 121.

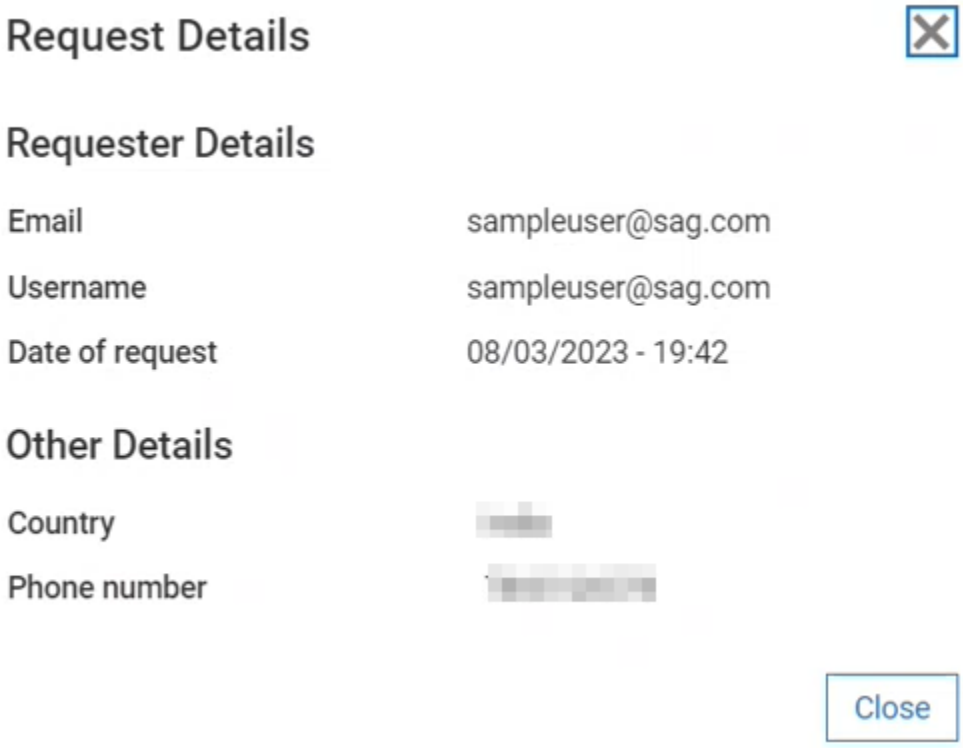
To approve a pending onboarding request

1. Click the menu options icon  from the title bar and click **Pending approvals**.
2. Select **User onboarding requests**.

   The list of onboarding requests appears.



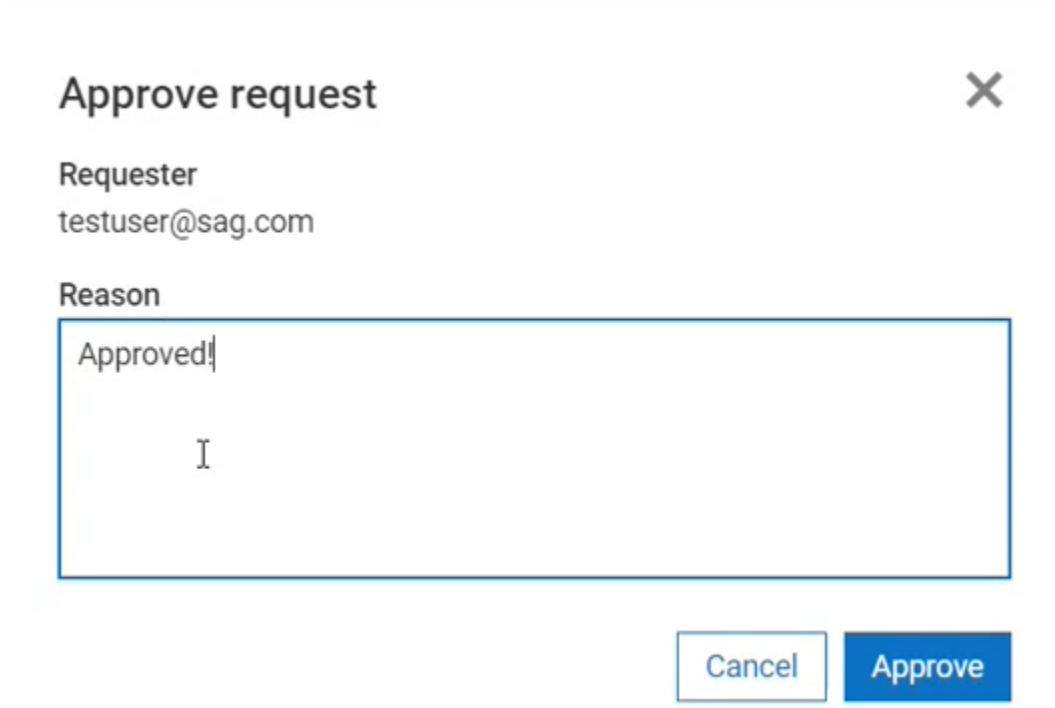
3. *Optional*. Click a request to view more details about the request.



This screen provides information that are gathered from the users during their sign-up.

4. Click next to the required onboarding request to approve the request.

   A confirmation screen appears.



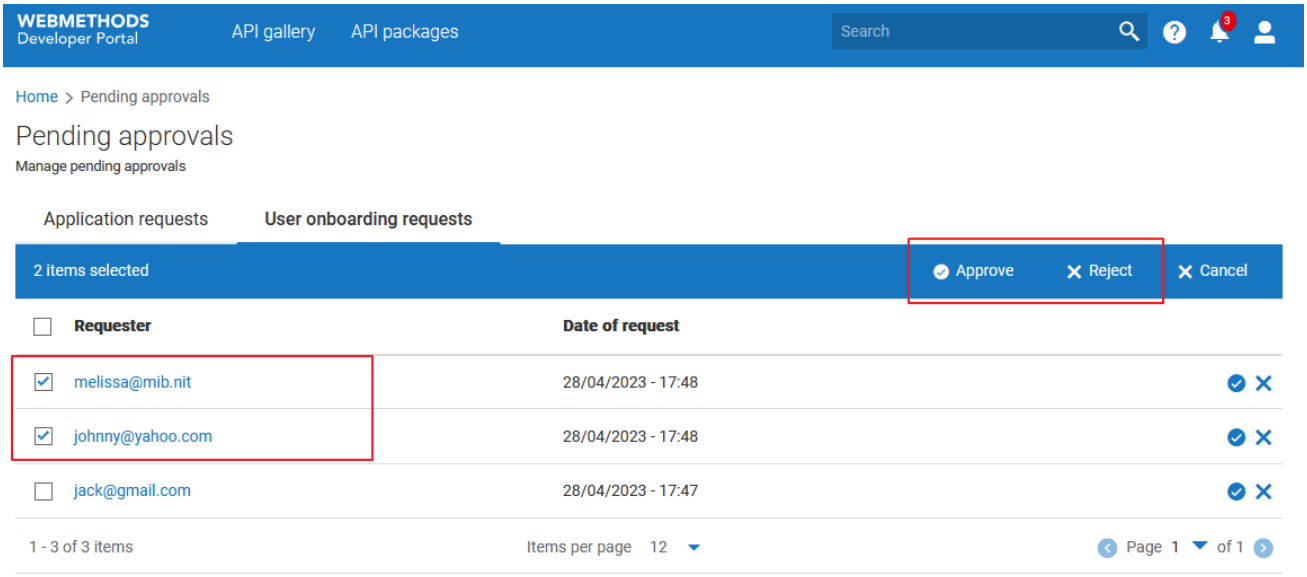
5. Provide your comments, if any in the **Reason** field, and click **Approve**.

Alternative steps:

- In *Step 3*, you can click ✕ to reject the request. Provide your comments, if any and click **Reject**.

  When you reject a user onboading request, a notification e-mail is sent to the requestor as per the **Approval result** e-mail notification template. If you have specified more than one approver, and if one of the approver rejects a request then the notification e-mail is sent to the requestor and the other approvers. For more information on the e-mail notification template, see "List of email notification templates available in Developer Portal" on page 21.

- To approve or reject multiple requests listed on the **Pending approvals** screen, select the checkboxes next to the required requests and click **Approve** or **Reject**. The selected pending requests are approved or rejected at once.



- To approve or reject all requests, select the checkbox in the grid header and click **Approve** or **Reject**. All pending requests are approved or rejected at once.

LDAP Users and Groups Onboarding

You can add LDAP users and their associated groups as Developer Portal users. You can provide LDAP server details by creating an LDAP connection and import users and user groups from the server. You can specify multiple LDAP servers.

The high level of LDAP configuration workflow is as follows:

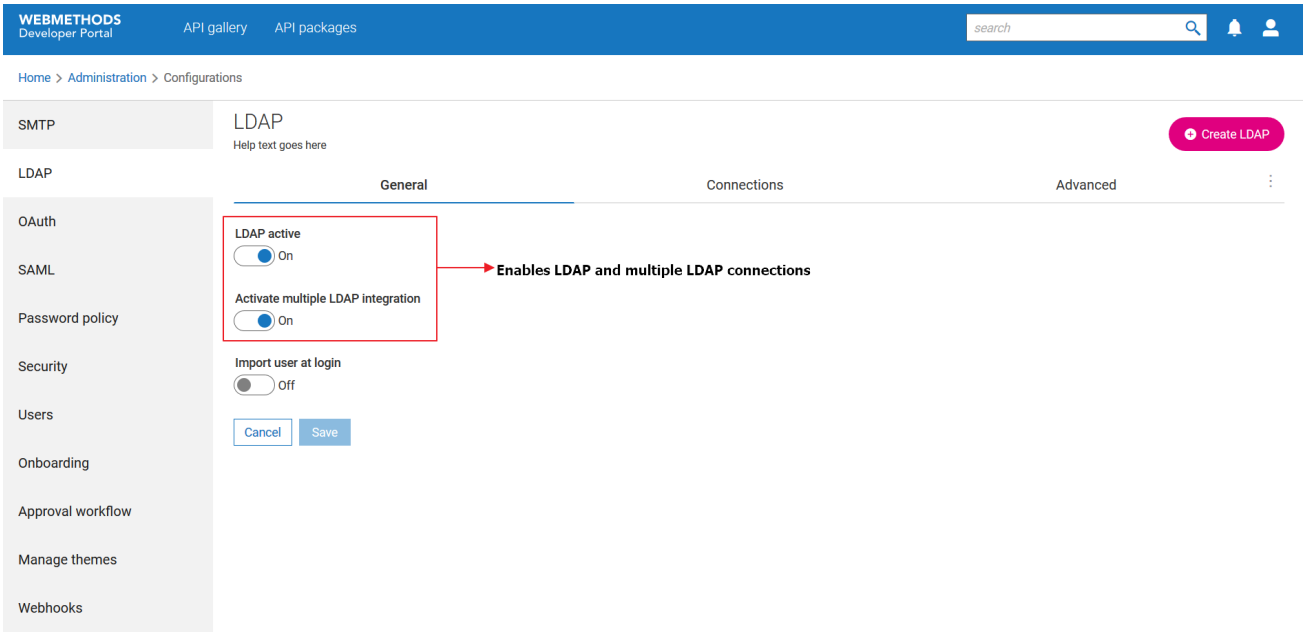
How do I create an LDAP connection to import users from a LDAP server?

This use case starts when you want to provide the LDAP server details and ends when you have successfully created a connection.

Before you begin

Ensure that:

- LDAP is enabled. You can enable LDAP by turning the **LDAP active** slider on.
- Multiple LDAP integration is enabled, if you want to specify more than one LDAP server.



- LDAP server details.

  Note:
  When you configure and enable the LDAP server, the Developer Portal will not allow users to sign up with a user name if that user name exists in the LDAP server. If there is already a local user with the same user name, the local user will be converted to LDAP user.

- **API Administrator** privilege.

To create an LDAP connection

1. Click the menu options icon from the title bar and click **Administration**.
2. Select **LDAP**.
3. Click **Create LDAP**.
4. In the **ID** field, provide a unique ID for the LDAP connection.
5. Provide the Server **Name**, **URL**, **Username**, and **Password** of the LDAP server.

   Note:
   If the server you provided here has LDAPS, then you must import the truststore even if do not enable any other security check.

6. In the **Simultaneous connections** field, provide the maximum number of simultaneous connections to the same LDAP server.
7. Provide the **Connection timeout** and **Read timeout** values in milliseconds.
8. Click **Save**.

The LDAP connection appears in the **Connections** tab.

9. Click  of the LDAP connection to verify if Developer Portal is able to connect successfully with the LDAP server.

   You can import users and user groups from the LDAP connection.

Alternative steps:

- Add a secured LDAP connection. For information on creating a secured LDAP connection, see "How do I create an LDAP connection to import users from a secured LDAP server?" on page 128.
- Configure the LDAP connection settings. For information on configuring the connection settings, see "How do I specify attributes for the LDAP connection established with an LDAP server?" on page 131.

Next steps:

- Import users or user groups from the LDAP server. For information on importing users, "How do I import users and user groups from an LDAP server?" on page 133.

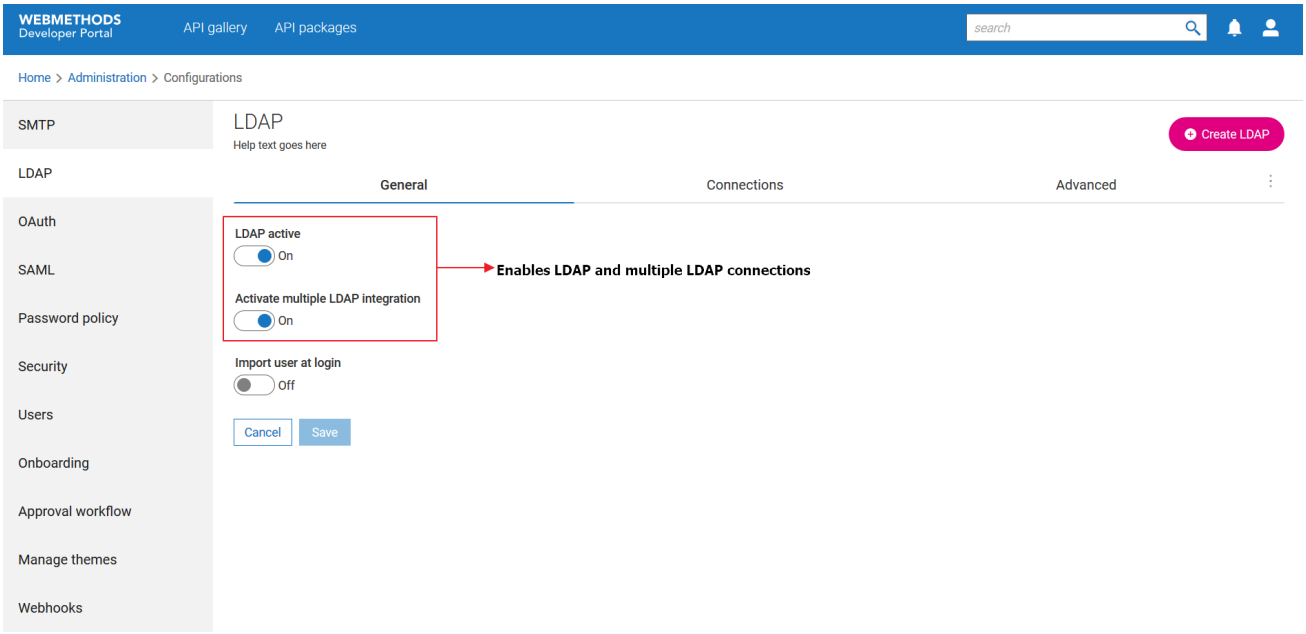
How do I create an LDAP connection to import users from a secured LDAP server?

This use case starts when you want to provide the secured LDAP server details and ends when you have successfully created a connection.

Before you begin

Ensure the following:

- LDAP is enabled. You can enable LDAP by turning the **LDAP active** slider on.
- Multiple LDAP integration is enabled, if you want to specify more than one LDAP server.



- LDAP server details.
- **API Administrator** privilege.
- Keystore, Truststore, and certificate generated from the LDAPS server side using the following commands:
  - keytool -genkey -alias server-alias -keyalg RSA -keypass changeit -storepass changeit -keystore keystore.jks
  - keytool -export -alias server-alias -storepass changeit -file server.cer -keystore keystore.jks
  - keytool -import -v -trustcacerts -alias server-alias -file server.cer -keystore cacerts.jks -keypass changeit -storepass changeit

To create a secured LDAP connection

1. Click the menu options icon  from the title bar and click **Administration**.
2. Select **LDAP**.
3. Click **Create LDAP**.
4. In the **ID** field, provide a unique ID for the LDAP connection.
5. Provide the **Server name**, **URL**, **Username**, **Password** of the LDAP server.

   Note:
   If the server you provided here has LDAPS, then you must import the truststore even if do not enable any other security check.
6. Based on your security requirements for the LDAP connection, enable the following checks:

- **Verify certificates**. Turn on to verify the SSL certificates provided by LDAP server. The LDAP connection fails if invalid certificates are provided.

  If you enable certificate verification, you can also turn on the hostname verification to ensure that the specified LDAP server host name matches the name in the SSL certificate received from the LDAP server when establishing the connection. The LDAP connection fails if the names do not match.

  To enable the hostname verification set the envirorment variable, **portal.server.config.disable-host-name-verifier**, to `false`. The default value of this setting is `true`.

- **Use SSL**. Turn on to specify that the connection to the LDAP server is secure. Enable this option or use an LDAPS URL for a secure connection. When you turn this on, the SSL mode list appears.

Note:
To configure a secured LDAP connection, you can either provide an LDAPS URL in the **URL** field (*Step 5*) or provide a LDAP URL and select **Use SSL**. You can enable the certificate verification and the hostname verification configurations for both these methods.

7. Select the required **SSL mode** from the list.

   This is applicable only when you have enabled the **Use SSL** field.

8. In the **Simultaneous connections** field, provide the maximum number of simultaneous connections to the same LDAP server.

9. Provide the **Connection timeout** and **Read timeout** values in milliseconds.

10. Click the **Truststore** tab.

11. Provide the required **jks** file and its corresponding **Password**.

12. Click **Save**.

    The LDAP connection appears in the **Connections** tab.

13. Click  of the LDAP connection to verify if Developer Portal is able to connect successfully with the LDAP server.

    You can import users and user groups from the LDAP connection.

Next steps:

- Import users or user groups from the LDAP server. For information on importing users, "How do I import users and user groups from an LDAP server?" on page 133.
- Configure the LDAP connection settings. For information on configuring the connection settings, see "How do I specify attributes for the LDAP connection established with an LDAP server?" on page 131.

How do I specify attributes for the LDAP connection established with an LDAP server?

This use case starts when you have created an LDAP connection and when you want to modify or specify the attribute mappings, user attribute mappings, group attribute mappings, and behavior of the LDAP connection.

Before you begin:

Ensure that you have:

- An LDAP connection.
- **API Administrator** privilege.

To specify attributes for the LDAP connection established with an LDAP server

1. From the **Connections** tab, click the edit icon  next to the connection.
2. Click the **Attribute mappings** tab.
3. Provide the following details:

Field	Description
objectClass	Attribute that contains the object class.
DN	Fully qualified name (distinguished name).
GUID	Globally unique Identifier of the LDAP server.

4. Click the **User attribute mappings** tab.
5. Provide LDAP user attributes:

Field	Description
Name, **First name**, and **Last name**	LDAP user name, first name, and last name.
E-mail address and **Telephone number**	Email address and telephone number of the LDAP user.
Picture	Location of the user's thumbnail picture.
memberOf	Attribute that references the groups of a user.
User-defined	List of LDAP attributes, separated by commas, that are to be imported as user-defined attributes of LDAP user.

6. Click the **Group attributes mappings** tab.
7. Provide the following LDAP group attributes:

Field	Description
Name	Group name.
hasMember	Attribute that references the members of a group.
User-defined	List of LDAP attributes, that you want to import as user-defined attributes of a group.

8. Click the **Behavior** tab.
9. Provide the following details:

Field	Description
Group object class.	Object class of the LDAP group.
User object class	Object class of the LDAP user.
Search paths	List of all LDAP search paths separated with semi-colons.
Group search paths	List of all LDAP search paths for user groups separated by semi-colons. The list provided here overwrites the list of general search paths.
User search paths	List of LDAP search paths for users separated using semi-colons. The list provided here overwrites the list of general search paths.
Group search filter	Query filter for LDAP groups.
User search filter	Query filter for LDAP users.
Recursion depth	Recursion depth that is to be used for nested groups and users.
Page size	Maximum number of entries that are loaded in a single LDAP query.
Refferals	Defines how referrals to other LDAP systems are processed.

10. Click **Save**.

    You have now completed providing LDAP details.

Next steps:

- Import users or user groups from the LDAP server. For information on importing users, "How do I import users and user groups from an LDAP server?" on page 133.

How do I import users and user groups from an LDAP server?

After creating an LDAP connection, you can import the users and user groups present in the LDAP server.

This use case begins when you have created an LDAP connection and ends when you have imported users and user groups from the specified server.

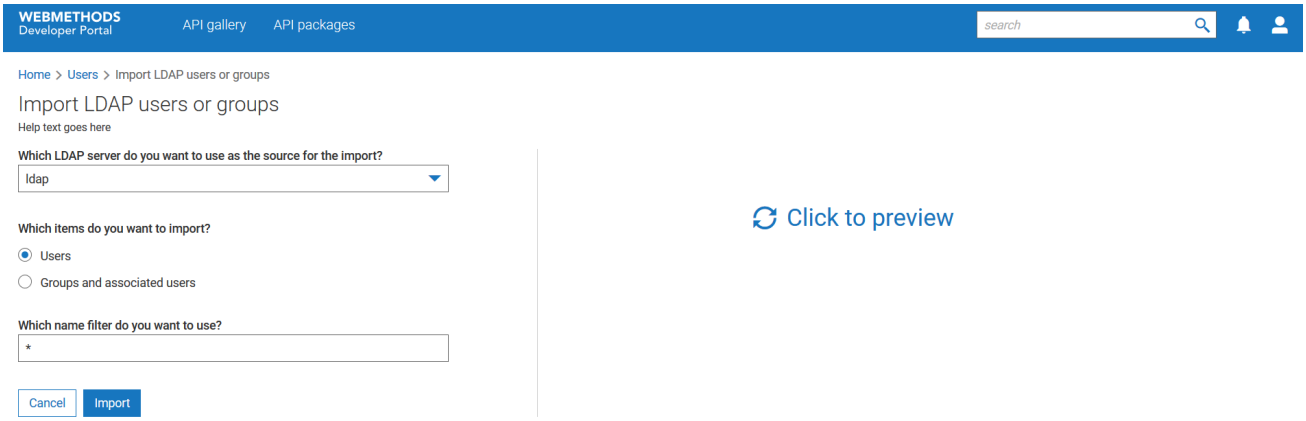
Before you begin:

Ensure that you have:

- An LDAP connection.
- **API Administrator** privilege.

To import users and user groups from an LDAP server

1. Click the menu options icon from the title bar and click **Manage users**.
2. Click **Import LDAP users or groups**.



3. From the list, select the LDAP connection from which you want to import.
4. Select one of the following:
   - **Users**. To import users from LDAP server.
   - **Groups and associated users**. To import user groups and their associated users.
5. In the text field, provide a value to filter users or groups, if required. Alternatively, type * to import all users or groups from the given LDAP server.
6. Click the right pane to preview users or groups.
7. Click **Import**.

   The list of users or groups are imported to Developer Portal.

Next steps:

- Imported users can sign in to Developer Portal using their LDAP credentials.

Single Sign-On Users Onboarding

Developer Portal uses SAML protocol to allow users to sign up with one of their following credentials:

- SAML Single Sign-On (SSO) identity provider accounts. The supported applications are:
  - Okta
  - PingIdentity
  - Azure Active Directory
- Social media accounts. The supported applications are:
  - Facebook
  - Google
  - GitHub

The onboarding strategy determines how the sign-up requests of users who sign up using their SSO credentials must be processed.

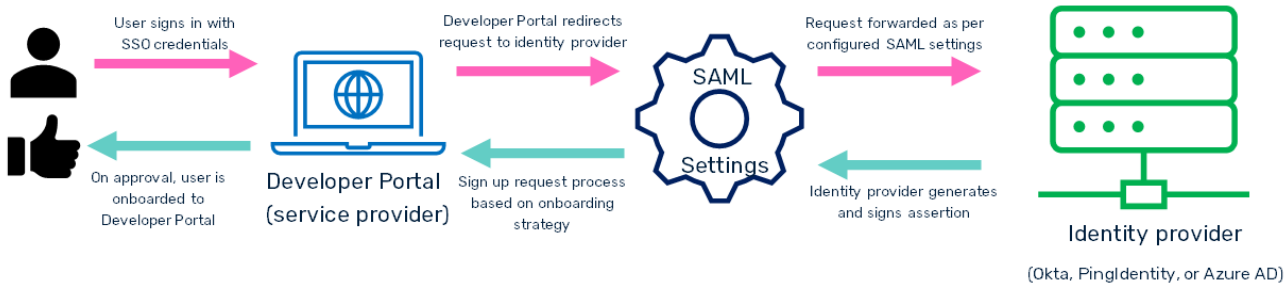
SAML SSO Onboarding

The SAML protocol is used to enable the SSO authentication. This authentication mechanism permits users to use one set of login credentials to access multiple applications. In addition to being a user-friendly option, implementing SSO makes user logins more secure as it uses SAML protocol for communication.

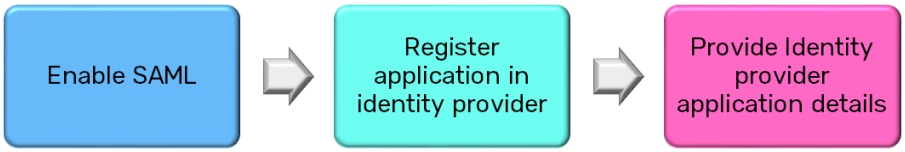
You can configure SAML settings and allow users to onboard using one of the following credentials:

- Azure Active Directory
- Okta
- PingIdentity

The SAML authentication workflow for onboarding users is as follows:



The high level of SAML configuration workflow is as follows:



How do I onboard users using their SAML service provider credentials?

You can enable SSO using one of the following applications:

- Okta
- PingIdentity
- Azure

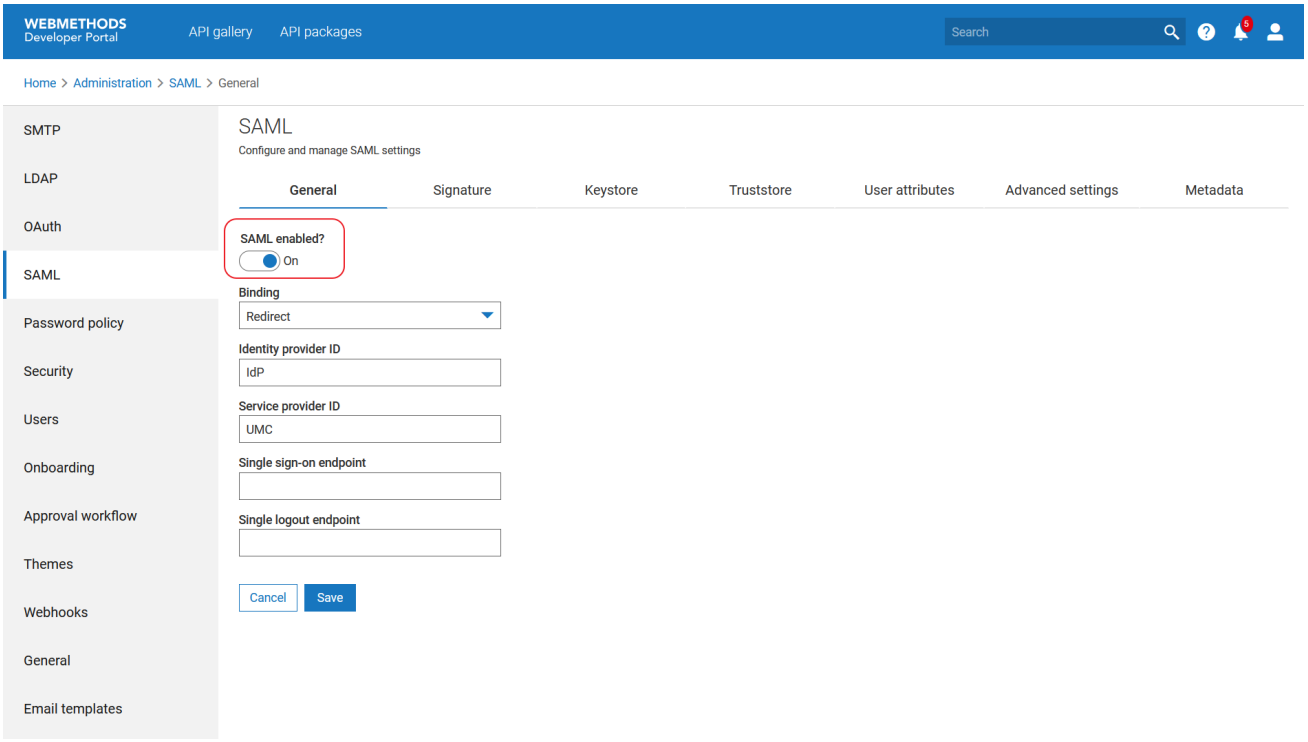
This use case begins when you want to allow users to onboard using their SSO credentials and ends when you have completed the configuration.

In this example, you enable SSO for user with their Okta credentials.

Before you begin:

Ensure that you:

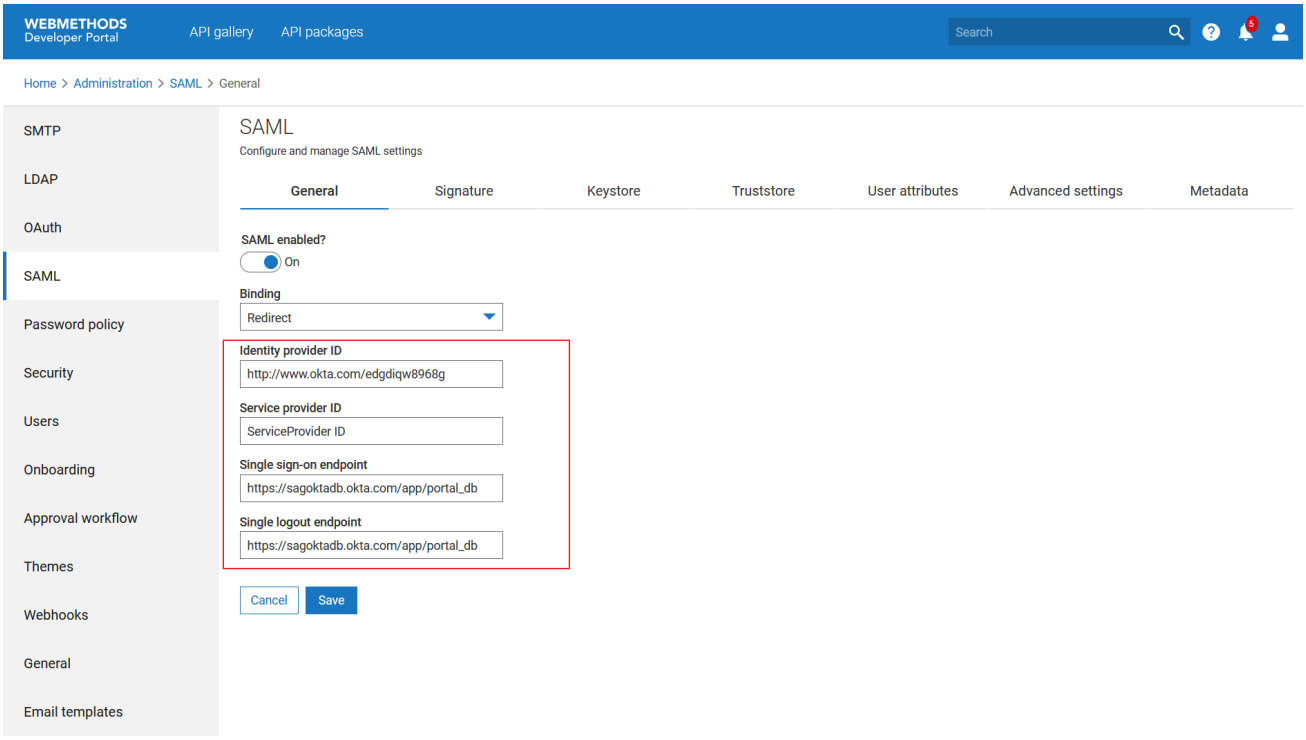
- Enable SAML. You can enable SAML by turning the **SAML active** slider on.



- Create an application in Okta to register the service provider application (Developer Portal) in the Okta, and keep the **Identity Provider Single Sign-on URL** and **Identity Provider Issuer** values ready. For information on creating an application in Okta, see https://developer.okta.com/docs/guides/add-an-external-idp/apple/register-app-in-okta/.

To enable SSO onboarding using Okta credentials:

1. Click the menu options icon from the title bar and click **Administration**.
2. Select **SAML**.
3. Select *Redirect* from the **Binding** list.
4. Provide the following values copied from Okta SSO application that you created for Developer Portal:
   - **Identity provider Id**. Id of the identity provider.
   - **Service provider Id**. Id of the service provider. This must be same as the value you specify in Okta.
   - **Single sign-on endpoint** and **Single logout endpoint**. Endpoints that the identity provider must use to send single sign-on and logout payloads.



5. Click **Save**.

   Your changes are saved.

Alternative steps:

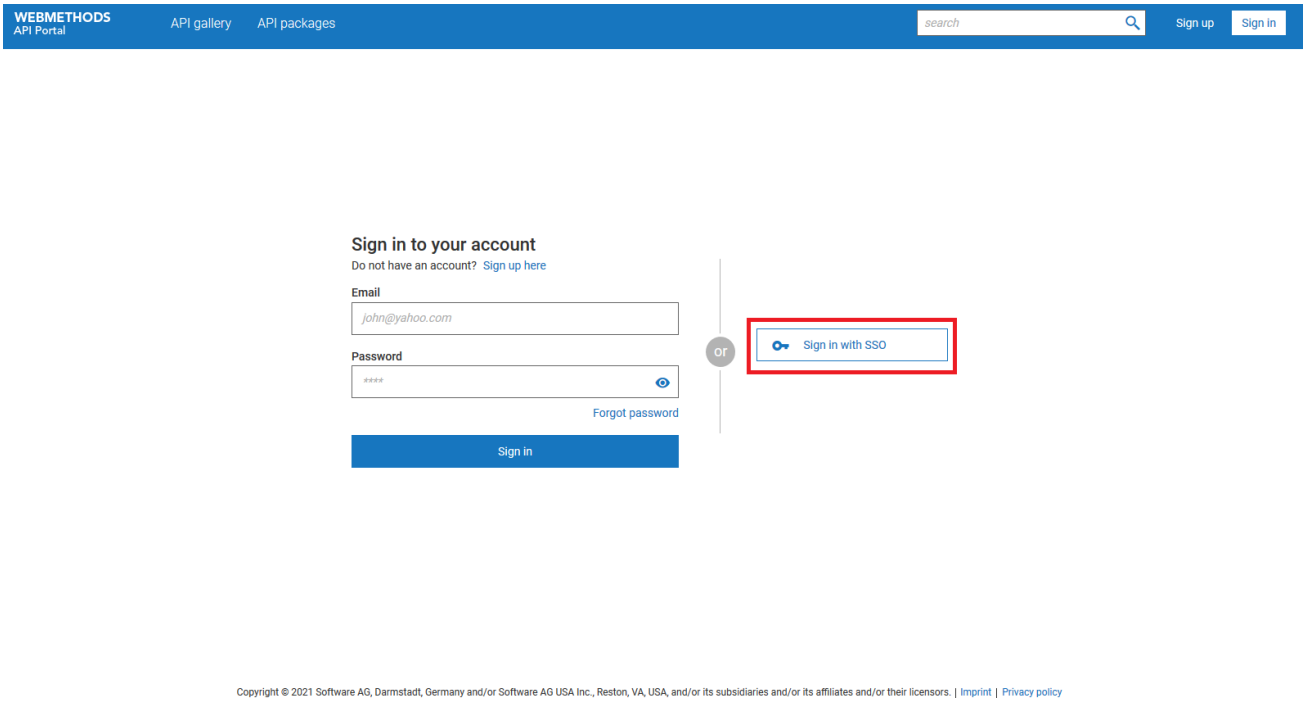
- Enable SSO to allow users to sign in to Developer Portal using their PingIdentity or Azure AD credentials.

  You must create an application in Okta to register the service provider application (Developer Portal) in the required service provider and provide the **Identity Provider Single Sign-on URL** and **Identity Provider Issuer** values in the SAML section:

  - For information on creating an application in PingIdentity, see https://docs.pingidentity.com/bundle/integrations/page/che1563995024790.html.
  - For information on creating an application in Azure Active Directory, see https://docs.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app.

Next steps:

- The **Sign in with SSO** button appears in the **Sign in** page.



User can click this button, provide their Okta credentials to sign up to Developer Portal. The sign-up request goes through the onboarding strategy.

- Configure advanced SAML settings. For information on configuring advanced SAML settings, "How do I configure SAML settings to specify user onboarding configurations?" on page 138.

How do I configure SAML settings to specify user onboarding configurations?

This use case starts when you want to configure SAML settings and ends when you have completed the configuration.

Before you begin:

Ensure you have

- Enabled the SAML feature.
- **API Administrator** privilege.

To configure SAML settings:

1. Click the menu options icon from the title bar and click **Administration**.
2. Select **SAML**.
3. Click the **Signature** tab.
4. Enable the following fields, if required:

- **Enforce signing of assertions**. Turn on to specify that the SAML assertions must be signed. If this is enabled, all assertions received by the application will be signed.
- **Enforce signing of requests**. Turn on to specify that the SAML authentication requests must be signed. If this field is enabled, all requests received by the application must be signed. Requests sent by the application are signed by the selected signature algorithm.
- **Enforce signing of responses**. Turn on to specify whether the SAML authentication response must be signed.
- **Enforce signing of metadata**. Turn on to specify whether the SAML metadata must be signed. If set, the service provider metadata file provided by the application is signed.

5. Select the required **Signature algorithm** from the drop-down list.
6. Click the **Keystore** tab.
7. Click **Browse** and select the SAML keystore file.
8. Provide the **Alias** name and **Password** required to access the keystore file in the corresponding fields.
9. Select the type of keystore file to be used from the **Type** drop-down list.
10. Click the **Truststore** tab.
11. Click **Browse** and select the SAML truststore file.
12. Provide the **Alias** name and **Password** required to access the truststore file in the corresponding fields.
13. Select the type of truststore file to be used from the **Type** drop-down list.
14. Click the **User attributes** tab.
15. Provide required values in the following fields:

Field	Description
First name	Attribute name to be used for reading the first name from a SAML assertion.
Last name	Attribute name to be used for reading the last name from a SAML assertion.
E-mail address	Attribute name to be used for reading the email addresses from a SAML assertion.
Telephone number	Attribute name to be used for reading the phone numbers from a SAML assertion.
memberOf	Attribute that references the groups of a user.

Field	Description
User-defined	List of attributes, separated by commas, to be imported as user-defined attributes of the user.

16. Click the **Advanced settings** tab.
17. Select **Create user automatically**.

    A user is created automatically using the details received from assertion.
18. Provide information in following fields:

Field	Description
Login using DN	Specifies whether sign in must be tried using the fully qualified name instead of the user name. The name in the assertion is assigned as the distinguished name of the user being created.
Decompose DN	Specifies whether the fully qualified name is to be decomposed. The name in the assertion is assigned as the distinguished name of the user being created only if the name is in an appropriate format.
Keyword	Specifies which part of the fully qualified name is to be used for login.
Authentication context comparison	Specifies the level of comparison that must be performed on the assertion context class against the authentication context. If this fails, the user is not authenticated.
Name ID format	Specifies the format in which the user ID must be saved.
Clock skew (in seconds)	Specifies the time offset between identity provider and service provider, in seconds. Assertions are accepted if they are received within the permitted time frame.
Assertion lifetime (in seconds)	Specifies the maximum lifetime of a SAML assertion, in seconds.
Assertion consumer service URL	Specifies the URL to which the identity provider must send the authentication response. The URL must be given in the format: `http(s)://hostname/portal/rest/v1/saml/initsso`
Default tenant	Specifies the default tenant that is to be used for the SAML-based login.

19. Click the **Extensions** tab.

    This tab includes options to allow you to configure settings to extract user information from the assertion sent by your SAML service provider to Developer Portal.

20. Turn the **Read multiple values from assertion** slider on to extract multiple values from the assertion.

21. Turn the **SAML assertion as a query string** slider on to <<info required>>.

22. Turn the **Include roles** slider on to <<info required>>.

23. Turn the **Enable assertion validation** slider on to validate the incoming assertion before extracting the required values.

24. Provide the names of attributes from which the following values to be extracted from the assertion :

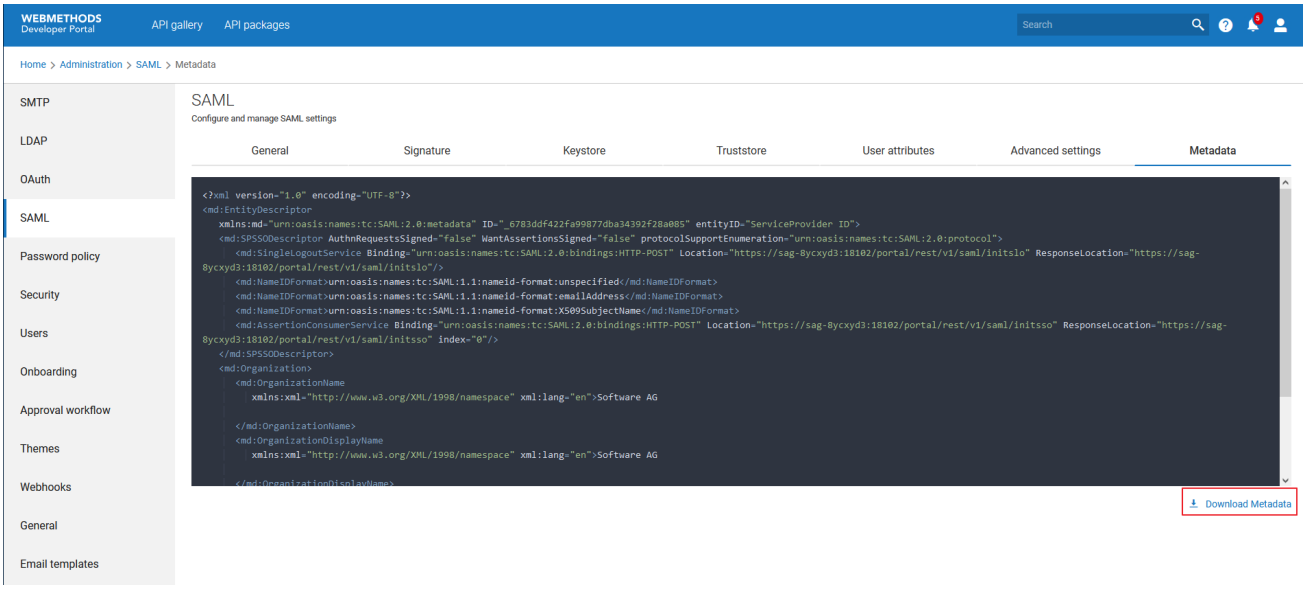
    - First name
    - Last name
    - E-mail address
    - Role
    - Sub-domains

25. Click **Save**.

    You have specified SAML configuration details. Users can sign up to Developer Portal using their SSO credentials.

Next steps:

- The service provider meta-data required for the registration is generated dynamically after SAML configuration. You can use the metadata and use it for configuring the identity provider. Download the metadata by clicking **Download Metadata** from the **Metadata** tab.



User Onboarding using Social Media Account

The OAuth section is used to configure onboarding using social media accounts. You can allow users to onboard using the following accounts:

- Google
- Facebook
- GitHub

To allow users to login through these accounts, you must register an OAuth application in their corresponding sites and provide the API Key and security token details in Developer Portal.

How do I onboard users using their Social media credentials?

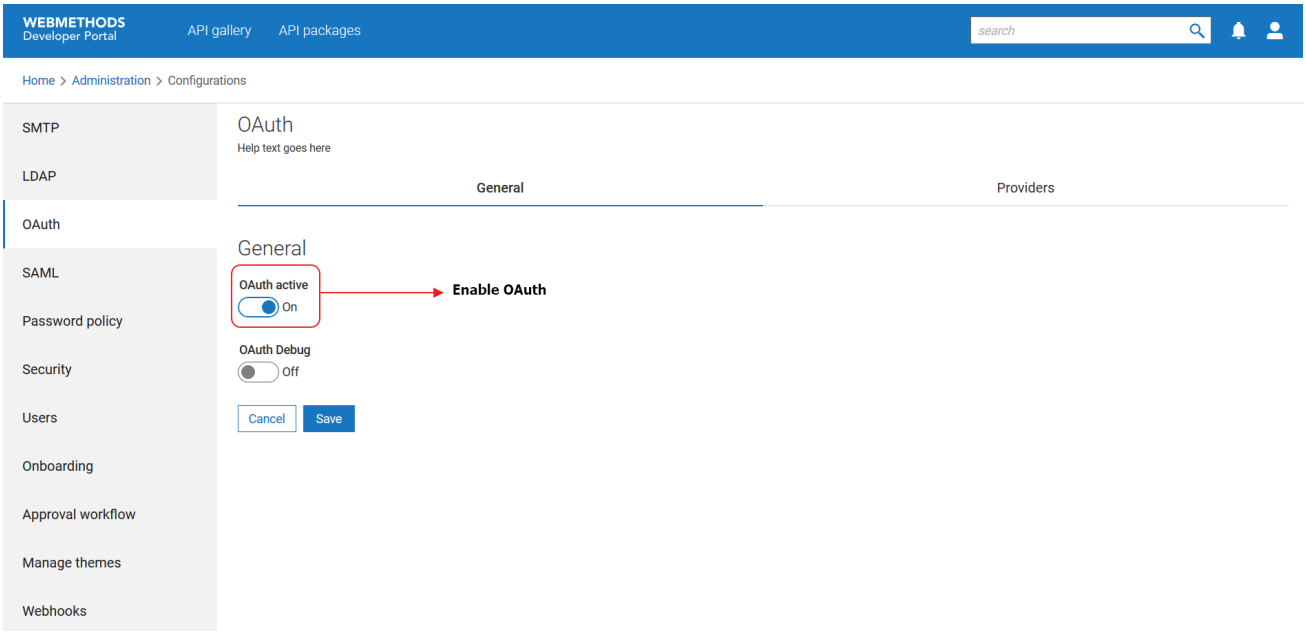
You can enable users to sign up using their Facebook, Google, or GitHub credentials.

This use case starts when want to allow user onboarding using their Social media account and ends when you have completed the configuration.

In this example, you enable users to sign in using their Facebook credentials.

Before you begin

- Ensure that the OAuth feature is enabled.



- Ensure you have registered an OAuth application in Facebook and have the **API key** and **API secret** values of the application. For information on registering an application in Facebook, see https://developers.facebook.com/docs/facebook-login/web/.
  - In the OAuth application that you create in Facebook, provide the Developer Portal URL in the following format:

    ```
    Developer_Portal_URL/portal/rest/v1/login/callback?provider=facebook&tenant=<tenant_name>
    ```

To enable SSO onboarding using Facebook credentials:

1. Click the menu options icon  from the title bar and click **Administration**.
2. Select **OAuth**.
3. Select *Facebook* from the **Providers** tab.
4. Provide the **API key** and **API secret** values from the OAuth application registered in Facebook.
5. Click **Save**.

   Your changes are saved.

Alternative steps

- Enable users to use their Google or GitHub credentials to sign in to Developer Portal. You must register an OAuth application in the required social media applications and provide the **API key** and **API secret** values of the application.
  - For information on registering an application in Google, see https://blog.rebex.net/howto-register-gmail-oauth.

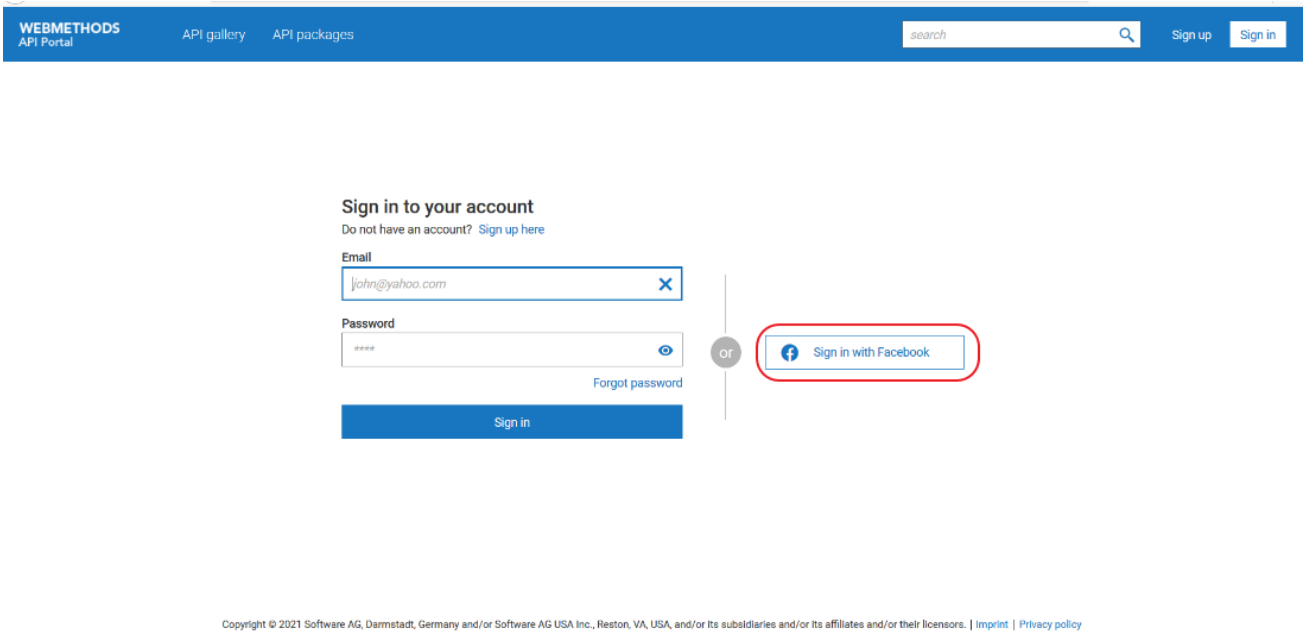
- For information on registering an application in GitHub, see https://docs.github.com/en/developers/apps/creating-an-oauth-app.

  Note:
  You must provide the calback URL for the GitHub application in the following format:

  ```
  Developer_Portal_URL/portal/rest/v1/login/callback
  ```

Next steps

- The **Sign in with Facebook** button appears in the **Sign in** page.



  User can click this button, provide their Facebook credentials to sign up to Developer Portal. The sign-up request goes through the onboarding strategy.

User Preferences

You can set your preferences by accessing your profile, where you can:

- Update your contact details
- Change your password
- View your communities
- Configure notification preferences
- Schedule the frequency of usage reports

Modifying user details

You can modify your user details in the **Profile** section.

To edit your profile

1. Click the menu options icon  from the title bar.
2. Click your *user name*.
3. Modify the required user details.
4. Click **Save**.

   Your changes are saved.

Changing your Password

Change your password in the user profile section after your first login.

To change your password

1. Click the menu options icon  from the title bar.
2. Click your *user name*.
3. Click **Change password**.
4. Provide your new password in the **New password** and **Confirm password** fields.
5. Click **Save**.

   Your password is changed.

Viewing communities

You can view the list of communities that you are a part of.

To view the list of communities

1. Click the menu options icon  from the title bar.
2. Click your *user name*.
3. Click **My communities**.

The list displays the names of all the communities that you are part of along with their descriptions.

Configuring notification preference

Configuring notification preference allows you to stay updated on important events and activities related to your assets.

To configure your notification preference

1. Click the menu options icon  from the title bar.
2. Click your *user name*.
3. Click **Notification preference**.
4. Select your preferred notification method for various events and activities.

   The available options are:

   - **In app**. Receive notifications directly within the application. These alerts appear while you are using the application, allowing you to stay informed without checking external platforms.
   - **Email**. Receive notifications through email. These alerts are sent to your registered email address, enabling you to stay updated even when you are not actively using the application.
5. Click **Save**.

   You receive notifications based on the options you choose.

Scheduling usage reports

Users can specify the frequency that they want to receive the **Application usage** report. This report displays the break-up of an API within an application and first five applications and first five APIs within the application.

To schedule usage reports

1. Click the menu options icon  from the title bar.
2. Click your *user name*.
3. Click **Usage report preference**.

4. Select a value for the frequency at which the report is generated and sent to the configured email. The available options are:

   - **Daily**: Specifies that the reports are generated daily and provides the usage of applications on daily basis. Reports are scheduled at12 AM every day.
   - **Weekly**: Specifies that the reports are generated every week and provides the usage of applications for the last seven days. Reports are scheduled at12 AM every Friday.
   - **Monthly**: Specifies that the reports are generated every month and provides the usage of applications for a month. Reports are scheduled at12 AM on the first day of the month.

5. Click **Save**.

   The reports are now scheduled and emailed to the configured email as per the frequency set. The report sent out has two sections.

   - **Summary**: Consists of a list of applications owned by the user along with their corresponding usage metrics.
   - **Details**: Contains the detailed information for each application, the API associated with the application, and the usage count.

Data Anonymization

Data protection laws and regulations, such as the General Data Protection Regulation (GDPR) requires specific handling of user's personal data, even after a user is removed. Additionally, employees or other clients with user accounts on Developer Portal may request that any user identifying information such as user name, email addresses, or client IP addresses be removed from Developer Portal.

When a user is deleted from Developer Portal, the audit events retain the information about the user, which should be deleted or anonymized.

When you anonymize user data, the corresponding user name is replaced with *anonymous* in all applicable instances of the UI.

You can anonymize user data automatically or manually as follows:

- **Automatic**. From the Developer Portal UI, you can configure to anonymize user accounts automatically as and when they are deleted from Developer Portal. The automatic anonymization is enabled by default. When you delete a user account, it is automatically anonymized after a delay of two minutes.
- **Manual**. If you do not want to automatically anonymize the deleted account, use a REST API to anonymize the required list of user accounts.

How do I enable or disable automatic anonymization of deleted user data through Developer Portal UI?

This section explains the steps to enable automatic anonymization of deleted user accounts.

To enable automatic anonymization user data using UI

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **General**.
3. Select **Enable automatic user anonymization** if not selected.

   This is selected by default. Clearing the check box disables automatic anonymization.
4. Click **Save**.

   Your configurations are saved.

How do I anonymize the deleted user data using REST API?

This section describes the REST API used to anonymize the deleted user accounts. You can use the REST API for anonymizing user accounts in bulk.

After you delete a user account, you must wait for a minimum of ten minutes to perform anonymization. This is to ensure that the background operations related to user account removal is completed before the anonymization.

To anonymize user data using REST API

1. Make a REST call with the required list of user names to the following endpoint:

   ```
   PUT  /users/anonymize
   ```

   You can provide the list of user accounts within quotes like shown here:

   ```
   ["user1","user2","user3"]
   ```

   The specified user accounts are anonymized.

5 Providers

Overview

API providers have the privileges to publish and manage APIs in Developer Portal. You can add and manage providers using the **Manage providers** section.

Developer Portal sends notifications to API providers regarding any event for an associated API, like a token request, through the callback URL of the provider.

The providers added to Developer Portal can publish their APIs to their consumers. When you configure a Developer Portal destination in an API Gateway instance, the corresponding API Gateway instance is added as a provider in Developer Portal.

How do I create a provider?

This use case starts when you want to create a provider and ends when you have successfully created one.

In this example, consider creating a provider *provider1* with the *pet_v1* and *pet_v2* APIs, and the *http://api-dev.xyz.com/rest/apigateway/accesstokens* callback URL.

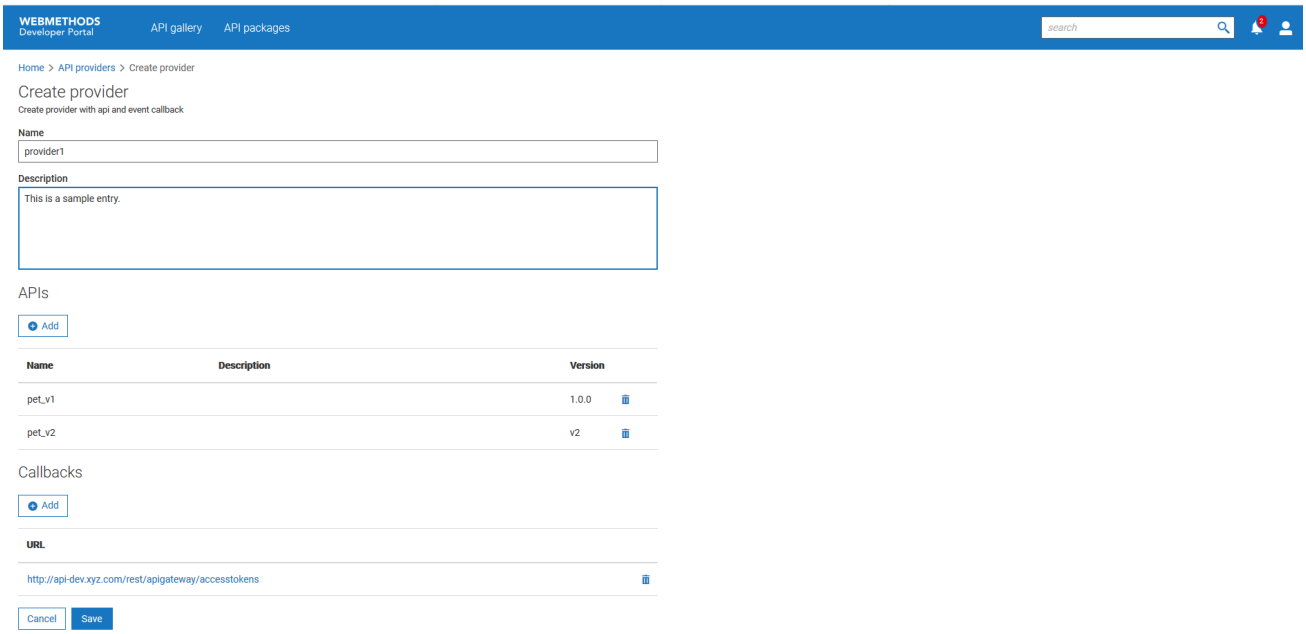
Before you begin:

Ensure that you have the **API Administrator** privilege.

To create a provider

1. Click the user menu icon  from the title bar and click **Manage providers**.
2. Click **Create provider**.
3. Provide *Provider1* in the **Name** field.
4. Select the *pet_v1* and *pet_v2* APIs from the **APIs** field.
5. Select the *http://api-dev.xyz.com/rest/apigateway/accesstokens* callback URL from the **Callback URLs** field.

   APIs that are not associated with a provider are displayed for selection because the assets associated with a provider cannot be associated with other providers.



6. Click **Save**.

   The new provider appears in the **API providers** page.

How do I map an API or a callback URL to a provider?

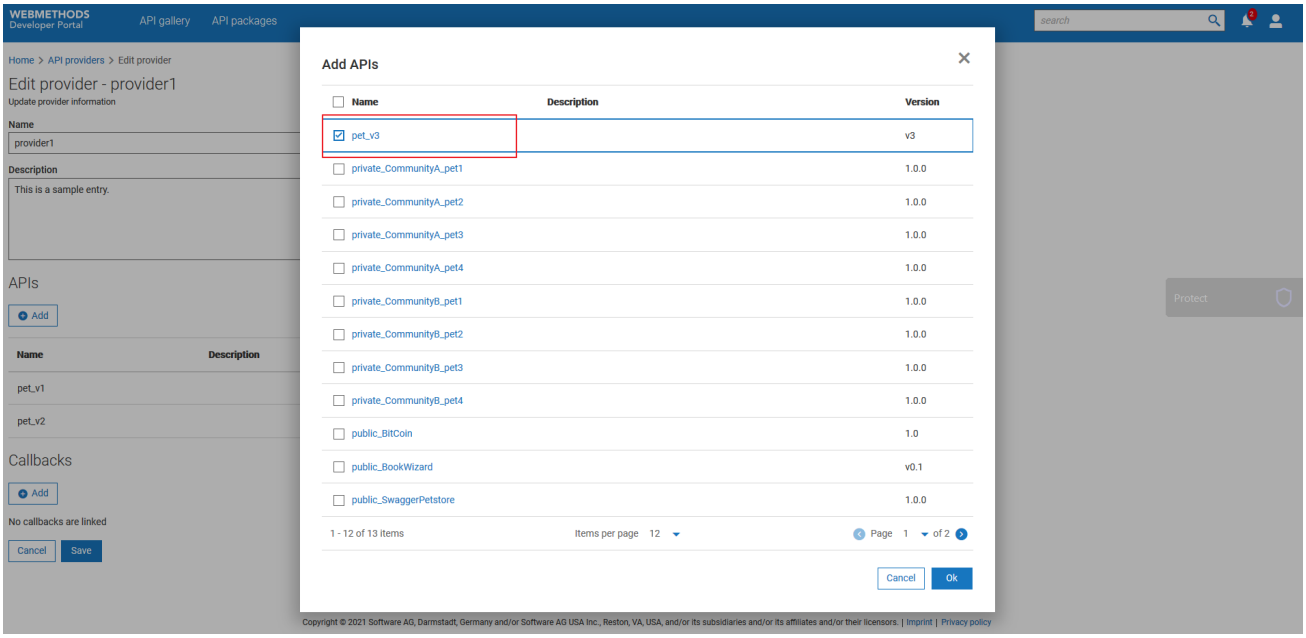
When an API Gateway user publishes an API, the corresponding instance is added as a provider automatically. You can edit the provider to map the required APIs and callback URLs to the provider.

This use case begins when you want to edit a provider and ends after you successfully save your changes.

In this example, consider mapping the *pet_v3* API to *provider1*.

To map an API or a callback URL to a provider

1. Click the user menu icon from the title bar and click **Manage providers**.
2. Click the edit icon next to the provider, *provider1*.
3. click **Add** in the **API** section.
4. Select *pet_v3* from **Add APIs**.



5. Click **Save**.

   Your changes are saved. The *pet_v3* API is mapped to *provider1*. The provider can now manage the assets assigned to them.

Alternative steps:

- To edit provider details, click provider name from the **Manage providers** page and click **Edit** from the provider details page that appears.

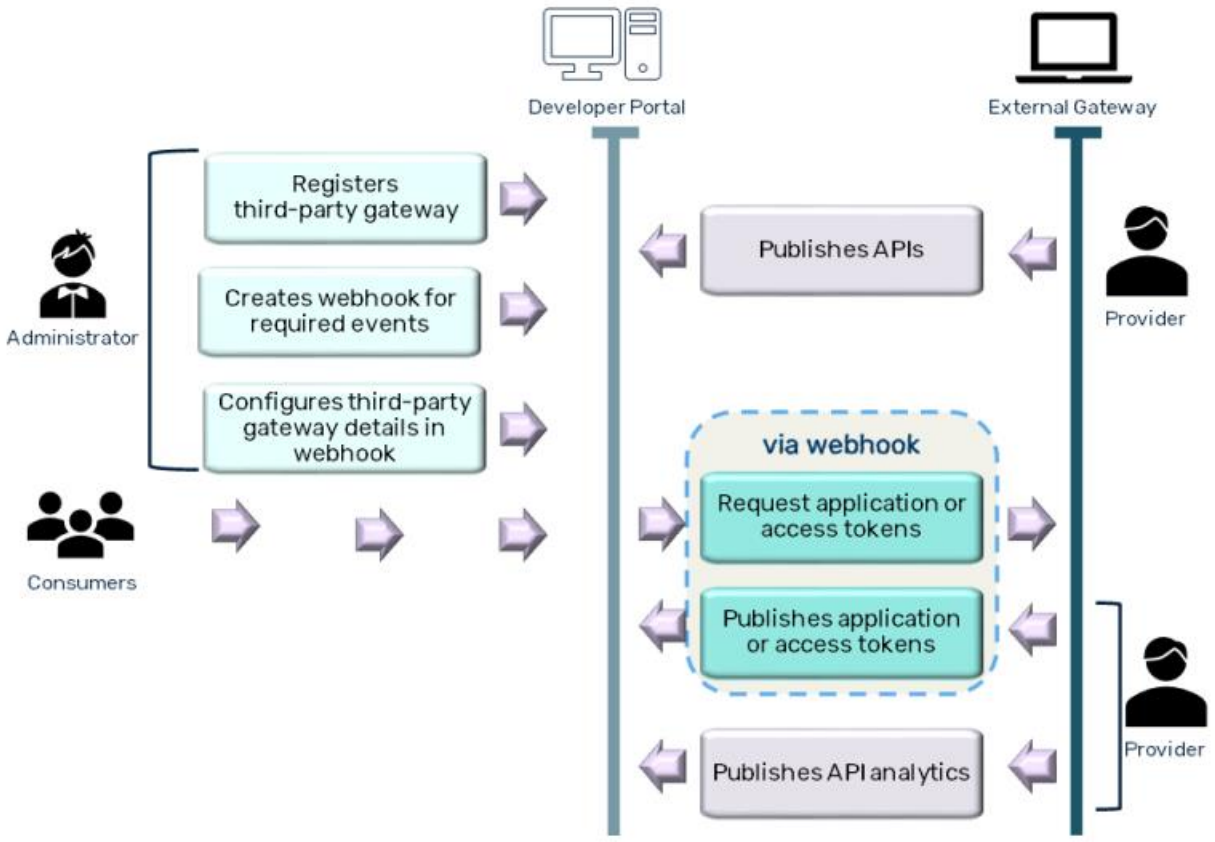
Next steps:

- The providers can now manage the APIs assigned to them.

Third-party API Gateway Configuration

You can integrate third-party gateways with Developer Portal to allow them publish their APIs to Developer Portal. You can add the third-party provider details in Developer Portal and establish communication between the two using webhooks. The registered third-party providers can also publish API analytics to Developer Portal.

The high-level workflow is:



You must also include the third-party provider URL as a CORS origin using the CORS setting in the **Administration** section. For information about specifying CORS origin URL, see "How do I specify Cross-Origin Resource Sharing (CORS) URLs?" on page 32.

Integrating Third-party Gateway with Developer Portal

You must provide the third-party gateway details and add it as a provider to integrate the gateway with Developer Portal.

To integrate third-party gateway with Developer Portal

1. Add the third-party gateway as an API provider using one of these methods:
   - Make a REST call to the following endpoint:

     ```
     POST  /rest/v1/providers
     ```

     Sample request

     ```
     {
         "name": "X Gateway",
         "shortDescription": "External Gateway",
         "description": "External Gateway",
         "version": "10.15",
         "external_key_provider": true
     ```

```
}
```

Sample response

```
{
    "id": "7ecefe01-efc7-43e4-9a19-57c3a761e252",
    "_self": "/portal/rest/v1/providers/7ecefe01-efc7-43e4-9a19-57c3a761e252"
}
```

For information about the *providers* REST API, see "Managing providers" on page 262.

- Create provider from the **Manage providers** page of Developer Portal UI. For detailed information about adding a provider using UI, see "How do I create a provider?" on page 150.

Next steps:

- Create and attach webhook to enable communication between the gateway and Developer Portal. Later, you can publish APIs from the registered gateway. For more information, see "Creating and Attaching Webhook" on page 154.

Creating and Attaching Webhook

After you specify third-party gateway details in Developer Portal, you must configure a webhook to enable communication between the two. Developer Portal uses the webhook to communicate with the gateway during the following events:

- Gateway application creation
- Gateway application scope increase
- Gateway application scope decrease
- Gateway application update

To create and attach webhooks

1. Create a webhook for the specified events by making a REST call to the following endpoint:

```
POST  /rest/v1/hooks
```

Sample request

```
{
    "url": "https://hookbin.com/NOjmeDg6o0ue8mNN8PxR",
    "configuration_type": "PROVIDER",
    "subscriptions": [
        {
            "id": "GATEWAY_APPLICATION_CREATION_REQUEST_EVENT"
        },
        {
            "id": "GATEWAY_APPLICATION_SCOPE_INCREASE_REQUEST_EVENT"
        },
        {
            "id": "GATEWAY_APPLICATION_SCOPE_DECREASE_REQUEST_EVENT"
```

```
        },
        {
            "id": "GATEWAY_APPLICATION_UPDATION_REQUEST_EVENT"
        }
    ]
}
```

Sample response

```
{
    "id": "07522ff8-fccd-45a0-b934-0ab67dffdf86",
    "_self": "/portal/rest/v1/hooks/07522ff8-fccd-45a0-b934-0ab67dffdf86"
}
```

For information about using the *hooks* REST API, see "Managing webhooks" on page 269.

You can also create webhooks using the Developer Portal UI. For information about creating webhooks using UI, see "How do I configure webhooks to notify events to an external system?" on page 23

2. Specify the provider Id in Developer Portal by making a REST call to the following endpoint:

```
PUT  /rest/v1/providers/id
```

Sample request

```
PUT /portal/rest/v1/providers/cf864096-75c9-4d46-8002-923d8ebbcd67
{
    "name": "X Gateway",
    "owner": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
    "id": "cf864096-75c9-4d46-8002-923d8ebbcd67",
    "providerRef": "cf864096-75c9-4d46-8002-923d8ebbcd67",
    "stage": "X Gateway",
    "type": "THIRD_PARTY",
    "summary": "External Gateway",
    "description": "External Gateway",
    "webhooks": [
        "07522ff8-fccd-45a0-b934-0ab67dffdf86"
    ]
}
```

Sample response

```
{
    "name": "X Gateway",
    "owner": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
    "id": "cf864096-75c9-4d46-8002-923d8ebbcd67",
    "documentType": "PROVIDER",
    "providerRef": "cf864096-75c9-4d46-8002-923d8ebbcd67",
    "stage": "X Gateway",
    "webhooks": [
        "07522ff8-fccd-45a0-b934-0ab67dffdf86"
    ],
    "type": "THIRD_PARTY",
    "summary": "External Gateway",
    "description": "External Gateway"
}
```

Next steps:

- Publish APIs for the gateway that you registered with Developer Portal. For more information, see "Publishing APIs from Third-party Gateway to Developer Portal" on page 156.

Publishing APIs from Third-party Gateway to Developer Portal

You can publish SOAP or REST APIs for the registered gateways to Developer Portal using any of these specifications - File, URL, or content.

To publish APIs from third-party gateway to Developer Portal

1. Publish APIs using any of the following methods:
   - Make a REST call to the following endpoint:

     ```
     POST  /rest/v1/apis
     ```

     Sample request

     ```
     POST /rest/v1/apis HTTP/1.1
     Host: localhost:18151
     Authorization: Basic QWRtaW5pc3RyYXRvcjpQQHNzdzByZEAxMg==
     Content-Length: 564
     Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

     ----WebKitFormBoundary7MA4YWxkTrZu0gW
     Content-Disposition: form-data; name="url"

     https://petstore.swagger.io/v2/swagger.json
     ----WebKitFormBoundary7MA4YWxkTrZu0gW
     Content-Disposition: form-data; name="type"

     swagger
     ----WebKitFormBoundary7MA4YWxkTrZu0gW
     Content-Disposition: form-data; name="async"

     false
     ----WebKitFormBoundary7MA4YWxkTrZu0gW
     Content-Disposition: form-data; name="providerid"

     cf864096-75c9-4d46-8002-923d8ebbcd67
     ----WebKitFormBoundary7MA4YWxkTrZu0gW
     Content-Disposition: form-data; name="name"

     petstore1
     ----WebKitFormBoundary7MA4YWxkTrZu0gW
     ```

     Sample response

     ```
     {
         "id": "842c3e33-5c4b-4abf-b827-1a194affde50",
         "_self": "/portal/rest/v1/apis/842c3e33-5c4b-4abf-b827-1a194affde50"
     }
     ```

     For information about the *providers* REST API, see "Managing APIs " on page 244.

- Publish APIs through the Developer Portal UI. For detailed information about publishing APIs, see "How do I create an API?" on page 178.

Next steps:

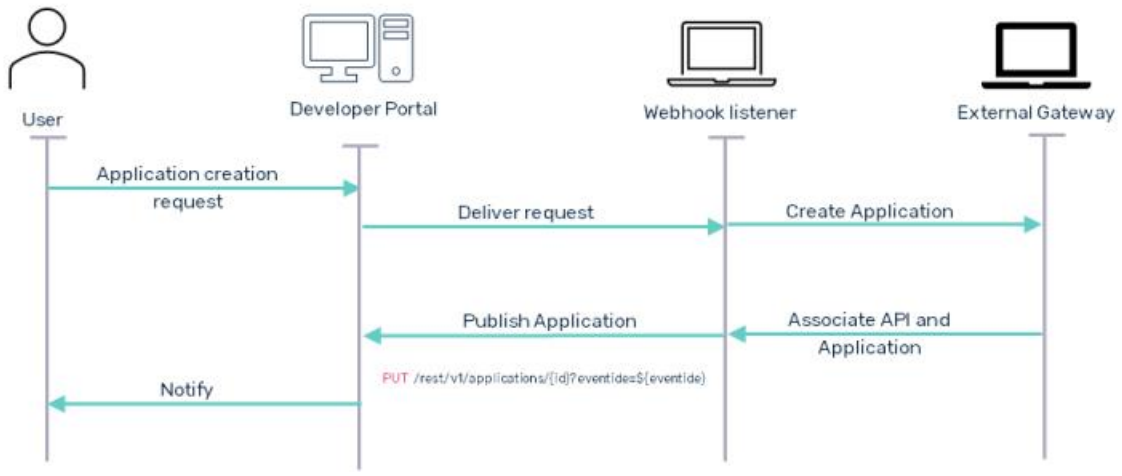
- View the published API from the **API gallery** page and the **Manage APIs** page.
- Request access tokens or application to invoke APIs published from third-party gateway. For more information, see "Requesting Application for Third-party Gateway APIs" on page 157.

Requesting Application for Third-party Gateway APIs

When you request for an application to access an API from a third-party gateway, the webhook attached to the gateway communicates the request to the gateway as an event payload.

The gateway then creates API key, JWT, or OAuth access token for the requested API. You can then update the access token credentials in Developer Portal.

The following diagram shows the basic flow of the application approval process:



To request application from third-party gateway:

1. Request for application to test an API using one of these methods:
   - Make a REST call to the following endpoint:

     ```
     POST  /rest/v1/requests
     ```

     For example, the following request creates an app *Petstore App* for invoking the API *petstore1*

     ```
     {
         "context": {
             "name": "Petstore App",
             "description": "This application is used for building petstore app",
             "apis": [
                 "842c3e33-5c4b-4abf-b827-1a194affde50"
             ]
         },
     ```

```
    "type": "APPLICATION_CREATION_REQUEST"
}
```

Sample response

```
{
    "id": "15fe0192-24e4-422a-afa5-58879161b9dc",
    "_self": "/portal/rest/v1/requests/15fe0192-24e4-422a-afa5-58879161b9dc"
}
```

Sample webhook payload sent to the callback URL of the third-party gateway

```
{
  "executor": {
    "id": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
    "email": null,
    "lastname": "System",
    "firstname": "Administrator",
    "name": "administrator"
  },
  "source": {
    "id": "842c3e33-5c4b-4abf-b827-1a194affde50",
    "externalref": "24372f80-072e-455f-9235-5c01fbe6f139"
  },
  "contextdata": {
    "consumer_appName": "Petstore App",
    "consumer_appDesc": "This application is used for building petstore app",
    "tenant": "default",
    "redirect_uris": [
      "https://hostname:18152/portal/rest/v1/oauth/callback"
    ],
    "provider_id": null,
    "api_id": "842c3e33-5c4b-4abf-b827-1a194affde50",
    "application_id": "6179b933-1382-4e1a-becc-239dcfeaaa6d"
  },
  "type": "REQUEST_ACCESS_TOKEN",
  "creationdate": "2022-01-10 14:36:14",
  "eventid": "5790ab9a-5312-4122-8aa8-8ca21cf98663",
  "status": "NEW"
}
```

- Request for an application through Developer Portal UI. For detailed information about creating an application request, see "Creating an application" on page 200.

2. Update the access token credentials (API Key, OAuth, and JWT) in Developer Portal by making REST call to:

```
PUT  /rest/v1/applications/{id}?eventId=${eventId}
```

For example, this request updates the access token required to access the API, *petstore1*.

```
PUT
/rest/v1/applications/6179b933-1382-4e1a-becc-239dcfeaaa6d?eventId=5790ab9a-5312-4122-8aa8-8ca21cf98663
{
    "name": "Petstore App",
     "credentials": [
        {
            "expiry": null,
            "apiKey": "709cb14b-f19f-4df9-98b8-eebe727f7844",
```

```
            "type": "APIKey"
        },
        {
            "clientId": "3d2404cb-d07d-4101-b38a-f13005b4cc53",
            "clientSecret": "caed6fb3-7fd3-479f-8db5-81b4aeeeba86",
            "scopes": [],
            "tokenLifeTime": 3600,
            "tokenRefreshLimit": 0,
            "authorizationUris": [
                "https://hostname:5543/invoke/pub.apigateway.oauth2/authorize"
            ],
            "accessTokenUris": [
              "https://hostname:5543/invoke/pub.apigateway.oauth2/getAccessToken"
            ],
            "redirectUris": [
                "https://hostname:18152/portal/rest/v1/oauth/callback"
            ],
            "refreshTokenUris": [
                "https://hostname:5543/invoke/pub.oauth/refreshAccessToken"
            ],
            "type": "OAuth2"
        },
        {
            "claimsets": [
                {
                    "name": "JWT default claims set",
                    "claims": [
                        {
                            "key": "app_id",
                            "value": "29cfdae1-c8c5-44d4-9214-5415496d3d4a"
                        }
                    ]
                }
            ],
            "accesstoken_uris": null,
            "type": "JWT"
        }
    ]
}
```

Sample response

```
{
    "code": 200,
    "message": "Application is updated successfully"
}
```

Next steps:

You can share the access tokens generated for an API to other APIs by increasing the scope of the application.

You must raise a request to the third-party gateway to add the new API details to an existing application. The gateway then updates the application, generates a new access token, and sends the token to Developer Portal.

The following example describes how to add an API, *petstore2*, to the previously created application, *Petstore App*.

- Make a REST call to the following endpoint:

```
POST  /rest/v1/requests
```

Sample request

```
POST /rest/v1/requests
{
    "context": {
        "application": "6179b933-1382-4e1a-becc-239dcfeaaa6d",
        "apis": [
            "717eaf03-3c59-448e-8acb-7f3a52667932"
        ]
    },
    "type": "APPLICATION_API_REGISTRATION_REQUEST"
}
```

Sample webhook payload sent to the third-party gateway

```
PUT
/rest/v1/applications/6179b933-1382-4e1a-becc-239dcfeaaa6d?eventId=ebdcef36-1bf6-4998-9556-9fa38b3785b9
{
    "name": "Petstore App",
     "credentials": [
        {
            "expiry": null,
            "apiKey": "709cb14b-f19f-4df9-98b8-eebe727f7844",
            "type": "APIKey"
        },
        {
            "clientId": "3d2404cb-d07d-4101-b38a-f13005b4cc53",
            "clientSecret": "caed6fb3-7fd3-479f-8db5-81b4aeeeba86",
            "scopes": [],
            "tokenLifeTime": 3600,
            "tokenRefreshLimit": 0,
            "authorizationUris": [
                "https://hostname:5543/invoke/pub.apigateway.oauth2/authorize"
            ],
            "accessTokenUris": [
              "https://hostname:5543/invoke/pub.apigateway.oauth2/getAccessToken"
            ],
            "redirectUris": [
                "https://hostname:18152/portal/rest/v1/oauth/callback"
            ],
            "refreshTokenUris": [
                "https://hostname:5543/invoke/pub.oauth/refreshAccessToken"
            ],
            "type": "OAuth2"
        },
        {
            "claimsets": [
                {
                    "name": "JWT default claims set",
                    "claims": [
                        {
                            "key": "app_id",
                            "value": "29cfdae1-c8c5-44d4-9214-5415496d3d4a"
                        }
                    ]
                }
```

```
            ],
            "accesstoken_uris": null,
            "type": "JWT"
        }
    ]
}
```

Update the application in Developer Portal by making a REST call to:

```
PUT  /rest/v1/applications/id?eventId=eventId
```

Sample request

```
PUT
/rest/v1/applications/6179b933-1382-4e1a-becc-239dcfeaaa6d?eventId=ebdcef36-1bf6-4998-9556-9fa38b3785b9
{
    "name": "Petstore App",
     "credentials": [
        {
            "expiry": null,
            "apiKey": "709cb14b-f19f-4df9-98b8-eebe727f7844",
            "type": "APIKey"
        },
        {
            "clientId": "3d2404cb-d07d-4101-b38a-f13005b4cc53",
            "clientSecret": "caed6fb3-7fd3-479f-8db5-81b4aeeeba86",
            "scopes": [],
            "tokenLifeTime": 3600,
            "tokenRefreshLimit": 0,
            "authorizationUris": [
                "https://hostname:5543/invoke/pub.apigateway.oauth2/authorize"
            ],
            "accessTokenUris": [
              "https://hostname:5543/invoke/pub.apigateway.oauth2/getAccessToken"
            ],
            "redirectUris": [
                "https://hostname:18152/portal/rest/v1/oauth/callback"
            ],
            "refreshTokenUris": [
                "https://hostname:5543/invoke/pub.oauth/refreshAccessToken"
            ],
            "type": "OAuth2"
        },
        {
            "claimsets": [
                {
                    "name": "JWT default claims set",
                    "claims": [
                        {
                            "key": "app_id",
                            "value": "29cfdae1-c8c5-44d4-9214-5415496d3d4a"
                        }
                    ]
                }
            ],
            "accesstoken_uris": null,
            "type": "JWT"
        }
    ]
}
```

- Decrease the scope of an application by making a REST call to the following endpoint:

```
POST  /rest/v1/requests
```

For example, to remove *petstore2* from the previously created application, *Petstore App*.

```
POST /rest/v1/requests
{
    "context": {
        "application": "6179b933-1382-4e1a-becc-239dcfeaaa6d",
        "apis": [
            "842c3e33-5c4b-4abf-b827-1a194affde50"
        ]
    },
    "type": "APPLICATION_API_DEREGISTRATION_REQUEST"
}
```

Sample webhook payload sent to the third-party gateway

```
{
  "executor": {
    "id": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
    "email": null,
    "lastname": "System",
    "firstname": "Administrator",
    "name": "administrator"
  },
  "source": {
    "id": "6179b933-1382-4e1a-becc-239dcfeaaa6d",
    "externalref": null
  },
  "contextdata": {
    "api": {
      "id": "842c3e33-5c4b-4abf-b827-1a194affde50",
      "externalRef": "24372f80-072e-455f-9235-5c01fbe6f139"
    }
    "application_id": "6179b933-1382-4e1a-becc-239dcfeaaa6d"
  },
  "type": "REVOKE_ACCESS_TOKEN",
  "creationdate": "2022-01-10 15:47:38",
  "eventid": "8115a10f-451f-4cce-9bb6-7beb1e4722d3",
  "status": "NEW"
}
```

Update the application in Developer Portal by making a REST call to:

```
PUT  /rest/v1/applications/id?eventId=eventId
```

Sample request

```
PUT
/rest/v1/applications/6179b933-1382-4e1a-becc-239dcfeaaa6d?eventId=8115a10f-451f-4cce-9bb6-7beb1e4722d3
```

- Check the status of an application and the list of APIs associated with it by making a REST call to:

```
GET  /rest/v1/applications/id
```

Sample request

```
GET /rest/v1/applications/6179b933-1382-4e1a-becc-239dcfeaaa6d

{
    "name": "Petstore App",
    "owner": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
    "id": "6179b933-1382-4e1a-becc-239dcfeaaa6d",
    "documentType": "APPLICATION",
    "providerRef": "7ecefe01-efc7-43e4-9a19-57c3a761e252",
    "access": {
        "teams": [],
        "users": []
    },
    "apis": [
        "842c3e33-5c4b-4abf-b827-1a194affde50",
        "717eaf03-3c59-448e-8acb-7f3a52667932"
    ],
    "credentials": [
        {
            "expiry": null,
            "apiKey": "709cb14b-f19f-4df9-98b8-eebe727f7844",
            "type": "APIKey"
        },
        {
            "clientId": "3d2404cb-d07d-4101-b38a-f13005b4cc53",
            "clientSecret": "caed6fb3-7fd3-479f-8db5-81b4aeeeba86",
            "scopes": [],
            "tokenLifeTime": 3600,
            "tokenRefreshLimit": 0,
            "authorizationUris": [
                "https://hostname:5543/invoke/pub.apigateway.oauth2/authorize"
            ],
            "accessTokenUris": [
              "https://hostname:5543/invoke/pub.apigateway.oauth2/getAccessToken"
            ],
            "redirectUris": [
                "https://hostname:18152/portal/rest/v1/oauth/callback"
            ],
            "refreshTokenUris": [
                "https://hostname:5543/invoke/pub.oauth/refreshAccessToken"
            ],
            "type": "OAuth2"
        },
        {
            "claimsets": [
                {
                    "name": "JWT default claims set",
                    "claims": [
                        {
                            "key": "app_id",
                            "value": "29cfdae1-c8c5-44d4-9214-5415496d3d4a"
                        }
                    ]
                }
            ],
            "accesstoken_uris": null,
            "type": "JWT"
        }
    ],
    "status": "LIVE",
    "slots": {
```

```
        "$stage": "X Gateway"
    },
    "deleted": false,
    "description": "This application is used for building petstore app",
    "app_type": "API"
}
```

- When any application creation, scope increase, or scope decrease transactions fail, you can retry the failed transaction using the following REST calls:

```
GET  /rest/v1/applications/applicationId/requests
```

This request returns the list of requests made for an application by the current user (signed in user or authorization used in the REST call). You must provide the required application Id.

Sample response

```
{
    "result": [
        {
            "owner": "200ceb26-807d-3bf9-9fd6-f4f0d1ca54d4",
            "id": "0d88fdbf-457b-489f-85bf-fb6678474649",
            "modified": "2022-02-10T12:43+0000",
            "created": "2022-02-10T12:43+0000",
            "documentType": "USER_REQUEST",
            "type": "APPLICATION_CREATION_REQUEST",
            "status": "DELIVERY_UNSUCCESSFUL",
            "application": "8d67856d-476f-492c-a82d-d274501ad08a",
            "state": {
                "7f8060c1-056e-47d1-8785-83a3eb8f5c88": "DELIVERY_UNSUCCESSFUL"
            }
        }
    ],
    "count": 1
}
```

From the list of requests returned in the response, you can view the failed requests and retry them using the below REST call with the request Id and state Id values:

```
PUT  /rest/v1/requests/requestId/retry?state=stateId
```

Sample response

```
{
    "code": 200,
    "message": "User request retried successfully"
}
```

Publishing API analytics to Developer Portal

The integrated gateways can provide the transactional events of published APIs in Developer Portal to allow users to view the API analytics in Developer Portal.

To publish API analytics in Developer Portal:

1. Make a REST call to the following endpoint:

```
POST  /rest/v1/events/transactions
```

Sample request

```
POST /rest/v1/events/transactions  {
    "request": null,
    "nativeHttpMethod": "get",
    "apiName": "petstore1",
    "responseCode": 200,
    "apiVersion": 1.0,
    "providerTime": 4711,
    "apiId": "842c3e33-5c4b-4abf-b827-1a194affde50",
    "applicationName": "Sample",
    "applicationIp": "127.0.0.1",
    "consumerId": "567f89cb-8182-3272-b69e-5d8532eb3ac5",
    "totalTime": 6264,
    "operationName": "/pet/findByStatus",
    "totalSize": 4663,
    "response": "******Some String response******",
    "applicationId": "6179b933-1382-4e1a-becc-239dcfeaaa6d",
    "consumerName": "john.doe@sample.com"
}
```

Next steps:

- You can view the analytics from the Developer Portal dashboard.

API Partner Isolation

API partners can be organizations or users who expose their APIs through Developer Portal. As a Developer Portal administrator, you can enable partners to isolate their APIs so that the partners can determine the users or user groups that can access their APIs. Isolation of assets will also restrict the APIs that the providers have access to.

To use this feature, users must register to Developer Portal as *API partners*.

API Partner Capabilities

API partners can

Manage APIs

Partner users can publish APIs from API Gateway or create APIs in Developer Portal, enrich them, and assign the APIs to the required private communities. To allow consumer users to view APIs, API partners can include them in the corresponding private communities. They can also unpublish or delete the required APIs.

Partners can publish their assets to:

- Private communities. Consumers, who are part of the community can view and request applications to try those APIs. Other consumers cannot view those assets.
- Public community. This does not support assets isolation based on partners.

API partners can also create their own communities. They cannot edit other communities.

Manage packages

Similar to APIs, partner users can publish packages from API Gateway to the required private communities. Consumers, who are a part of the published communities can view the packages and subscribe to them.

View Dashboard

API partners can view the analytics of APIs and applications usage from their **Dashboard** section. This section includes only the analytics of assets that the partners have published or have been assigned as owners.

Manage Communities

Once an API partner registers with Developer Portal, administrators can associate them with private communities as administrators. As community administrators, partners can manage their communities, assets associated with their communities, and the community members.

Partners can also create their own communities and manage them.

API gallery

API partners can view:

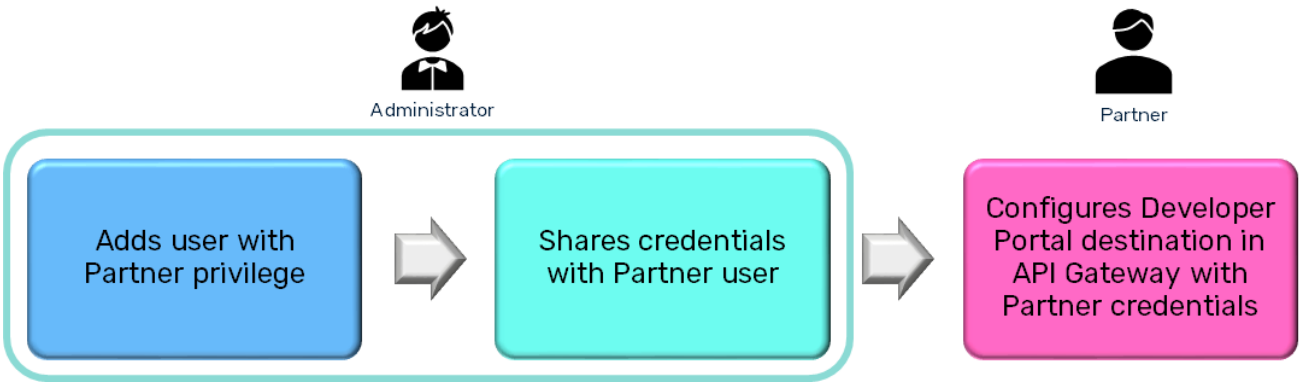
- APIs published to the private communities that they are members of.
- APIs that they published (regardless of their communities).
- APIs that are assigned to them by other partners.

Change asset ownership

API partners can assign their assets to other partners; and the partners can manage the assigned to them by other partners. For more information, see "Assigning Asset Ownership" on page 167.

Onboarding an API partner

As an administrator, you must register users with **Partner** privilege, and add them as administrators of the required communities. As stated earlier, partners can manage their assigned communities, assets associated with their communities, and the community members.



To onboard an API partner

1. Create a user with the **Partner** privilege.

   For information about adding a user, see "How do I add a user?" on page 113.

2. Provide the user credentials to the required users.

Next steps:

- Partners can configure the Developer Portal destination from API Gateway with the credentials shared by administrators.
- Partners can publish APIs and packages from API Gateway to the required private communities so that they can isolate their assets from the assets published by other partners in their private communities.
- Partners can also create APIs from Developer Portal and add these APIs to their private community to isolate the APIs from other partners.

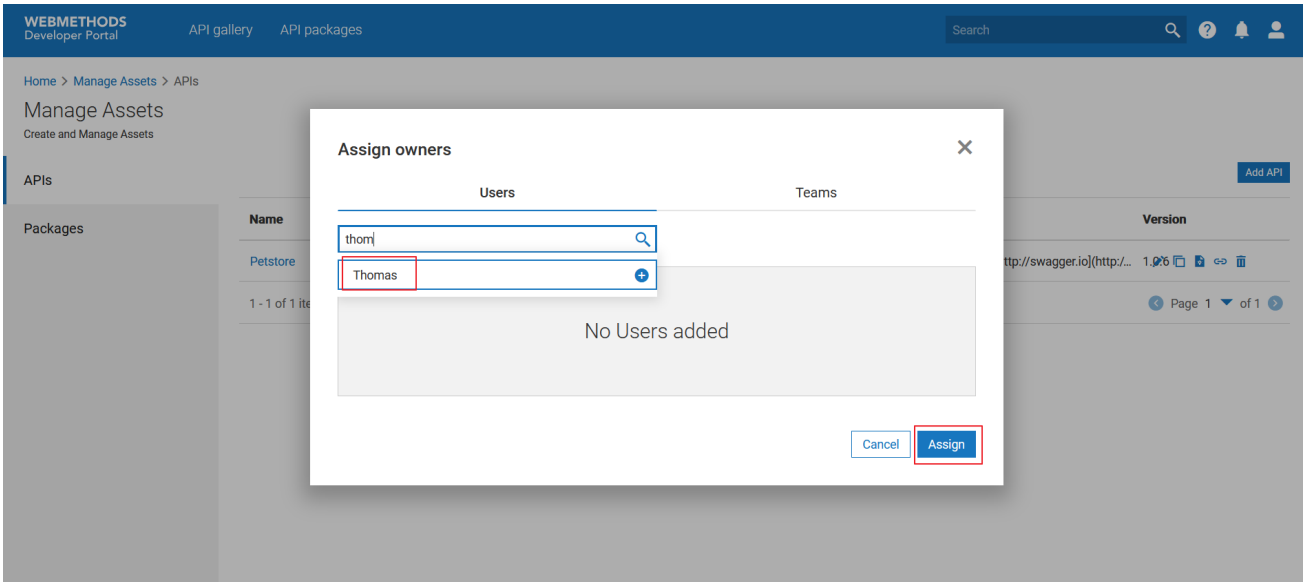
Assigning Asset Ownership

As an API partner, you can assign APIs to other partners; and they can make these assets available to their community members. You can assign an API to the required users or teams. The owners thus assigned to an API can manage the API.

To assign asset ownership

1. Click the menu options icon  from the title bar and click **Manage assets**.
2. Click **APIs** or **Packages** depending on the asset you want to assign.
3. Click  next to the required asset.
4. In the **Assign owners**, select the required partner from the **Users** tab.

   This field displays only the users who have the *Partner* role assigned to them. Users who have the *API Provider* or *Administrator* role can view all APIs. Hence, only the users with the *Partner* role are displayed.



5. Click **Assign**.

   The selected partners become owners of the asset.

Next steps:

- Partners can manage the assets assigned to them by other partners.

6 Communities

Overview

Community facilitates API Administrators or API Providers in handling API visibility among the Developer Portal users.

Users with **API Administrator** and **API Provider** privileges can create communities and manage the members of a community.

There are two types of communities in Developer Portal - private community and public community.

Private community

As the name suggests, a private community contains APIs that are available only for its members.

When publishing an API from API Gateway, providers can select specific communities to restrict the access of asset by other users.

Users can access only the assets that are assigned to their communities. For example, consider your portal provides APIs to the users of multiple domains such as Banking, Healthcare, Telecom, and so on. In such cases, you can group the users of each domain as a community and offer them the required APIs.

As an administrator, you can restrict the visibility of consumer users within their communities. That is, when a consumer searches for other consumers to share an application or package subscription, or during team creation they can view only the consumer users who belong to their communities. For more information about this setting, see "Configuring visibility of users based on communities" on page 174.

Public community

The public community comes along with the product installation and is open to all users, registered or non-registered.

By default, the APIs published from API Gateway and the ones created in Developer Portal are posted as a part of the public community. If an API that is not a part of the public community, you can add the APIs to the public community using the Edit option.

To restrict the access of assets, users must assign them to a private community.

Manage communities in Developer Portal

The **Manage communities** section allows you to group the users who work in a project or users with similar roles as a community to assign APIs only for the access of that community. This feature is helpful for providers that offer APIs to a wide range of customers from various disciplines.

API Administrators have the privileges to assign community administrators, add users to, and remove users from a community. APIs are assigned to specific communities by API Providers, and the packages are published to communities from API Gateway. API consumers have access to APIs and packages depending on whether they belong to a specific community or not. Consumers can view the API(s) that belong to a community by grouping the APIs by Communities in the **API gallery** page.

How do I create a community?

You can create any number of communities.

This use case begins when you want to create a community and ends when you have created one.

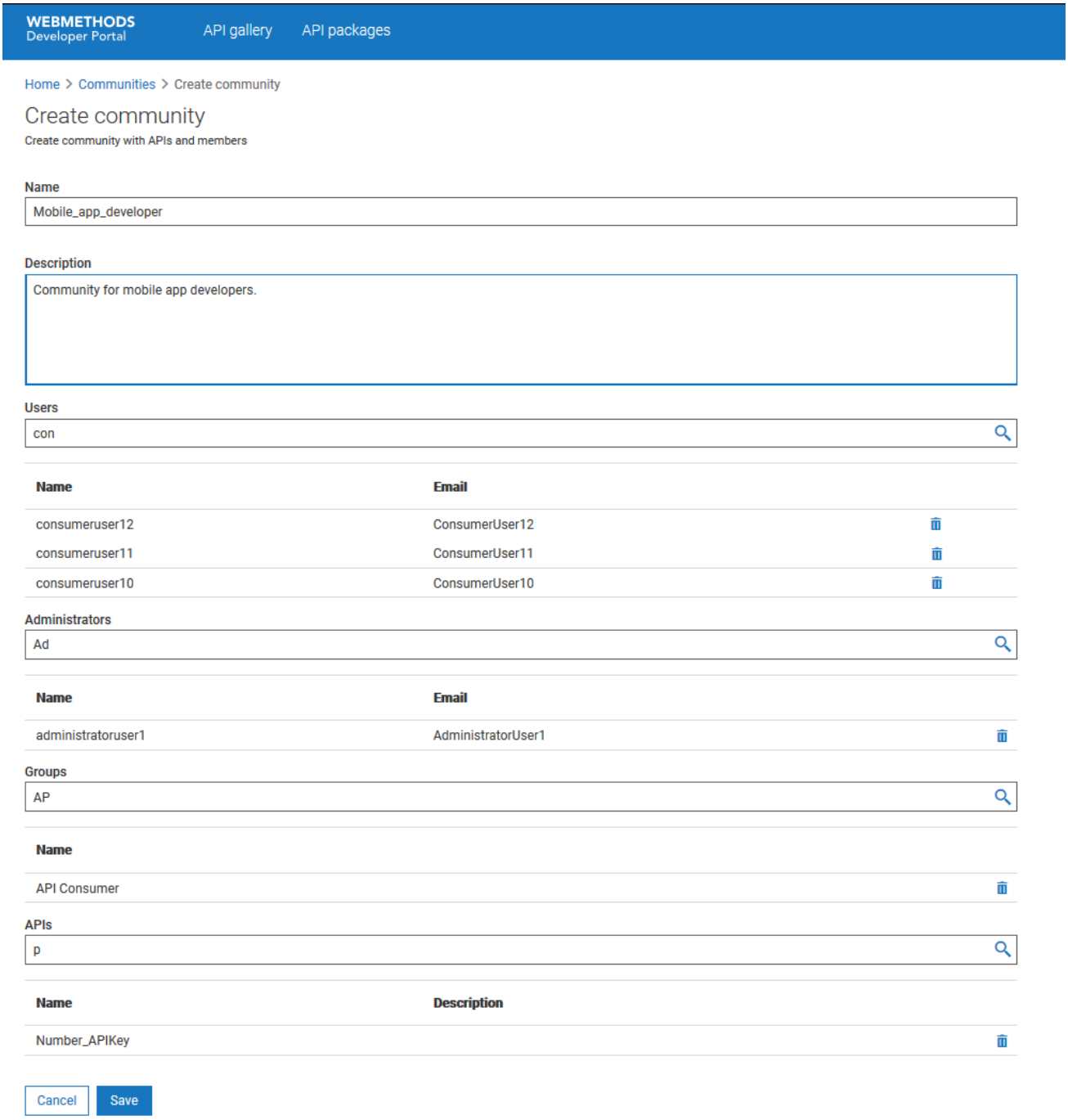
In this example, a private community *Mobile_app_developer* is created with users *consumeruser10*, *consumeruser11*, and *consumeruser12* as members, the user *administratoruser1* as community administrator, the user group *API Consumer*, and the API *Number_APIKey*.

Before you begin:

Ensure that you have the **API Administrator** privilege.

To create a community

1. Click the menu options icon  from the title bar and click **Manage communities**.
2. Click **Create community**.
3. Provide *Mobile_app_developer* in the **Name** field.
4. Select *consumeruser10*, *consumeruser11*, and *consumeruser12* from the **Users** field.
5. Select *administratoruser1* from the **Administrators** field.
6. Select *API Consumer* from the **Groups** field.
7. Select *Number_APIKey* from the **APIs** field.



8. Click **Save**.

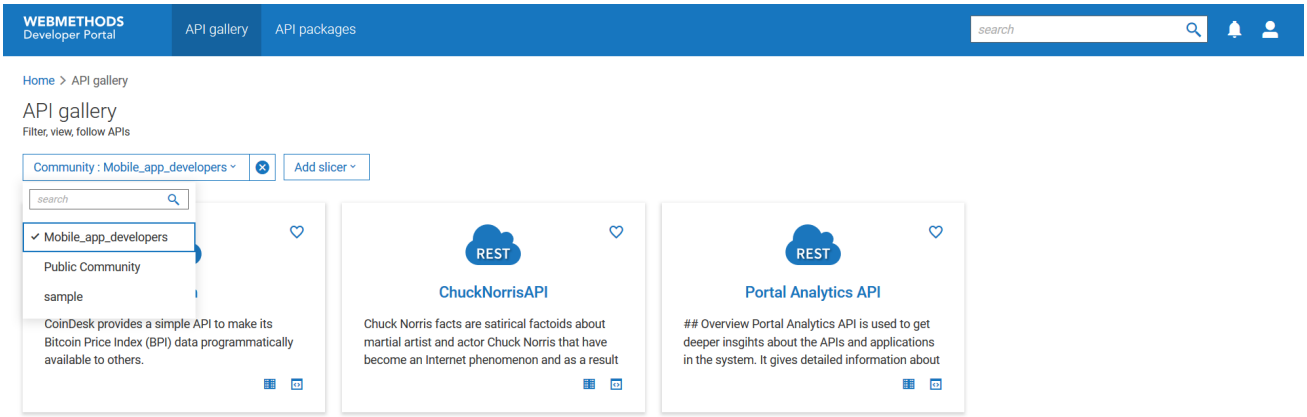
   The *Mobile_app_developer* community is created.

Alternative steps:

- Modify the list of the mapped users and APIs using the **Edit communities** option.

Next steps:

- Select a community in the **API gallery** page to view the APIs of that community.



How do I map the required user, group, or API to a community?

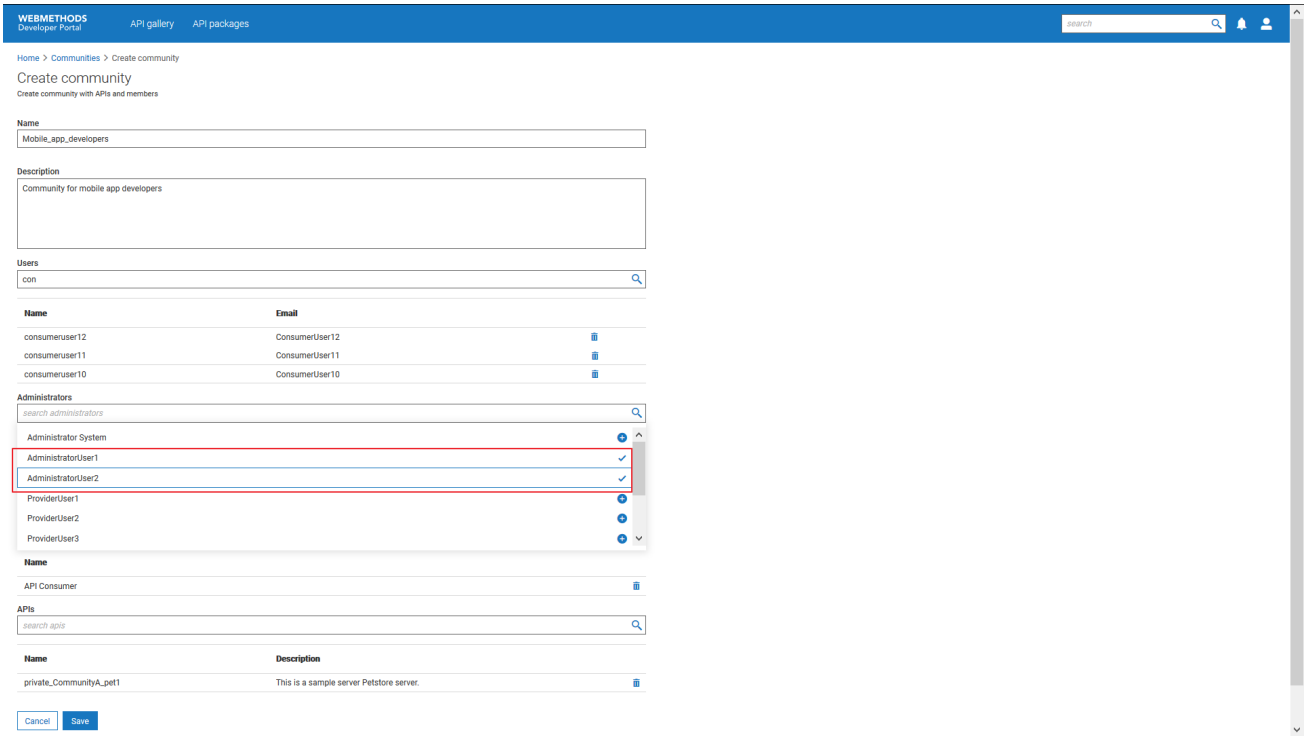
You can edit the community to map the required users, user groups, and APIs to the community.

This use case begins when you want to edit any existing communities and ends when you have saved the changes.

In this example, consider the community *Mobile_app_developers* that has one administrator *administratoruser1* must be mapped to one more administrator, *administratoruser2*.

To map users, user groups, or APIs to a community

1. Click the menu options icon from the title bar and click **Manage communities**.
2. Click the edit icon next to **Mobile_app_developers**.
3. Select *administratoruser2* in the **Groups** field.
4. Click the add icon .



5. Click **Save**.

   Your changes are saved. The *API providers* user group is mapped to the *Mobile_app_developers* community.

Alternative steps:

- Modify the list of the mapped users and APIs from the corresponding fields.

Configuring visibility of users based on communities

As an administrator you can specify that the consumer users can view only the consumers who belong to their communities. That is, the consumer users cannot view consumers from other communities when they share an application or package subscription, or during team creation.

To configure visibility of users based on community

1. Navigate to the location, *SAGInstallDir*`/DeveloperPortal/configuration`.
2. Open the `dpo_wrapper.conf` or the `custom_wrapper.conf` file to edit the same.
3. Add the following to file:

   ```
   wrapper.java.additional.2101=-Dportal.server.config.user-search-community-restricted
   ```

4. Provide one of these values to the entry:

- *true*. Restricts the visibility of other users based on communities. That is, consumers can only view other consumers from their communities.
- *false*. No restrictions based on communities. Consumer users can view any consumer irrespective of their communities.

Sample

```
wrapper.java.additional.2101=-Dportal.server.config.user-search-community-restricted=false
```

Note:
If the variable *2101* is already used for some other entry in the `dpo_wrapper.config` file, then replace the value with a number greater than that. For example, *2102*

5. Save the changes.
6. Restart the server for the changes to take effect.

7 Assets

APIs

Developer Portal offers you an exclusive platform to safely expose your APIs to your target developers and partners.

Developer Portal also allows developers to self-register, learn about these APIs, and use the APIs in their applications.

To prepare to manage the APIs that you plan to make available in Developer Portal, consider the following questions:

- How many Developer Portal instances you might need?
- Which organizations might use Developer Portal?
- Which users in the organization might use Developer Portal to consume the published APIs?
- Which taxonomies and categories are required to organize the APIs?

For each Developer Portal instance, there is a Developer Portal object registered with the API Provider. A Developer Portal is associated with an organization. Multiple Developer Portal instances can share the same organization.

An API can be published to multiple Developer Portal instances. Developer Portal is capable of managing APIs published from API Gateway or any other provider application.

When an API is unpublished (removed) from Developer Portal, its metadata is deleted from Developer Portal, and the API is no longer available for access.

In addition to APIs published from a provider, you can also create APIs in Developer Portal by providing a corresponding specification. The APIs created in Developer Portal need not be associated with any providers.

Developer Portal also has the Lifecycle management capability for APIs and packages to control the visibility of these assets for consumers based on their active state.

By default, this Lifecycle Management feature is disabled. You can enable this feature from the **Administration** section. For more information about API and package lifecycle management, see "Lifecycle Management of APIs and Packages" on page 224.

You can download the published APIs from Developer Portal to have a copy of the API specification.

How do I create an API?

You can create an API using a file, URL, or by providing the source content.

Developer Portal supports the publishing of OData APIs from provider applications such as API Gateway. However, you cannot create OData APIs in Developer Portal by providing a specification.

This use case starts when you want to create an API and ends when you have successfully created an API.

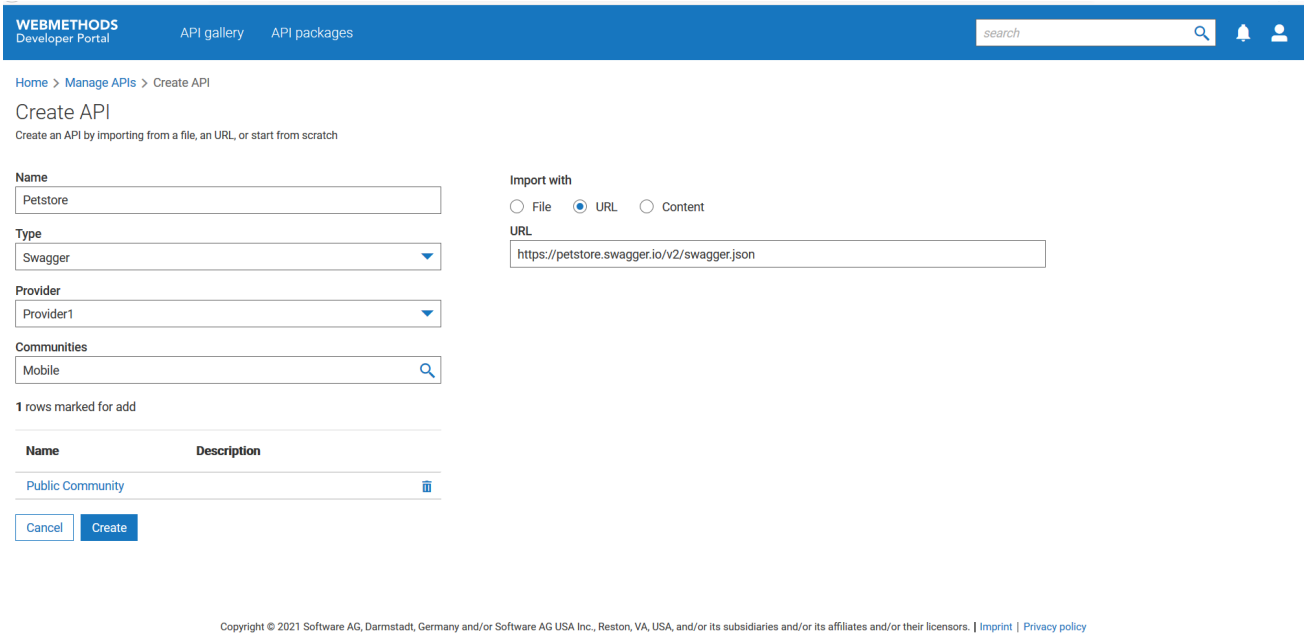
Before you begin:

Ensure that you have the file, URL, or the required content for creating the API.

In this example, you create an API *Petstore* using the `https://petstore.swagger.io/v2/swagger.json` URL. The API is assigned to the provider, *Provider1* and the community, *Mobile_app_developer*.

To create an API

1. Click the menu options icon from the title bar and click **Manage assets**.
2. Click **APIs**.
3. Click **Create API**.
4. Provide *Petstore* in the **Name** field.
5. Select *Swagger* from the **Type** list.
6. Select *Provider1* in the **Provider** field.
7. Select *Mobile_app_developer* from the **Community** field.
8. Select *URL* from the **Import with** section and provide `https://petstore.swagger.io/v2/swagger.json` in the field.



9. Click **Create**.

   The API is created. You can view the API from the **API gallery** page and the **Manage APIs** page.

Alternative steps:

1. In *Step 5*, select any of the following API types from the **Type** list:

- **Open API** . To create a REST API using an Open API specification.
- **RAML**. To create a REST API using a RAML specification.
- **WSDL**. To create a SOAP API using a WSDL specification.

2. In *Step 6*, select the required provider.

   If you are creating the API for a third-party API gateway, ensure that the gateway is registered for the value to be available in the **Provider** field.

3. In *Step 7*, when you select a private community remove the **Public community** which is selected by default. This is to ensure that the API is available only for the selected private community members.

4. In *Step 8*, provide the following inputs to create an API from the **Import with** section:

   - **File**. To create API from a file. Click **Browse** and select the required file.
   - **Content**. To create API from the given content. Provide the required parser content using which the API has to be created. Ensure that the content does not have references to external files.

Next steps:

- If the API lifecyle feature is enabled, then the newly created API is kept in the *Draft* state. For information about the API lifecycle, see "Lifecycle Management of APIs and Packages" on page 224.
- Move the API to the Live state to make it available for consumers. For information on moving APIs to the *Live* state, see "How do I change the state of an API to expose it to consumers?" on page 230.
- Configure rate limit for your APIs. For details steps, see "Configuring rate limit for APIs" on page 191.

How do I edit the basic attributes of an API?

Basic attributes of APIs include the API name, description, type, providers and communities associated with an API, and the API source.

This use case starts when you want to edit the basic attributes of an API and ends when you saved your changes.

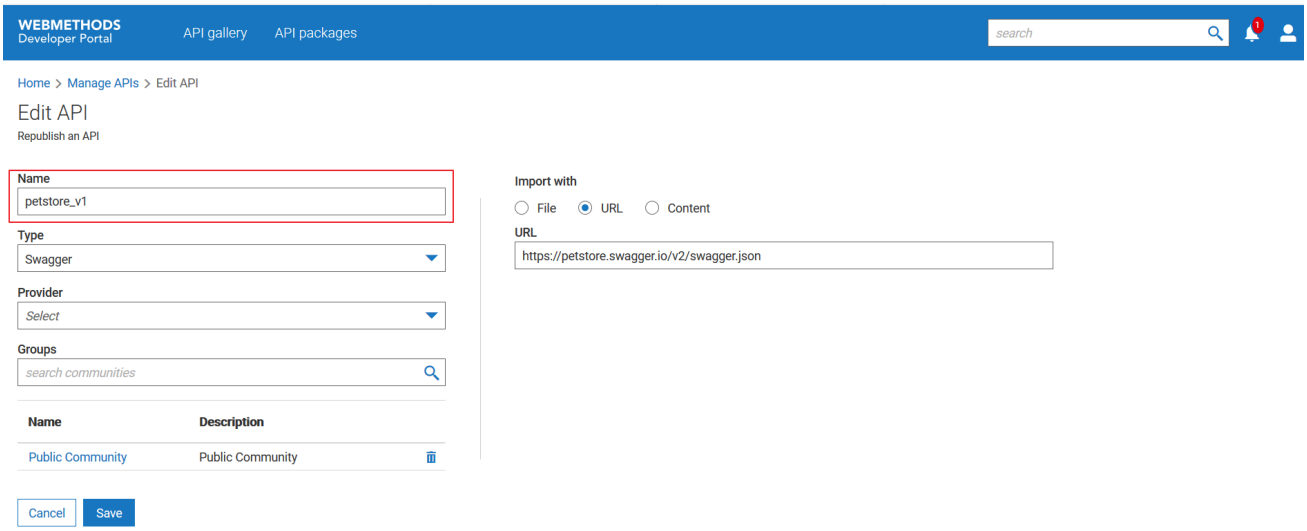
In this example, consider renaming the *Pet_v1* API as *Petstore_v1*.

To edit the basic details of an API

1. Click the menu options icon  from the title bar and click **Manage assets**.
2. Click **APIs**.

   The list of APIs appears.

3. Click the edit icon next to the *Pet_v1* API.



4. Click **Save**.

   Your changes are saved. The API is removed from the public community. So, only the users who are a part of the *Mobile_app_developer* community can view or try the API.

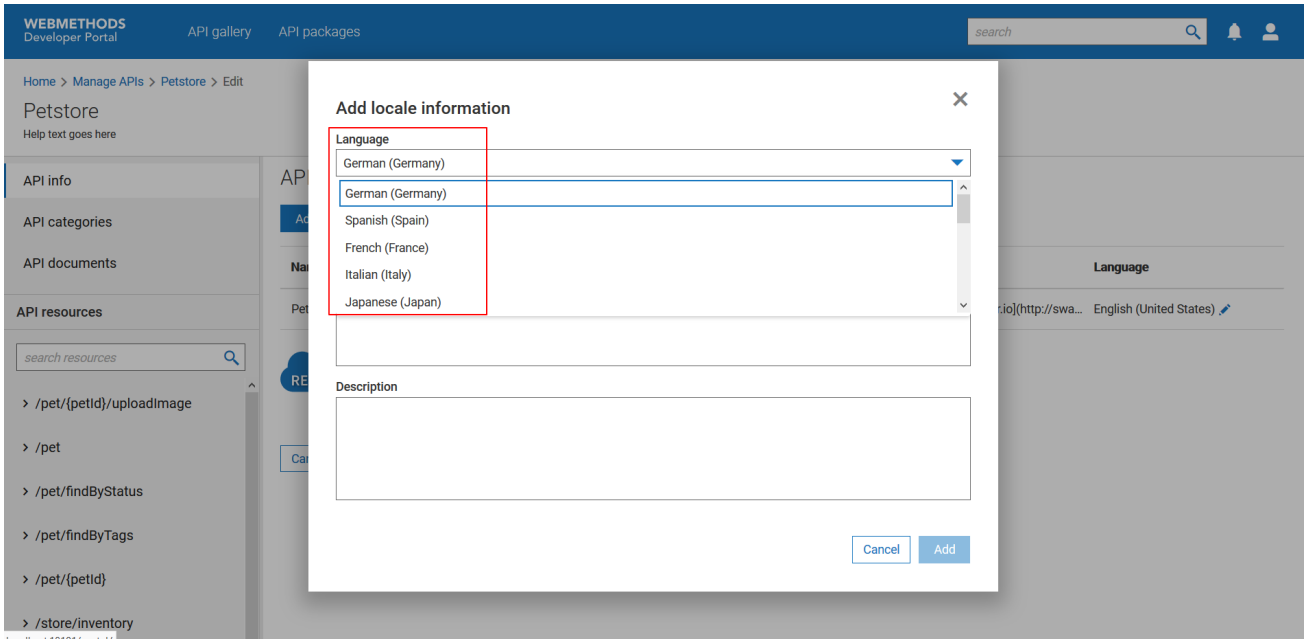
How do I edit the advanced attributes of an API?

Advanced attributes of APIs include the API logo or icon, supporting documents, categories, and summary and description of the API resources.

Developer Portal supports providing of Markdown text and text in the supported languages for the following API attributes:

- API summary and description
- API categories such as tags, business terms, API grouping, and maturity status
- API documents
- Resource summary and description
- Method summary and description

Note:
For security reasons, it is recommended that you add authentic and valid Markdown content.



Sample Markdown text:

This use case starts when you want to edit the advanced attributes of an API and ends when you saved your changes.

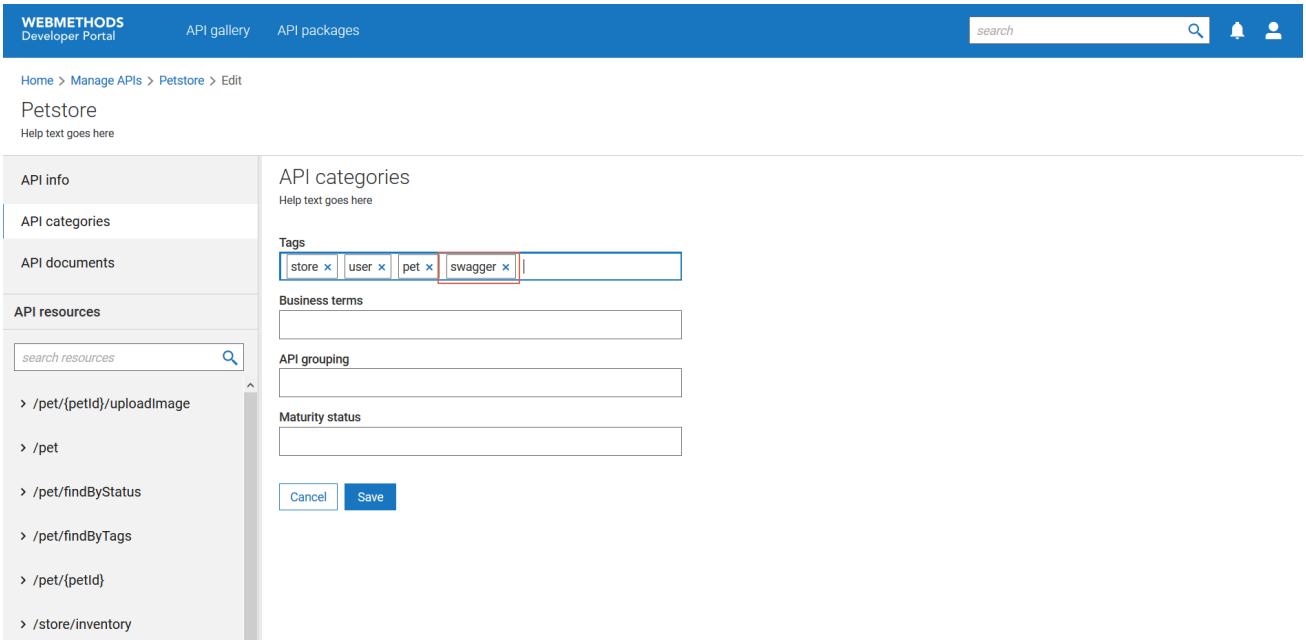
In this example, consider adding API tags under API categories of the API, *Petstore*.

To edit the advanced details of an API

1. Click the menu options icon from the title bar and click **Manage assets**.
2. Click **APIs**.

   The list of APIs appears.

3. Click the enrich icon next to the API that you want to edit.
4. Select **API categories** from the left pane and provide the new tag, *Swagger* and press Enter.



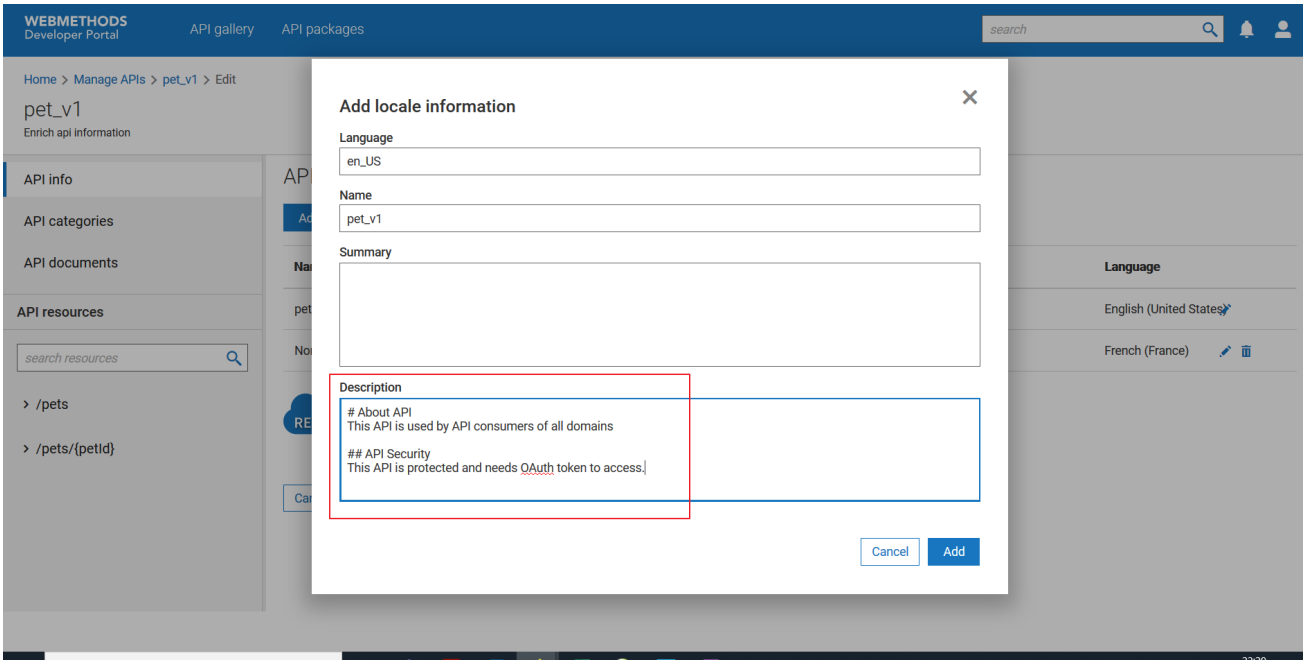
5. Click **Save**.

   The tag is added to the API. Users can filter APIs based on tags in the **API gallery** page.

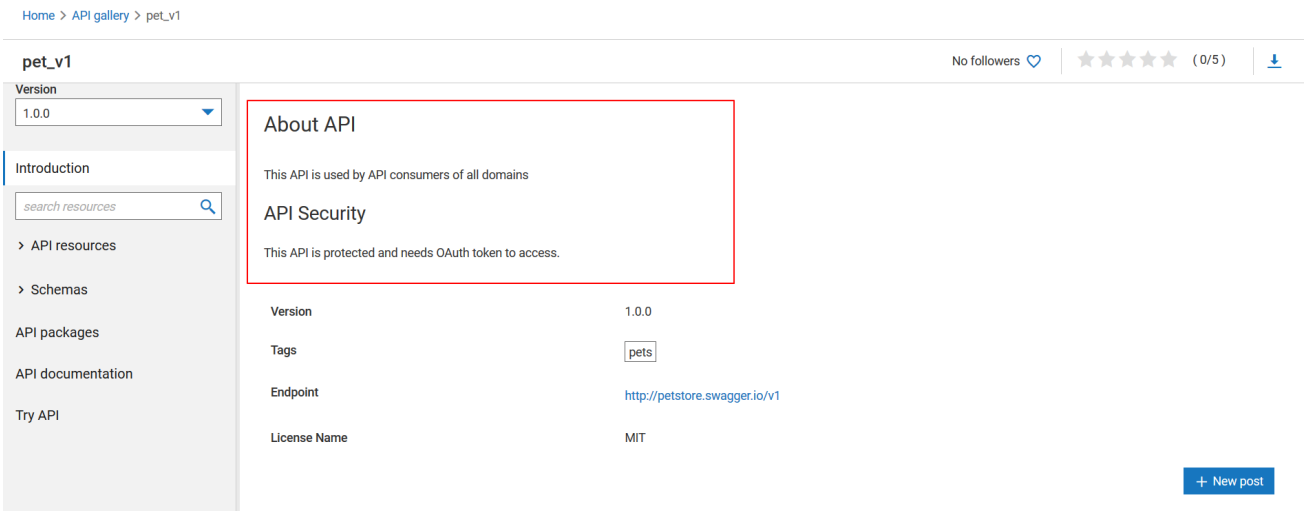
Alternative flow

- You can provide or modify the following details of the API:
  - API info
  - API summary and description
  - Resource summary and description
  - Method summary and description
- **Sample Markdown text**:

  In this example, information is given under two different levels of headings.



Then, the output looks like this.



For information on Markdown text, see https://www.markdownguide.org/extended-syntax/.

How do I create a new version of an API?

When you create a new version of an API, the **API gallery** page displays the latest version of the API. You can view all versions of the from the **Manage APIs** page. The API details page has a drop-down list that displays all API versions.

You can create new versions of an API for the use of a different set of consumers or with different security policies. New data can also be updated to the existing meta-data when you create new versions of your APIs.

The new API has the same metadata but with an updated version. The version can either be a number or a string.

Different versions of an API can be added to different communities and can be associated with different packages.

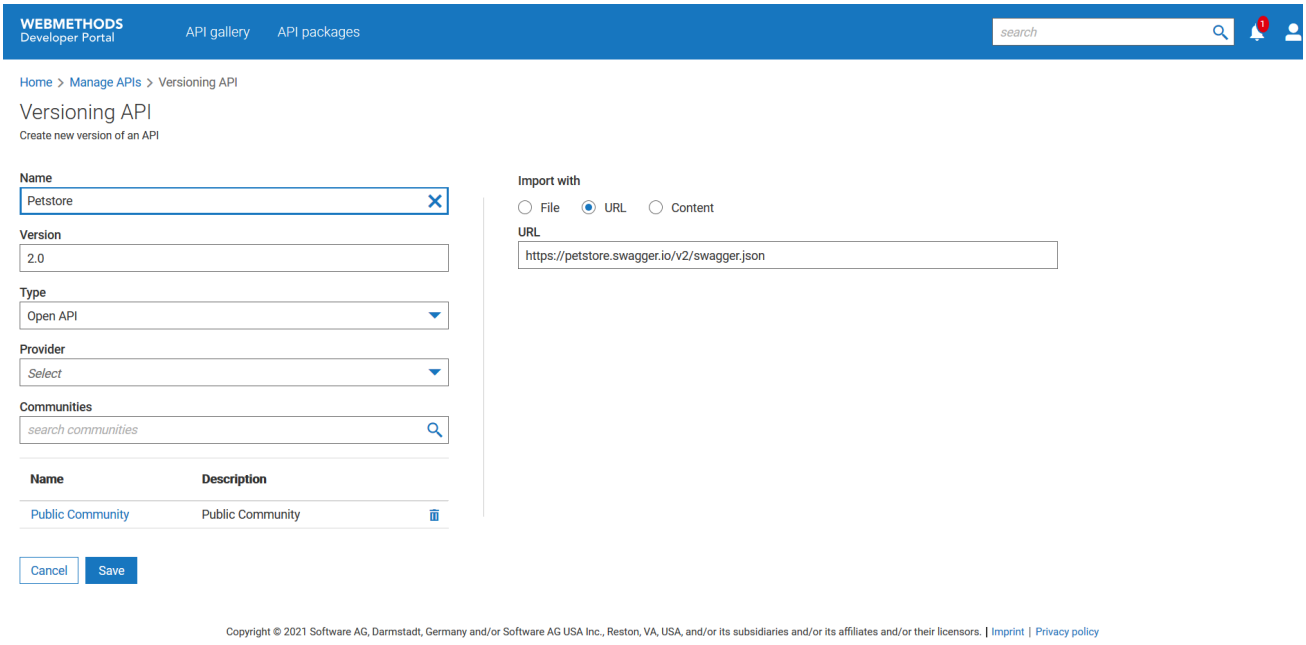
This use case starts when you want to create a new version of an existing API and ends when you created a new version of an API.

In this example, consider creating a new version of the API, *Petstore 2.0* and map the newer version to the *Public* community.

To create a new version of an API

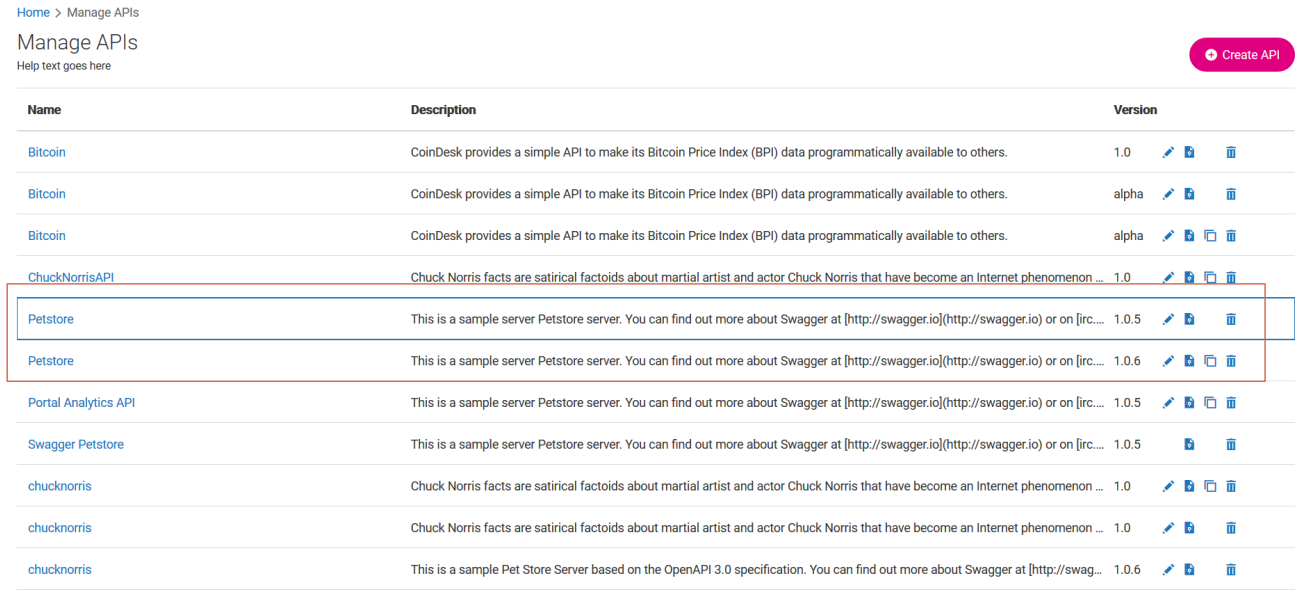
1. Click the menu options icon  from the title bar and click **Manage assets**.
2. Click **APIs**.

   The list of APIs appears.
3. Click the version icon  next to the API, *Petstore*.
4. Provide *2.0* in the **Version** field.
5. Select *Swagger* from the **Type** list.
6. Select *Provider1* in the **Provider** field.
7. Select *Public* from the **Community** field.
8. Select *URL* from the **Import with** section and provide `https://petstore.swagger.io/v2/swagger.json` in the field.



9. Click **Save**.

A new version of the API, *Petstore* is created.



Note:
When you version an API, you can create versions from the newer version of the API and not from the older version.

In the example, note that the older API does not have the version icon appearing next to it.

Alternative steps:

1. You can update the API by uploading a new file or by providing a new URL or pasting new content.
2. You can also map the API to another provider.

Next steps:

- All consumers can view the newer verison of the API as it is a part of the public community.
- If the API lifecyle feature is enabled, then the newly created version is kept in the *Draft* state. For information about the API lifecycle, see "Lifecycle Management of APIs and Packages" on page 224.
- Move the API to the Live state to make it available for consumers. For information on moving APIs to the *Live* state, see "How do I change the state of an API to expose it to consumers?" on page 230.

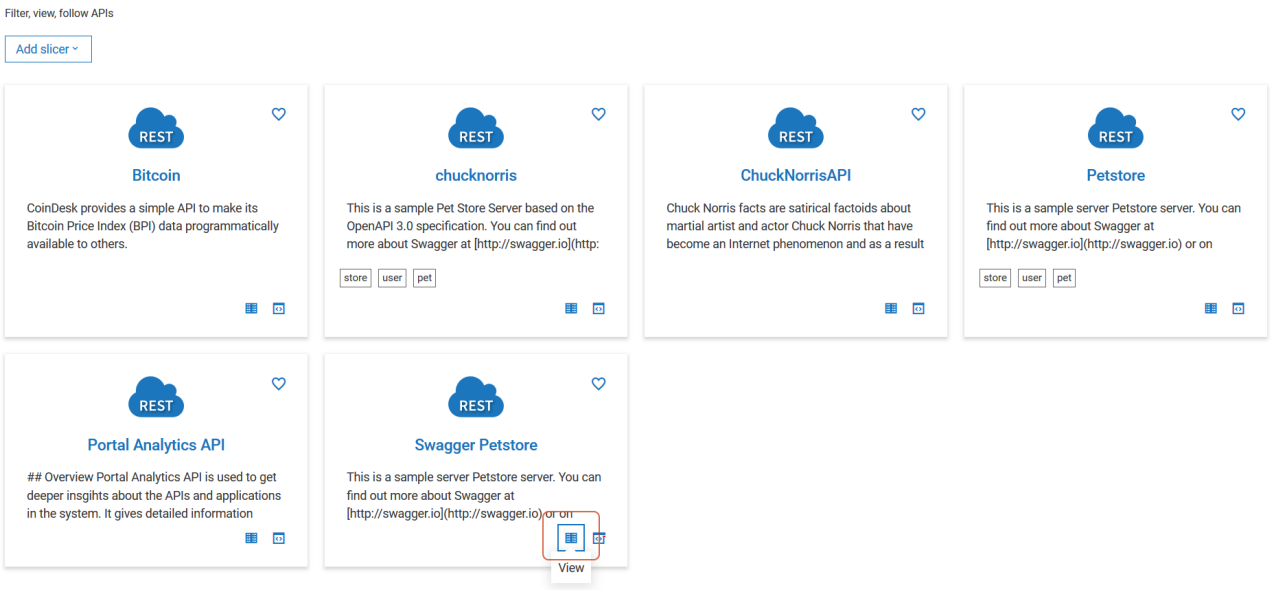
Viewing Code Snippets of REST APIs

You can view the sample code snippets for the required REST API resources.

When you want to invoke any endpoint from a REST API, you can view the sample code to invoke the endpoint in a language of your choice.

To view client code snippets

1. From the **API gallery** page, click the view icon  next to the required API.

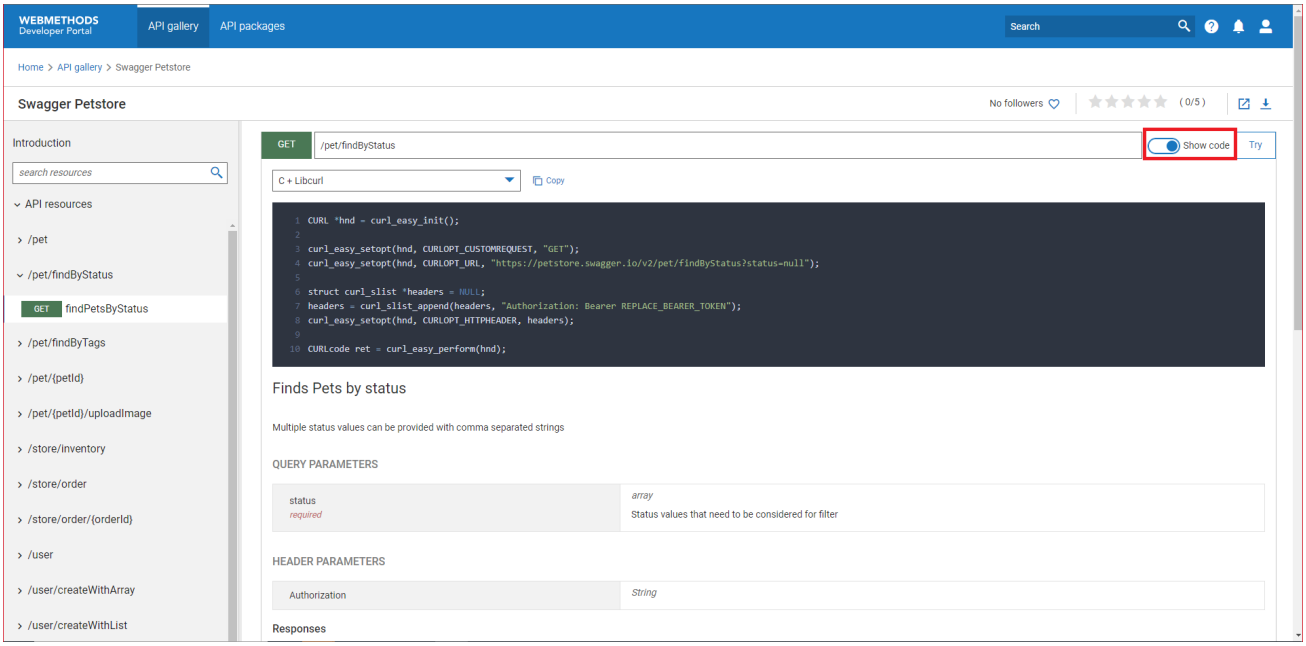


   The API details screen appears.

2. Select the required API resource.

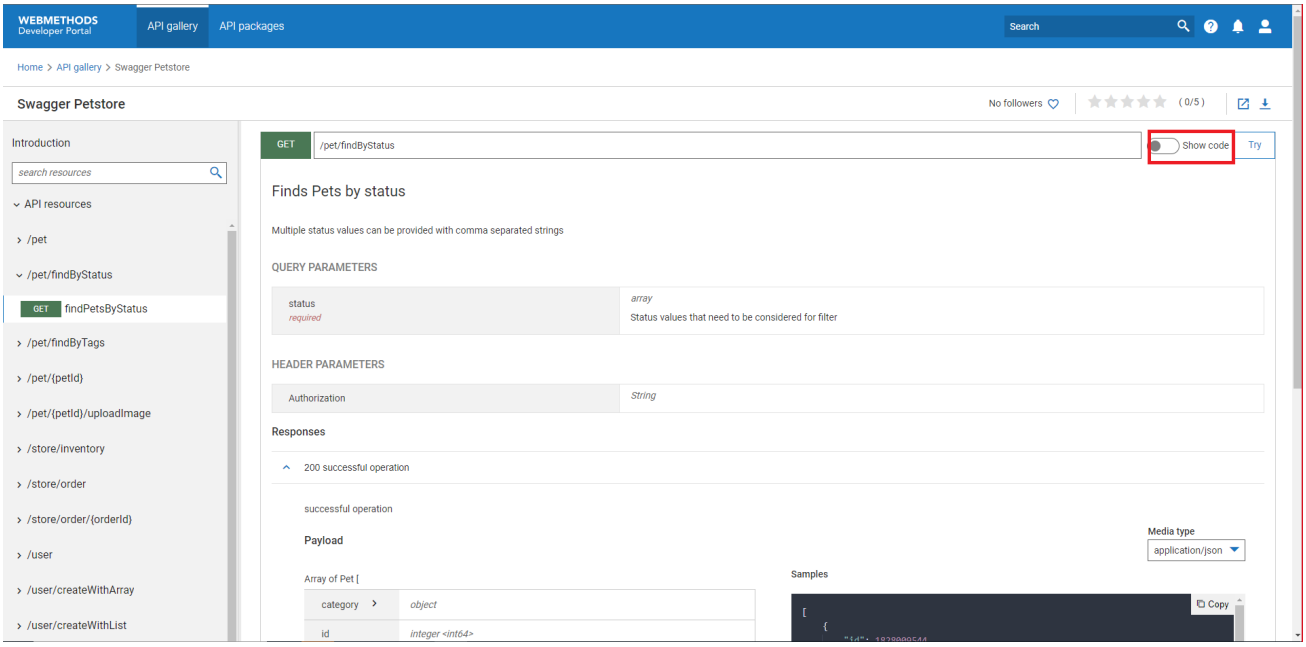
3. Turn the **Show code** slider on to view the sample code for the selected resource.

   You can also select the required language for the code snippet.



4. [*Optional*] Click **Copy** to copy the sample code.

5. Turn the **Show code** slider off to hide the sample code.

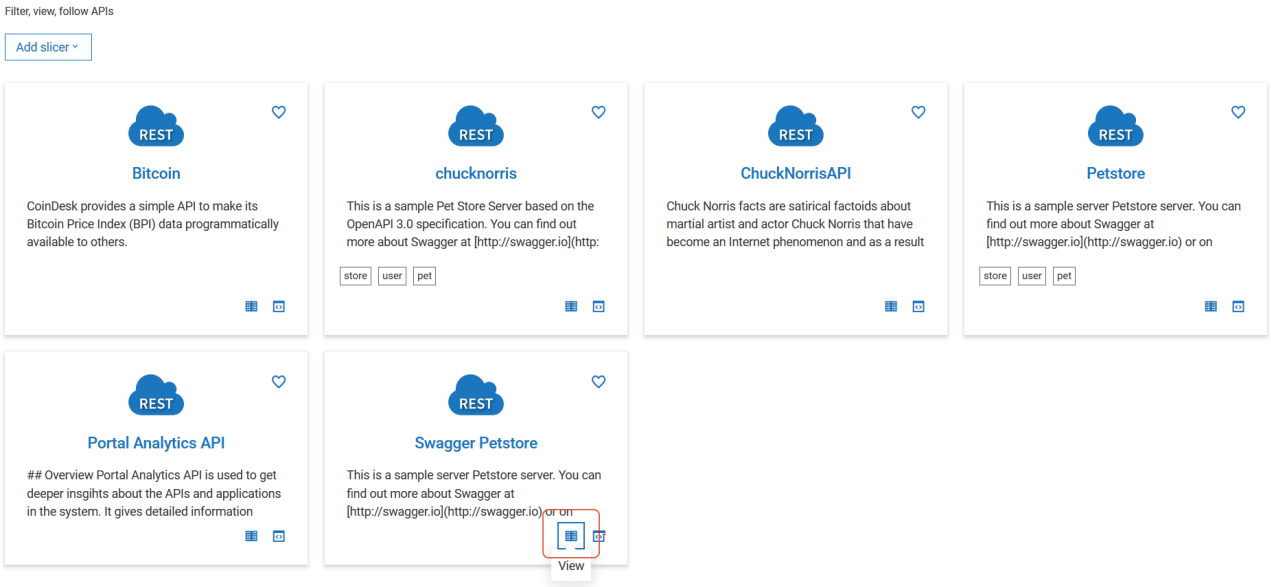


Downloading Client Software Development Kit (SDK) for REST API resources

As an API consumer, after you have selected the required REST API, you would require the software development kit (SDK) of the API to consume the API resources to build your custom applications(web or mobile). You can download client SDK for the required REST API.

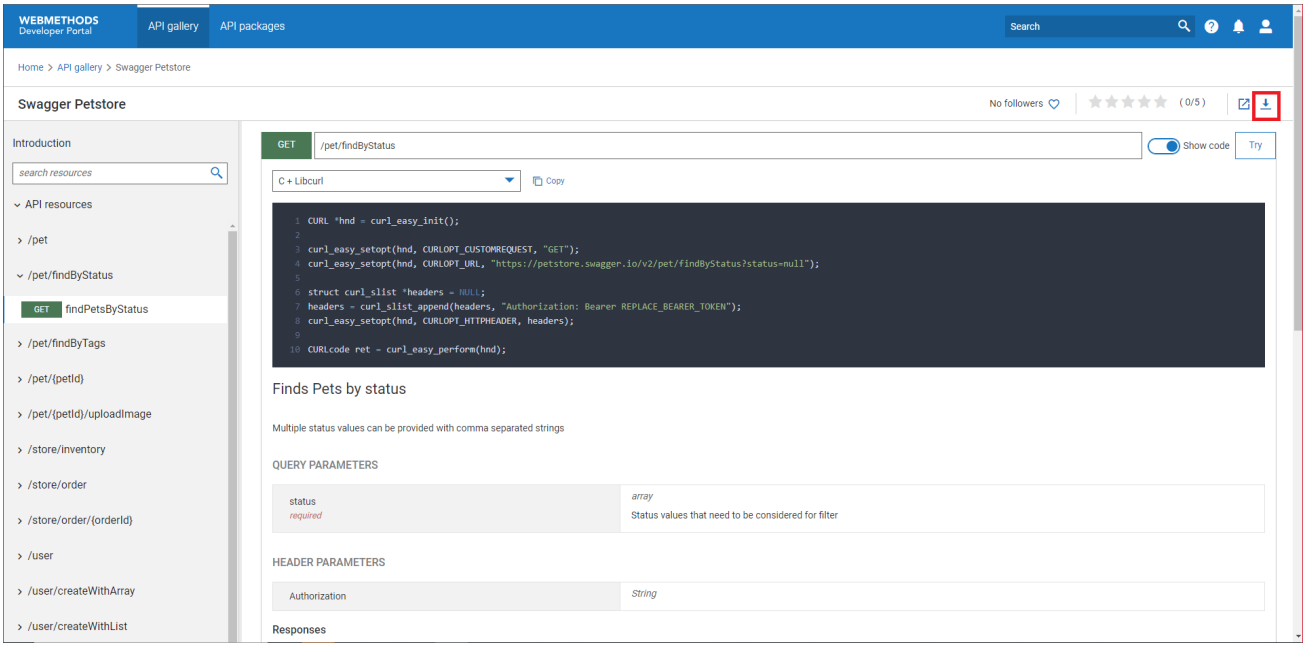
To download client SDK

1. From the **API gallery** page, click the view icon next to the required API.

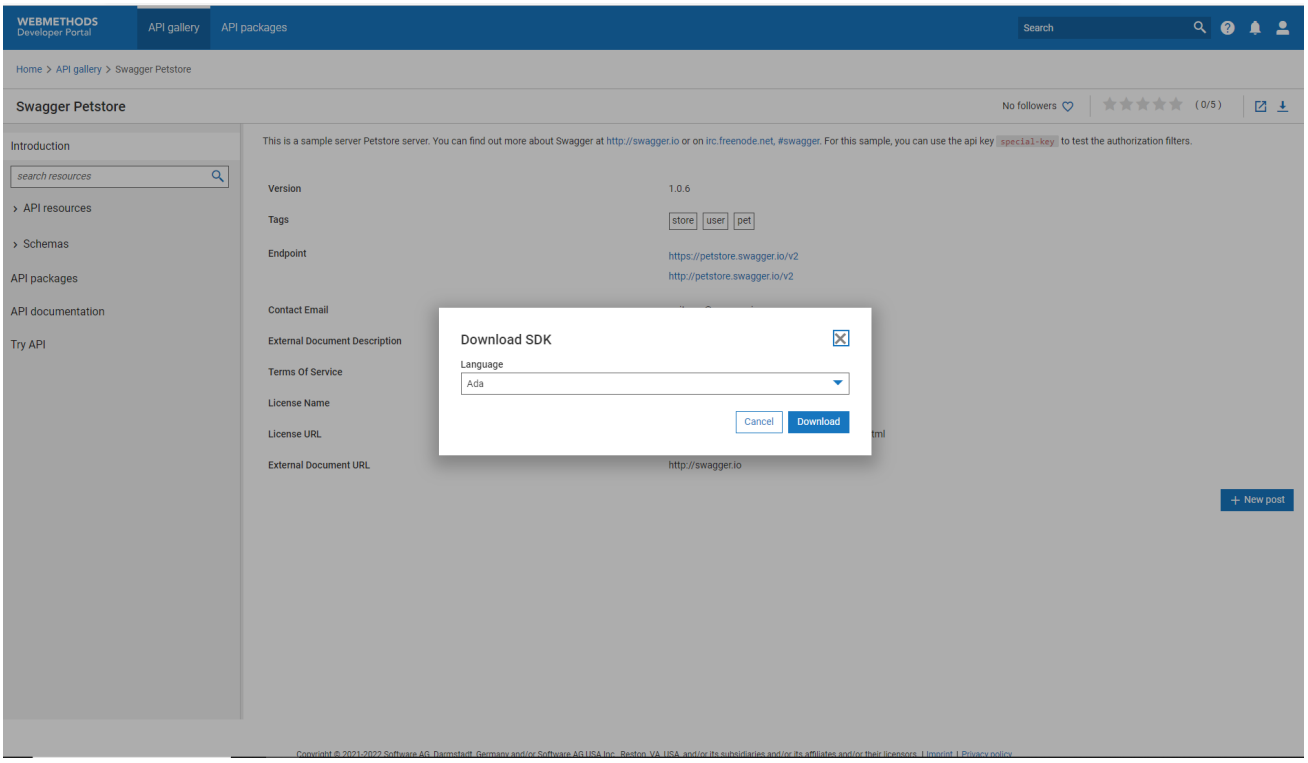


   The API details screen appears.

2. Click the download SDK icon to download the client SDK for the API.



3. Select the language that you require for your development kit.



4. Click **Download**.

   The SDK is downloaded as zip file. The downloaded file includes the dependencies of the HTTP client for the selected language.

Next steps:

- Unzip the downloaded SDK from the default download location of your browser to view the SDK file contents.

Configuring rate limit for APIs

Rate limit is the number of API calls a user can make during a given time period. Once a rate limit is reached, the API will return an error code until the specified rate limit time frame resets. Rate limit helps to avoid overloading the backend service and their infrastructure and to limit the specific clients in terms of resource usage and so on.

Developer Portal allows you to specify rate limit to the list of required APIs.

To configure the rate limit for APIs

1. Navigate to the location, `SAGInstallDir/DeveloperPortal/configuration`.
2. Open the **dpo_wrapper.conf** file to edit the same.
3. Add the following to file:

```
wrapper.java.additional.2011=-Dportal.server.config.enable-rate-limiting=
wrapper.java.additional.2012=-Dportal.server.config.rate-per-minute=
wrapper.java.additional.2013=-Dportal.server.config.rate-limited-apis=
```

4. Provide the required values in:
   - **Dportal.server.config.enable-rate-limiting**. Provide `true` to enable rate limit for APIs.
   - **Dportal.server.config.rate-per-minute**. Provide the number of invocations that you want to allow per minute for the given APIs.
   - **Dportal.server.config.rate-limited-apis**. Provide the list of APIs, separated by commas, that you want to configure rate limit.

   Sample values:

```
wrapper.java.additional.2011=-Dportal.server.config.enable-rate-limiting=true
wrapper.java.additional.2012=-Dportal.server.config.rate-per-minute=3
wrapper.java.additional.2013=-Dportal.server.config.rate-limited-apis=/rest/v1/signup,/rest/v1/forgotpassword
```

5. Save the changes.

Next steps:

- You can edit the **dpo_wrapper.conf** file to include more APIs or modify the specified the number of invocations, when required.

Trying APIs

Developer Portal allows you to try an API with required parameters.

You can try the required resource of an API by providing corresponding parameter values. You must also provide sufficient authorization details to have access to the API.

As an API consumer, it is important to test the endpoints of an API before putting them to use. Hence, Software AG recommends that you always test an API before consuming.

Trying a REST API

This section explains how you can test a REST API.

Pre-requisites

- If the API that you are trying is protected with API key, JWT, or OAuth, then create an application to establish your identification. For information about creating applications, see "Creating an application" on page 200.

To try a REST API

1. Navigate to the Try API page using one of the following ways:
   - Click the Tryout icon  in the corresponding API tile.
   - Click **Try API** from the API details page of the corresponding API.

   The Try API page appears. The resources that you can test appear in the left pane.
2. Select a resource and the required method.
3. Provide the following values in the **Parameters** tab:
   - **Key**. Key value of the parameter.
   - **Value**. Value of the corresponding key parameter.

   You can click **Add new** and add multiple entries.
4. Provide the following required values in the **Headers** tab:
   - **Key**. Value of the header key.
   - **Value**. Value of the corresponding key parameter.

     You can click **Add new** and add multiple entries.
5. Provide the authorization details from the **Authorization** tab.

   For information about the authorization types and required inputs, see "Authorization fields" on page 196.
6. Select the required request form from the **Request body** tab and provide your request:
   - **form-data**. To send your request in the form of the key-value pairs. You can select **Text** and provide your form key and value or select **File** and attach the request file.

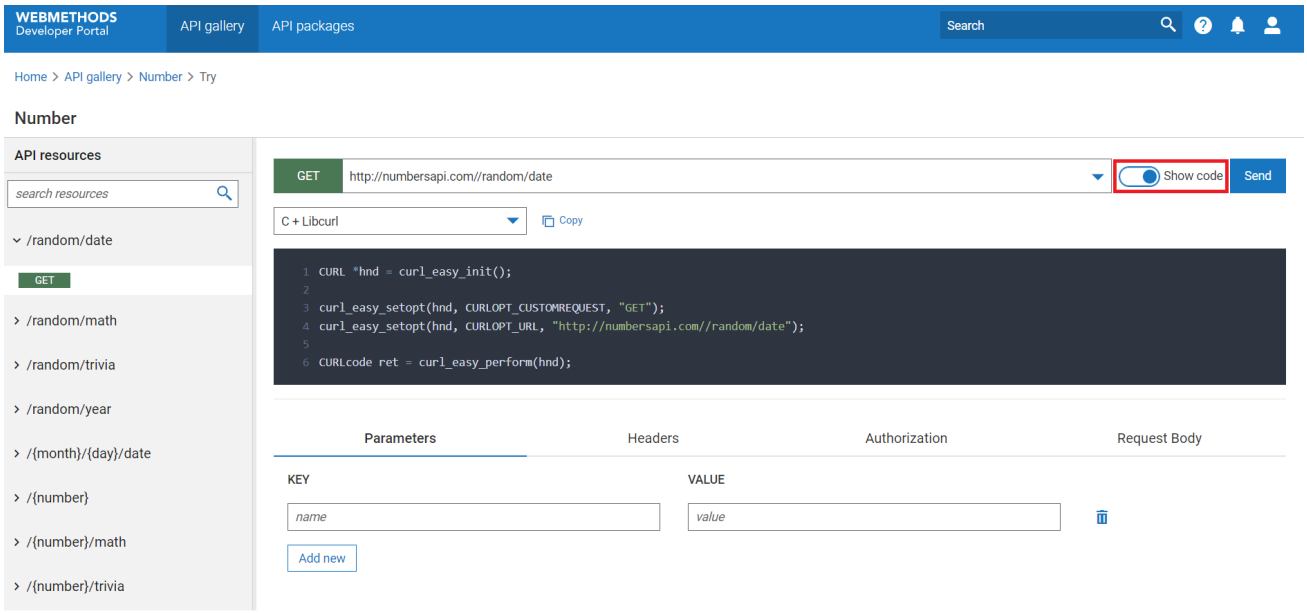
- **x-www-form-encoded**. To send your request in the form of the encoded key-value pairs.
- **raw**. To provide any request. You can provide the request in JSON or XML format.
- **None**. To send the test request without a request body.

7. Click **Send**.

   The response body and headers appear in their respective sections.

Alternative steps

After you perform *Step 2*, you can turn the **Show code** slider on to view the sample code for the selected resource.



You can also download client SDK for the REST API you have selected. For information about downloading client SDK, see "Downloading Client Software Development Kit (SDK) for REST API resources" on page 189.

Trying a SOAP API

This section explains how you can test a SOAP API.

Pre-requisites

- If the API that you are trying is protected with API key, JWT, or OAuth, then create an application to establish your identification. For information about creating applications, see "Creating an application" on page 200.

To try a SOAP API

1. Navigate to the Try API page using one of the following ways:

- Click the Tryout icon  in the corresponding API tile.
- Click **Try API** from the API details page of the corresponding API.

The Try API page appears. The SOAP methods that you can test appear in the left pane.

2. Select the required method.
3. Provide the following values in the **Parameters** tab:
   - **Key**. Key value of the parameter.
   - **Value**. Value of the corresponding key parameter.

   You can click **Add new** and add multiple entries.
4. Provide the required header details from the **Headers** tab:
   - To include header key value combinations, provide the following:
     - **Key**. Value of the header key.
     - **Value**. Value of the corresponding key parameter.

       You can click **Add new** and add multiple entries.
   - To include a SOAP header:
     - **Name**. A unique name to distinguish the provided SOAP headers, if there are multiple headers.
     - Click **Add new namespace URI** to provide unique **Key** elements and corresponding **Value** attributes for the header content. You can provide multiple key and value combinations by clicking **Add new namespace URI**.
     - **Header content**.

       You can click **Add SOAP header** to provide more headers.
5. Provide the authorization details from the **Authorization** tab.

   For information about the authorization types and required inputs, see "Authorization fields" on page 196.
6. If the SOAP API is protected with web service security (WSS) select the required authorization method from the **Web Service Security** tab:
   - **WSS user token**. Select this to provide WSS user name and password.
     - Provide the user name and password required for WSS authorization.
     - Select the **Add nonce** check box to include nonce, which is a randomly generated number that is included in the request header sent to the API server. Nonce is generated every time a request is sent to the API server and thus, adding a nonce to the request along with WSS user token prevents reuse of old communications.

- Select the **Add creation time** check box to add a timestamp that the nonce was created. This value helps server to ensure that the outdated nonce values are not reused.
- Select **Password text** to send password as plain text. Else, select **Password digest** to encrypt the password.

- **WSS SAML assertion**. Select this to include SAML assertion for WSS authorization. You can either paste the assertion or upload the assertion file.

  You can include both, WSS user token and SAML assertion, in your request. You can select either of these options, provide the required details, and then the other option without pressing **Clear**. So, the details provided for both options are included to the request.

7. In the **Request body** tab, provide your request in the form of key-value pairs or raw text.
8. Click **Send**.

   The response body and headers appear in their respective sections.

Trying a OData API

This section explains how you can test a OData API.

Pre-requisites

- If the API that you are trying is protected with API key, JWT, or OAuth, then create an application to establish your identification. For information about creating applications, see "Creating an application" on page 200.

To try a OData API

1. Navigate to the Try API page using one of the following ways:

   - Click the Tryout icon  in the corresponding API tile.
   - Click **Try API** from the API details page of the corresponding API.

   The Try API page appears. The resources that you can test appear in the left pane.
2. Expand the required resource and click the required method.
3. In the **Query builder** tab, select the required query parameters and click **Add** .

   The specified parameters are included as a query in the request URL.
4. Provide the following values in the **Parameters** tab:

   - **Key**. Key value of the parameter.
   - **Value**. Value of the corresponding key parameter.

   You can click **Add new** and add multiple entries.

5. Provide the following required values in the **Headers** tab:
   - **Key**. Type a header key value.
   - **Value**. Type the corresponding value of the key parameter.

   You can click **Add new** and add multiple entries.
6. Provide the authorization details from the **Authorization** tab.

   For information about the authorization types and required inputs, see "Authorization fields" on page 196.
7. Select the required request form from the **Request body** tab and provide your request:
   - **form-data**. To send your request in the form of the key-value pairs. You can select **Text** and provide your form key and value or select **File** and attach the request file.
   - **x-www-form-encoded**. To send your request in the form of the encoded key-value pairs.
   - **raw**. To provide any request. You can provide the request in JSON or XML format.
   - **None**. To send the test request without a request body.
8. Click **Send**.

   The response body and headers appear in their respective sections.

Authorization fields

This section lists the steps to provide authorization details, for testing APIs, based on the authorization type applied to the corresponding APIs:

Authorization type	Procedure
For APIs protected with API key	■ Select the required application from the **Applications** field. ■ Select *API Key* from the **Authorization type** field. The test request is sent to the API server with the API key of the selected application.
For APIs that require basic authentication	■ Select *Basic auth* from the **Authorization type** field. ■ Provide the user credentials and click **Update**. The test request is sent with the selected authorization details.
For APIs protected with JWT	■ Select the required application from the **Applications** field. ■ Select *JWT* from the **Authorization type** field.

Authorization type	Procedure
	The JWT tokens available for the selected application appear in the **Available tokens** section. You can select a token from the list or request for a new one. ■ Click + to get a new access token. The **Create new token** page appears. ■ Provide the credentials associated with the JWT authorization and click **Get token**. The token appears in the **Available token** section. Validity of token's are based on the duration specified in API Gateway. ■ Select the created token. The test request is sent to the API server with the selected token details.
For APIs protected with OAuth	■ Select the required application from the **Applications** field. ■ Select *OAuth 2.0* from the **Authorization type** field. The tokens available for the selected application appear in the **Available tokens** section. You can select a token from the list or request for a new one. ■ Click + to get a new access token. The **Create new token** page appears. ■ Provide a **Token name**. ■ Select the required **Grant type** and the required scope. ■ Click **Get token**. If you had selected the *Authorization code* or *Implicit* grant type, the **webMethods API Gateway Resource access approval** page appears. ■ Select the required scopes and click **Approve**. ■ Provide your API Gateway credentials and click **Sign in**. You will be redirected to Developer Portal Try API page. The token appears in the **Available token** section. Validity of token's are based on the duration specified in API Gateway. ■ Select the created token. If you had selected *Client credentials* grant type, then the request token appears without any further action. ■ Select the created token.

Authorization type	Procedure
	The test request is sent to the API server with the selected token details.
For APIs that do not require any authentication	■ Select *None* from the **Authorization type** field. The test request is sent to the API server without any authorization.

Troubleshooting Tips: Testing APIs published from API Gateway

Error message appears when you test APIs

When the HTTP ports of an API Gateway instance are configured with a self-signed certificate, and if you test an API published from that instance, then the following error message appears.

```
-1 PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target; nested exception is
javax.net.ssl.SSLHandshakeException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid
certification path to requested target
```

Resolution:

Run the following command to add the self-signed certificated used in API Gateway to the Developer Portal truststore.in the following location: `SAGInstallDir/jvm/jvm/lib/security/cacerts`:

```
keytool -import -trustcacerts -alias alias_name -file certificate_path -keystore
cacerts
```

Sample command

```
keytool -import -trustcacerts -alias mdecert -file C:\apigw\mdeCert.cer -keystore
cacerts
```

Note:
If SSL exception message appear even after you import the self-signed certificate to the mentioned path, then ensure that the **disable-host-name-verifier** setting in the application-dev.yml file is set to *true*. This setting allows you to turn off the host name verification during Developer Portal communication with other systems.

Applications

When you invoke a protected API from Developer Portal, you need to create an application requesting access to the API from the provider. Through the application, you can request access tokens from the provider by revealing a corresponding authentication such as username and password, OAuth, or JWT details.

During runtime, the provider recognizes the consumer's identity through the application. The details passed from Developer Portal to the provider enables them to:

- Control access to an API at run time (that is, allow only authorized applications to invoke an API.
- Monitor an API for violations of a Service-Level Agreement (SLA) for a specified application.
- Indicate the application to which a logged transaction event belongs.

You can create, view, and share applications from the **Manage applications** section.

Developer Portal sends the application requests from consumers to providers using the callback URL of providers. Configuring an incorrect callback URL can lead to communication issues between the provider and Developer Portal.

If an application onboarding strategy is configured, then the applications that you create will be processed based on the configured strategy. For information on configuring an application onboarding strategy, see "How do I configure onboarding strategy to process application or subscription requests?" on page 199.

If you do not configure an onboarding strategy, then the registration requests are automatically approved.

You can view the status of requested applications from the **Manage applications** section. Once an application is approved, you can share the application with other users or user groups to allow them use the application and invoke the corresponding APIs.

From the **Trace application** section of an application, you can view the stages that the application has passed through and their corresponding timestamp. These stages include application creation request,

You can trace the status and stages of an application using the **Trace application** section of an application. This section displays the various stages including application creation request, application request submit, application publish, application scope increase or decrease, and so on with the corresponding time stamp.

How do I configure onboarding strategy to process application or subscription requests?

This use case starts when you want to configure onboarding process to approve or reject the application or subscription registration requests.

Before you begin:

Ensure that you have:

- Configure an approval workflow. For information on configuring an approval workflow, see "How do I configure an approval workflow to process an internal approval onboarding strategy?" on page 121.
- **API Administrator** privilege.

To configure onboarding strategy to process application or subscription requests

1. Click the menu options icon  from the title bar and click **Administration**.
2. Select **Onboarding**.
3. From the **Application/ subscription onboarding** section, enable any or all of the required strategies:
   - **Internal approval**. Select the required approval workflow from the **Select a flow** window that appears when you enable this strategy.
   - **External approval**. Select this if you want to process the requests using an external approval system. You can notify the required external approving system by creating a webhook. For information on configuring user sign-up notifications to your external approving system, see "How do I configure webhooks to notify user sign-up and application requests to an external approval system?" on page 27.
4. Use the arrow keys next to these strategies to change their order. The strategies are followed by the order they appear.
5. Click **Save**.

   The onboarding strategy to process application or subscription requests configuration is saved.

Next steps:

- Application and package subscription requests are processed based on the onboarding strategy.

Creating an application

You must create applications to invoke protected APIs.

You can create applications for an API from one of the following:

- **Manage applications** screen
- **API details** screen
- **Try API** screen

You can create more than one application for an API.

How do I create an application from Manage applications screen?

This use case starts when you want to create an application and ends when you have created one.

In this example, consider creating an application *app1* to access the API, *Petstore* that is published by the provider, *Provider1*

Before you begin:

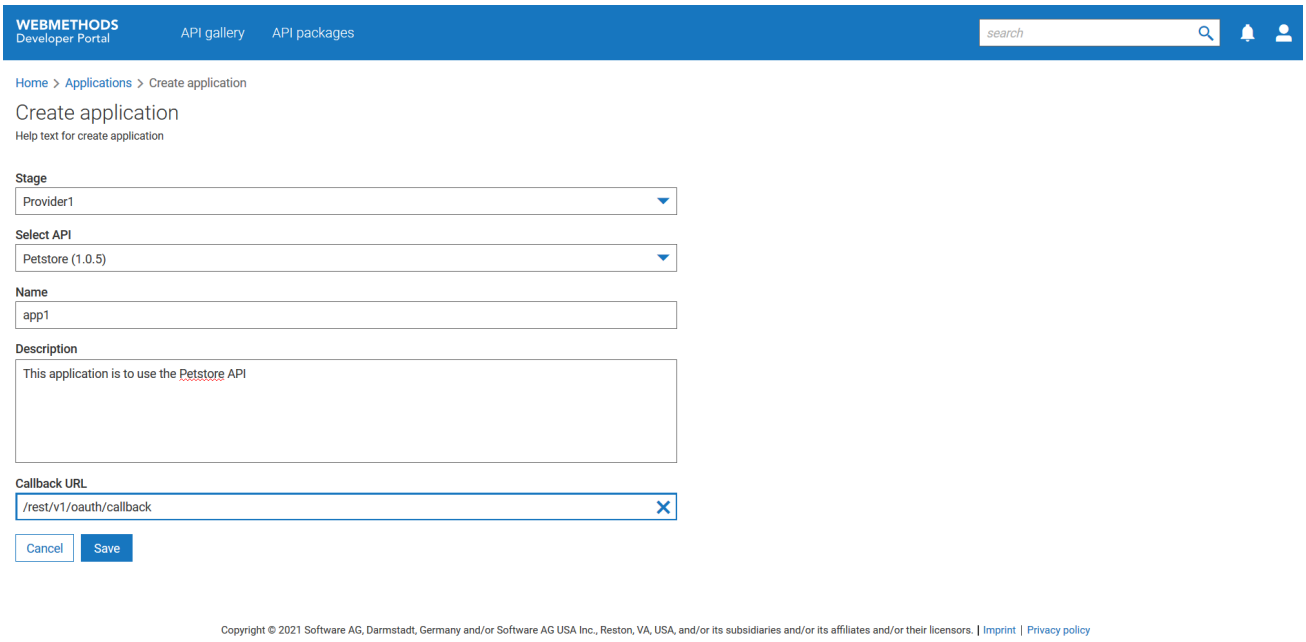
Ensure you have the **API Administrator** or the **API Provider** privilege.

To create an application from the Manage applications screen

1. Click the menu options icon from the title bar and click **Manage applications**.

   The list of applications appear.

2. Click **Create application**.
3. Select *Provider1* from the **Stage** list.
4. Select *Petstore* from the **Select API** field.
5. Provide app1 in the **Name** field.
6. Select the required callback URL of the provider.



7. Click **Save**.

   The application is created.

Next steps:

- Application is approved based on the configured application onboarding strategy. If there is no onboarding strategy configured to onboard applications, then the application is approved automatically.
- Use application to invoke the API.
- Perform the following to share an application:
  - Click the share icon next to the application you want to share.

- From the **Assign user/ team** screen, select the users or teams with whom you want to share the application.

  By default, this field displays only the users who belong to your communities. You can modify this restriction. For more information, see "Configuring visibility of users based on communities" on page 174.

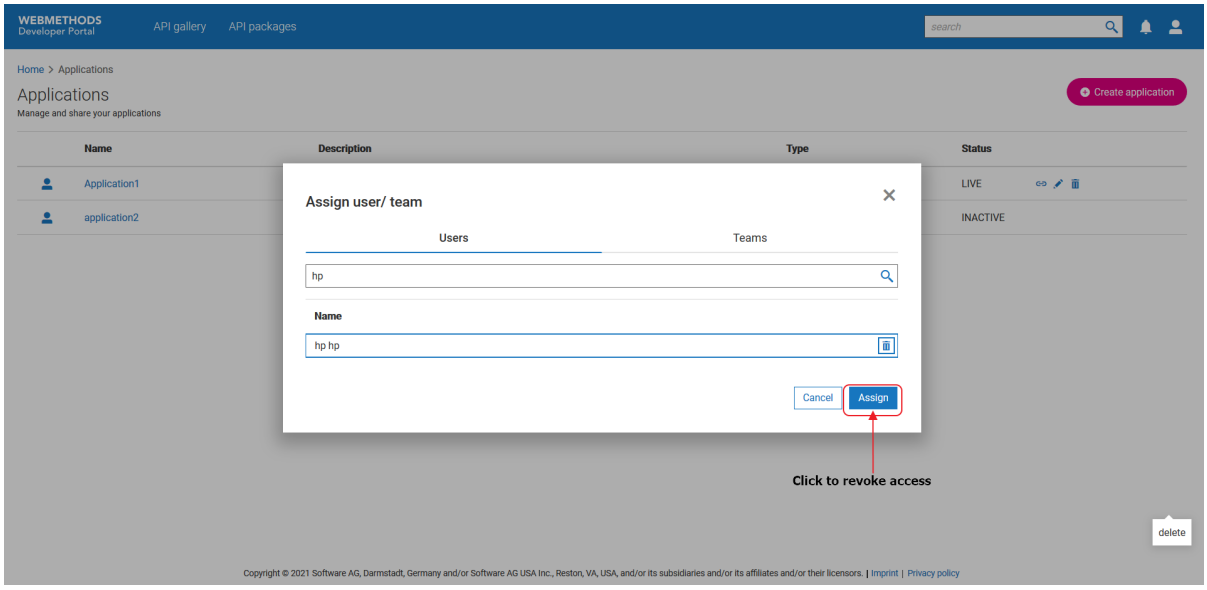
- Click **Assign**.

  The application is shared with the selected users and teams.

- Perform the following to revoke access to an application from specific users or teams:

  - Click the share icon next to the application you want to share.

  - From the **Assign user/ team** screen, click the delete icon next to the user or team from whom you want to revoke access.

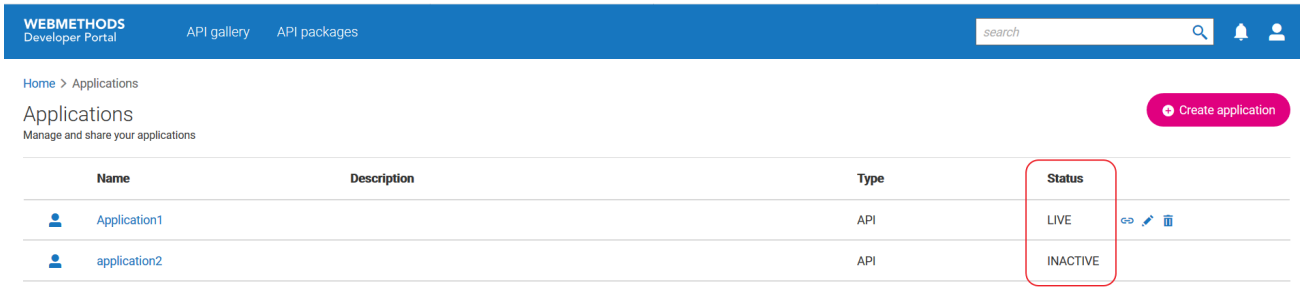


Note:
Only the users who created an application can share or delete the application. When an application is deleted, the users with whom the application was shared can no longer access the application.

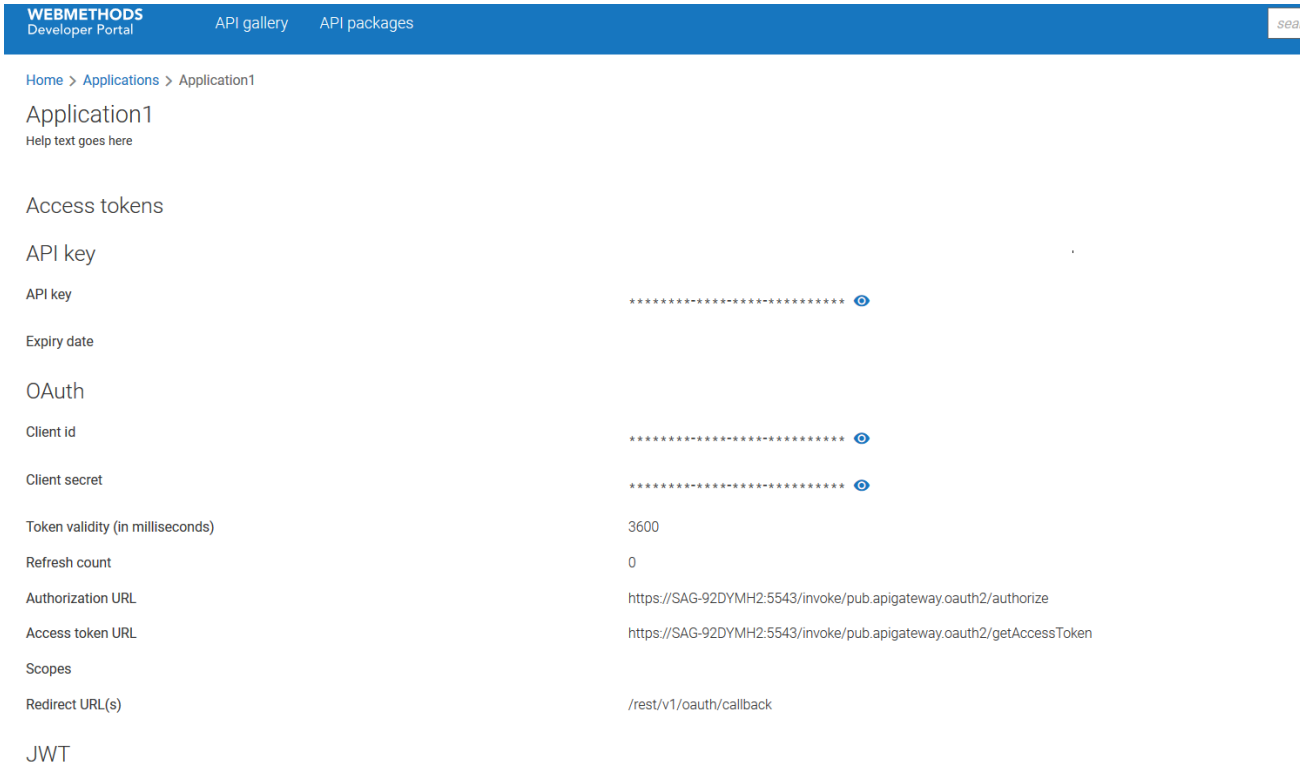
- Click **Assign**.

  The access to the application is revoked from the selected users and teams.

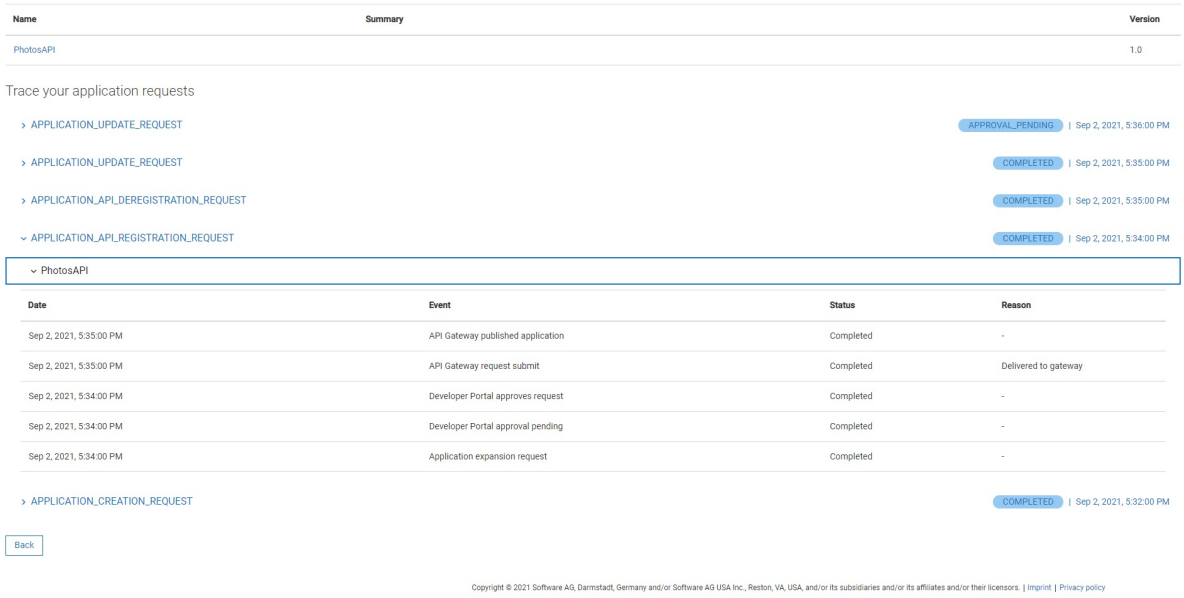
- When you create an application, the request is sent to the corresponding provider and approved based on the settings configured by provider. Applications in Developer Portal are approved based on the configured application onboarding strategy. If there is no onboarding strategy configured to onboard applications, then the application is approved automatically. After an application is approved, the application is ready to use. Check the **Status** field to ensure that an application is active.



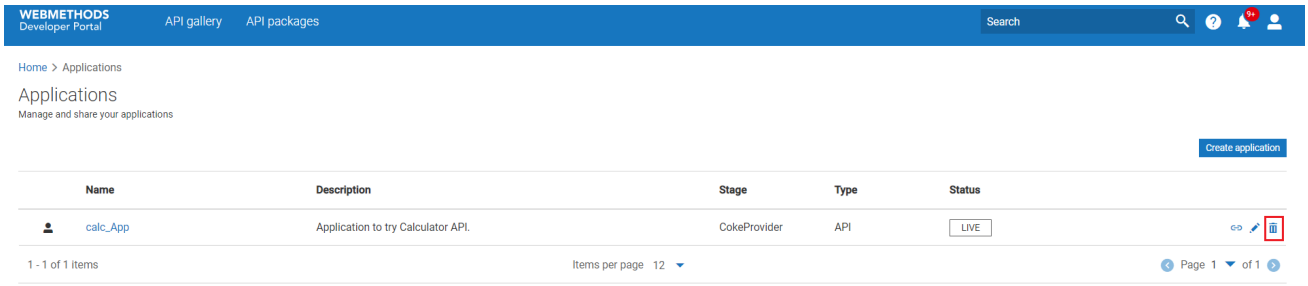
To view the details of an application, click the application name.



You can trace the application's stage from the **Trace your application requests** section of the page.



- You can delete applications that you do not use anymore.
  - From the **Manage applications** page, click the delete icon next to the application that you want to delete.



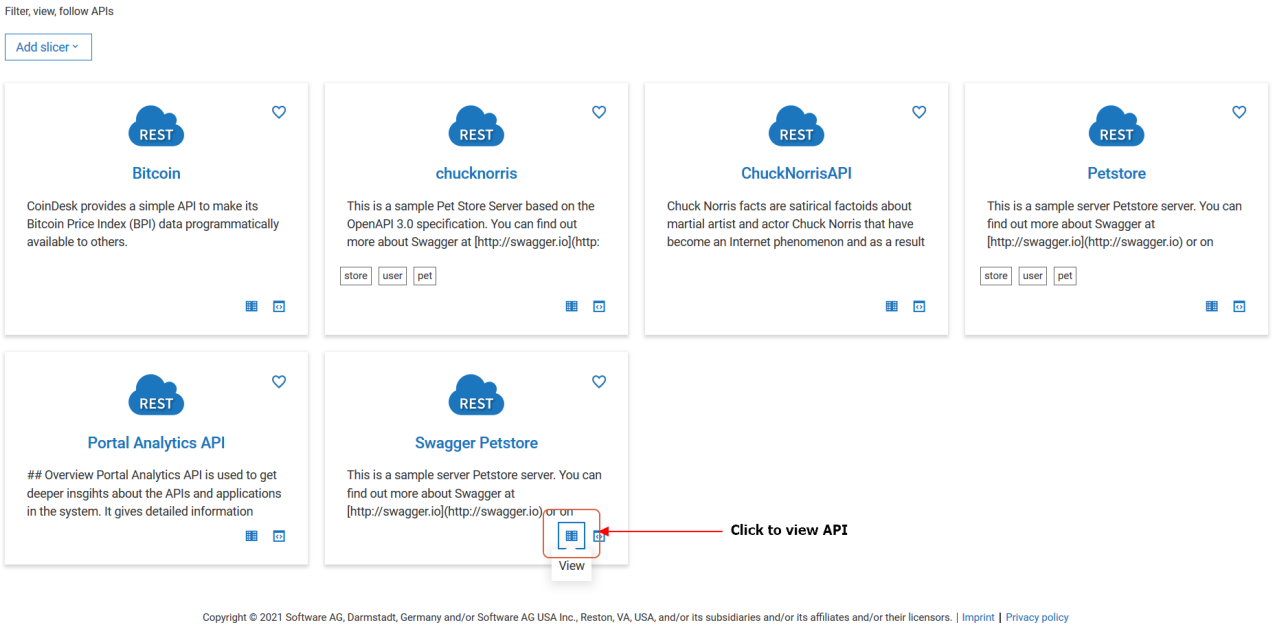
> **Note:**
> Only the users who created an application can share or delete the application. When an application is deleted, the users with whom the application was shared can no longer access the application.

How do I create an application from API details screen?

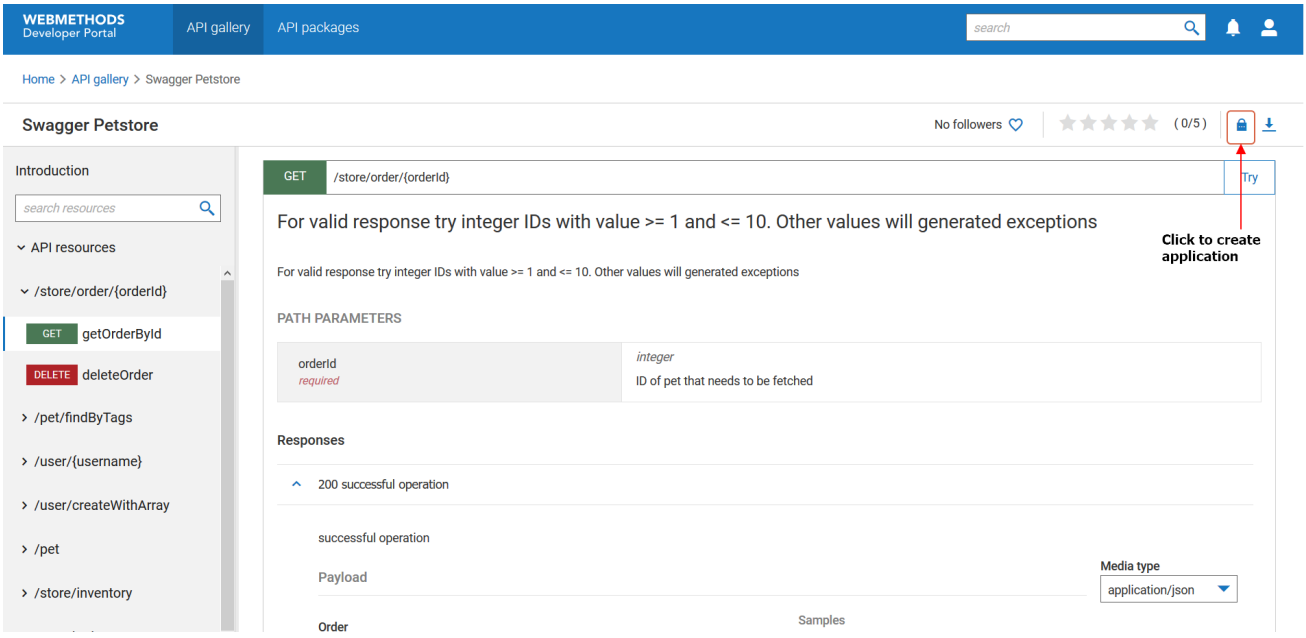
This use case starts when you want to create an application and ends when you have created one.

To create an application from the API details screen:

1. From the **API gallery** page, click the view icon next to the API for which you want to create an application.



2. In the **API details** page, click the request application icon .

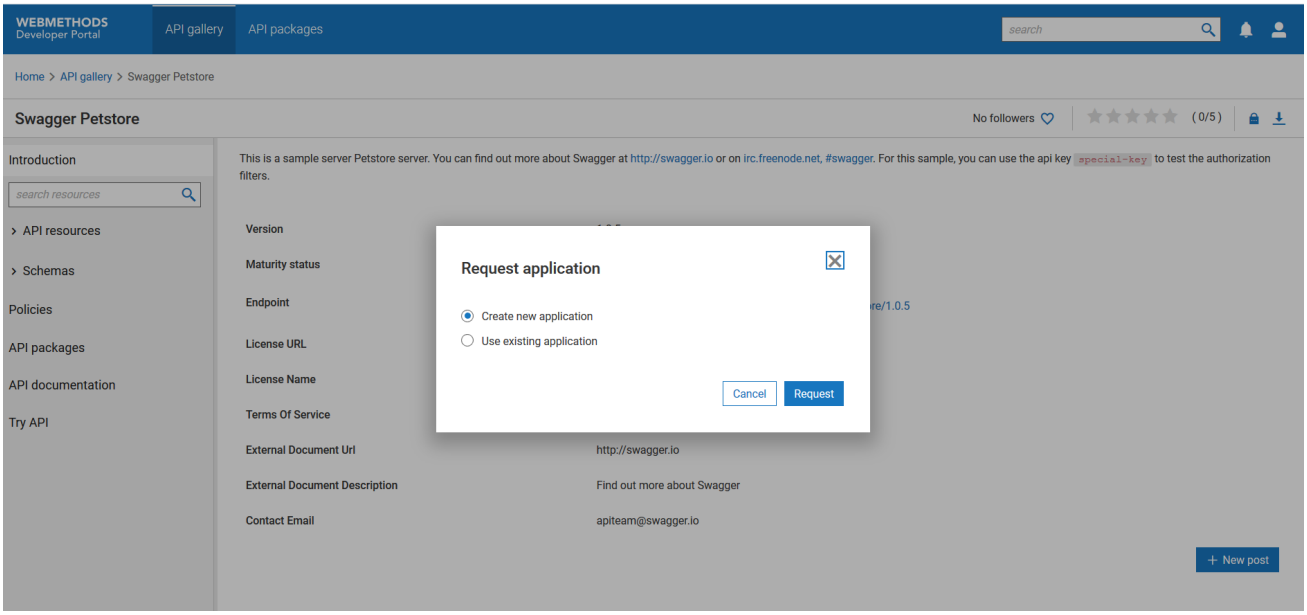


Note:

The request application icon appears only for the protected APIs.

3. In the **Request application** screen, click **Request**.

If there is an existing application for the selected API, then the following screen appears:



You can select an existing one or create a new one. The application stage and API name appear in the corresponding fields on the **Create application** page. If you select to use an existing application, the scope of the application increases and you can invoke the API using an existing application.

4. Provide the application name and description.
5. Click **Save**.

   The application is created.

Next steps:

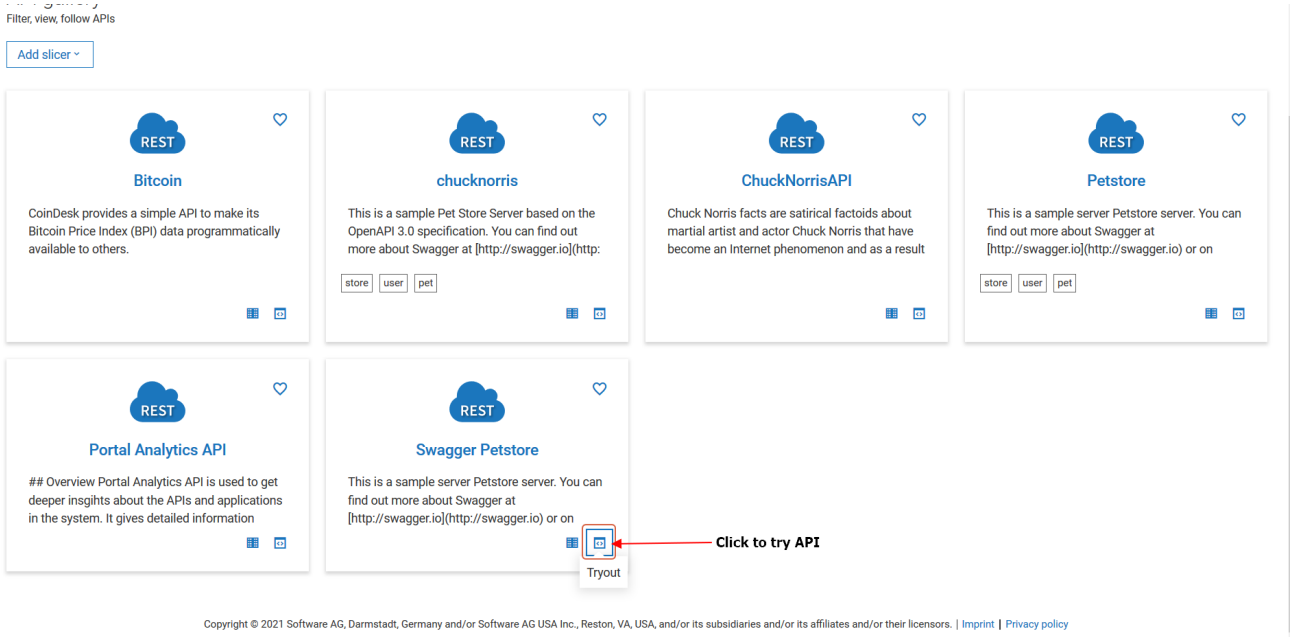
- Application is approved based on the configured application onboarding strategy. If there is no onboarding strategy configured to onboard applications, then the application is approved automatically.
- Use application to invoke the API.

How do I create an application from Try API screen?

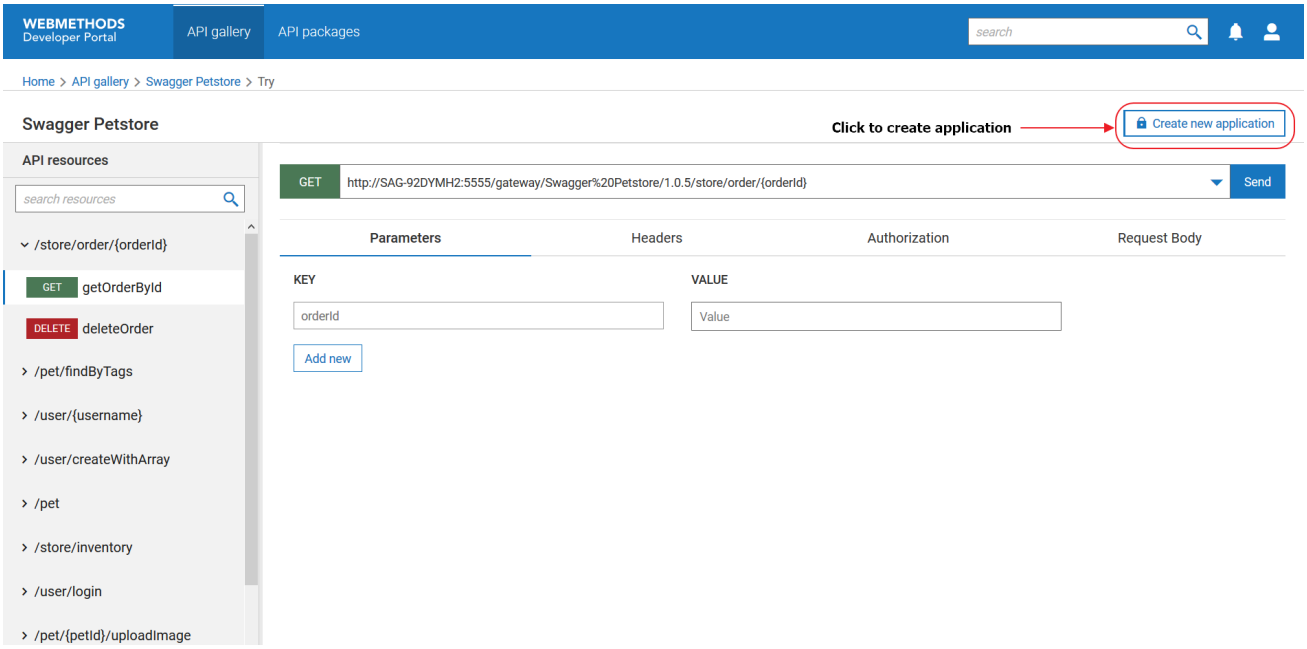
This use case starts when you want to create an application and ends when you have created one.

To create an application from the Try API screen

1. From the **API gallery** page, click the Try API icon  next to the API for which you want to create an application.



2. In the **Try API** page, click **Create new application**.



The application stage and API name appear in the corresponding fields on the **Create application** page.

3. Provide the application name and description.

4. Click **Save**.

The application is created.

Next steps:

- Application is approved based on the configured application onboarding strategy. If there is no onboarding strategy configured to onboard applications, then the application is approved automatically.
- Use application to invoke the API.

Approving Application or Package Subscription Requests

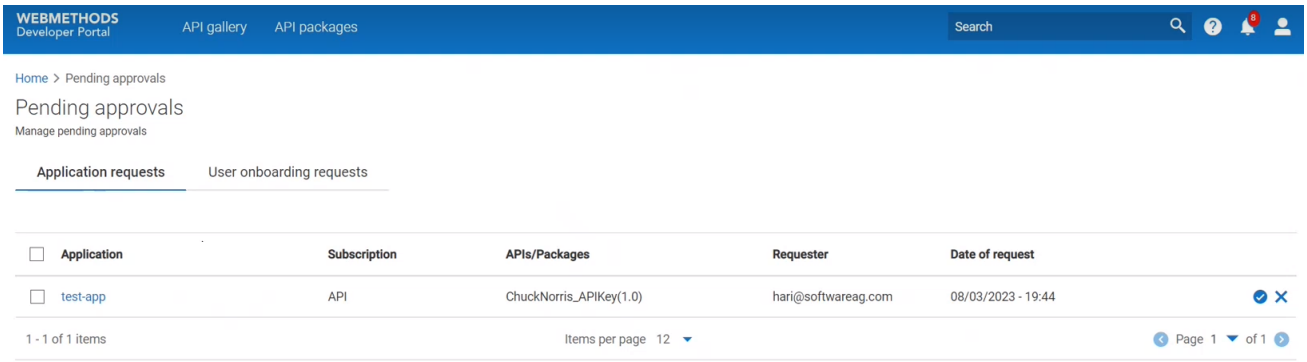
If you have configured an approval workflow for approving application and package subscription requests, then approvers can view the pending requests from the **Pending approval** screen, and approve or reject them.

For information about configuring application or package subscription approval workflow, see "How do I configure onboarding strategy to process application or subscription requests?" on page 199.

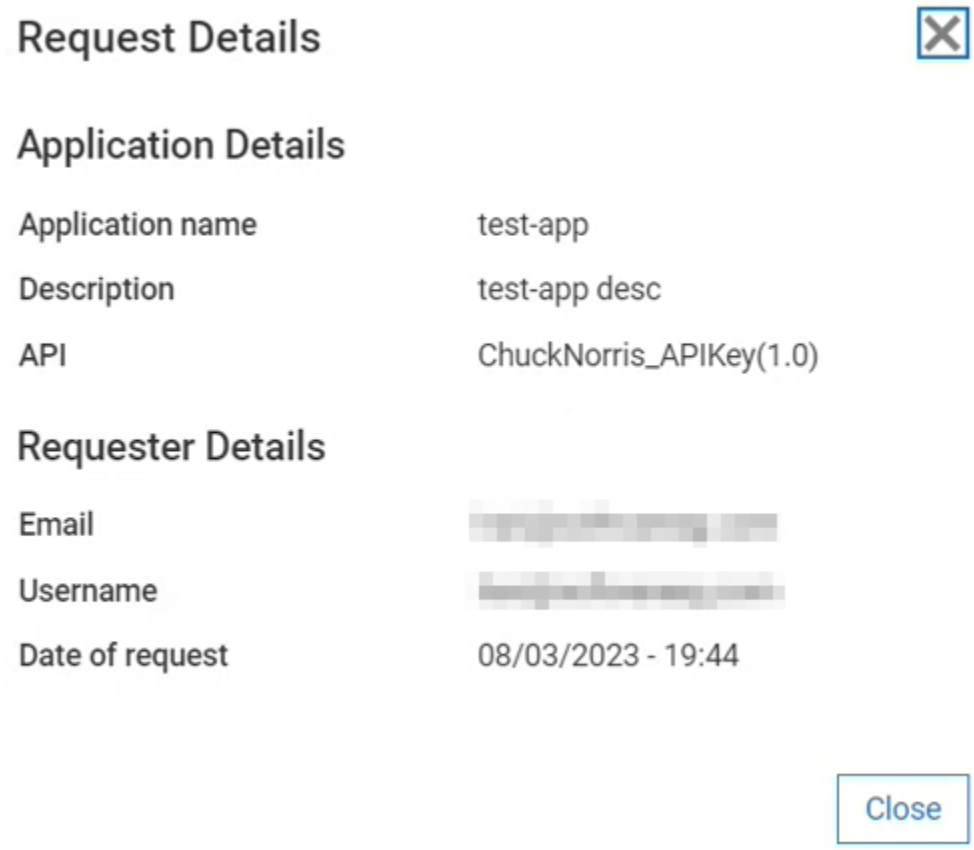
To approve a pending application or package subscription request

1. Click the menu options icon from the title bar and click **Pending approvals**.

   The list of application requests appears in the **Application requests** tab.

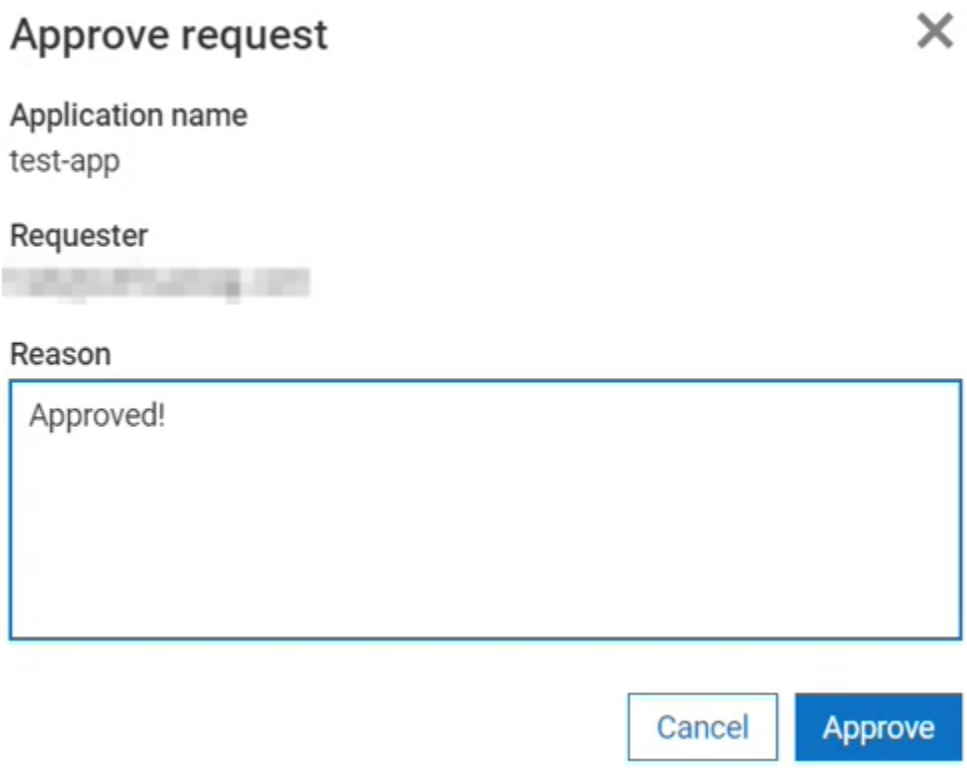


2. *Optional*. Click a request to view more details about the request.



3. Click next to the required request to approve the request.

   A confirmation screen appears.



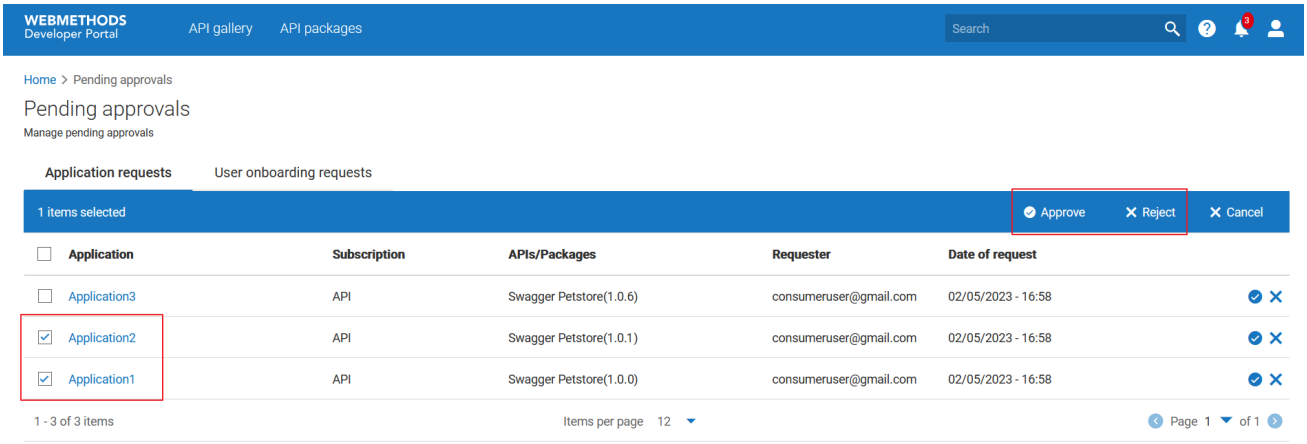
4. Provide your comments if any in the **Reason** field, and click **Approve**.

Alternative steps:

- In *Step 3*, you can click ✕ to reject the request. Provide your comments, if any and click **Reject**.

  When you reject a application onboarding request, package subscription, or an application scope increase request, a notification e-mail is sent to the requestor as per the **Approval result** e-mail notification template. If you have specified more than one approver, and if one of the approver rejects a request then the notification e-mail is sent to the requestor and the other approvers. For more information on the e-mail notification template, see "List of email notification templates available in Developer Portal" on page 21

- To approve or reject multiple requests listed on the **Pending approvals** screen, select the checkboxes next to the required requests and click **Approve** or **Reject**. The selected pending requests are approved or rejected at once.



- To approve or reject all requests, select the checkbox in the grid header and click **Approve** or **Reject**. All pending requests are approved or rejected at once.

Custom Asset Management

In addition to APIs and packages, Developer Portal allows you to publish assets of your choice. This empowers providers to publish end-to-end development reference to their consumers.

As a provider, you can publish any kind of reference for your consumers using this feature.

You can add required asset types and define their properties. You can also create a gallery for the custom assets and display on the UI. Like APIs and packages, your end-users can view the custom assets and collaborate over them.

Custom asset publishing involves the following steps:

- **Add asset type**. You have to first add the asset type, and configure its properties. You would have information that you require to describe the assets you publish to Developer Portal. You can specify the corresponding properties to provide information about each asset that you add to your custom gallery.
- **Add asset details**. After defining the asset type, you can add assets along with their corresponding details.
- **Add asset gallery to Developer Portal**. After you add the asset type, you can add the asset gallery to the UI for your consumers to view the custom assets.

The following use cases demonstrate the process of publishing custom assets using an example. These use cases consider *Microservices* as custom assets and list the steps of publishing them to Developer Portal.

How do I configure the asset types to publish custom assets to Developer Portal?

Before publishing custom assets to Developer Portal, you must configure the asset type. You can add an asset type and configure its properties from the **Manage asset types** page.

This use case starts when you want to add a custom asset type and ends when you add the require asset type and configure its properties.

In this example, consider adding the asset type, *Microservice* with the following properties:

- **Overview**. A rich text field that states the overview of the asset (Microservice). This is an optional field.
- **Microservice document**. An option to upload a file (Microservice document). This is an optional field.
- **Published On**. A date time field to provide the date of the asset publish. This is an optional field.
- **Author**. A text field that provides the asset's author name. This is a mandatory field.

To add the asset type with the specified details

1. Click the menu option icon  from the title bar and click **Administration**.
2. Click **Manage asset types**.
3. Click **Add type**.
4. Provide *Microservice* in the **Name** field.
5. Provide a description in the **Description** field.
6. Add the fields based on the information about custom assets that you want to publish.

   When you add an asset of the defined type, you must specify values for the configured properties.

   To add the **Overview** field:

   a. Click **Add properties**.

   b. Provide *Overview* in the **Name** field.

   c. Provide *Microservice Overview* in the **Display name** field.

   d. Select *Rich text* from the **Select property type** field.

e. Click **Add**.

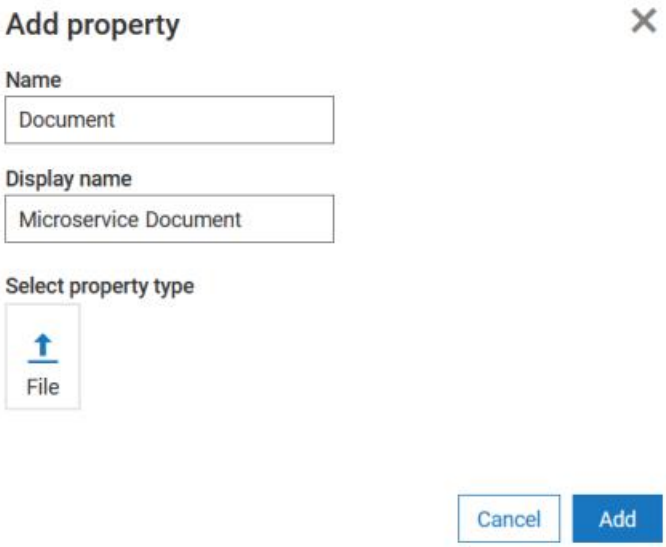
To add the **Microservice document** field:

a. Click **Add properties**.

b. Provide *Document* in the **Name** field.

c. Provide *Microservice Document* in the **Display name** field.

d. Select *File* from the **Select property type** field.



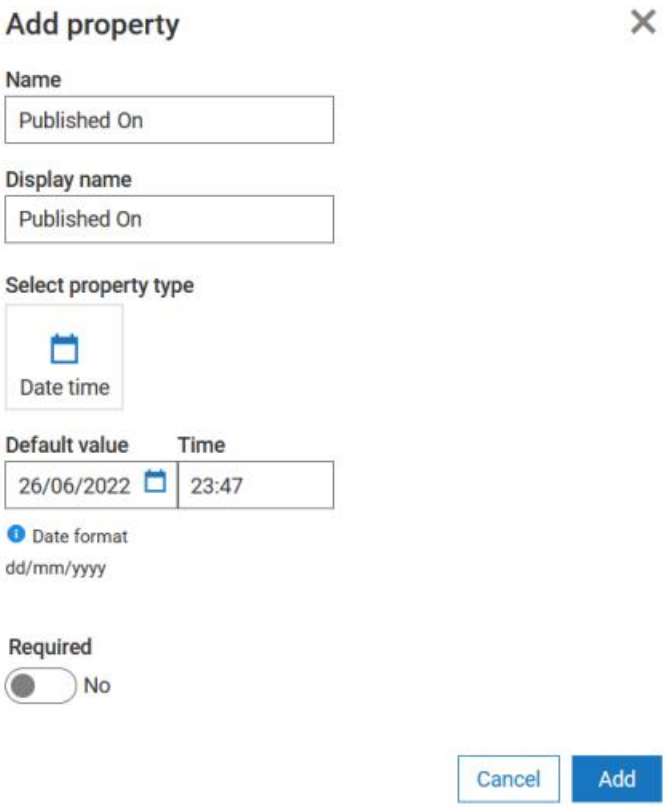
e. Click **Add**.

To add the **Published on** field:

a. Click **Add properties**.

b. Provide *Published On* in the **Name** and **Display name** fields.

c. Select *Date* from the **Select property type** field.



d. Click **Add**.

To add the **Author** field:

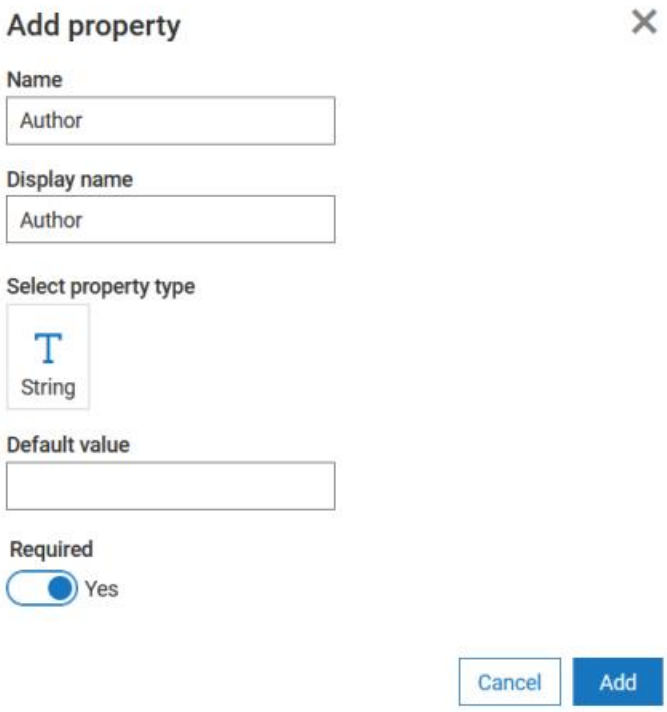
a. Click **Add properties**.

b. Provide *Author* in the **Name** field.

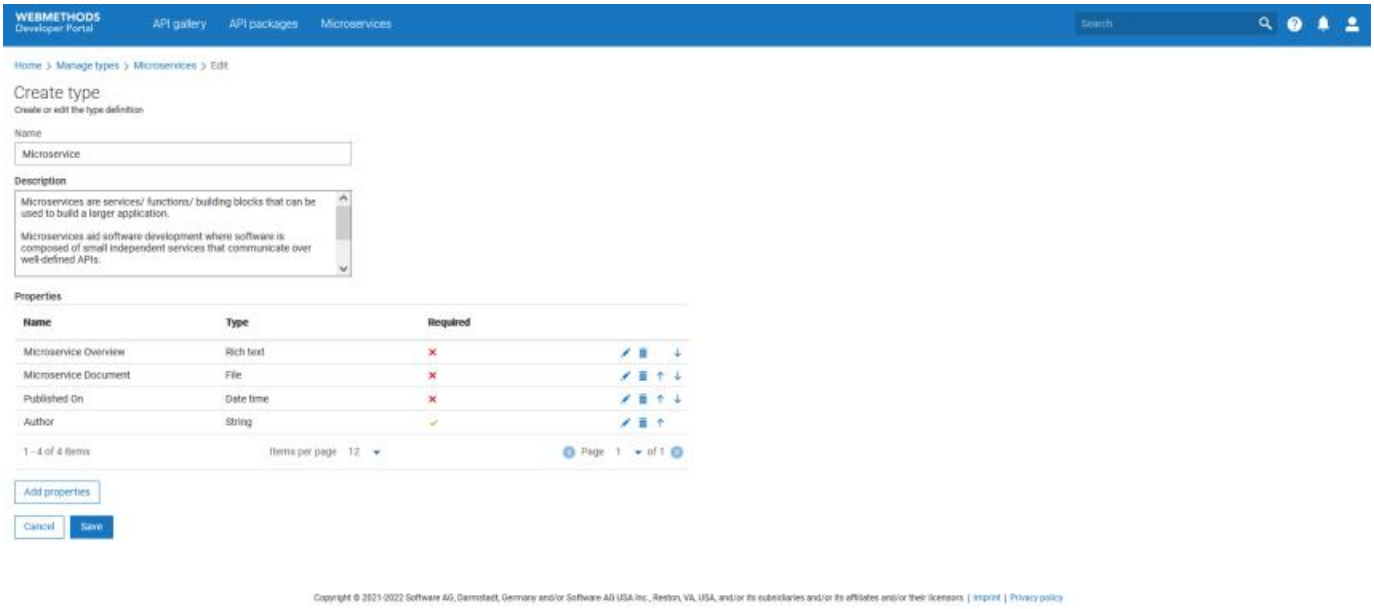
c. Provide *Author* in the **Display name** field.

d. Select *String* from the **Select property type** field.

e. Turn the **Required** slider on.



f. Click **Add**.



7. Click **Save**.

   The asset type is added.

Alternative steps

- In *Step 6*, you can select one of the following from the **Select property type** to add the corresponding property for the asset type being created:

Property type	Description	Inputs to be provided
Boolean	Includes a boolean value field with options, *Yes* or *No*.	Select the default value that must be displayed for the field. Select whether this property is mandatory when adding an asset.
Date time	Includes a date time field.	Select the default date and time that must be displayed in the field. Select whether this property is mandatory when adding an asset.
Enum	Includes a drop-down list.	Provide the values that must be displayed in the **Possible values** field. You must provide a value and type a comma to provide the next value. Select the default value that must be displayed in the field. Select whether this property is mandatory when adding the asset.
File	Includes a file upload button that you can click to upload a file when you add the asset.	Select whether this property is mandatory when adding the asset. **Note:** You cannot provide a default value for this field.
Image	Includes an image upload button that you can click to upload an image when you add the asset.	Select whether this property is mandatory when adding the asset. **Note:** You cannot provide a default value for this field.
Number	Includes a number field.	Provide the default value that must be displayed in the field. Select whether this property is mandatory when adding an asset.
Rich text	Includes a rich text field.	Provide the default value, with required formatting, to be displayed in the field.

Property type	Description	Inputs to be provided
		Select whether this property is mandatory when adding the asset.
String	Includes a text box.	Provide the default value that must displayed in the field. Select whether this property is mandatory when adding the asset.
Tags	Includes the tags field for the asset.	Provide the default value that must displayed in the field. Select whether this property is mandatory when adding the asset.

Note:
When you create an asset type, the following properties are added by default:

- Icon
- Name
- Version
- Summary
- Tags

Ensure that you do not add duplicate of these default properties.

- After you add the required list of properties, you can:
  - Edit the details of the field by clicking  .
  - Move the field up or down by clicking ↑ or ↓. These fields appear in the order they are created. You can modify the properties as per requirement.
  - Remove a field by clicking  .

Next steps

- Add assets for the added type from the **Manage asset** page. For the procedure to add assets, see "How do I add a custom asset to publish to Developer Portal?" on page 219.
- Customize the custom asset gallery and asset details pages. For more information, see "How do I add custom asset gallery to UI?" on page 220.
- Configure accessibility of the custom asset gallery page. For information on configuring accessibility, see "How do I assign access to a custom page?" on page 77.

How do I add a custom asset to publish to Developer Portal?

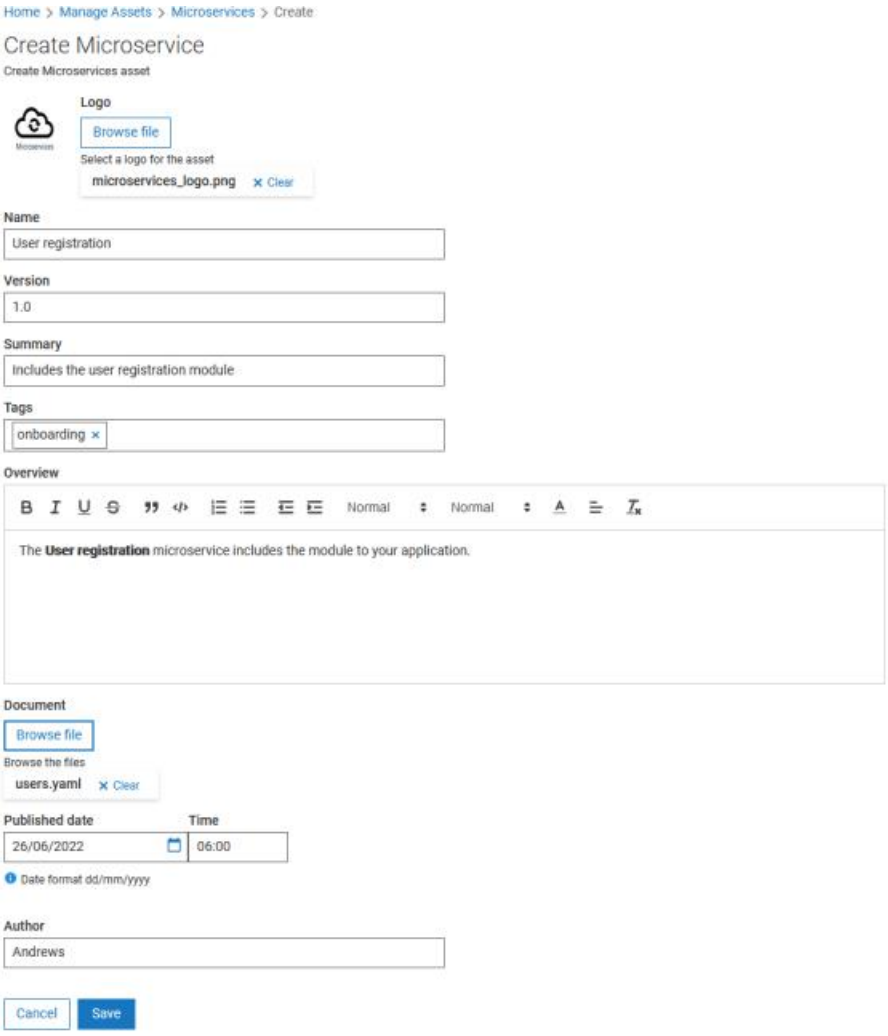
After you have added the required asset type, you can add the custom assets of the configured type.

This use case starts when you want to add a custom asset and ends when you add the required assets.

In this example, consider adding an asset for the type, *Microservice*, that is already configured.

To add a custom asset

1. Click the menu option icon  from the title bar and click **Administration**.
2. Click **Manage assets**.
3. Click **Microservice**.
4. Click **Add Microservice**.
5. Provide information for the following default fields:
   a. Click **Browse** and select the logo file.
   b. Provide *User registration* in the **Name** field.
   c. Provide *1.0* in the **Version** field.
   d. Provide a summary for the asset.
   e. Provide *sign in* in the **Tags** field and press **Enter**.
6. Provide information for the properties configured for the custom asset type:
   a. Provide an overview for the microservice in the **Overview** field.
   b. Click **Browse** from the **Document** field and select the required microservices document.
   c. Select a date and time from the **Published On** field.
   d. Provide an author name in the **Author** field.



7. Click **Save**.

Next steps

- Add the asset gallery and asset details pages to the UI. For more information, see "How do I add custom asset gallery to UI?" on page 220.

How do I add custom asset gallery to UI?

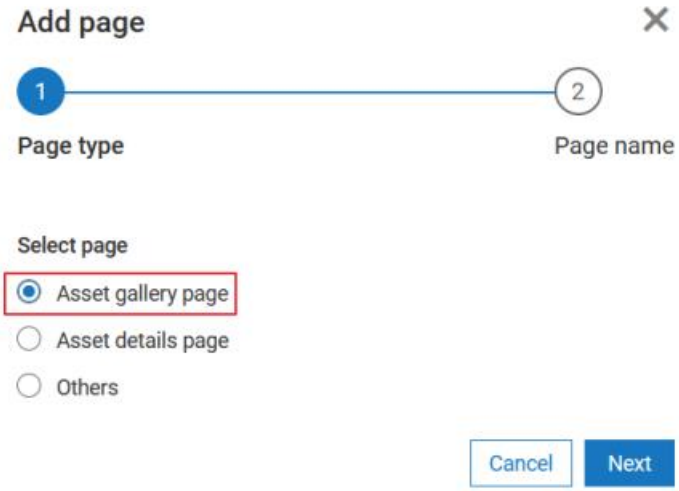
You must add the custom asset gallery to UI for the assets to appear in UI. If you have not added this to UI, then it is not exposed to your consumers. For more details, see "How do I add a new item to the top navigation bar?" on page 79.

The use case starts when you have added custom assets that has to be displayed in a gallery and ends when you have completed adding the page.

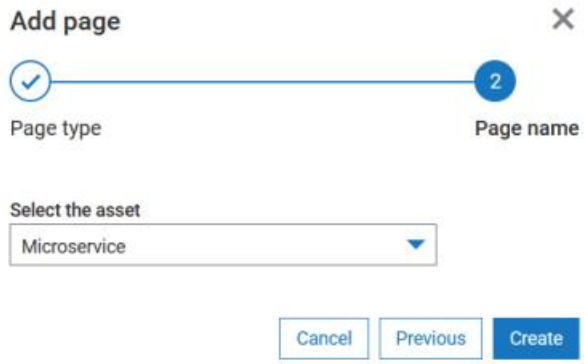
In this example, consider creating a gallery for the *Microservice* assets.

To add the custom asset gallery to the UI

1. Click the menu option icon from the title bar and click **Administration**.
2. Click the customize icon next to the active theme.
3. Select **Pages** and click **Add**.
4. From the **Add page** screen that appears, select **Asset gallery page**.

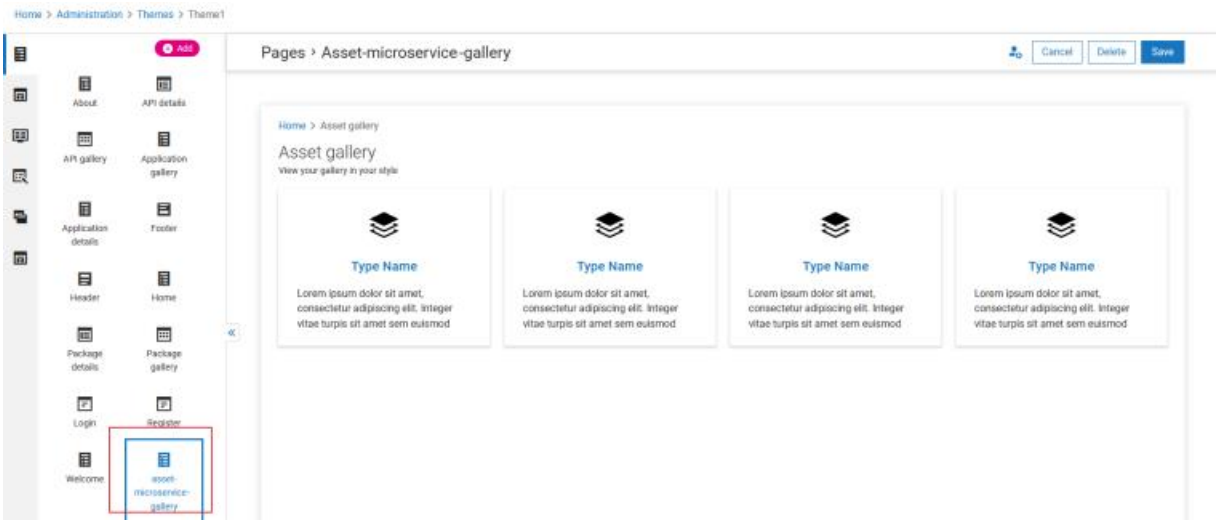


5. Click **Next**.
6. Select *Microservice* from the **Select the asset** field.

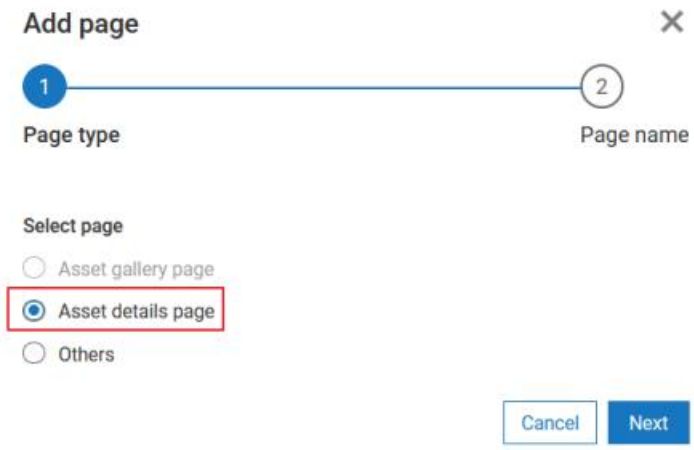


7. Click **Create**.

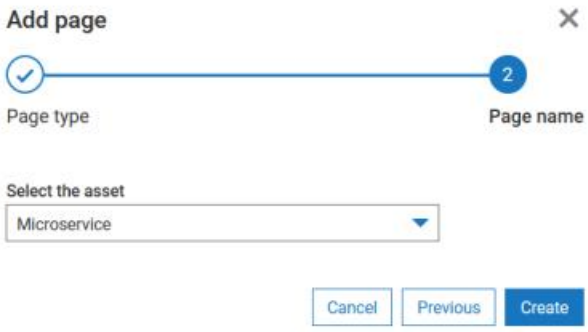
   The asset gallery is added to the list of pages.



8. Click **Add** to add the asset detail page.
9. From the **Add page** screen that appears, select **Asset details page**.

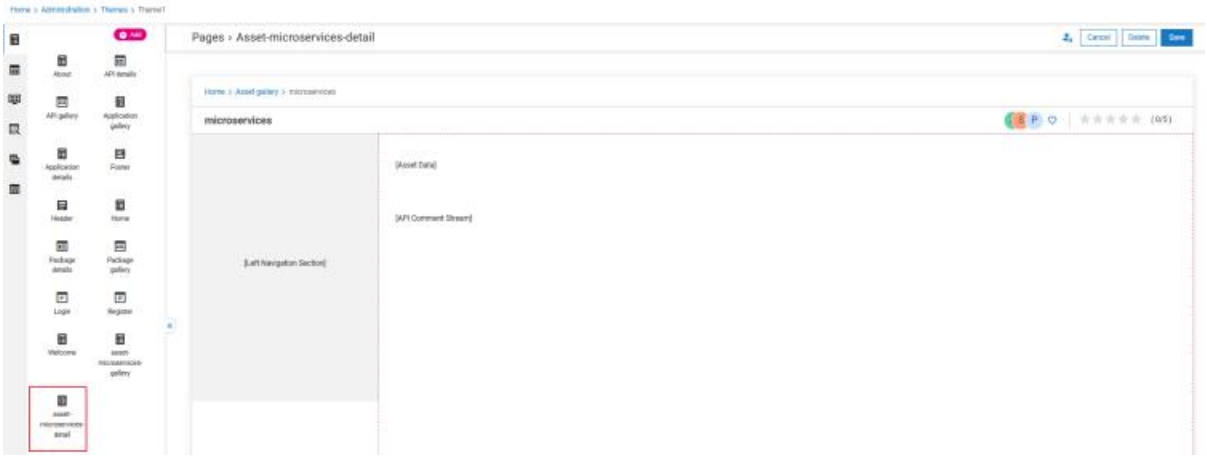


10. Click **Next**.
11. Select *Microservice* from the **Select the asset** field.



12. Click **Create**.

    The asset details is added to the list of pages.



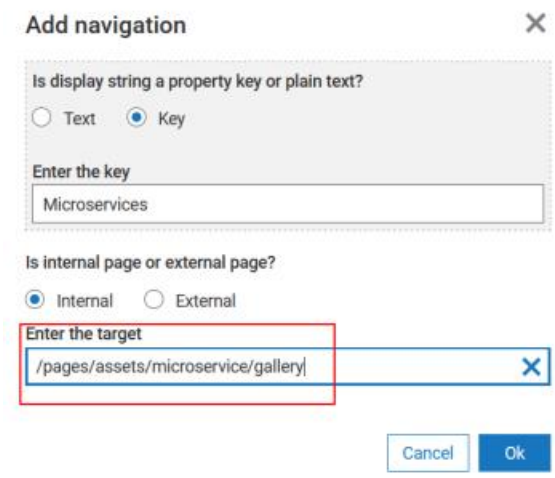
Next steps

- Add custom assets gallery to the top navigation bar. For information about adding a page to the top navigation bar, see "How do I add a new item to the top navigation bar?" on page 79.

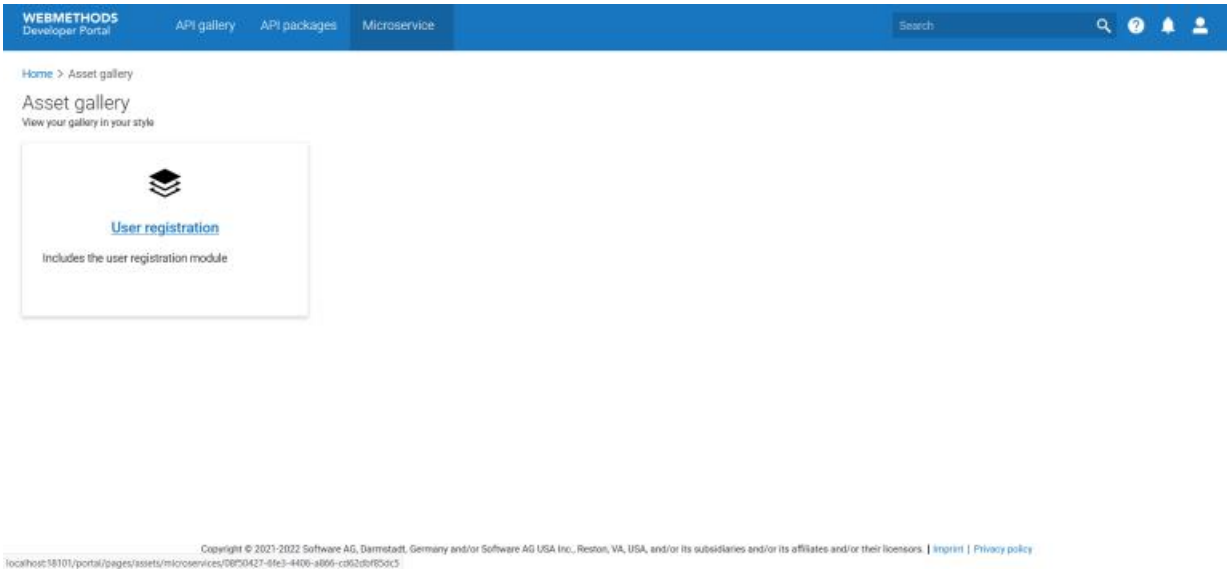
  To add the custom asset gallery to the top navigation, provide the page target URL in the following format:

  ```
  /pages/assets/custom_asset_type/gallery
  ```

  For example,



  The added custom assets appear in the gallery.



Note:
Ensure that you make the changes to the active theme or you apply the theme in which you have made the changes. Also, note you cannot modify the default theme for including custom asset gallery. You must create a new theme and customize it accordingly.

- Click the asset to view its details.
- Customize the appearance of the asset gallery and asset details pages as per your requirements. For information about customizing pages, see "How do I customize a block on a page?" on page 62.
- Customize the accessibility of the asset gallery and asset details pages. For information about customizing access of custom pages, see "How do I assign access to a custom page?" on page 77

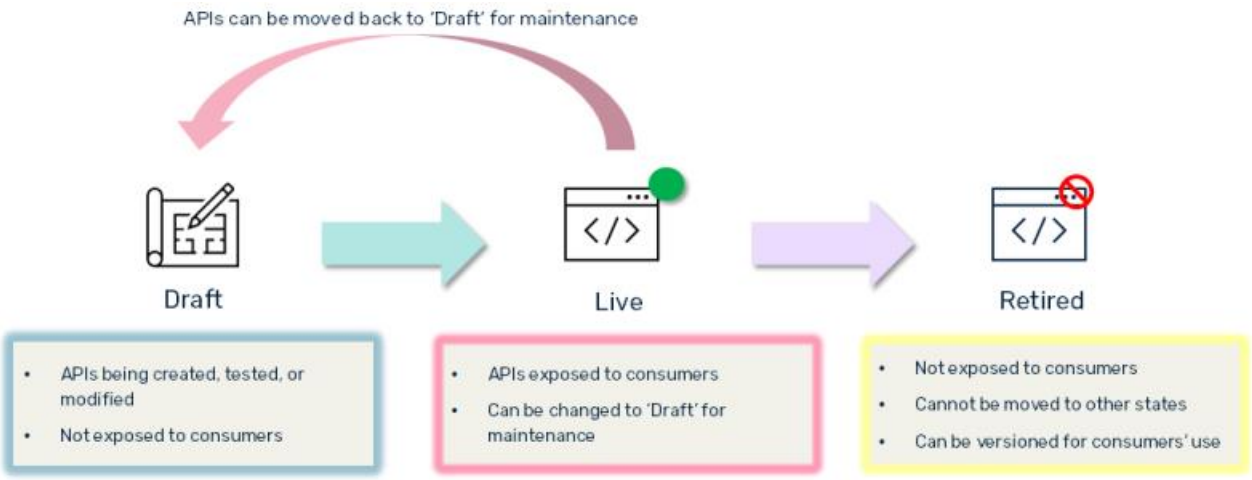
Lifecycle Management of APIs and Packages

By default, all APIs and API packages are accessible by consumers as and when they are published to Developer Portal. But, as an API provider, you may have a lot of APIs or API packages that are under construction or maintenance, and you may want to expose only the assets that are mature, tested, or well-documented for consumers. In such cases, the API lifecycle feature in Developer Portal allows you to expose the APIs that are ready for consumption. You can enable or disable this feature as per your requirement.

An API undergoes a lot of stages before it is ready to be exposed. The following section takes you through these stages and the states that you can assign to those APIs:

In general, API developers develop the skeleton of an API, start implementing, mock it for testing responses, and then activate the API and publish it to Developer Portal. After this, API providers can use the **Enrich API** feature in Developer Portal to include the required logo or localized descriptions to API and its resources, and then test it again to ensure the performance of the API.

You can move API states in the following linear way based on the development of your APIs:



The default lifecycle states available in Developer Portal are:

- **Draft**. Default state of all APIs. The APIs published to the Developer Portal are kept in the *Draft* state initially. APIs in this state do not appear in **API gallery**. These are not available for consumers through applications and packages.
- **Live**. APIs that are moved to the *Live* state appear in **API gallery**. If you want to prevent consumers from using an API, you can move the particular API in the *Live* state to the *Draft* state. Hence, you can move APIs to the *Draft* state for the maintenance of those APIs.
- **Retired**. You can move the APIs that are no more used to the *Retired* state . APIs that are moved to this state cannot be moved back to the other states. Only administrators and providers can view the retired APIs and their details. However, you can create new versions from a retired API and expose them to consumers.

Note:
You can modify the names of the states, and include additional states or remove them using the **application-lcm.yml** file that comes with the installation. For information about customizing lifecycle states, see "Customizing Lifecycle States" on page 227.

Lifecycle Feature Considerations

As an administrator or a provider, you must consider the following when you enable the lifecycle feature for APIs and packages in Developer Portal.

Visibility of assets

The following table lists the visibility and accessibility of assets on the UI based on user persona.

Use case	State of asset	API consumer	API provider	Administrator
View API in API gallery	Draft	✗	✗	✗
	Live	✓	✓	✓

Use case	State of asset	API consumer	API provider	Administrator
	Retired	✗	✗	✗
View package in API packages (Package gallery)	Draft	✗	✗	✗
	Live	✓	✓	✓
	Retired	✗	✗	✗
Request application (for an API) or subscription (for a package)	Draft	✗	✓	✓
	Live	✓	✓	✓
	Retired	✗	✓	✓
Search an API or package using global search	Draft	✗	✗	✗
	Live	✓	✓	✓
	Retired	✗	✗	✗
Accessing an API or package using deep link URL	Draft	✗	✓	✓
	Live	✓	✓	✓
	Retired	✗	✓	✓

Note:
API consumers can access the APIs or packages that are in *Draft* or *Retired* state, if they have a valid application or valid subscription associated with the asset.

State of restored APIs

When you restore APIs from an archive, the restored APIs retain the same state from the backup archive. If APIs are backed up from an instance in which the API lifecycle feature is not enabled, then the restored APIs are kept in the *Draft* state.

State of migrated APIs

When you migrate APIs to 10.15 from an earlier version of Developer Portal, the migrated APIs are kept in the *Draft* state. You can change their state as per your requirement.

Creating applications for APIs in Draft **state**

Only administrators or providers can create applications for the APIs that are in the *Draft* state, in order to test them before exposing to consumers. Consumers cannot view or use those APIs.

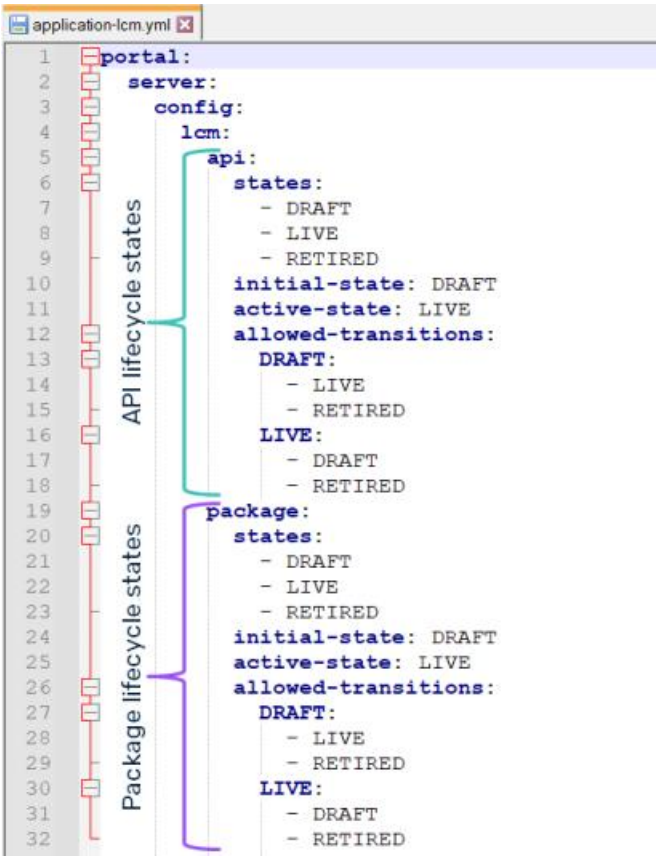
Customizing Lifecycle States

The Lifecycle Management of APIs and Packages section explains the default lifecycle and the possible transition that Developer Portal offers. However, you can modify names of those states, the default state that must be assigned to an API or package, and the transition of the states.

To customize the default states names and their transition

1. Navigate to the following location in the installation folder and open **application-lcm.yml** in *Edit* mode.

   SAGInstallDir/DeveloperPortal/workspace/webapps/portal.war/WEB-INF/classes



2. Make changes as per your requirement in the corresponding section:
   - Add, edit, or remove the available states from the **states** section.
   - Specify the initial state of an API or package in the **initial-state** section.
   - Specify the state that has to be applied to an API or package to make it active (expose it to consumers) in the **active-state** section.
   - Specify the flow of states from one state in the **allowed-transitions** section.

3. If you add a new state or modify a state, add the custom label under the **Properties** section of your active theme. For information on adding a new label, see "How do I add new UI labels?" on page 96.

4. Save the changes. Restart Developer Portal for the changes to take effect.

Sample scenario

Consider adding a new state called *Review*. This new state must be in between *Draft* and *Live* states.

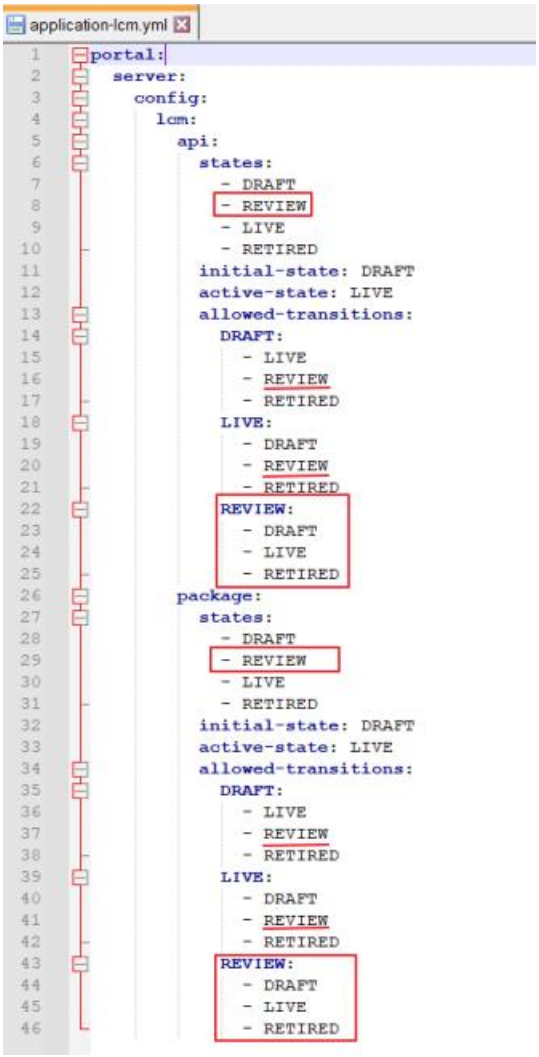
To add the new state

1. Navigate to the following location in the installation folder and open **application-lcm.yml** in *Edit* mode.

   `SAGInstallDir/DeveloperPortal/workspace/webapps/portal.war/WEB-INF/classes`

2. Add *Review* in the **states** section under the **api** and **package** sections.

3. Add the states for the possible transitions from the *Review* state as shown in the following image:



4. Save the changes and restart Developer Portal.

5. Click the menu options icon from the title bar and select **Administration**.

6. From the **Manage themes** page, click the customize icon next to the active theme.

7. Select **Properties** and select **Custom**.

8. Click the **Click here** link.

   The **Add property** screen appears.

9. In the **Key** field, provide the new state name with the prefix as *base.lifecycle.state* and suffix as *.LBL*.

   For the example considered, you must provide *base.lifecycle.state.review.LBL*.

10. Provide *REVIEW* in the **Key** field.

11. Click **Ok**.

    The label is added to the lifecycle states.

    Important:
    Ensure that you add the custom label in the active theme for the changes to take effect.

How do I enable the lifecycle feature for APIs and packages?

By default, the lifecycle management feature is not enabled. You must enable the feature from the **Administration** section of Developer Portal to use the feature.

To enable the lifecycle feature

1. Click the menu options icon  from the title bar and click **Administration**.
2. Click **General**.
3. Turn the **Enable lifecycle** slider on.
4. Click **Save**.

   The lifecycle feature for APIs and packages is enabled.

   The APIs and packages in the *Draft* state are not available for the use of consumers.

   When you enable the lifecycle feature, all APIs and pacakges in your Developer Portal instance are moved to the *Draft* state.

Next steps:

- Move the required APIs and packages to the *Live* state. For information about moving APIs and packages to the *Live* state, "How do I change the state of an API to expose it to consumers?" on page 230 and "How do I change the state of a package to expose it to consumers?" on page 234 respectively.

Modifying State of APIs

How do I change the state of an API to expose it to consumers?

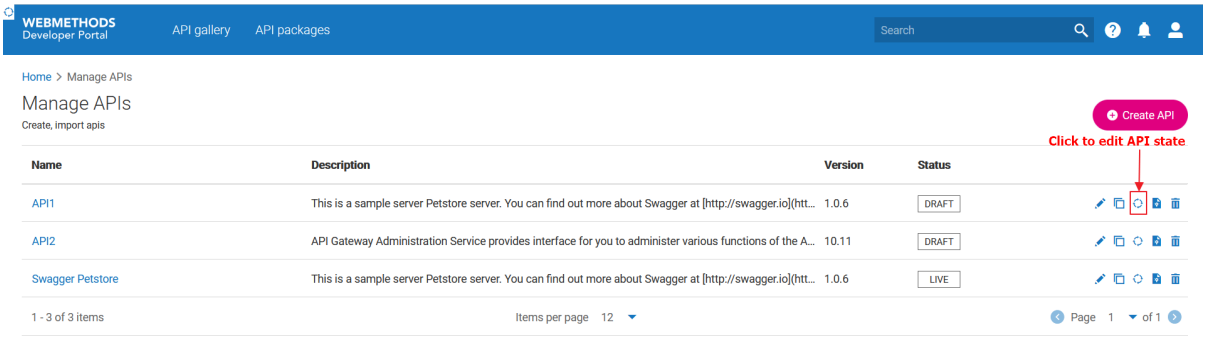
This use case explains the steps to move an API to the Live state in order to expose it to consumers.

This use case starts when you want to move an API that is in *Draft* state to the *Live* state.

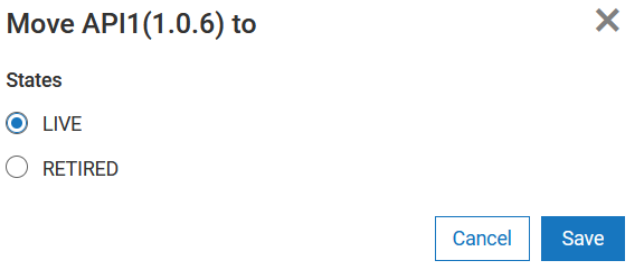
In this example, consider the API, *API1* that is in the *Draft* is state moved to the *Live* state.

To move an API to the *Live* state

1. Click the menu options icon from the title bar and click **Manage APIs**.

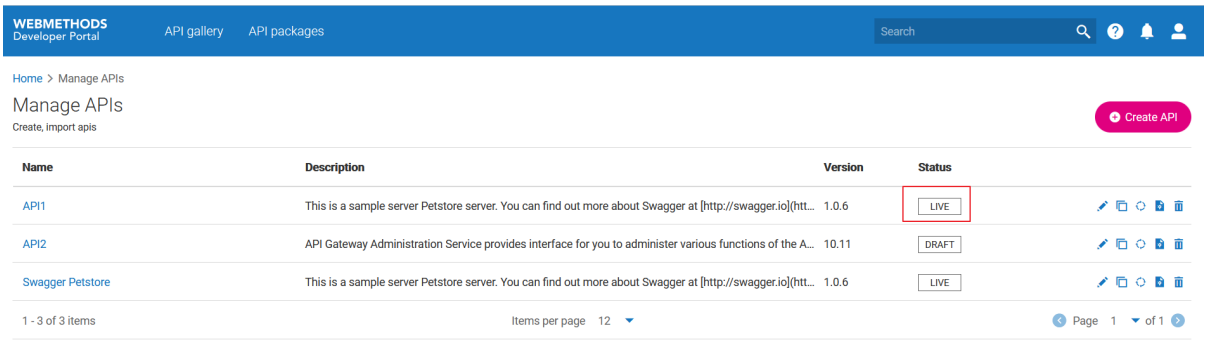


2. Click the change state icon icon next to the *API1*.



3. Select *Live*.
4. Click **Save**.

   The state of API is changed to *Live*.



   APIs that are moved the *Live* state appear in the **API gallery** page.

Next steps:

- Consumers can try the APIs that are in the *Live* state .
- Providers can include the *Live* APIs to API packages and publish them for the use of consumers.

How do I change the state of a Live API for maintenance?

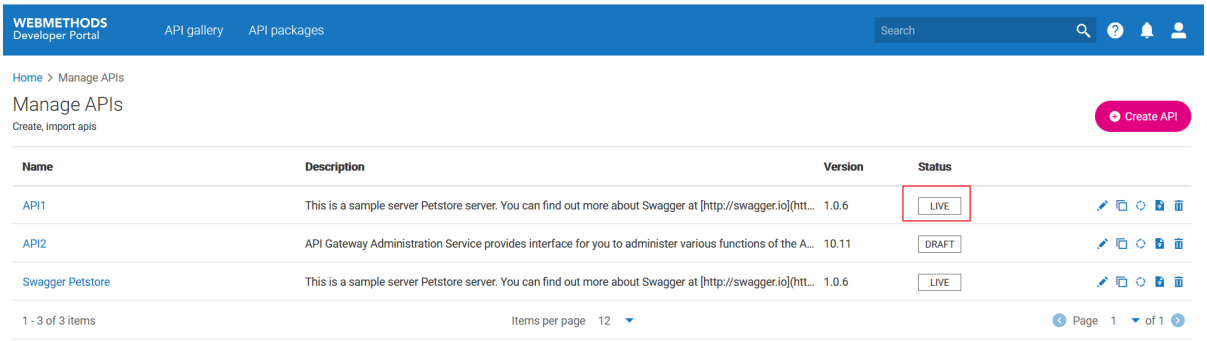
This use case explains the steps to move an API to the *Draft* state for performing any maintenance activity on that API.

This use case starts when you want to move an API to the *Draft* state from the *Live* state .

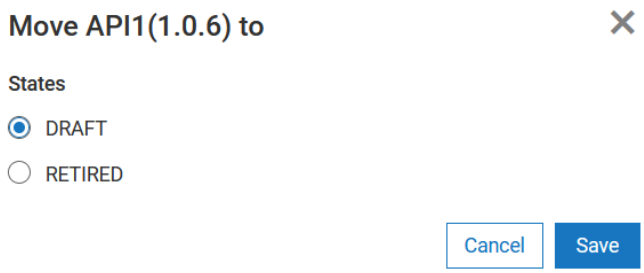
In this example, consider the same API, *API1* that is in *Live* state moved to the *Draft* state.

To move an API to the *Draft* state

1. Click the menu options icon from the title bar and click **Manage APIs**.

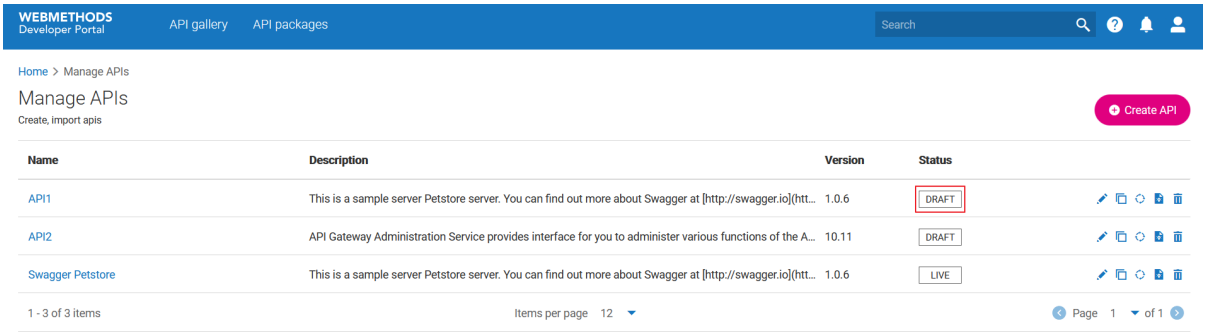


2. Click the change state icon icon next to *API1*.



3. Select *Draft*.
4. Click **Save**.

   The state of API is changed to *Draft*.



APIs that are moved to the *Draft* state do not appear in the **API gallery** page.

Next steps:

- Providers or administrators can create applications to test the APIs that are in the *Draft* state.
- Providers can perform the required maintenance activities to the *Draft* APIs and then change their state to *Live* for the use of consumers.

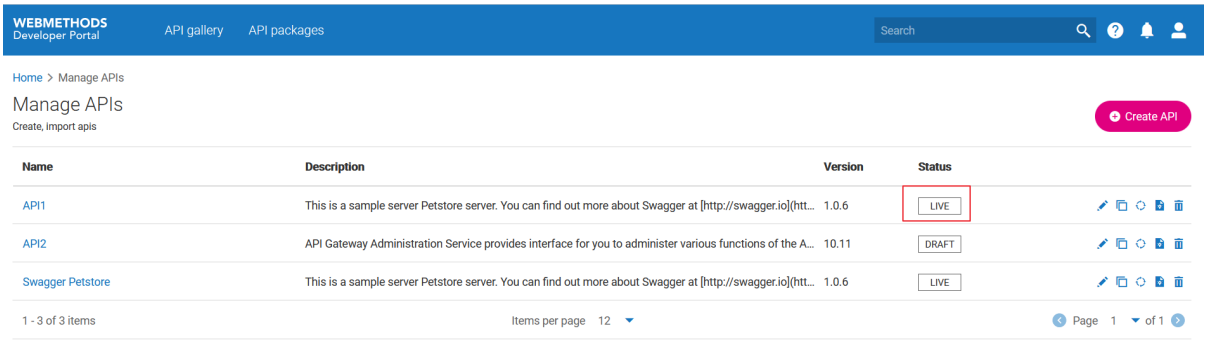
How do I change an API's state to retire it from usage?

This use case starts when you want to retire an API that is in the *Live* or *Draft* state to prevent consumers from viewing it.

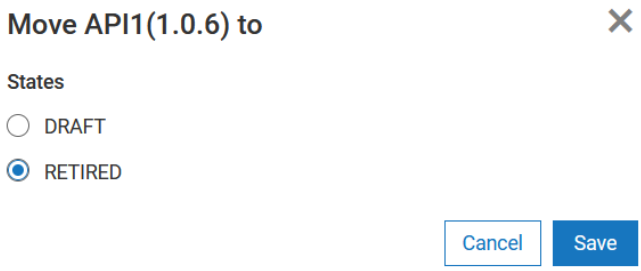
In this example, consider the API, *API1* that is in *Live* state.

To move an APIs to the *Retired* state

1. Click the menu options icon from the title bar and click **Manage APIs**.

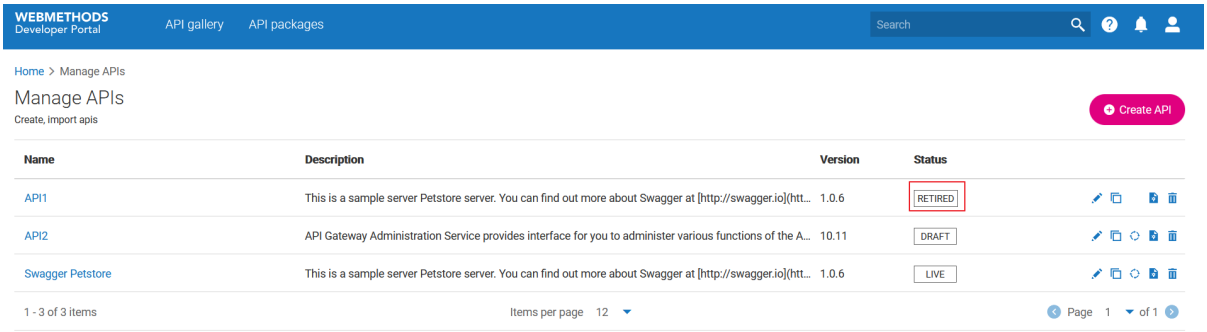


2. Click the change state icon icon next to the *API1*.



3. Select *Retired*.
4. Click **Save**.

   The state of API is changed to *Retired*.



Next steps:

- You cannot move the retired APIs to the other states.
- You can create new versions of the retired API and expose to your consumers.

Modifying State of Packages

How do I change the state of a package to expose it to consumers?

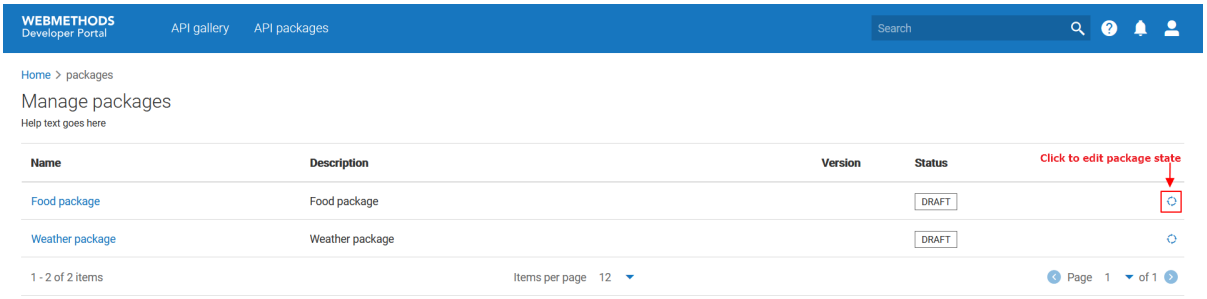
This use case explains the steps to move a package to the Live state to expose it to consumers.

This use case starts when you want to move an package that is in *Draft* state to the *Live* state.

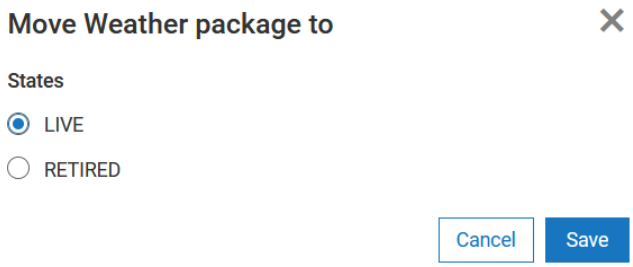
In this example, consider the package, *Weather* that in the *Draft* state moved to the *Live* state.

To move a package to the *Live* state

1. Click the menu options icon  from the title bar and click **Manage packages**.

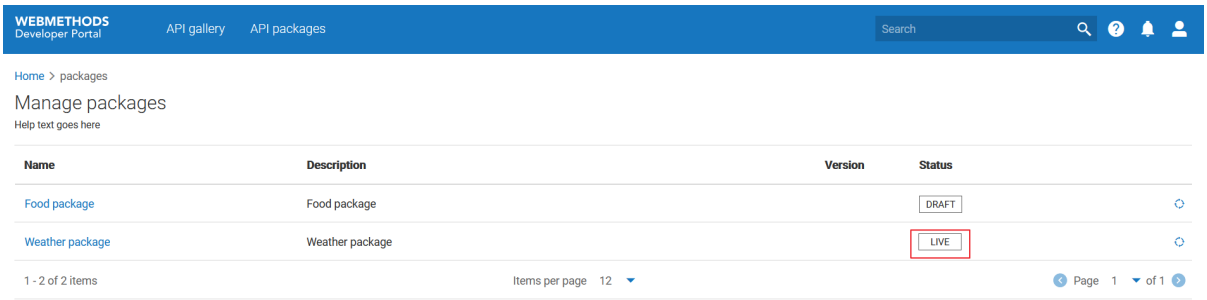


2. Click the change state icon icon next to the package.



3. Select *Live*.
4. Click **Save**.

   The state of package is changed to *Live*.



   Packages that are moved to the *Live* state appear in the **API packages** page.

Next step:

- Consumers can view the packages that are in the *Live* state.

How do I change the state of a Live package for maintenance?

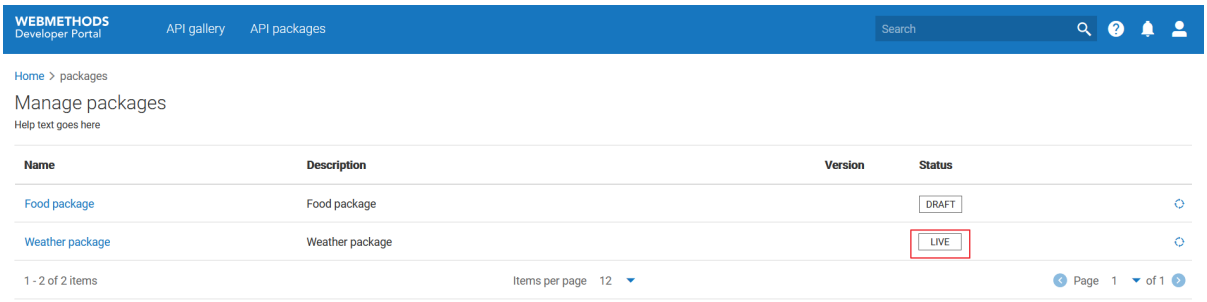
This use case explains the steps to move a package to the *Draft* state for performing any maintenance activity on that API.

This use case starts when you want to move a package to the *Draft* state from the *Live* state .

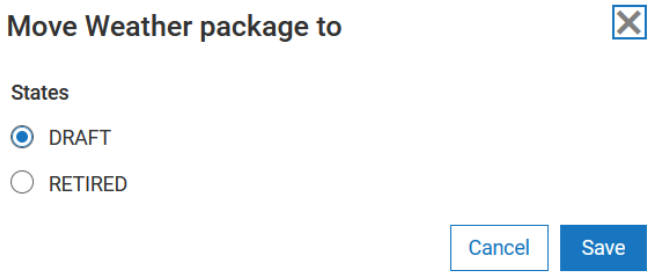
In this example, consider the same package, *Weather* that is in *Live* state moved to the *Draft* state.

To move an package to the *Draft* state

1. Click the menu options icon from the title bar and click **Manage packages**.

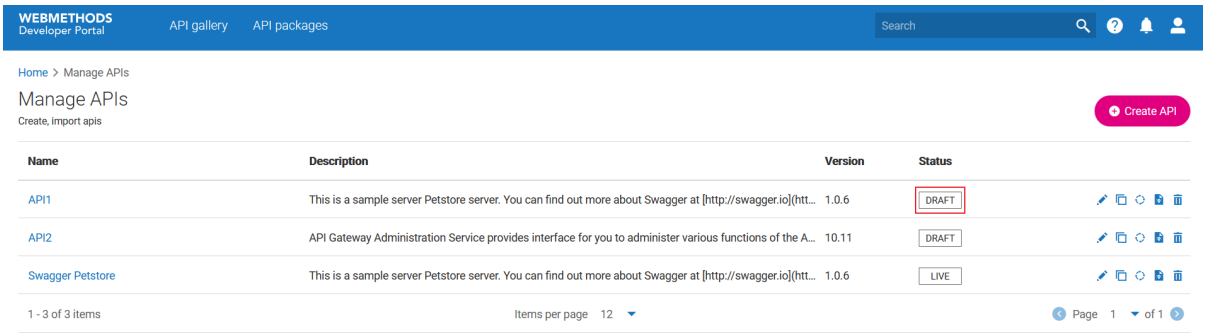


2. Click the change state icon icon next to the package.



3. Select *Draft*.
4. Click **Save**.

   The state of package is changed to *Draft*.



   Packages that are moved to the *Draft* state do not appear in the **API packages** page.

Next steps:

- Providers can perform the required maintenance activities to the *Draft* packages and then change their state to *Live* for the use of consumers.

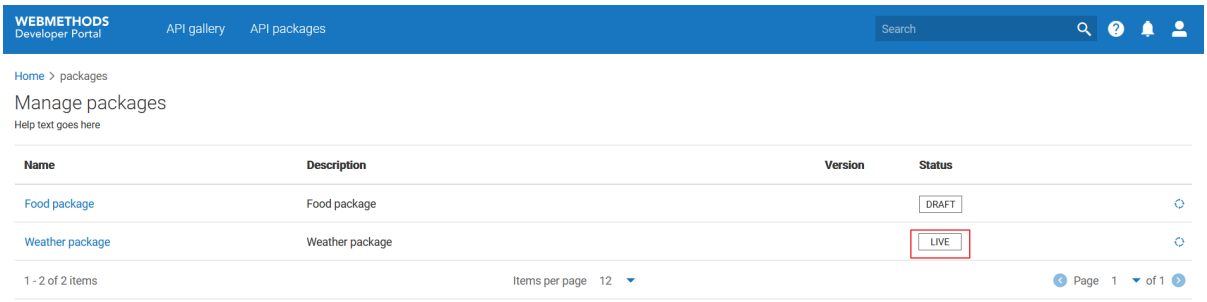
How do I change a package's state to retire it from usage?

This use case starts when you want to retire a package that is in the *Live* or *Draft* state to prevent it from the view of consumers.

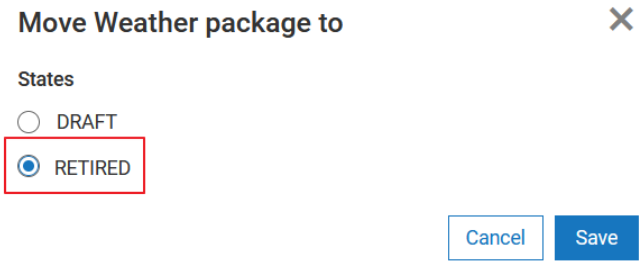
In this example, consider the package, *Weather* that is in *Live* state.

To move a package to the *Retired* state

1. Click the menu options icon from the title bar and click **Manage packages**.

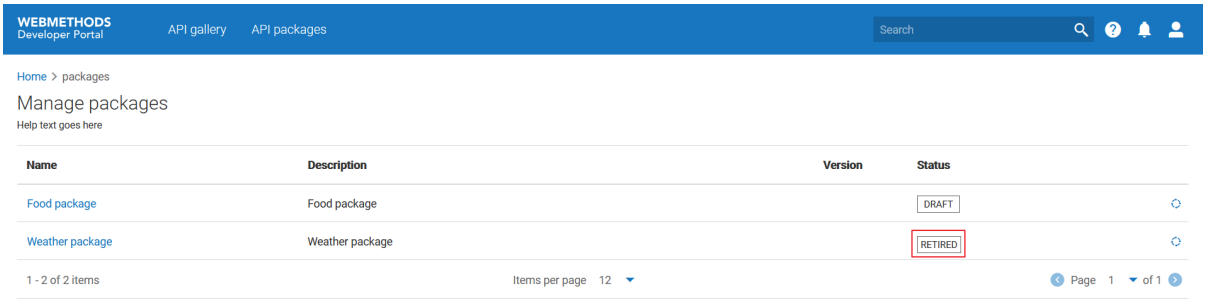


2. Click the change state icon icon next to the package.



3. Select *Retired*.
4. Click **Save**.

   The state of package is changed to *Retired*.



You cannot move the retired packages to the other states.

8 REST APIs

Overview

The headless architecture feature of Developer Portal equips you with the APIs that you would require to build an application with the Developer Portal functionality.

The REST APIs that come along with the Developer Portal installation allow you to perform all functions that you can accomplish through the application's user interface.

Developer Portal REST APIs are located at *SAGInstallDir*\DeveloperPortal\developers\openapis.

Authentication to use Developer Portal REST APIs

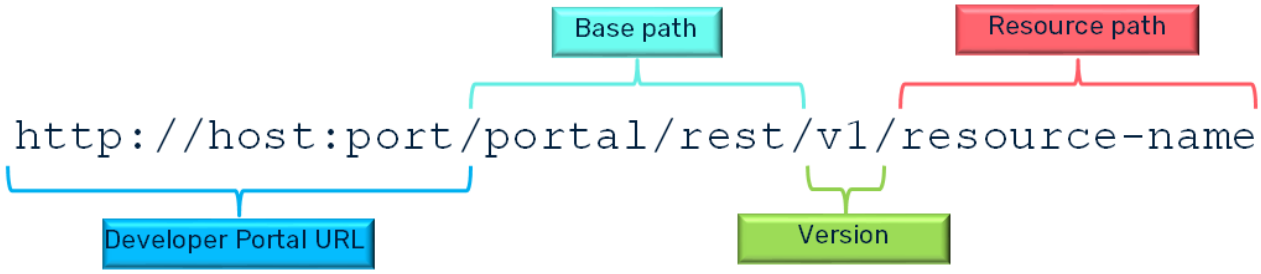
The authentication method applicable for the Developer Portal REST APIs is HTTP Basic authentication.

Access privileges based authorization

The availability of API resources is based on the user privilege. Typically, users who have the **API Administrator** privilege have access to all APIs, whereas, the users who have any of the **API Provider** and **API Consumer** privileges have limited access of API resources.

REST APIs URI format

Developer Portal allows you to access the API resources using URI paths. The REST call URI must be specified with the required method in the format shown below:



Possible response messages

The following table lists the possible response messages and what they indicate:

Response message	Description
200 - OK	Indicates the request sent and response received successfully.
201 - Created	Indicates that the request is fulfilled, resulting in the creation of a new resource.
400 - Bad request	Indicates that a parameter is missing in the request due to which it could not be processed.

Response message	Description
401 - Unauthorized	Indicates authorization problem.
403 - Forbidden	Indicates that there is no permission to the requested resource.
404 - Not found	Indicates that the requested resource does not exist.
500 - Internal server error	Indicates that an unexpected condition is encountered and no more specific message is suitable.

List of APIs

The REST APIs allow you to perform the following:

- "Managing webhooks" on page 269.

Viewing analytics

The Analytics API is used to view the performance analytics of APIs and applications in the system. This API provides detailed information on the trends, top hits, usage based on the filters applied. There are variety of feeds and each is responsible for providing the aforesaid specific information.

List of resources

- **POST** /analytics/{id}

  Retrieves the analytic insights for the given feed Id and matching filter criteria.

  Possible values are:

Value	Description
TRANSACTION_SUMMARY_FEED	Retrieves the summary of all transactions.
API_ACCESS_TREND_FEED	Retrieves the API access trend.
APP_ACCESS_TREND_FEED	Retrieves the application access trend.
API_RESPONSE_TIME_TREND_FEED	Retrieves the API response time trend.
TOP_API_HITS_BY_ACCESS_FEED	Retrieves the list of top APIs based on the number of hits.
TOP_API_BY_FOLLOWERS_FEED	Retrieves the list of top APIs based on the number of followers.
TOP_RESOURCE_HITS_BY_ACCESS_FEED	Retrieves the list of top resources based on the number of hits.
TOP_STATUSCODE_HITS_BY_ACCESS_FEED	Retrieves the list of top status codes based on the number of hits.
TOP_APP_HITS_BY_ACCESS_FEED	Retrieves the list of top applications based on the number of hits.
APP_DISTRIBUTION_IN_API_FEED	Retrieves the app distribution data from APIs.
API_DISTRIBUTION_IN_APP_FEED	Retrieves the API distribution data from applications.
SIGN_UP_TREND_FEED	Retrieves the user sign-up trend.
LOGIN_TREND_FEED	Retrieves the sign-in trend.
ACCESS_LOGS	Retrieves access logs.

Value	Description
TOP_APP_TYPE_HITS_BY_ACCESS_FEED	Retrieves the list of application types based on the number of hits.
USER_SIGNUP_SUMMARY_FEED	Retrieves the user sign-up summary.
USER_LOGIN_SUMMARY_FEED	Retrieves the user sign-in summary.
TOP_CONSUMER_LOGINS	Retrieves the list of top consumer sign ins.
TOTAL_USER_SIGNUP_AND_DELETE_FEED	Retrieves the total number sign-ups and deletes.

- **POST** /analytics/ACCESS_LOGS

  Retrieves the access logs for the matching filter criteria.

Sample cURL Command

```
curl --location --request POST
'developer_portal_rest_base/analytics/TRANSACTION_SUMMARY_FEED' \
--header 'Authorization: Basic basic_auth' \
--header 'Content-Type: application/json' \
--data-raw '{
    "apis":["api_id"]
}'
```

The `analytics.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing APIs

Developer Portal provides the capability to retrieve and manage all its APIs. You can use the resources of the API to update or retrieve the API details such as the associated applications, versions, topics, communities and so on.

List of resources

- **GET** /apis

  Retrieves the list of APIs in Developer Portal.

- **POST** /apis

  Publishes an API to Developer Portal.

- **GET** /apis/_count

  Retrieves the count of APIs in Developer Portal.

- **GET** apis/available

  Retrieves the list of APIs that are not mapped with any provider.

- **GET** /apis/filter

  Retrieves the possible filter criteria of the given Id.

- **GET** /apis/{id}

  Retrieves API based on the given Id.

- **PUT** /apis/{id}

  Updates API based on the given Id

- **DELETE** /apis/{id}

  Unpublishes API based on the given Id.

- **GET** /apis/{id}/communities

  Retrieves the list of communities associated with the given Id.

- **GET** /apis/{id}/topic

  Retrieves the list of topics for the given Id.

- **GET** /apis/{id}/applications

  Retrieves the list of applications associated with the given Id.

- **GET** /apis/{id}/subscriptions

  Retrieves the list of subscriptions mapped with the given Id.

- **GET** /apis/{id}/versions

  Retrieves the versions of the API.

- **GET** /apis/{id}/stages

  Retrieves the stages of the API.

- **GET** /apis/{id}/followers

  Retrieves the list of followers of the given Id.

- **PUT** /apis/{id}/followers

  Subscribes or unsubscribes to receive the updates of the given Id.

- **GET** /apis/{id}/packages

  Retrieves the list of packages associated with the given Id.

- **GET** /apis/{id}/groups

  Retrieves the list of groups associated with the given Id.

- **GET** /apis/{id}/applications/available

Retrieves the list of applications that can be associated with the given Id.

- **PUT** /apis/{id}/rate

  Allows to provide a rating for the given Id.

- **GET** /apis/{id}/rate

  Retrieves the rating of the given Id.

- **GET** /apis/{id}/followers/_count

  Retrieves the number of followers for the given Id.

- **GET** /apis/{id}/bookmarks

  Retrives the list of topics of the given Id that are saved as bookmarks.

- **POST** /apis/{id}/try

  Allows to test the given Id.

- **POST** /apis/{id}/fileTypeTry

  Allows to test the multipart or binary type resources of the given Id.

- **GET** /apis/{id}/try/history

  Retrieves the details of tests performed for the given Id.

- **GET** /apis/status/{referenceId}

  Retrieves the published status of an API based on the given reference Id.

- **POST** /apis/search

  Searches for an API based on the given search criteria.

- **GET** /apis/{id}/try/history

  Retrieves the details of tests performed for the given Id.

- **GET** /apis/{id}/export

  Exports the given Id.

- **PUT** /apis/{id}/preferences

  Updates the view preference of the given Id.

- **PUT** /apis/{id}/edits

  Allows to edit the details of the given Id and its resources.

- **PUT** /apis/{id}/logo

  Allows to update the logo of the given Id.

- **PUT** /apis/{id}/attachments

  Allows to update the attachments of the given Id.

- **PUT** /apis/{id}/state

  Allows to modify the lifecycle state of the specific API.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/apis' \
--header 'Authorization: Basic basic_auth'
```

The `apis.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing applications

This REST API is used to create applications, retrieve application information, share applications with require users or teams, and delete the existing applications using a REST API.

List of resources

- **GET** /applications

  Retrieves the list of applications in Developer Portal for the current user.

- **POST** /applications

  Creates an application with the given data.

- **GET** /applications/{id}

  Retrieves the application associated with the given Id.

- **PUT** /applications/{id}

  Updates the application associated with the given Id.

- **DELETE** /applications/{id}

  Deletes the application associated with the given Id.

  Note:
  To delete an inactive or obsolete applications, make a REST call to the following endpoint with the required application Id:

  ```
  DELETE  /applications/{id}?force=true
  ```

- **PUT** /applications/{id}/share

  Allows to share the specified application with a user or team.

- **DELETE** /applications/{id}/share

  Allows to revoke the access of the specified application.

- **GET** /applications/_all

  Retrieves the list of applications in Developer Portal irrespective of the users associated with them.

- **GET** /applications/{id}/requests

  Retrieves the users requests received for the specified application.

- **GET** /applications/{id}/tokens

  Retrieves the list of specified tokens generated for the specified application.

- **DELETE** /applications/{id}/tokens/{tokenId}

  Deletes the specified tokens generated for the specified application.

- **GET** /applications/{id}/scopes

  Retrieves the list of scopes mapped with the specified application.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/applications' \
--header 'Authorization: Basic basic_auth'
```

The `applications.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing approvals

This REST API is used to retrieve details of approval processes that are used to onboard a user, application, or subscription. You can also retrieve, approve. or reject the user or application onboarding requests.

List of resources

- **GET** /approvals

  Retrieves the list of configured approval workflows in Developer Portal.

- **POST** /approvals

  Creates a new approval workflow with the given details.

- **GET** /approvals/{id}

  Retrieves the specified approval.

- **PUT** /approvals/{id}

  Updates the specified approval workflow with the given details.

- **DELETE** /approvals/{id}

  Deletes the specified approval workflow.

- **GET** /approvals/policy

  Retrieves the approval workflow policy assigned to onboarding process.
- **PUT** /approvals/policy

  Assign an approval workflow policy to onboarding process.
- **GET** /approvals/request

  Retrieves the list of pending approval requests.
- **GET** /approvals/request/{id}

  Retrieves the specified pending approval request.
- **PUT** /approvals/request/{id}/approve

  Approves the specified approval request.
- **PUT** /approvals/request/external/{id}/approve

  Approves the specified external approval request.
- **PUT** /approvals/request/approve

  Approves the set of specified pending approval requests.
- **PUT** /approvals/request/{id}/reject

  Rejects the specified approval request.
- **PUT** /approvals/request/external/{id}/reject

  Rejects the specified external approval request.
- **PUT** /approvals/request/reject

  Rejects the set of specified pending approval requests.
- **GET** /approvals/instance

  Retrieves the list of approval request details by their status.
- **PUT** /approvals/instance/step/{id}/request

  Retrieves the approval request associated with the given Id.
- **GET** /approvals/instance/step/{id}/trace

  Retrieves the approval request trace logs of the given approval Id.
- **PUT** /approvals/instance/step/{id}/logs

  Retrieves the approval request logs for the given Id.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/approvals' \
--header 'Authorization: Basic basic_auth'
```

The `approvals.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing backup and restore

This REST API is used to perform data backup and restore.

List of resources

- **POST** /data/backup

  Creates a backup of the specified modules.

- **POST** /data/restore

  Restores the specified modules in the specified data file.

- **GET** /data/status/{handle}/backup

  Downloads the specified backup file.

- **GET** /data/status/{handle}

  Retrieves the download status of the specified backup file.

Sample cURL Command

```
curl --location --request POST 'developer_portal_rest_base/data/backup/' \
--header 'xsrf-token: VbOuuiSA6K8YNWvN2D8Dl-I9sUU9GVM8bFZBMjVScFo' \
--header 'Authorization: Basic basic_auth' \
--header 'Content-Type: application/json' \
--data-raw '{
   "modules": ["User","Collaboration","Theme","Core"]
}'
```

The `backup-restore.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing comments

This REST API is used to add, retrieve, upvote or downvote, and delete comments under a topic.

List of resources

- **GET** /topics/{id}/comments

  Retrieves the list of comments for the given topic.

- **POST** /topics/{id}/comments

Creates a comment under the given topic.

- **GET** /topics/{topicId}/comments/{id}

  Retrieves the details of a specified comment under the given topic.

- **PUT** /topics/{topicId}/comments/{id}

  Updates the specified comment under the given topic.

- **DELETE** /topics/{topicId}/comments/{id}

  Deletes the specified comment under the given topic.

- **PUT** /topics/{topicId}/comments/{id}/upvote

  Upvotes the specified comment under the given topic.

- **PUT** /topics/{topicId}/comments/{id}/downvote

  Downvotes the specified comment under the given topic.

- **PUT** /topics/{topicId}/comments/{id}/flag

  Adds a flag to the specified comment under the given topic.

Sample cURL Command

```
curl --location --request
GET 'developer_portal_rest_base/topics/topic_Id/comments' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic basic_auth' \
```

The `comment.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing communities

This REST API is used to add, view, update, or delete communities and their details.

List of resources

- **GET** /communities

  Retrieves the list of communities in Developer Portal.

- **POST** /communities

  Creates a new community with the given details.

- **GET** /communities/{id}

  Retrieves the details of the specified community.

- **PUT** /communities/{id}

Updates the specified community.

- **DELETE** /communities/{id}

  Deletes the specified community.

- **GET** /communities/{id}/apis

  Retrieves the list of APIs associated with the specified community.

- **PUT** /communities/{id}/apis

  Add the given APIs to the specified community.

- **DELETE** /communities/{id}/apis

  Delete the given APIs from the specified community.

- **GET** /communities/{id}/users

  Retrieves the list of users associated with the specified community.

- **PUT** /communities/{id}/users

  Add the given users to the specified community.

- **DELETE** /communities/{id}/users

  Delete the given users from the specified community.

- **GET** /communities/{id}/groups

  Retrieves the list of user groups associated with the specified community.

- **PUT** /communities/{id}/groups

  Add the given user groups to the specified community.

- **DELETE** /communities/{id}/groups

  Delete the given user groups from the specified community.

- **PUT** /communities/{id}/owner

  Update the owner of the specified community.

- **GET** /communities/{id}/packages

  Retrieves the list of packages associated with the specified community.

- **PUT** /communities/{id}/packages

  Add the given packages to the specified community.

- **DELETE** /communities/{id}/packages

  Delete the given packages from the specified community.

- **GET** /communities/{id}/ispublic

  Allows you to verify whether the specified community is a public community.

- **GET** /communities/{id}/administrators

  Retrieves the list of administrators of the specified community.

- **PUT** /communities/{id}/packages

  Add the given administrators to the specified community.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/communities/' \
--header 'Authorization: Basic basic_auth'
```

The `communities.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing configurations

This REST API is administerDeveloper Portal settings such as SAML, LDAP, OAuth, user account settings, email notification templates and so on.

List of resources

- **GET** /configurations/smtp

  Retrieves the SMTP email server configuration.

- **PUT** /configurations/smtp

  Updates the SMTP email server configuration with the given details.

- **DELETE** /configurations/smtp

  Deletes the SMTP email server configuration.

- **GET** /configurations/password_policy

  Retrieves the password policy settings.

- **PUT** /configurations/password_policy

  Updates the SMTP password policy settings with the given details.

- **DELETE** /configurations/password_policy

  Resets the password policy settings to default.

- **GET** /configurations/saml/metadata

  Retrieves SAML Service provider metadata that can be used to configure Identity provider.

- **GET** /configurations/saml

Retrieves the SAML settings.

- **PUT** /configurations/saml

  Updates the SAML settings with the given details.

- **DELETE** /configurations/saml

  Resets the SAML settings to default.

- **GET** /configurations/mfa

  Retrieves the multi-factor authentication settings.

- **PUT** /configurations/mfa

  Updates the multi-factor authentication settings with the given details.

- **DELETE** /configurations/mfa

  Resets the multi-factor authentication settings to default.

- **GET** /configurations/oauth/{id}

  Retrieves the specified OAuth provider configuration.

- **GET** /configurations/oauth

  Retrieves the OAuth configuration settings.

- **PUT** /configurations/oauth

  Updates the OAuth configuration settings with the given details.

- **DELETE** /configurations/oauth

  Resets the OAuth configuration settings to default.

- **GET** /configurations/lcm

  Retrieves the lifecycle model of the specified asset (API or package).

- **GET** /configurations/ldap_connection/{id}

  Retrieves the specified LDAP configuration settings.

- **PUT** /configurations/ldap_connection

  Updates the LDAP configuration settings with the given details.

- **DELETE** /configurations/ldap_connection

  Deletes the LDAP configuration settings.

- **GET** /configurations/ldap_settings

  Retrieves the LDAP configuration settings.

- **PUT** /configurations/ldap_settings

  Updates the LDAP configuration settings with the given details.

- **DELETE** /configurations/ldap_settings

  Deletes the LDAP configuration settings.

- **POST** /configurations

  Creates a configuration payload with the given details.

- **GET** /configurations/{category}

  Retrieves the configurations payload for the specified category.

- **POST** /configurations/{category}

  Updates the specified configurations payload with the given details.

- **DELETE** /configurations/{category}

  Deletes the specified configurations payload.

- **GET** /configurations/email-templates

  Retrieves the list email notifications templates.

- **GET** /configurations/email-templates/{templateId}

  Retrieves the specified email notifications template.

- **PUT** /configurations/email-templates/{templateId}

  Updates the specified email notifications template with the given details.

- **GET** /configurations/system_locale

  Retrieves the default language that is currently configured.

- **POST** /configurations/system_locale

  Specifies the given language as the default language.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/configurations/smtp' \
--header 'Authorization: Basic basic_auth'
```

The `configurations.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing custom assets

This REST API is used to manage custom assets. You can define an asset type and add assets of the defined type.

List of resources

- **POST** /types

  Creates a custom asset type with the given details.

- **GET** /types/ticket

  Retrieves the details of the specified custom asset type.

- **GET** /types

  Retrieves the list of custom asset types in Developer Portal.

- **POST** /types/search

  Searches for an asset type based on the given search criteria.

- **PUT** /types/ticket

  Updates the specified custom asset type with the given details.

- **PUT** /types/ticket/fields/Issue_details_doc

  Provides the given value as the default value for the specified asset type property.

- **PUT** /types/ticket/fields/Issue_screenshot

  Provides the given image as the default image for the specified asset type property.

- **GET** /types/_count

  Retrieves the number of asset types added in Developer Portal.

- **DELETE** /types/{id}

  Deletes the specified asset type.

  Note:
  To delete an inactive or obsolete asset type, make a REST call to the following endpoint:

  ```
  DELETE  /types/{id}?force=true
  ```

  with the required application Id:

- **POST** /types/ticket/instances

  Creates an asset of the specified asset type.

- **GET** /types/feature/instances/{id}

  Retrieves the details of the specified asset.

- **GET** /types/feature/instances

  Retrieves all assets of the specified asset type.

- **POST** /types/ticket/instances/search

  Retrieves the details of assets matching the given search keyword.

- **PUT** /types/ticket/instances/{id}

  Updates the specified asset with the given details.

- **PUT** /types/ticket/instances/{id}/fields/Issue_details_doc

  Uploads the given file for the specified file field.

- **PUT** /types/ticket/instances/{id}/fields/Issue_screenshot

  Uploads the given image file for the specified image field. .

- **PUT** /types/ticket/instances/{id}/logo

  Uploads the given logo for the specified asset.

- **GET** /types/ticket/instances/_count

  Retrieves the number of assets for the specified asset type.

- **GET** /types/ticket/instances/{id}/topics

  Retrieves the list of topics for the given asset.

- **PUT** /types/ticket/instances/{id}/followers

  Subscribes or unsubscribes to receive the updates of the given asset.

- **GET** /types/ticket/instances/{id}/followers

  Retrieves the list of followers of the specified asset.

- **GET** /types/ticket/instances/{id}/followers/_count

  Retrieves the number of followers of the specified asset.

- **GET** /types/ticket/instances/{id}/bookmarks

  Retrives the list of topics of the specified asset that are saved as bookmarks.

- **DEL** /types/ticket/instances/{id}

  Deletes the specified asset.

- **PUT** /types/{id}/rate

  Allows to provide a rating for the specified asset.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/types' \
--header 'Authorization: Basic basic_auth'
```

The `typedefinition.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Viewing Developer Portal events

This REST API is used to view the following events:

- **Audit events**. Events that involve the lifecycle of an API, Application, Packages, Plans, Community, Provider, Topic, Comment and monitor the subscriptions per package, access token requests per API, Developer Portal system usage and so on.
- **User events**. Events that are created based on your settings on the **Advanced** tab of the **Security** screen. For the list of log events that you can configure, see "How do I configure advanced security settings?" on page 18.

List of resources

- **GET** /events

  Retrieves the list of events based on the specified query parameter.

  Specify the required query parameter value for the variable, **system**:

  - **Core**. To view the events related to APIs and other assets such as packages, communities, and so on.
  - **UMC**. To view the events related to user activities such as user creation, user login, and so on.

- **DELETE** /events

  Deletes the list of events based on the specified query parameter.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/events' \
--header 'Authorization: Basic basic_auth'
```

The `events.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Viewing Developer Portal health

This REST API is used to view the Developer Portal health, build, and metrics.

List of resources

- **GET** /info

  Retrieves the Developer Portal build details.

- **GET** /metrics

  Retrieves the Developer Portal application metrics.

- **GET** /health

  Retrieves the Developer Portal health details.

- **GET** /health/liveness

  Retrieves the Developer Portal health liveness status.

- **GET** /health/readiness

  Retrieves the Developer Portal health readiness status.

- **GET** /env

  Retrieves the Developer Portal environment details.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/info' \
--header 'Authorization: Basic basic_auth'
```

The `health.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing notifications

This REST API is used to view, update, and delete notification preferences of the current user.

List of resources

- **GET** /users/preferences

  Retrieves the notification settings of the currently signed in user.

- **PUT** /users/preferences

  Updates the notification settings of the current user with the given details.

- **GET** /notifications

  Retrieves the paginated notifications for the currently signed in user.

- **PUT** /notifications

  Updates the status of notifications as *Read* or *Unread*.

- **DELETE** /notifications

  Deletes the notification settings of the currently signed in user.

- **GET** /notifications/_count

  Retrieves the number of unread notifications for the current user.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/notifications/' \
--header 'Authorization: Basic basic_auth'
```

The `notifications.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Getting OAuth token

This REST API is used to request access token and to receive authorization code for invoking APIs.

List of resources

- **POST** /oauth/tokens

  Requests token for the specified application using client credential grant.

- **GET** /oauth/callback

  Acts as the callback URL for receiving authorization code.

Sample cURL Command

```
curl --location --request POST 'developer_portal_rest_base/oauth/tokens' \
--header 'Authorization: Basic basic_auth' \
--header 'Content-Type: application/json' \
--data-raw '{
    "name": "cc grant token",
    "scope": "Read",
    "applicationId": "application_id"
}'
```

The `oauth.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing packages

This REST API is used to manage packages. You can view, add, edit, and delete packages. You can also add, update, and delete the APIs and Plans associated with packages.

List of resources

- **GET** /packages

  Retrieves the list of packages in Developer Portal.

- **POST** /packages

  Create a package with the given details.

- **GET** /packages/{id}

  Retrieves the details of the specified package.

- **PUT** /packages/{id}

  Updates the specified package with the given details.
- **DELETE** /packages/{id}

  Deletes the specified package.
- **GET** /packages/{id}/plans

  Retrieves the list of plans for the specified package.
- **PUT** /packages/{id}/plans

  Updates the plans of the specified package with the given details.
- **DELETE** /packages/{id}/plans

  Deletes the given plans from the specified package.
- **GET** /packages/{id}/apis

  Retrieves the list of APIs in the specified package.
- **PUT** /packages/{id}/apis

  Updates the APIs of the specified package with the given details.
- **DELETE** /packages/{id}/apis

  Deletes the given APIs from the specified package.
- **GET** /packages/{id}/communities

  Retrieves the list of communities in the specified package.
- **GET** /packages/{id}/topics

  Retrieves the list of topics in the specified package.
- **GET** /packages/{id}/rate

  Retrieves the rating details of the specified package.
- **PUT** /packages/{id}/rate

  Updates the rating details the specified package with the given details.
- **GET** /packages/{id}/followers

  Retrieves the list of followers of the specified package.
- **PUT** /packages/{id}/followers

  Updates the list of followers in the specified package with the given details.
- **GET** /packages/{id}/followers/_count

Retrieves the number of followers for the specified package.

- **PUT** /packages/{id}/state

  Allows to modify the lifecycle state of the specific package.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/packages' \
--header 'Authorization: Basic basic_auth'
```

The `packages.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing plans

This REST API is used to manage plans. You can view, add, update, and delete the required plans.

List of resources

- **GET** /plans

  Retrieves the list of plans in Developer Portal.

- **POST** /plans

  Creates a plan with the given details.

- **GET** /plans/{id}

  Retrieves the details of the specified plan.

- **PUT** /plans/{id}

  Updates the specified plan with the given details.

- **DELETE** /plans/{id}

  Deletes the specified plan.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/plans' \
--header 'Authorization: Basic basic_auth'
```

The `plans.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing providers

This REST API is used to manage providers. You can view, add, update, and delete providers. You can also view, update and delete the APIs and packages associated with providers.

List of resources

- **GET** /providers

  Retrieves the list of providers in Developer Portal.

- **POST** /providers

  Creates or publishes a provider with the given details.

- **GET** /providers/{id}

  Retrieves the details of the specified provider.

- **PUT** /providers/{id}

  Updates the specified provider with the given details.

- **DELETE** /providers/{id}

  Deletes the specified provider.

- **GET** /providers/{id}/apis

  Retrieves the list of APIs associated with the specified provider.

- **PUT** /providers/{id}/apis

  Updates the specified APIs with the given details.

- **DELETE** /providers/{id}/apis

  Deletes the specified APIs.

- **GET** /providers/{id}/packages

  Retrieves the list of packages associated with the specified provider.

- **PUT** /providers/{id}/packages

  Updates the specified packages with the given details.

- **DELETE** /providers/{id}/packages

  Deletes the specified packages.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/providers' \
--header 'Authorization: Basic basic_auth'
```

The `providers.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Performing search

This REST API is used to search for an asset.

List of resources

- **GET** /search

  Searches for the APIs and packages that include the provided keyword.

- **POST** /search/basic

  Searches based on the given search criteria.

- **GET** /search/advanced

  Searches for assets with provided keyword. Advanced search provides a metric which captures the number of assets matching given criteria in each asset type and also provides a detailed result on specific asset identified by `type`.

Sample cURL Command

```
curl --location --request
GET 'developer_portal_rest_base/search?q=search_keyword' \
--header 'Authorization: Basic basic_auth'
```

The `search.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing teams

This REST API is used to manage teams. You can view, add, update, and delete the required teams. You can also view, update, and delete the users associated with the team.

List of resources

- **GET** /teams

  Retrieves the list of teams in Developer Portal.

- **POST** /teams

  Creates a team with the given details.

- **GET** /teams/{id}

  Retrieves the details of the specified team.

- **PUT** /teams/{id}

  Updates the specified team with the given details.

- **DELETE** /teams/{id}

Deletes the specified team.

- **GET** /teams/{id}/users

  Retrieves the list of users associated with the specified team.

- **PUT** /teams/{id}/users

  Updates the specified users associated with the specified team.

- **DELETE** /teams/{id}/users

  Updates the specified users associated with the specified team.

- **GET** /teams/{id}/applications

  Retrieves the list of applications associated with the specified team.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/teams' \
--header 'Authorization: Basic basic_auth'
```

The `teams.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing topics

This REST API is used to manage topics. You can add topics for required APIs and packages. You can also upvote, downvote, flag, or bookmark a topic.

List of resources

- **POST** /apis/{id}/topics

  Creates a topic for the given API.

- **POST** /packages/{id}/topics

  Creates a topic for the given package.

- **GET** /topics/{id}

  Retrieves the details of the given topic.

- **PUT** /topics/{id}

  Updates the specified topic with the given details.

- **DELETE** /topics/{id}

  Deletes the specified topic.

- **PUT** /topics/{id}/upvote

  Upvotes the specified topic.

- **PUT** /topics/{id}/downvote

  Downvotes the specified topic.

- **PUT** /topics/{id}/flag

  Adds or removes flag from the specified topic.

- **PUT** /topics/{id}/pin

  Adds or removes pin from the specified topic.

- **PUT** /topics/{id}/bookmarks

  Adds or removes bookmark from the specified topic.

- **GET** /topics/{id}/upvote/_count

  Retrieves the number of upvotes for the specified topic.

- **GET** /topics/{id}/downvote/_count

  Retrieves the number of downvotes for the specified topic.

- **GET** /topics/{id}/flag/_count

  Retrieves the number of flags for the specified topic.

- **GET** /topics/{id}/_count

  Retrieves the number of upvotes, downvotes, and flags for the specified topic.

- **GET** /collaboration/flags

  Retrieves the number of topics and comments that are flagged.

- **GET** /collaboration/flagged

  Retrieves the number of topics and comments that are flagged by administrators.

Sample cURL Command

```
curl --location --request
GET 'developer_portal_rest_base/topics/59d97968-7b25-4964-a0b7-d783ba910db2' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic basic_auth' \
--data-raw ''
```

The `topic.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing application and subscription requests

This REST API is used to manage application and subscription requests. You can view and retry application and subscription requests.

List of resources

- **GET** /requests

  Retrieves the list of requests that belong to the specified type.

- **POST** /requests

  Creates an application or subscription request with given details.

- **GET** /requests/pending

  Retrieves the list of pending requests that belong to the specified type.

- **GET** /requests/{id}

  Retrieves the specified request.

- **DELETE** /requests/{id}

  Deletes the specified request.

- **PUT** /requests/{id}/retry

  Retries the pending request for approval.

- **GET** /requests/{id}/trace

  Retrieves the log trace of the specified request.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/requests' \
--header 'Authorization: Basic basic_auth'
```

The `userequests.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing users

This REST API is used to manage users and user groups. You can view, add, update, and delete users and their details.

List of resources

- **POST** /login

  Signs in to Developer Portal with the provided credentials. This API supports the Native or LDAP users login only.

- **POST** /forgotpassword

  Allows to recover a forgotten password via the registered email.

- **POST** /resetpassword

Allows to reset your password via the registered email.

- **GET** /passwordpolicy

  Retrieves the details configured password policy.

- **POST** /signup

  Allows to sign up to Developer Portal.

- **PUT** /users/updatepassword

  Updates the password with the provided value.

- **PUT** /users/updateemail

  Updates the email with the provided value.

- **PUT** /users/updatepicture

  Updates the profile picture with the provided image.

- **GET** /users

  Retrieves the list of users.

- **POST** /users

  Creates a new user with the given details.

- **PUT** /users/anonymize

  Anonymizes the specified user accounts that are deleted.

- **GET** /users/self

  Retrieves the details of the currently signed in user.

- **GET** /users/self/privileges

  Retrieves the privileges of the currently signed in user.

- **GET** /users/{id}

  Retrieves the details of specified user.

- **PUT** /users/{id}

  Updates the user name such as the first name, last name, email of specific user.

- **DELETE** /users/{id}

  Deletes the specified user.

- **POST** /users/delete

  Deletes the account of current user.

- **GET** /groups/{id}/users

  Retrieves the users of the specified group.

- **PUT** /groups/{id}/users

  Updates the users of the specified group with the given details.

- **DELETE** /groups/{id}/users

  Deletes the given users of the specified group.

- **GET** /groups

  Retrieves the list of groups.

- **POST** /groups

  Creates a group with the given details.

- **GET** /groups/{id}

  Retrieves the details of the specified group.

- **POST** /groups/{id}

  Updates the specified group with the given details.

- **DELETE** /groups/{id}

  Deletes the specific group.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/users/' \
--header 'Authorization: Basic basic_auth'
```

The `users.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

Managing webhooks

This REST API is used to manage webhooks. You can view, add, update, and delete webhooks.

List of resources

- **GET** /hooks

  Retrieves the list of webhooks.

- **POST** /hooks

  Creates a webhook with the given details.

- **GET** /hooks/{id}

Retrieves the details of the specified webhook.

- **POST** /hooks/{id}

  Updates the specified webhook with the given details.

- **DELETE** /hooks/{id}

  Deletes the specific webhook.

- **GET** /hooks/events

  Retrieve the list of events to which a webhook can be created.

Sample cURL Command

```
curl --location --request GET 'developer_portal_rest_base/hooks/' \
--header 'Authorization: Basic basic_auth'
```

The `webhooks.yaml` file is located at *`SAGInstallDir`*`\DeveloperPortal\developers\openapis`.

9 Docker support

Docker support

Docker is an open-source technology that allows users to deploy applications to software containers. A Docker container is an instance of a Docker image, where the Docker image is the application, including the file system and runtime parameters.

You can create a Docker image from an installed and configured Developer Portal instance and then run the Docker image as a Docker container.

To facilitate running Developer Portal in a Docker container, Developer Portal provides a script to build a Docker image and then load or push the resulting Docker image to a Docker registry.

Generating Docker Image from Installation

Using the **generateDockerfile** script file, you can generate Docker image from Developer Portal installation.

You can generate a Docker image for the operating system of your preference.

Prerequisites

Prior to building a Docker image for Developer Portal, you must complete the following:

- Install Docker client on the machine on which you are going to install Developer Portal and start Docker as a daemon. The Docker client should have connectivity to Docker server to create images.
- Install Developer Portal packages, and fixes on a Linux or UNIX system using the instructions in Installing Software AG Products, and then configure Developer Portal and the hosted products.

To generate a Docker file and image

1. Run the following command with the required parameters. You can provide any or all parameters depending on your required:

   ```
   sudo sh generateDockerfile
   ```

   Based on your requirement, you can provide any of the parameters listed in the **List of parameters that can be provided with the generateDockerfile script** section with the above command.

   Sample command

   ```
   sudo sh generateDockerfile --os.image ubuntu:20.04
   --installation.path fs/fslocal/Portal1015
   ```

   You can run this command with the **--build** parameter to build the Docker image in one step or generate a Docker file and then generate the Docker image.

   Tip:

Run the **generateDockerfile** command with the parameter, `--h` to view the list of parameters that you can provide with the command:

```
sudo sh generateDockerfile --h
```

2. Run the following command to generate a Docker image from the Docker file:

```
docker build -t image_name
```

Sample command

```
docker build -t dpo_image
```

List of parameters that can be provided with the generateDockerfile script

Developer Portal offers the following parameters that let you provide your preferences for generating Docker file and image:

Parameter	Description	Default value
--extern.ES	Specifies whether the generated Docker image should point an external Elasticsearch. Possible values are: ■ `true`. Uses an external Elasticsearch. ■ `false`. Uses the Elasticsearch that comes with Developer Portal installation.	■ false
--os.image	Specifies the base operating system of the Docker image. **Note:** Ensure that the operating system name and version provided for this parameter matches with the name and version of the corresponding operating system (tag) in the Docker hub.	■ centos:7
--build	Specifies whether a Docker image needs to be generated. Possible values are: ■ `true`. Docker image is generated when you run the command. ■ `false`. Docker image is not generated.	■ false

Parameter	Description	Default value
--installation.path	Specifies the Developer Portal installed directory.	/opt/SoftwareAG
--extern.ES.url	Specifies the external Elasticsearch URL. This is mandatory if your Docker image has to point to an external Elasticsearch.	
--docker.image.name	Specifies the name for the Docker image. You need to provide this value if you are generating a Docker image. If you provide the **--build** parameter and do not provide this parameter, then the generated the default value `wm_developer_portal` is given as name the of the generated Docker image.	wm_developer_portal
--keep.ES.data	Specifies whether the data in the Elasticsearch of the base installation must be included in the generated Docker file.	false

Running the Developer Portal Docker Image

To run the Docker image

1. Run the following command to start the Developer Portal Docker image:

   ```
   docker run -p host_numer:port_numer image_name
   ```

   Sample commad

   ```
   docker run -p 18101:18101 dpo_final
   ```

2. Run the following command to check the status of the Docker container:

   ```
   docker ps
   ```

   Sample output (Docker image created with internal Elasticsearch)

   ```
   CONTAINER ID   IMAGE
   COMMAND                  CREATED        STATUS
   PORTS
                         NAMES
   dd5f0c541d25   softwareag/devportal:10.11.0.20                          "/bin/sh -c
    /usr/loc…"   25 hours ago   Up 25 hours
   0.0.0.0:80->8083/tcp, :::80->8083/tcp   unruffled_ardinghelli
   87ad8248dbd7   docker.elastic.co/elasticsearch/elasticsearch:7.16.3   "/bin/tini
   -- /usr/l…"   26 hours ago   Up 26 hours
   9200/tcp, 9300/tcp                     serene_babbage
   ```

```
[sv@daeyapcent01 ~]$
```

10 API Programs

About API Programs

API providers and partners can organize the following programs for their consumers using the API programs feature:

- **Hackathons**. You can create hackathons and expose them to the required communities.
- **Beta programs**. You can create a beta program with selected APIs for beta testing and seek feedback from the participants.

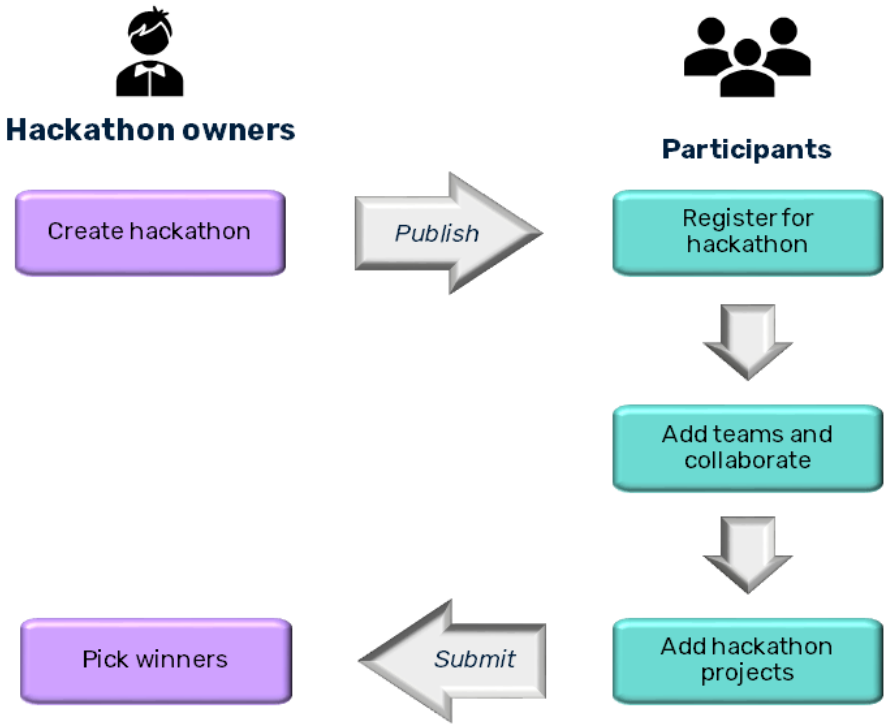
The following sections explain the feature in detail.

Hackathons

Organizations conduct hackathons to widen the reach of their APIs to a network of skilled individuals, fostering future collaborations and opportunities within the tech community. Hackathons also promote innovation by bringing together diverse talents to solve problems creatively and swiftly.

Developer Portal allows you to create hackathons by including the set of required APIs. You can expose your hackthons to the required audience.

The following image outlines the hackathon workflow:



Important stages in a hackathon lifecycle include:

- **Hackathon creation**. As an API provider or partner, you have the ability to initiate hackathons by furnishing essential details such as the hackathon duration, participant APIs, and

corresponding rewards. Organizers can define a theme or problem statement to challenge participants, shaping the context of the hackathon. The program is then featured on the **API programs** page, allowing relevant users to access and engage with it throughout the hackathon period. For information about creating hackathons, see "Creating a hackathon" on page 279.

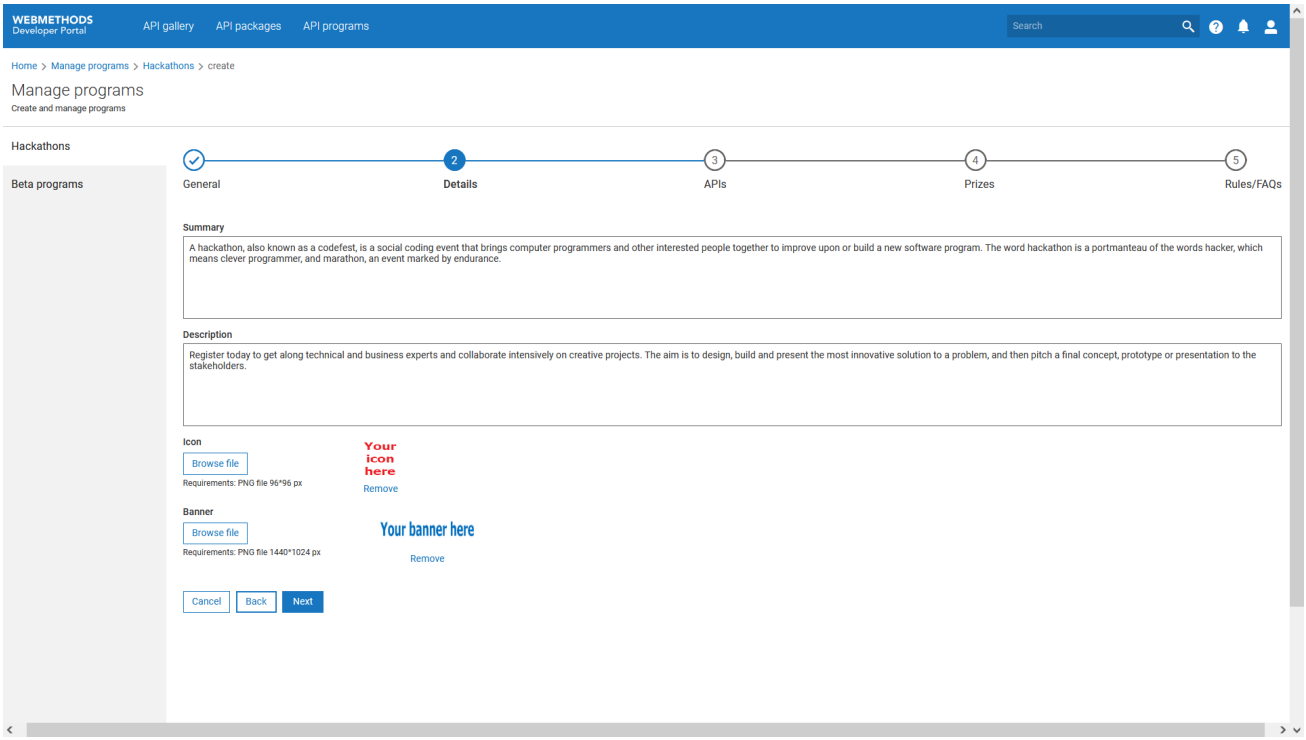
- **Participants registration**. Participants can register for the programs that are available to them through the **API programs** page. Participants can create teams of required users who are onboard and work on developing their hackathon projects. For information about registering for a hackathon, see "Registering for a hackathon" on page 285.
- **Hackathon project submission**. On completion of their projects, participants can submit them to the hackathon organizers, providing essential project details. For information about submitting a hackathon project, see "Adding a hackathon project" on page 287.
- **Winner selection and conclusion**. Hackathon organizers can view projects submitted by participants and pick winners.

Creating a hackathon

You can create a hackathon with your specifications from the **Hackathons** tab under the **Manage programs** page.

To create a hackathon

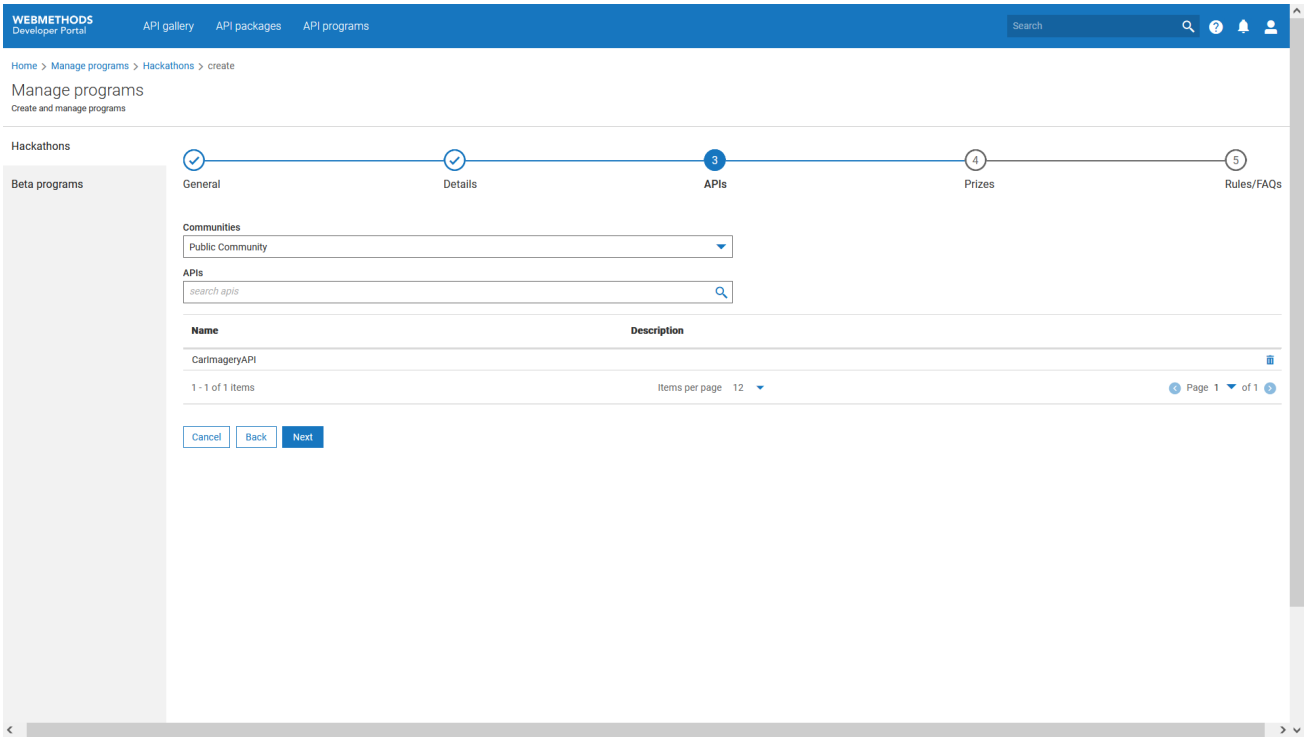
1. Click the menu options icon  from the title bar and click **Manage programs**.
2. Click **Create hackathon**.
3. Provide a name for the hackathon.
4. Select the **Start date**, **Time** and the **End date**, **Time** from the corresponding fields.
5. Click **Next** tab.
6. Provide the **Summary** and **Description** of the hackathon in the corresponding fields.
7. Select the required **Icon** and **Banner** for the hackathon by clicking **Browse file** buttons next to the corresponding fields.



8. Click **Next**.
9. Select the required **Community** from the list.

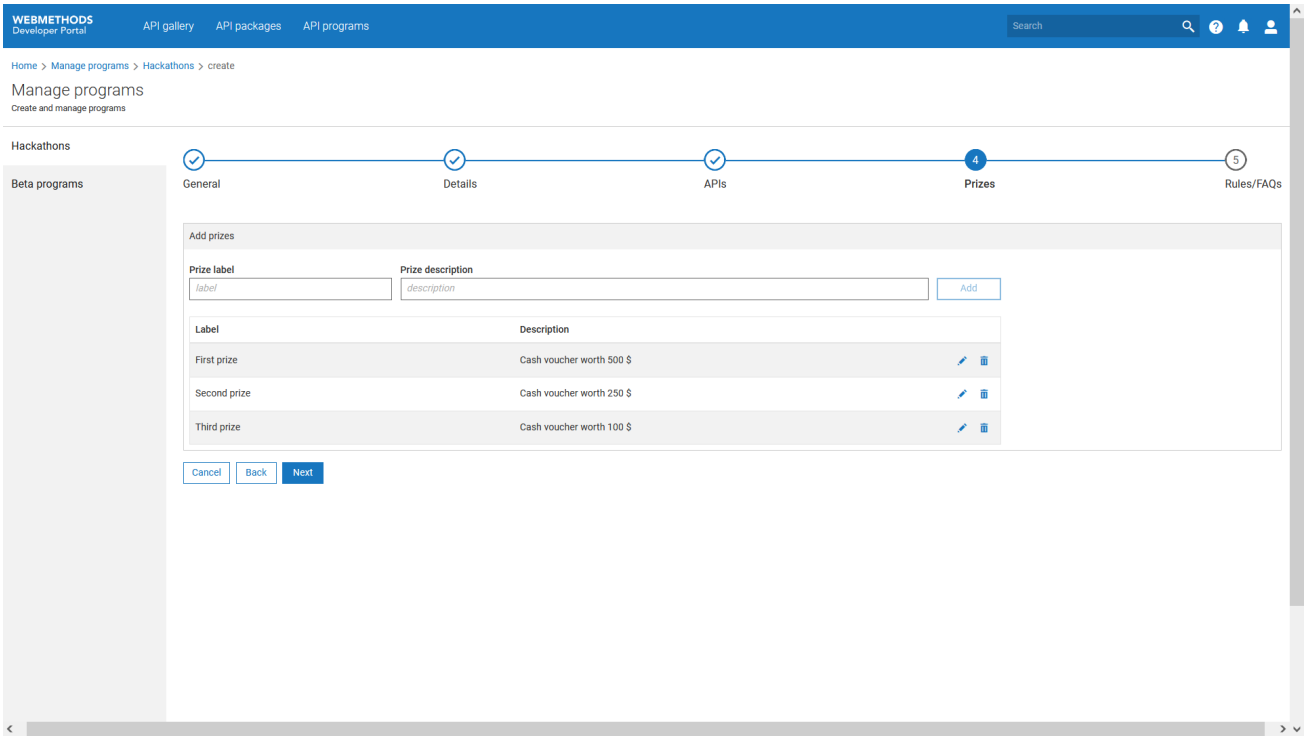
   Only the users from the selected community can view the hackathon. If you do not select a community, the **Public community** is selected by default.
10. Select the required **APIs** for the hackathon.

    The APIs displayed are based on the community you have selected. You must select at least one API.

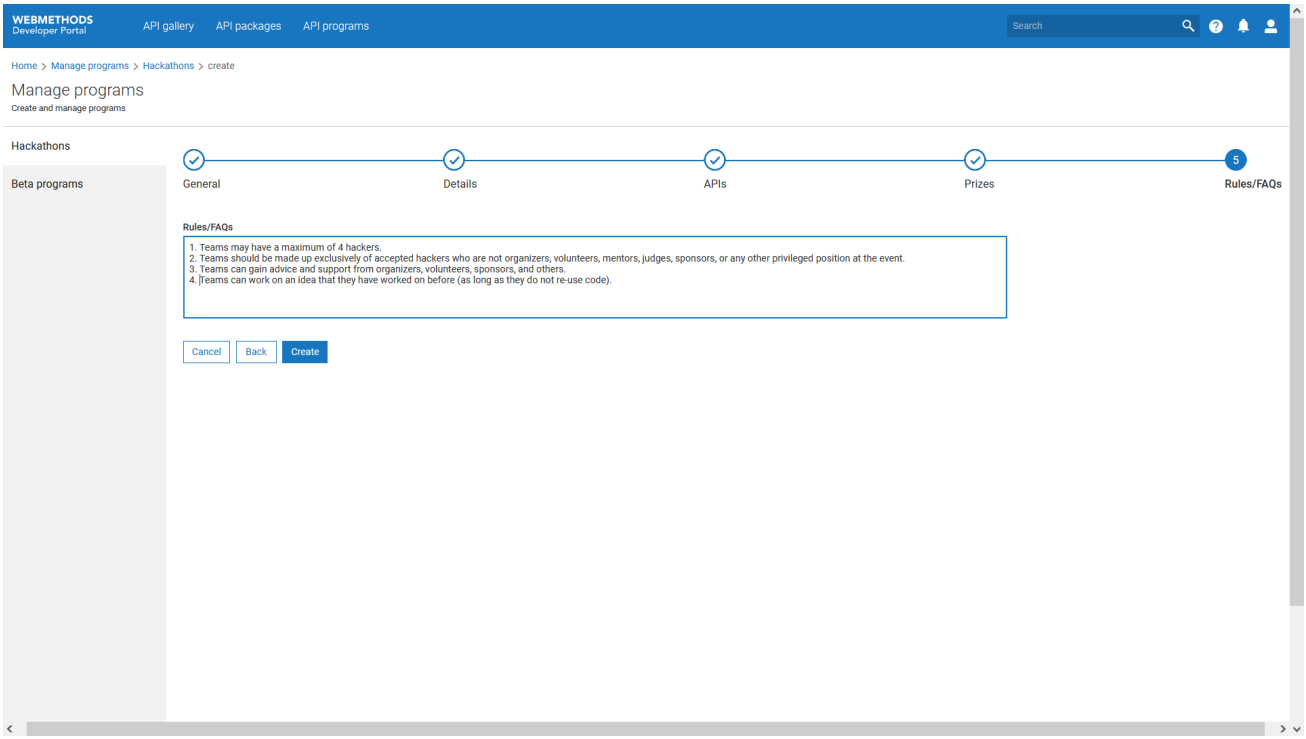


11. Click **Next**.
12. Provide the required **Prize label** and **Prize description** and click **Add**.

    You can repeat this step to add the required number of prizes.



13. Click **Next**.
14. Provide the required **Rules/ FAQ** of the hackathon.



15. Click **Create**.

    Hackathon is created.

Next steps:

- You, as an administrator, API provider, or API partner, can
  - View all details of a hackathon such as the Overview, APIs included in the hackathon, Prizes associated with the hackathon, Participants who have registered for the hackathon, and projects submitted by the participants.
  - Add more hackathons by clicking **Add hackathon** from the **Hackathons** section under the **Mange programs** page.
  - Invite participants to take part in a hackathon. For more details, see "Inviting participants for an API program" on page 283.
  - Include users as owners to co-organize a hackathon. For more details, see "Assigning owners for an API program" on page 284.
  - Click the edit icon next to a hackathon to edit the hackathon details. You can only edit the details of upcoming hackathons and not the ongoing and completed hackathons.
  - Click the delete icon next to a hackathon to delete the hackathon

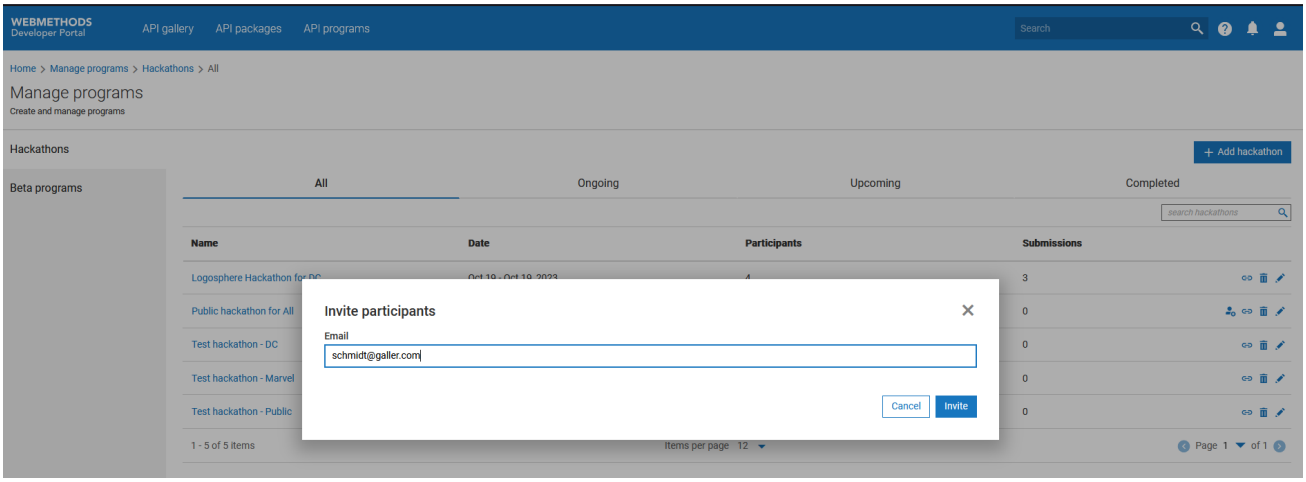
- Other eligible users can
  - View the ongoing and upcoming hackathons from the **Hackathons** section of the **API programs** page and register for them.
  - Submit their hackathon projects. For more information, see "Adding a hackathon project" on page 287.
  - Add team members to collaborate on their hackathon projects. For more information, see "Adding team members for a hackathon" on page 286

Inviting participants for an API program

As an administrator, API provider, or API partner, you can invite participants to work on an API program (Hackathon or Beta program).

To invite a participant

1. Perform one of the following:
   - Click the menu options icon from the title bar, and click **Manage programs**. Click the invite participants icon next to the required API program from the required tab, **Hackathons** or **Beta programs**
   - Click **API programs**, and select the required API program (Hackathon or Beta program). Click **Invite participants** from the **Participants** section of the program details page.
2. Provide the e-mail addresses of the participants you want to invite.



3. Click **Invite**.

   A mail is sent to the participants inviting them to use Developer Portal. The mail includes a link that the users can click to sign up to Developer Portal.

Next steps:

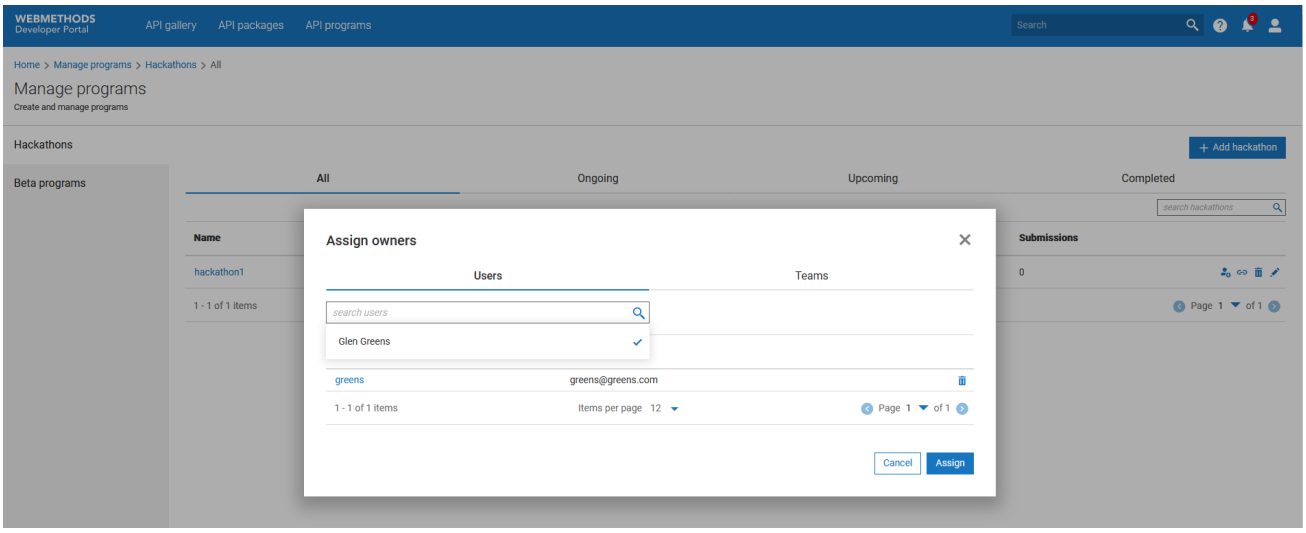
- Users who received the invite mail can click the link and follow the instructions to sign up to Developer Portal. The users thus sign up are automatically registered for the API program to which they were invited.
- Users who sign up using the mail invite are assigned with the *Consumer* role, by default.
- Newly signed up users can participate in the API programs that they were invited; and use other features of the application.
- Users invited by administrators using this feature do not undergo the onboarding approval process, if any. In addition, if you have enabled **Email verification** as a part of your onboarding process, it is ignored for users who sign up using the mail invite. For information about onboarding strategy and email verification, see "Onboarding Strategy" on page 119.

Assigning owners for an API program

You can add other API providers or partners as co-owners for your API programs (Hackathon or Beta program).

To assign owners

1. Click the menu options icon from the title bar and click **Manage programs**.
2. Select the required tab, **Hackathons** or **Beta programs**.
3. Click the assign owners icon next to the required hackathon or beta program.
4. Select the required users or teams.



5. Click **Assign**.

   The newly assigned owners can view and manage their API programs.

Registering for a hackathon

You, as a consumer, can view the hackathons available to you from the **Hackathons** section under the **API programs** page. You can view their details and register for them.

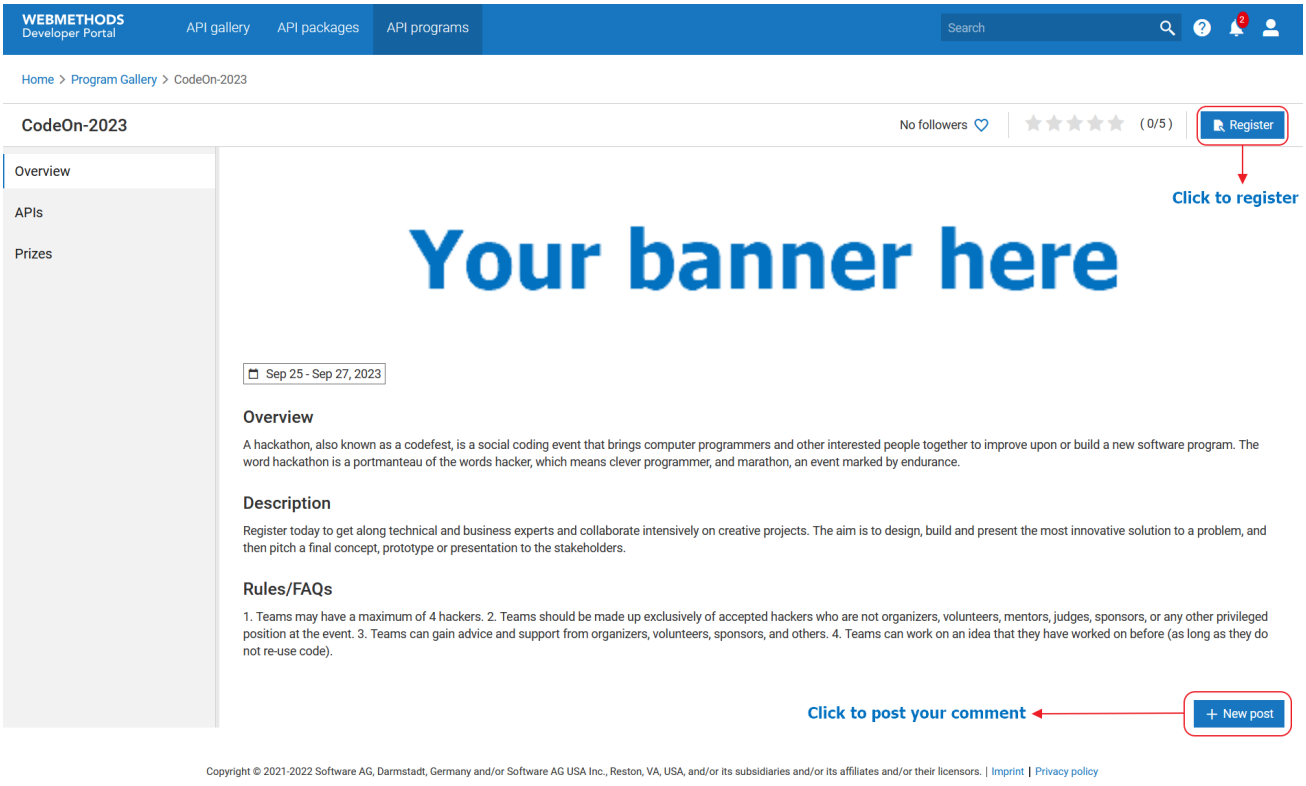
You can also create posts and share your suggestions and feedback about the API programs and collaborate with other consumers of similar interests.

To view and register for a hackathon

1. Click **API programs** from the title bar.

   The **Hackathons** page appears. This page lists the ongoing and upcoming hackathons.

2. Click the view icon in a tile to view the details of the hackathon.



3. Click **Register** to register for the hackathon.

   You are registered for the hackathon and a mail is sent to the registered e-mail address.

Next steps:

- You, as an registered participant, can
  - View hackathon details.

- Add team members to participate with you in the hackathon. For more information, see "Adding team members for a hackathon" on page 286.
- Create your hackathon projects. For more information, see "Adding a hackathon project" on page 287.

Adding team members for a hackathon

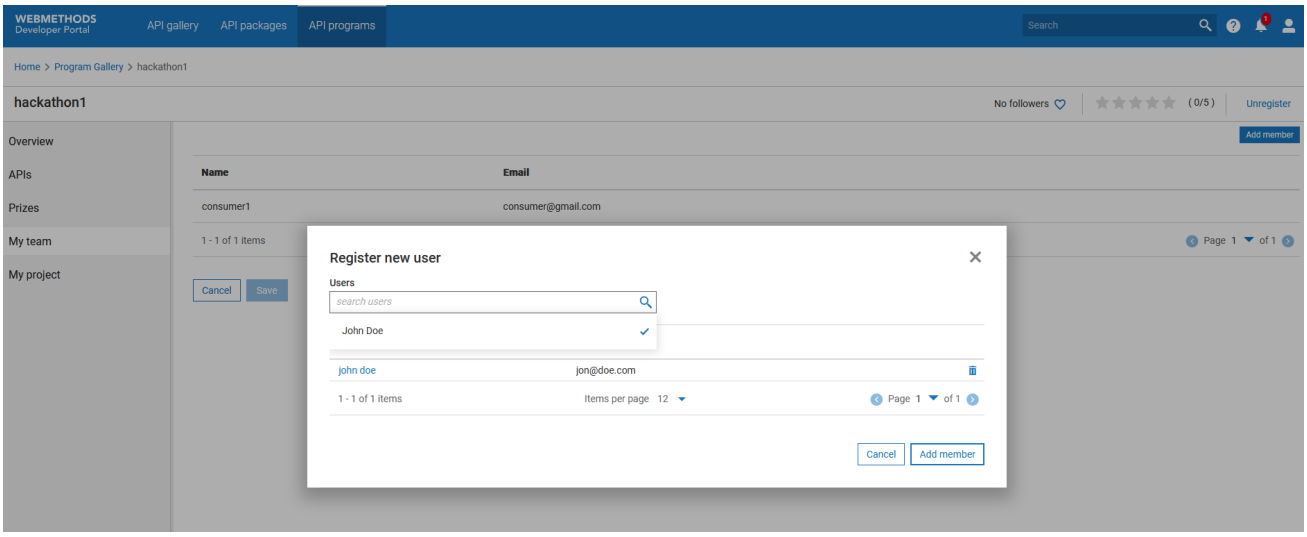
After you, as a consumer, register for a hackathon, you can add other consumers as team members to work with you.

To add team members for a hackathon

1. Click **API programs** from the title bar.

   The **Hackathons** page appears. This page lists the ongoing and upcoming hackathons.

2. Click the view icon in a tile to view the details of the hackathon.
3. Click **My teams**.
4. Click **Add member**.
5. Select the required user from **Users** field.
6. Click **Add member**.



The selected user is added to your team.

Next steps:

- Repeat *Step 6* to add required number of team members.
- The newly added members can

- View the hackathon from the **API programs** page.
- Can add more team members.
- Submit hackathon projects to hackathon owners. For more information, see "Adding a hackathon project" on page 287.

Adding a hackathon project

After you, as a consumer, register for a hackathon, and start working for it, you can add your project details and submit it to the hackathon owner to view.

To add a hackathon project

1. Click **API programs** from the title bar.
2. Click the view icon  next to the required hackathon.
3. Click **My projects**.
4. Click **Add project**.
5. Provide your project details in **General**, **Details**, and **Links** tabs and click **Create**.

   Your project details are saved.

Next steps:

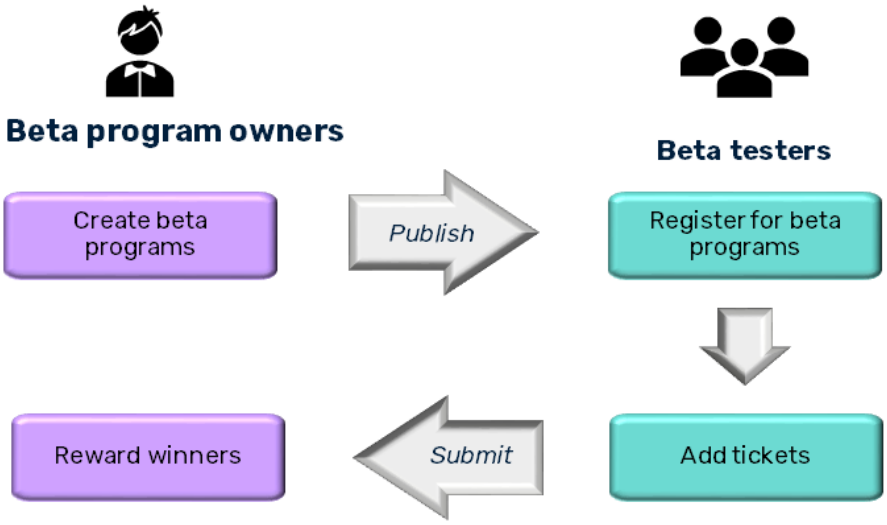
- You can add any number of projects.
- Click the edit icon  next to a project in the **My projects** page to edit the project details.
- Click the submit icon  to submit your project to the hackathon owner.
- Click the delete icon  next to a project to delete the project.

Beta programs

Organizations conduct beta programs to test their APIs with a smaller audience before they launch those APIs to a larger audience.

The beta programs feature in Developer Portal allows you to select the required APIs for beta testing and seek feedback from the participants. As an API provider, you can amend the APIs based on the feedback received from your beta testers and reward them accordingly. Thus, this helps you to improvise your APIs for better performance and reach.

The following image outlines the beta program workflow:

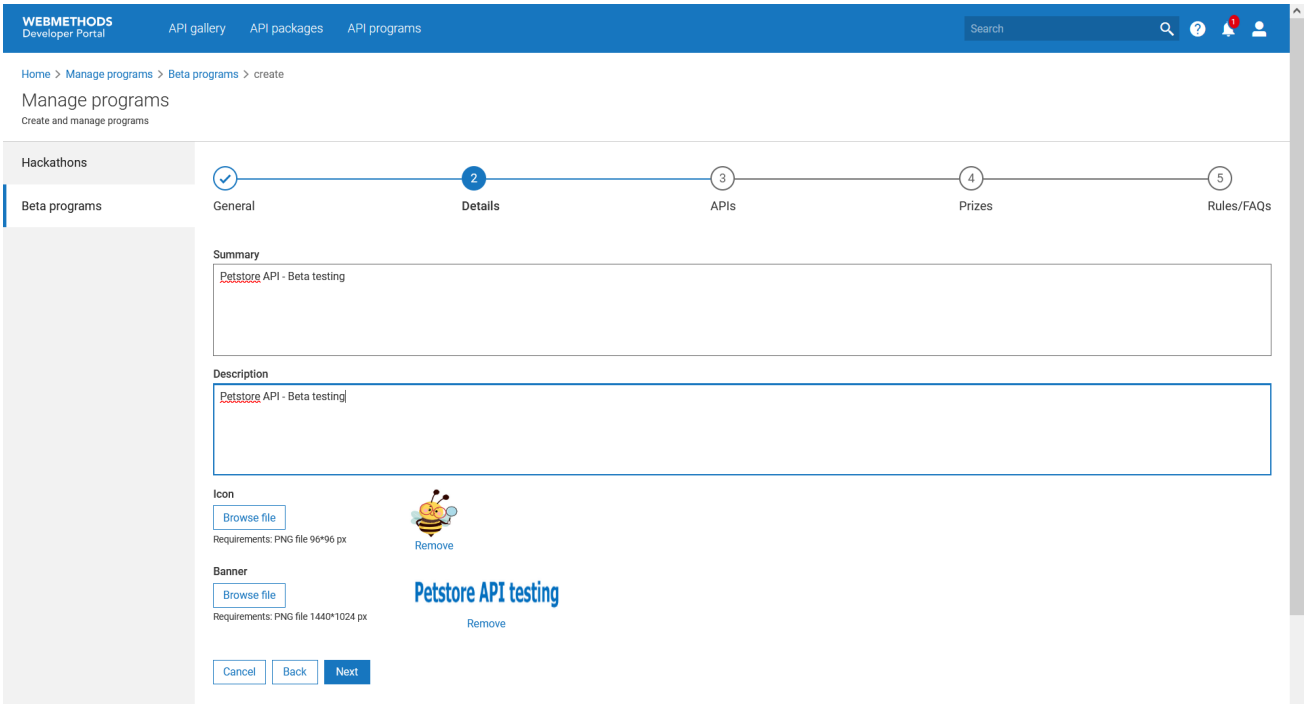


Creating a beta program

You can create a beta program with your specifications from the **Beta programs** tab under the **API programs** page.

To create a beta program

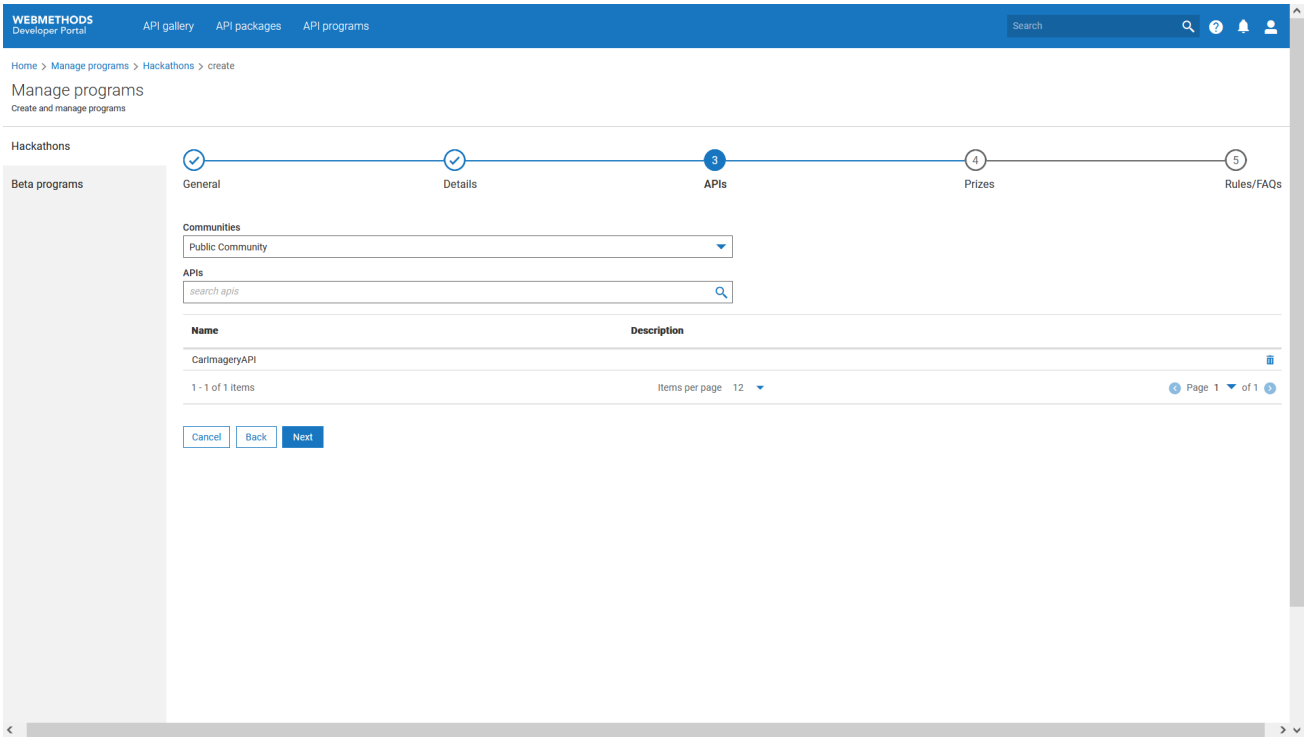
1. Click the menu options icon from the title bar and click **Manage programs**.
2. Click **Beta programs**.
3. Click **Create Beta program**.
4. Provide a name for the program.
5. Select the **Start date**, **Time** and the **End date**, **Time** from the corresponding fields.
6. Click **Next**.
7. Provide the **Summary** and **Description** of the beta program in the corresponding fields.
8. Select the required **Icon** and **Banner** for the beta program by clicking **Browse file** buttons next to those corresponding fields.



9. Click **Next**.
10. Select the required **Community** from the list.

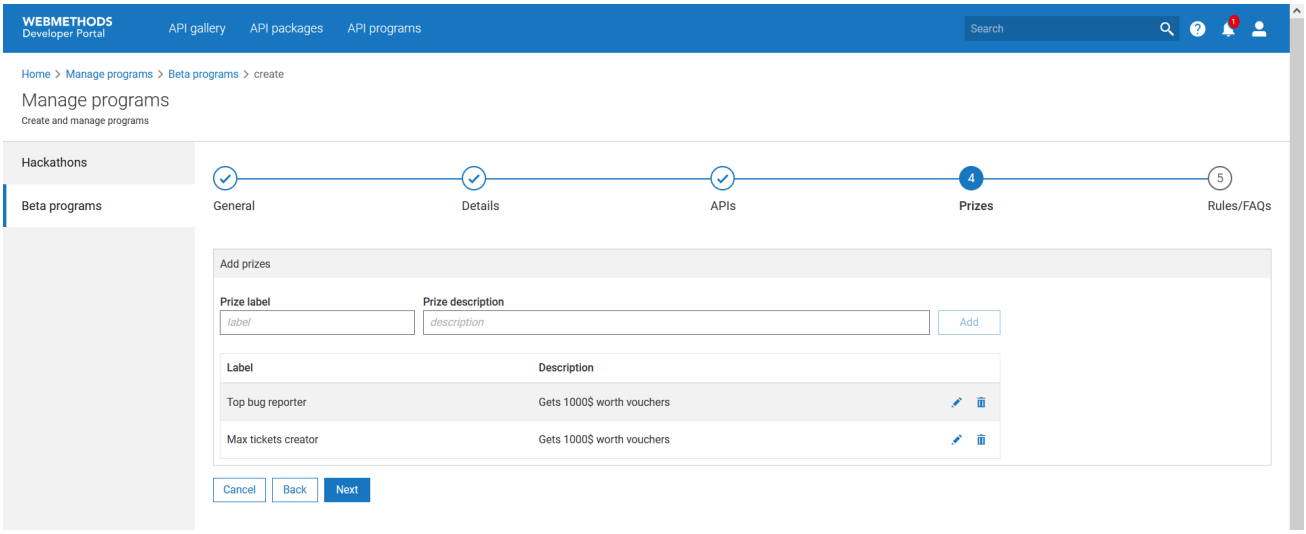
    Only the users from the selected community can view the beta program. If you do not select a community, the **Public community** is selected by default.
11. Select the required **APIs** for the beta program.

    The APIs displayed are based on the community you have selected. You must select at least one API.

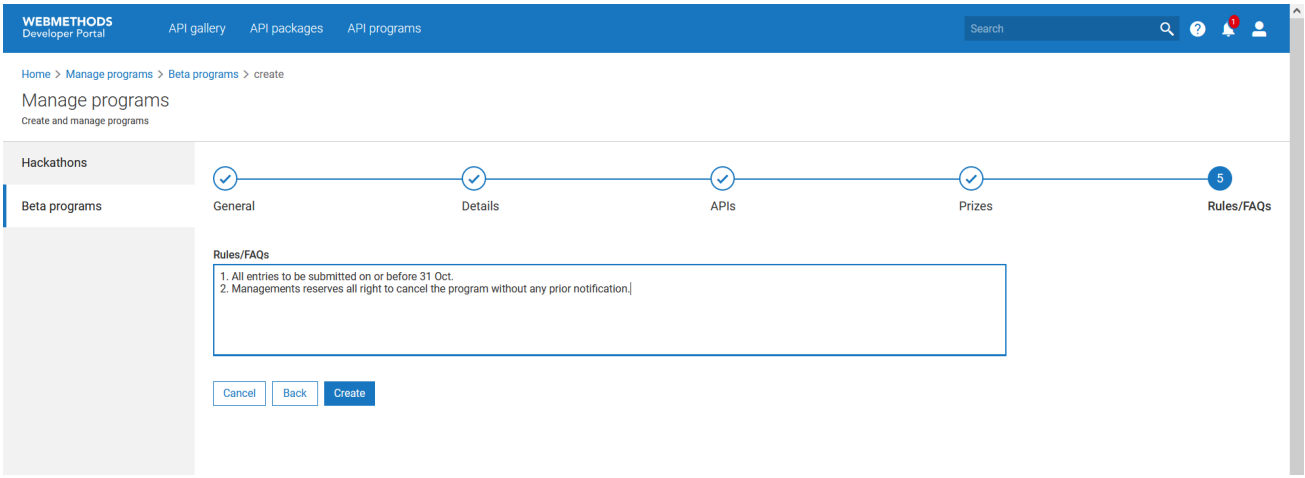


12. Click **Next**.
13. Provide the required **Prize label** and **Prize description** and click **Add**.

    You can repeat this to required number of prizes.



14. Click **Next**.
15. Provide the required **Rules/ FAQ** of the beta program.



16. Click **Create**.

    Beta program is created.

Next steps:

- You, as an administrator, API provider, API partner, can
  - View all details of a beta program such as the overview, APIs included in the program, prizes associated with the program, participants who have registered for the program, and tickets created by the participants.
  - Add more programs by clicking **Add beta program** from the **Beta programs** section under the **Manage programs** page.
  - Invite participants to take part in an API program. For more details, see "Inviting participants for an API program" on page 283.
  - Include users as owners to co-organize a program. For more details, see "Assigning owners for an API program" on page 284.
  - Click the edit icon next to a beta program to edit its details. You can only edit the details of upcoming programs and not the ongoing and completed ones.
  - Click the delete icon next to a program to delete the program.
- Other eligible users can view the ongoing and upcoming programs from the **Beta programs** section of the **API programs** page and register for them. For more information, see "Registering for a beta program" on page 291.

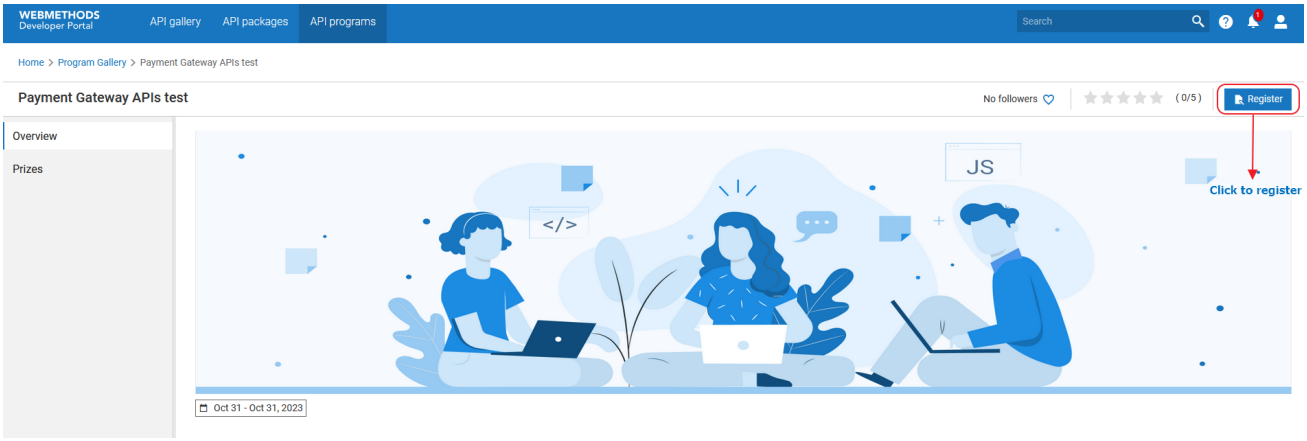
Registering for a beta program

You, as a consumer, can view the beta programs available to you from the **Beta programs** section under the **API programs** page. You can view their details and register for them. You can also create posts and share your suggestions and feedback about the API programs and collaborate with other consumers of similar interests.

To view and register for a beta program

1. Click **API programs** from the title bar.
2. Click **Beta programs**.

   The **Beta programs** page appears. This page lists the ongoing and upcoming beta programs.
3. Click the view icon in a tile to view the details of the beta program.



4. Click **Register** to register for the beta program.

   You are registered for the beta program and a mail is sent to your registered e-mail address.

Next steps:

- You, as a registered participant, can
  - View beta program details.
  - Create test tickets. For more information, see "Adding ticket for a beta program" on page 292.

Adding ticket for a beta program

After you register for a beta program, and start testing the participant APIs, you can add your tickets and submit them for the program owners to view.

To add a ticket

1. Click **API programs** from the title bar.
2. Click **Beta programs**.
3. Click the view icon next to the required beta program.
4. Click **Tickets**.

5. Click **Add ticket**.
6. Provide a **Title** for your ticket.
7. Select the type of ticket. The available options are:
   - Bug
   - Feature
   - Enhancement
8. Select the **APIs** for which you want to create the ticket.
9. Provide the **Summary** and **Description** about the ticket.
10. Provide any required attachments by clicking **Browse**.
11. Click **Create**.

    Your ticket details are saved.

Next steps:

- You can add any number of tickets.
- Once you create a ticket, the status of the ticket is set to *Open*.
- Click the change status icon next to a ticket to change the status of the ticket. Only the API partner who created the program, an administrator, or an API provider can change the ticket status. The available statuses are *Fixed*, *Accepted*, and *Closed*.
- Click the edit icon next to a ticket in the **Tickets** page to edit the ticket details.
- Based on the nature of your ticket, an administrator, a provider, or the API partner who created the program can modify the ticket type. For example, if you have selected `Bug` as your ticket type but the ticket suggests an enhancement, then the program owner can modify the ticket type as `Enhancement`.
- Click the change type icon next to a ticket to change the type of the ticket.