



# Red Hat 3scale API Management 2.14

## Creating the Developer Portal

A good developer portal is a must have to assure adoption of your API. Create yours in no time.



## Red Hat 3scale API Management 2.14 Creating the Developer Portal

---

A good developer portal is a must have to assure adoption of your API. Create yours in no time.

## Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

This guide documents the developer portal on Red Hat 3scale API Management 2.14.

## Table of Contents

<b>PROVIDING FEEDBACK ON RED HAT DOCUMENTATION</b> .....	<b>4</b>
<b>CHAPTER 1. OVERVIEW OF CREATING YOUR DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS</b> .....	<b>5</b>
1.1. TOURING THE 3SCALE API MANAGEMENT EDITING ENVIRONMENT FOR CREATING YOUR DEVELOPER PORTAL	5
1.2. MODIFYING THE 3SCALE API MANAGEMENT NATIVE DEVELOPER PORTAL	7
1.3. DESCRIPTIONS OF ADDITIONAL 3SCALE API MANAGEMENT FEATURES FOR MODIFYING THE NATIVE DEVELOPER PORTAL	9
1.4. HOW 3SCALE API MANAGEMENT LAYOUTS AND PARTIALS REUSE DEVELOPER PORTAL CONTENT	12
1.5. ADDING IMAGES AND OTHER ASSETS TO YOUR DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS	12
1.6. CONSIDERATIONS FOR CUSTOMIZING THE NATIVE DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS	13
1.7. REQUIREMENTS FOR ALLOWING ACCESS TO YOUR DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS	14
1.8. OPTIONAL STEPS FOR CUSTOMIZING YOUR DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS	16
1.9. UPDATING DEVELOPER PORTALS USED IN 3SCALE API MANAGEMENT RELEASES EARLIER THAN 2.8	17
<b>CHAPTER 2. CUSTOM SIGNUP FORM FIELDS</b> .....	<b>20</b>
<b>CHAPTER 3. CONFIGURING SIGNUP FLOWS</b> .....	<b>26</b>
3.1. REMOVING ALL APPROVAL STEPS	26
3.2. ENABLING ALL POSSIBLE DEFAULT PLANS	27
3.3. TESTING THE WORKFLOW	28
<b>CHAPTER 4. MULTI-SERVICE SIGNUP</b> .....	<b>29</b>
4.1. PREREQUISITES	29
4.2. INTRODUCTION	29
4.3. MULTI-SERVICE SIGNUP	29
4.3.1. Retrieving information about services	29
4.3.2. Configuring the signup columns	30
4.3.3. Configuring the subscription	30
4.3.4. Styling	30
<b>CHAPTER 5. DEVELOPER PORTAL AUTHENTICATION</b> .....	<b>32</b>
5.1. ENABLING AND DISABLING USERNAME/EMAIL AND PASSWORD	33
5.2. ENABLING AND DISABLING AUTHENTICATION VIA GITHUB	33
5.3. ENABLING AND DISABLING AUTHENTICATION VIA AUTH0	34
5.4. ENABLING AND DISABLING AUTHENTICATION THROUGH RED HAT SINGLE SIGN-ON	35
5.4.1. Before You Begin	35
5.4.2. Configuring RH SSO to authenticate the Developer Portal	35
5.4.3. Configuring 3scale API Management to authenticate the Developer Portal	39
<b>CHAPTER 6. RED HAT SINGLE SIGN ON FOR DEVELOPER PORTAL</b> .....	<b>41</b>
6.1. CREATING USERS IN THE 3SCALE API MANAGEMENT PLATFORM	41
6.2. REQUESTING A LOGIN LINK	41
6.3. REDIRECTING USERS WITH AUTOMATIC LOGIN	41
<b>CHAPTER 7. RESTRICTED CONTENT</b> .....	<b>43</b>
7.1. RESTRICTED PAGES	43
7.2. RESTRICTED BLOCKS OF CONTENT	44
7.3. AUTOMATING THE CONFIGURATION OF EXTRA FIELDS	44
7.4. REQUIRING USER LOGIN	45

<b>CHAPTER 8. EMAIL TEMPLATES</b> .....	<b>46</b>
8.1. CUSTOMIZING EMAIL TEMPLATES	46
8.1.1. Define your workflows before email configuration	46
8.1.2. Test your workflow and identify active email templates	46
8.1.3. Edit and save your custom template	46
8.1.4. Repeat for all templates in your workflows	47
8.2. MORE INFORMATION	47
<b>CHAPTER 9. LIQUIDS: DEVELOPER PORTAL</b> .....	<b>48</b>
9.1. USING LIQUIDS IN THE DEVELOPER PORTAL	48
9.1.1. Enabling Liquids	48
9.1.2. Different use on pages, partials, and layouts	48
9.1.3. Use with CSS/JS	49
9.2. USAGE OF LIQUIDS IN EMAIL TEMPLATES	49
9.2.1. Differences from Developer Portal	49
9.3. TROUBLESHOOTING	49
9.3.1. Debugging	49
9.3.2. Typical errors and ways to solve them	49
9.3.3. Contacting support	50
<b>CHAPTER 10. LIQUIDS: EMAIL TEMPLATES</b> .....	<b>51</b>
10.1. ACCOUNT MANAGEMENT	51
10.2. CREDIT CARD NOTIFICATIONS	51
10.3. LIMIT ALERTS	52
10.4. APPLICATIONS	52
10.5. INVOICING	53
10.6. SERVICES	54
10.7. SIGNUP	55
<b>CHAPTER 11. CUSTOMIZING THE DEVELOPER PORTAL LAYOUT</b> .....	<b>56</b>
11.1. CREATING A NEW CSS FILE	56
11.2. LINKING THE STYLESHEET INTO YOUR PAGE LAYOUT	56
11.3. DEFINING PAGE LAYOUT TEMPLATES	56
<b>CHAPTER 12. CHANGE BUILT-IN PAGES</b> .....	<b>57</b>
12.1. IDENTIFY THE ELEMENTS	57
12.2. MODIFY OR HIDE THE ELEMENTS	57
12.3. OPTION A: CSS	58
12.4. OPTION B: JQUERY	58
<b>CHAPTER 13. SETTING TERMS AND CONDITIONS</b> .....	<b>60</b>
13.1. TERMS AND CONDITIONS	60
13.2. CREDIT CARD POLICIES	61
<b>CHAPTER 14. EXPORTING AND IMPORTING THE DEVELOPER PORTAL</b> .....	<b>63</b>



## PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation.

To propose improvements, open a Jira issue and describe your suggested changes. Provide as much detail as possible to enable us to address your request quickly.

### Prerequisite

- You have a Red Hat Customer Portal account. This account enables you to log in to the Red Hat Jira Software instance. If you do not have an account, you will be prompted to create one.

### Procedure

1. Click the following link: [Create issue](#).
2. In the **Summary** text box, enter a brief description of the issue.
3. In the **Description** text box, provide the following information:
  - The URL of the page where you found the issue.
  - A detailed description of the issue.  
You can leave the information in any other fields at their default values.
4. Click **Create** to submit the Jira issue to the documentation team.

Thank you for taking the time to provide feedback.



# CHAPTER 1. OVERVIEW OF CREATING YOUR DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS

Your 3scale Developer Portal is the website that application programming interface (API) consumers use to:

- Sign up for access to your 3scale-managed, upstream API
- Read documentation about how to use your upstream API

3scale provides a sample Developer Portal with features that most API providers want to implement in their Developer Portal. This native Developer Portal uses the sample Echo API to demonstrate the structure of a typical Developer Portal. After you explore the native Developer Portal, there are instructions for changing it so you can learn how to create your own Developer Portal.

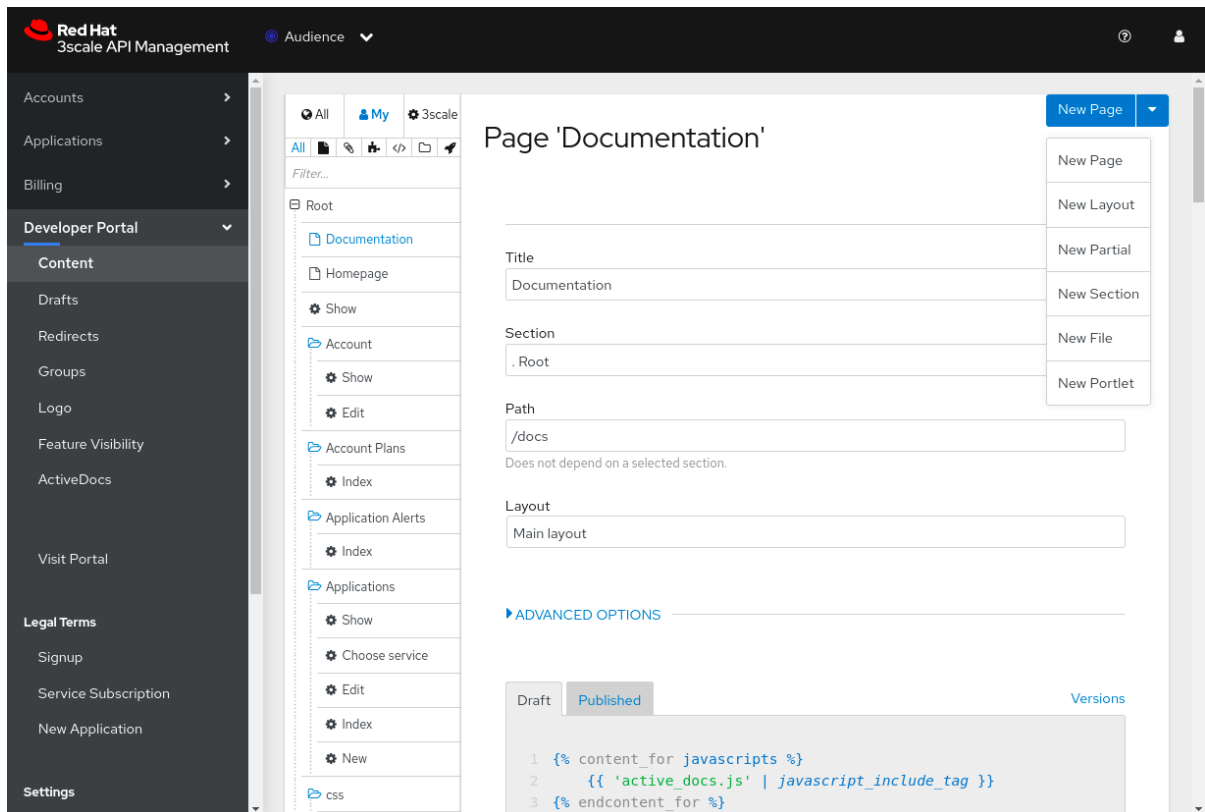
There are no prerequisites for exploring and modifying the native Developer Portal. However, after you modify the native Developer Portal to create your own Developer Portal, you must also implement sign-in workflows and authentication before you open your Developer Portal to API consumers.

## 1.1. TOURING THE 3SCALE API MANAGEMENT EDITING ENVIRONMENT FOR CREATING YOUR DEVELOPER PORTAL

Before you start to create your Developer Portal, explore the sample Echo API Developer Portal provided with 3scale. The Echo API Developer Portal is your starting point for creating your own Developer Portal. You do not create a Developer Portal from scratch. Instead, you modify the native Echo API Developer Portal to create a Developer Portal that has the look and feel that you want.

### Procedure

1. In the 3scale Admin Portal, expand the context selector at the top and click **Audience**.
2. In the navigation tree on the left, expand **Developer Portal** and click **Content**. This displays the main editing environment for creating your Developer Portal:



Under **Root**, 3scale displays the Developer Portal resources hierarchy:

- a. **Documentation, Homepage, and Show** are the foundation pages for the Developer Portal. For each page, scroll down to see the HTML that defines the page content.
  - b. Folders below those pages contain pages that pull your 3scale resources into the Developer Portal. For example, the **Account** folder contains pages for displaying and editing 3scale accounts that a 3scale administrator created in the Admin Portal. Use these pages as a starting point and modify them as needed.
  - c. In the upper right, the **New Page** drop-down lets you add a page, layout, partial, section, file, or portlet. Select each one to see the information you provide to create one.
3. With **Developer Portal > Content** selected, scroll down the resources hierarchy to almost the bottom until you see **Layouts** and click **Main layout**.
  4. Scroll back up to the top so that the **Layout 'Main Layout'** heading is visible. After the internal title and system name, you can see that **Liquid** is enabled. Liquid is the framework that 3scale uses to display and process most of the data from the 3scale system. The code that defines page content contains Liquid markup as well as HTML. You can see this in the **Draft** tab, which contains the code for the main layout for Developer Portal pages.
  5. In the navigation tree on the left, below **Developer Portal > Content**, click each sub-category in turn to explore it: **Drafts, Redirects, Groups, Logo, Feature Visibility, and ActiveDocs**.
  6. In the navigation tree on the left, below **Developer Portal**, click **Visit Portal**, which is the last entry. A new browser tab displays the website for the development version of the 3scale-provided Echo API Developer Portal. You use this development version to view the 3scale native Developer Portal. You can then iteratively customize and view changes that you make to the native Developer Portal to create your own Developer Portal.

You know that this is the development version of the Developer Portal because it has a dark

gray panel on the right with **Draft|Published** at the top. The Draft view supports iterative/incremental improvements. When the Draft version looks and behaves the way you want it to, you can publish it.

The panel on the right lists the elements that provide the content for the current page:

- a. **Page Homepage**
  - b. **Layout Main layout**
  - c. **Partial Submenu**
  - d. **Partial analytics**
7. Click **Page Homepage**. A new browser tab displays the Developer Portal editing environment with **Page 'Homepage'** open for editing.
  8. Go back to the development version of the Echo API Developer Portal and in the upper right, click **SIGN IN**, which displays the **SIGN IN** page that API consumers use to sign in to your Developer Portal.

In the gray panel on the right, below the list of templates, is a username and a password that you can use to simulate signing in to your Developer Portal:

- a. In the **SIGN IN** page, in the **USERNAME OR EMAIL** field, enter **John**, which is the username listed in the gray panel on the right.
- b. In the **PASSWORD** field, enter **123456**, which is the password listed in the gray panel.
- c. Click **Sign In** to display the Developer Portal as your API consumers would see it.

### Next steps

Continue to explore the native Developer Portal as much as you want. When you are comfortable navigating the editing environment and the development version of the Echo API Developer Portal, follow the procedure for [Modifying the 3scale native Developer Portal](#).

## 1.2. MODIFYING THE 3SCALE API MANAGEMENT NATIVE DEVELOPER PORTAL

After exploring the Echo API Developer Portal, make some changes to it before you start to create your own Developer Portal. These practice steps are helpful preparation for creating your Developer Portal.

This procedure replaces the sample 3scale Echo API landing page heading with a heading for the common Swagger Petstore API. It also shows you how to update your Developer Portal to display documentation for the Petstore API.

### Procedure

1. In the 3scale Admin Portal, expand the context selector at the top and click **Audience**.
2. In the navigation tree on the left, expand **Developer Portal** and click **Content**.
3. Under **Root**, click **Homepage**, which is the internal title of the Developer Portal landing page, and change the landing page heading that appears in the development version of your Developer Portal:

- a. In **Page 'Homepage'**, scroll down to see the code that renders the landing page.
  - b. In line **5**, change

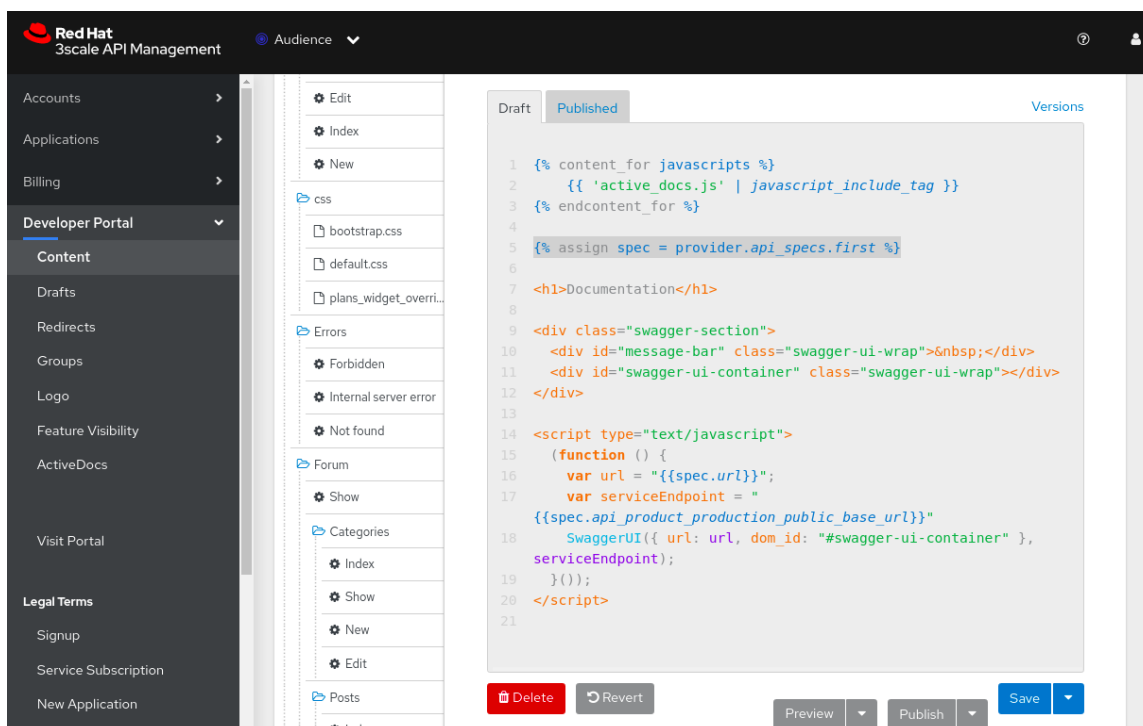
```
<h1>Echo API</h1>
```

to

```
<h1>Petstore API</h1>
```
  - c. At the bottom of the page, click **Publish**.
  - d. In the navigation tree on the left, below **Developer Portal**, click **Visit Portal** to display the development version of your Developer Portal and see that the landing page heading is now **Petstore API**.
4. Remain in the development version of your Developer Portal and click **Documentation**, which is in the top menu bar. The Developer Portal displays ActiveDocs for the Echo API.
  5. Return to the 3scale Admin Portal and select **Developer Portal > ActiveDocs** to see an entry for the Echo API. 3scale provides an OpenAPI document that defines the Echo API. 3scale uses this OpenAPI document to display ActiveDocs for the Echo API.
  6. Import an OpenAPI document that defines the Swagger Petstore API:
    - a. Go to <https://petstore.swagger.io/v2/swagger.json> and copy the JSON content to your clipboard.
    - b. Return to the 3scale Admin Portal with **Developer Portal > ActiveDocs** selected.
    - c. On the **ActiveDocs** page, click **Create a new spec**
    - d. In the **Name** field, enter **Petstore**.
    - e. Select **Publish?**.
    - f. Click in the **API JSON Spec** window and paste the Swagger Petstore JSON content from your clipboard.
    - g. Click **Create Spec** at the bottom of the page. 3scale displays ActiveDocs for the Petstore API.
    - h. In the navigation tree on the left, under **Developer Portal**, click **ActiveDocs**. After **Echo API**, there is a second entry for **Petstore**.
  7. Display documentation for the Petstore API in your Developer Portal:
    - a. In the navigation tree on the left, under **Developer Portal**, click **Content**.
    - b. Under **Root**, click **Documentation**, which is the internal title of the Developer Portal documentation page.
    - c. In **Page 'Documentation'**, scroll down to see the code that renders the documentation landing page. Line **5** identifies the OpenAPI document for which your Developer Portal displays ActiveDocs. The default value of line **5** is:

```
{% assign spec = provider.api_specs.first %}
```

Default behavior is that your Developer Portal displays ActiveDocs for the first entry in the **Developer Portal > ActiveDocs** page, which is initially the Echo API. This figure highlights line 5:



- d. Modify line 5 to change **provider.api\_specs.first** to have an index that identifies the second entry in the ActiveDocs page:

```
{% assign spec = provider.api_specs[1] %}
```

Default behavior is that your Developer Portal displays ActiveDocs for only one OpenAPI document. To display ActiveDocs for more than one OpenAPI document, you must modify the **Documentation** page beyond this simple change.

- e. At the bottom of the page, click **Publish**.
8. In the navigation tree on the left, below **Developer Portal**, click **Visit Portal** to display the development version of your Developer Portal.
9. In the top menu bar, click **Documentation** to see documentation for Swagger Petstore.

## Next steps

Start creating your Developer Portal by changing the native Developer Portal pages so they display information about access to your 3scale-managed API and documentation for your API.

## 1.3. DESCRIPTIONS OF ADDITIONAL 3SCALE API MANAGEMENT FEATURES FOR MODIFYING THE NATIVE DEVELOPER PORTAL

3scale provides many features for customizing the native Developer Portal to create your own Developer Portal.

During development of your Developer Portal, anyone who needs to view your Developer Portal must specify an access code. Keep this access code in place while API providers are performing the required

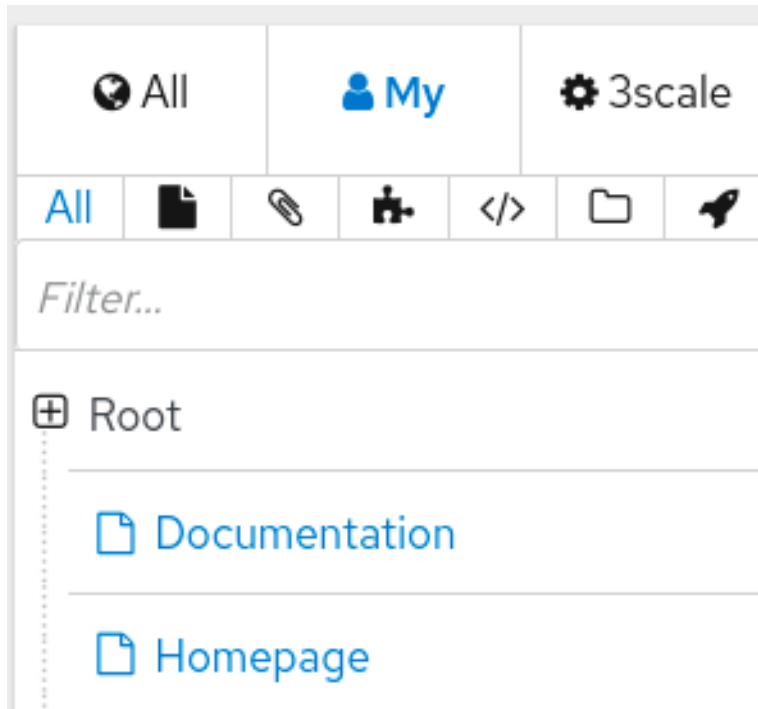
tasks for opening your Developer Portal to access by API consumers. The access code ensures that only you or anyone with this code can view your Developer Portal. The access code is in the **Domains & Access** page, which is available in the Admin Portal by selecting **Audience > Developer Portal > Domains & Access**.

The following list introduces some of the features for modifying the native Developer Portal:

- In the **Developer Portal > Content** environment, the **New Page** drop-down in the upper right lets you add the following elements to your Developer Portal:
  - **Page** - basic unit of Developer Portal content.
  - **Layout** - a template that multiple pages can use.
  - **Partial** - content that can be reused in other partials, in layouts, and in pages.
  - **Section** - functionally equivalent to a directory. Create sections to organize Developer Portal content.
  - **File** - a resource that you want your Developer Portal to use, such as a stylesheet, an image, or a script.
  - **Portlet** - external RSS feed, table of contents, or latest forum posts.
- You can input code in HTML, Markdown, or Textile. The code editor supports code highlighting, tabulations, line numbers, and other features.
- You can preview the draft or published version of a page. Below a page's text editor, clicking **Preview** displays the development version of the Developer Portal. The preview has a dark gray vertical bar on the right side. At the top of the dark gray bar, one of **Draft | Published** is highlighted to indicate which version you are viewing. This bar contains links to the editing environment for the following:
  - The page.
  - The layout that the page uses.
  - Any partials that the page uses. A partial is code that can be reused in many places on different pages.

The Draft view of the development version of the Developer Portal supports iterative/incremental improvements. When the Draft version looks and behaves correctly, you can publish it. When the access code is in place, publishing a page means that you can view what an API consumer would see if your Developer Portal was live. When the access code is no longer in place, publishing a page updates that page in your live Developer Portal.

- The filter field lets you search for resources in your Developer Portal environment. 3scale displays only the elements that you are searching for, which makes it easier to find just what you want to work on. The filter field is just below the icons:



- The **Developer Portal > Redirects** page lets you set up redirects from one Developer Portal URL to another. For example, if you deprecate a page that you created, you can redirect requests to the new page.
- Placeholders are editable fields. For example, the **Developer Portal > Content > Login > New** page has several placeholders:

```
<input id="session_username" name="username" tabindex="1"
  autofocus="autofocus" type="text"
  placeholder="Authenticate with your email"
  class="form-control">
...
<input id="session_password" name="password" tabindex="2"
  type="password"
  placeholder="...and password"
  class="form-control">
```

This code generates the following fields in your Developer Portal:

<b>USERNAME OR EMAIL</b>	<input type="text" value="Authenticate with your email"/>
<b>PASSWORD</b>	<input type="password" value="...and password"/>

You can update the placeholder text and publish it to see the update in your Developer Portal. For example, you can change **Authenticate with your email** to **Authenticate email**. After you publish the page, you can see the updated prompt in your Developer Portal, for example:

USERNAME OR EMAIL	<input type="text" value="Authenticate email"/>
PASSWORD	<input type="text" value="...and password"/>

## 1.4. HOW 3SCALE API MANAGEMENT LAYOUTS AND PARTIALS REUSE DEVELOPER PORTAL CONTENT

In the **Developer Portal > Content** environment, when you scroll down in the resource hierarchy, you can see headings for **Layouts** and **Partials**. Layouts and partials let you reuse content in your Developer Portal:

- A layout defines the basic structure for a page and functions as a page template. Every page that uses a particular layout contains the content that the layout defines. As you can see in the resource hierarchy under **Layouts**, your native Developer Portal provides an **Error layout** and a **Main layout**. You can modify the code for these layouts and you can create layouts.
- A partial defines code that you use in more than one place in your Developer Portal. For example, a partial can define the footer in all layouts or the sidebar on multiple pages. You can use a partial in a page, a layout, another partial, or an email template. In the resource hierarchy, under **Partials**, you can see the partials that the native Developer Portal provides. There are partials for gathering analytics, subscribing to application plans, displaying submenus, and more. Again, you can modify these partials and you can create partials.

In a page, layout, partial, or email template, to use a partial, you specify an **include** command with the name of the partial. For example, to use the **submenu** partial, you would specify:

```
{% include "submenu" %}
```

Layouts and partials have draft and published states as well as a version history. For example, you can save a draft, publish it, and revert to the last published version.

To learn about editing and writing code for layouts and partials, see the [3scale Liquid Reference](#).

## 1.5. ADDING IMAGES AND OTHER ASSETS TO YOUR DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS

Typically, you want to use your own images, files, or other assets to customize the native Developer Portal. To do this, add the file to the Developer Portal content library, make a note of its path in this environment, and then in code you add a link to the location of the file in the content library.

### Prerequisites

- Access to the image, file, or other asset that you want to use in your Developer Portal

### Procedure

1. Add your asset to the Developer Portal content library:
  - a. In the **Developer Portal > Content** environment, in the upper right, expand **New Page** and select **New File**.



- b. Under **Upload File**, click in the **Section** field to display a list of the directories in which you can store, and click the appropriate directory to select it. For example, you might select **|-images**.
  - c. Optional. In the **Path** field, enter any additional path levels that are useful for you.
  - d. Select **Downloadable**.
  - e. Click **Choose File**, navigate to the file you want to add, select it and click **Open**.
  - f. Optional. In the **Tag list** field, enter a Liquid tag that you want to use to include this new content in code, for example, you might enter **my-logo**.
  - g. Click **Create File** to add your asset to the Developer Portal content library.
2. Make a note of the path to the asset you added. For example, if you added the **MyLogo.png** image to the **images** section, the path would be **/images/MyLogo.png**.
  3. To use the asset in code, insert an HTML **<a>** element. For example:

```
<a href="/images/MyLogo.png"/>
```

Alternatively, if you added a Liquid tag to your asset, you can use the asset in code by specifying the tag, for example:

```
{% my-logo %}
```

## 1.6. CONSIDERATIONS FOR CUSTOMIZING THE NATIVE DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS

Before you start customizing the native Developer Portal to create your own Developer Portal, you should have a plan for its basic elements, reusable content, page hierarchy, page headers and footers, and branding.

- Basic Developer Portal elements - The plan for your Developer Portal should include:
  - Site map: Skeleton of how your portal is organized.
  - Top menu bar: Navigation that is repeated on each page.
  - Side menu bars: Access to individual pages in each section.
  - Page layout guidelines: For a consistent look and feel.
  - Reusable content - Layouts and partials ensure consistency in your Developer Portal. Create them before you start creating pages.
- Page hierarchy - Beginning at the root level in your site map, create a section for each of your top menu items.
 

Next, create the pages needed for each top menu item. When you create a page, be sure to associate it with the correct section. This creates a consistent structure for URL paths in your Developer Portal. To code a page with a markup language such as Textile or Markdown, expand the page's **ADVANCED OPTIONS**. The **Handler** field identifies the markup language.

- Page headers and footers - Repetitive page elements are typically defined in partials. When you have only a few layouts, you might prefer to define headers and footers directly in your layouts rather than in partials.
- Branding - The default native Developer Portal stylesheet, **default.css**, is large and complex. Rather than extending this stylesheet, create your own stylesheet with customizations that overwrite the defaults.  
You can create a new stylesheet in the same way you create a page. Choose the **css** section and an appropriate MIME content type in the advanced page settings. Then in your layouts, after the link to **default.css**, add a link to your custom stylesheet. For example, if you create **my-custom.css**, you would insert:

```
<link rel="stylesheet" href="/css/my-custom.css" />
```

- Built-in features that you cannot modify, include:
  - Pagination.
  - Message menu.
  - Forums - You can change only the layout.
  - Payment - Only a few text fields in these pages are editable.

## 1.7. REQUIREMENTS FOR ALLOWING ACCESS TO YOUR DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS

Before you can give API consumers access to your Developer Portal, API providers must perform the following tasks. These tasks can be performed concurrently:

- Change the 3scale native Developer Portal to have the look and feel that you want:
  - [Liquids: Developer Portal](#) shows how to use Liquid markup to display and process 3scale system data related to your API. Liquid markup is the primary way to add logic to your Developer Portal pages.
  - [Customizing the Developer Portal layout](#) describes how to change the native Developer Portal to match your own branding. A standard cascading stylesheet (CSS) is available to provide an easy starting point for your customizations.
  - [Change built-in pages](#) explains how to use CSS and JavaScript to modify or hide an element in a system-generated page. All pages provided as part of the native Developer Portal are referred to as system-generated pages.

A developer who is familiar with HTML, CSS, Liquid, and websites in general can modify the native Echo API Developer Portal to create your Developer Portal web pages. This developer can create pages and modify code in system-generated pages to create whatever you want API consumers to see in your Developer Portal.

- Define 3scale API products, backends, and application plans, and assign policies to products. A 3scale product is the 3scale resource that exposes your API to consumers. In your Developer Portal, API consumers read the documentation for a product and subscribe to use the API provided by a product. A 3scale product has the following characteristics:
  - Bundles one or more backends, which are internal 3scale APIs that you have created.

- Has an application plan that defines the rules for using the product with regard to limits, pricing, and available features.
- Has an APIcast gateway configuration for how the gateway should process API consumer calls before sending them to your API. Policies that you add to a product change the default APIcast gateway behavior.

See the [Admin Portal Guide](#).

- Import the OpenAPI document that defines and documents your 3scale-managed API. The foundation for your Developer Portal is an OpenAPI document that defines your API. When you import an OpenAPI document into 3scale, 3scale creates or updates ActiveDocs so you immediately have functional documentation for your API. In your Developer Portal, API consumers use this documentation to explore, test, and integrate with your API.

For each operation defined in an OpenAPI document, importing the document causes 3scale to create a method and a mapping rule. Methods and mapping rules help enforce the limits and rules for API consumer access to your API.

The 3scale guide, [Providing APIs in the Developer Portal](#) contains information and procedures for working with 3scale and OpenAPI documents. In particular, see [How to write an OpenAPI document for use as a 3scale specification](#) and [Adding ActiveDocs to 3scale](#).

- Configure the workflow for how an API consumer signs up for access to your 3scale-managed API.

Signup workflows are a critical aspect of API consumer experience with your Developer Portal. The workflow can range from self-service to total control over who gains access to what. Accounts, services, and application plans offer multiple levels of granularity. At each level, you control whether there is an approval gate that you operate and whether the API consumer is required to make any choices.

For maximum automation and self-service, you can remove all approval steps and enable all possible default plans. Immediately after signup, your Developer Portal can issue a key that provides API consumer access to your Developer Portal.

The native Developer Portal provides commonly used fields for user, account, and application signups. You might need to add custom fields to these typically used fields. For details, see [Configuring signup flows](#) and [Custom signup form fields](#). Be sure to configure workflows before you customize email templates.

- Implement authentication of API consumers. Authentication of API consumer access to your Developer Portal secures your Developer Portal resources and your APIs. You can use any of the following methods to authenticate access to your Developer Portal:
  - [Username, email and password](#)
  - [GitHub](#)
  - [Auth0](#)
  - [Red Hat Single Sign-On \(RH-SSO\)](#)

See [Developer Portal authentication](#).

- Customize 3scale native templates for email communication between your Developer Portal and API consumers.

Many different events require communication between your Developer Portal and API consumers, including providing an account activation link upon signup, password recovery, service charges, change notifications, and many more. 3scale provides templates for each typical email type that a Developer Portal sends to API consumers.

After you have defined signup workflows, customize the content of email messages. This lets you closely match the workflows that you set up for your Developer Portal.

See [Email templates](#) and [Liquids: Email templates](#).

- Specify terms, conditions, and policies that API consumers must agree to for access to your 3scale-managed API.  
When you allow API consumers to sign up and call your API, you typically want them to agree to your terms, conditions and policies before you grant access. You can have different versions of terms and conditions, for example, for signup, for using particular applications, or for using particular services, if your Developer Portal provides more than one service.

If you are charging for use of your API, you probably want agreement to your credit card policies.

See [Setting terms and conditions](#).

- Set up API consumer billing and credit card gateways.  
The 3scale billing process runs daily. It creates invoices for each API consumer account that is subscribed to a paid service. An invoice can be in one of the following states: open, finalized, pending, unpaid, paid, failed, canceled. 3scale uses the payment gateway that you configure to process invoices.

The 3scale billing process can run in a prepaid or postpaid mode. Billing in 3scale is based on the calendar month, and there are special events that happen on the first day of the month.

See [Admin Portal Guide, Billing](#).

The last task you must perform before your Developer Portal goes live is to remove the access code. Do this only after authentication for access to your Developer Portal is in place, and after you have thoroughly tested your Developer Portal to confirm that it is behaving as you want it to.

To delete the access code, display the **Developer Portal > Content** environment. In the lower right, click **Open your Portal to the world** and confirm this action.

## 1.8. OPTIONAL STEPS FOR CUSTOMIZING YOUR DEVELOPER PORTAL FOR 3SCALE-MANAGED APIS

Beyond the requirements for opening your Developer Portal to API consumers, you might want to do the following:

- Provide multiple APIs in your Developer Portal.  
Each OpenAPI document that you import into 3scale provides the foundation for a separate API offering, also referred to as a separate service. To configure your Developer Portal to provide multiple services, the main task is to create a page that lets an API consumer choose which service or services to subscribe to.

For details, see [Multi-service signup](#).

- Mark Developer Portal pages or portions of pages as visible to only the API consumers you specify.

You might need to have parts of your Developer Portal accessible to only a specific group of API consumers. You can restrict access to a page, part of a page, or a menu selection, which typically corresponds to a section.

A handy way to restrict access to a section is to map each section to a logical group of API consumers. For example, suppose there are API consumers who are partners. You can create a group called partners and give only that group access to a particular section.

You can provide access to restricted content to API consumers based on a change in state. For example, when an API consumer upgrades to a new application plan additional pages might automatically become visible to that API consumer.

Another way to restrict access is to require an API consumer to log in to view certain content.

For details, see [Restricted content](#).

- Implement webhooks.  
Webhooks let you tightly integrate 3scale with your back-office workflow. When specified events happen in the 3scale system, your back-office applications can be notified with a webhook message. Your application can then use that data, for example, information about a new account, to populate your Developer Portal.

For details, see [Webhooks](#).

## 1.9. UPDATING DEVELOPER PORTALS USED IN 3SCALE API MANAGEMENT RELEASES EARLIER THAN 2.8

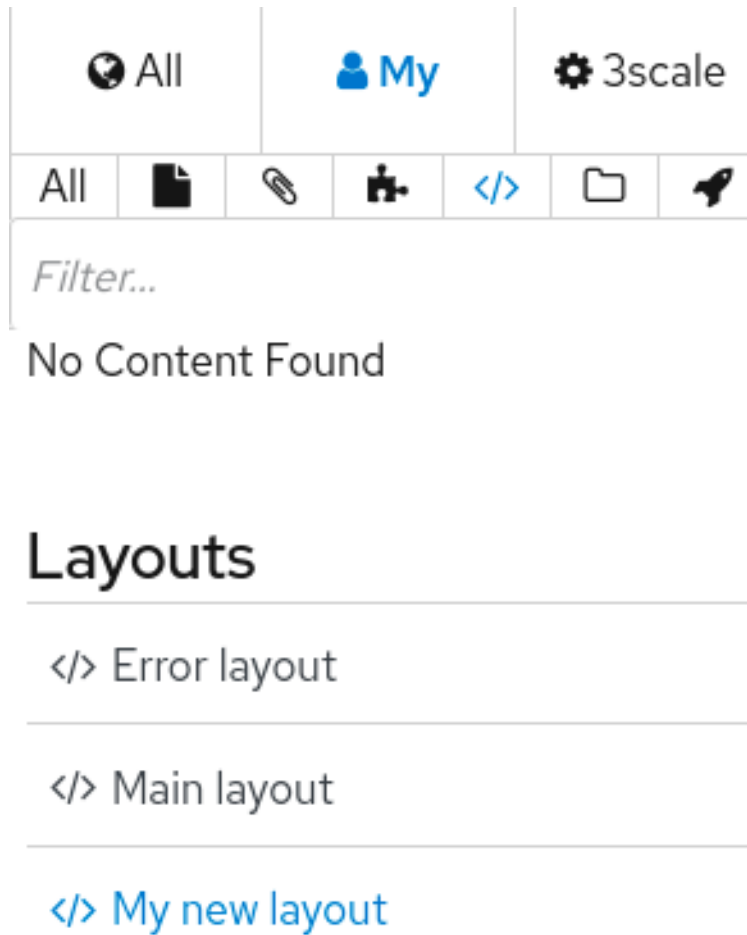
In 3scale 2.8, external assets were moved from Content Delivery Networks (CDN) to the 3scale codebase. Consequently, starting with 3scale 2.8, the native Developer Portal is created by using the **cdn\_asset** Liquid tag. If you are upgrading from a 3scale release that is earlier than 2.8, you must update your Developer Portal to use the new assets with the **cdn\_asset** tag. With the use of this tag, there is no longer a dependency for downloading assets from external websites.

### Prerequisites

- An installation of 3scale 2.8 or later.
- A Developer Portal that you created with 3scale 2.7 or earlier.

### Procedure

1. In the 3scale Admin Portal, with **Audience > Developer Portal > Contents** selected, at the top of the resource hierarchy, click **</>** to display only layouts:



2. Under **Layouts**, click **Main layout**.
3. In the code editor for the main layout, on or around line **17**, replace this line:

```

{{ '//netdna.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.css' |
stylesheet_link_tag }}

```

with this line:

```

{% cdn_asset /font-awesome/4.3.0/css/font-awesome.css %}

```

4. On or around line **19**, replace this line:

```

{{ '//ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js' | javascript_include_tag }}


```

with this line:

```

{% cdn_asset /jquery/1.7.1/jquery.min.js %}

```

5. Scroll down and click **Publish**.
6. Go to the top of the resource hierarchy and click  to display partials.
7. Under **Partials**, click **stats/chart**.
8. In the code editor for **stats/chart**, on or around line **3**, replace this line:

```
  {{ '//ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/themes/ui-lightness/jquery-ui.css' |  
  stylesheet_link_tag }}
```

with this line:

```
  {% cdn_asset /jquery-ui/1.11.4/jquery-ui.css %}
```

9. Scroll down and click **Publish**.
10. Optional. If you created files based on the main layout or based on the **stats/chart** partial, you must similarly update those files.

### Additional resources

- [Liquid drops, tags, and filters](#) in the 3scale Liquid Reference

## CHAPTER 2. CUSTOM SIGNUP FORM FIELDS

Learn how to add custom signup fields and the different options around this feature.

By default, 3scale provides commonly used fields at user/account/application signup. You may need to add your own custom fields to these common defaults.

In your Admin Portal, go to **Audience > Accounts > Field Definitions** where you can see the default form fields and define new ones.

**RED HAT 3SCALE** API MANAGEMENT Audience

### Fields Definitions

Here you can manage all the information you gather from your users. You can add new fields and change the existing ones; making them Hidden, Read Only, or Required. You can change the text your users see when viewing or entering data (shown here between quotes). Drag and drop the fields to set the order in which they will be shown.

**Account** Create

<b>org_name</b>	"Organization/Group Name"	Required	Edit
-----------------	---------------------------	----------	------

**User** Create


<b>username</b>	"Username"	Required	Edit
<b>email</b>	"Email"	Required	Edit

**Application** Create

<b>name</b>	"Name"	Required	Edit
<b>description</b>	"Description"	Required	Edit

The new account/user signup page is actually an amalgamation of the first two sections. The account fields appear at the top, followed by the user fields, followed by the password fields which don't need to be configured.



 SIGN UP

ORGANIZATION/GROUP NAME

USERNAME

EMAIL

PASSWORD

PASSWORD CONFIRMATION

By signing up you agree to the following Legal Terms and Conditions ([show](#))

[Sign up](#)

Try adding 3 extra fields, 2 to the user signup section and 1 to the account section. Click create, add the following new field definition and then create it. The required checkbox will, of course, make it mandatory on the signup form. There are also options to make things hidden and read only. A hidden field may be added, for example, when you want new signups to have fields set that you don't necessarily want to highlight to them, such as `access_restricted_areas` which would be empty by default. As an admin, you can update this to true later on a per-user basis. Your page logic could read it in to determine what to display. A read-only field might be, for example, browser location, which you could use JavaScript on page load to set.

## New Field definition for User

Add a field to store information about your developers on signup or at any other time. Make the fields Hidden, Read Only, Required. The label is the text developers will see when viewing or entering their data.

FIELD FROM SCRATCH OR BASED ON EXISTING FIELD

[new field]

FIELD DETAILS

Name

last\_name

The low level system name.

Label

Last Name

The field title your developers will see.

Required

Makes the field required for developers.

Hidden

Developers won't be able to see this field.

Read only

Developers won't be able to change this field.

Choices

Full time, Part time, Contract

Separate the predefined options for this field by commas or enter each option on a new line.

Create

Now try adding a drop-down to the user signup form. Call it "employment type". Add these comma-separated values into the choices field: full time, part time, contract. The drop-down will be populated with these values.

FIELD FROM SCRATCH OR BASED ON EXISTING FIELD

FIELD DETAILS

Name

The low level system name.

Label

The field title your developers will see.

 Required

Makes the field required for developers.

 Hidden

Developers won't be able to see this field.

 Read only

Developers won't be able to change this field.

Choices

Separate the predefined options for this field by commas or enter each option on a new line.

Now add a pre-defined field to the account. Usually the fields you add have no system functionality – they simply hold data that you can access later. See [restricted content](#).

Create a field as normal. Then on the drop-down above "name", choose `po_number`. With this field, a PO number will appear on 3scale-generated invoices sent to this developer account. System-generated fields can be overridden by your admins at any time. Give the field a name – something like "PO number" – and create it.

## New Field definition for Account

Add a field to store information about your developers on signup or at any other time. Make the fields Hidden, Read Only, Required. The label is the text developers will see when viewing or entering their data.

FIELD FROM SCRATCH OR BASED ON EXISTING FIELD

✓ [new field]  
org\_legaladdress  
org\_legaladdress\_cont  
telephone\_number  
vat\_code  
vat\_rate  
fiscal\_code  
state\_region  
city  
country  
zip  
primary\_business  
business\_category  
po\_number  
billing\_address

Required

Makes the field required for developers.

Hidden

Developers won't be able to see this field.

Read only


Developers won't be able to change this field.

Choices

Separate the predefined options for this field by commas or enter each option on a new line.

Create

Now take a look at your work. You can see the free text last name and the employment type drop-down have been added to the User section. The PO number system field, also free text, has been added to the Account section.

 SIGN UP

ORGANIZATION/GROUP NAME

PO NUMBER

USERNAME

EMAIL

LAST NAME

EMPLOYMENT TYPE

PASSWORD

PASSWORD CONFIRMATION

By signing up you agree to the following Legal Terms and Conditions ([show](#))

Finally, these custom fields can be set using the 3scale ActiveDocs; for example, **application create**.

## CHAPTER 3. CONFIGURING SIGNUP FLOWS

In this section, you will see which settings to configure to adjust signup workflows.

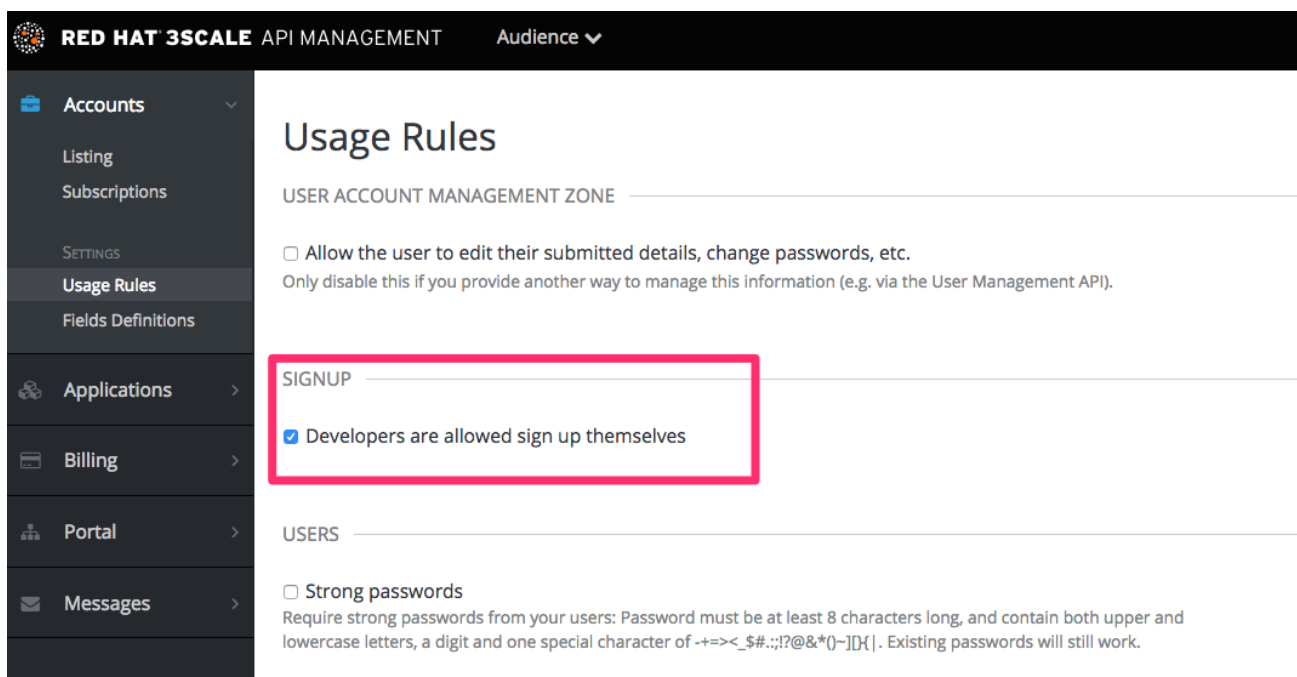
Signup workflows are a critical aspect of the developer experience you provide through your Developer Portal. The process can range from being completely automatic and self-service to the other extreme of requiring total control over who gains access to what, with various levels of granularity.

The 3scale platform allows you to model your API with a combination of account (optional), service (optional), and application plans. For each of these plans, you can control whether there is an approval gate that you operate. For each one, you also determine whether there is a default, or the developer is required to take the next step and make a choice.

For the extreme of maximum automation and self-service, remove all approval steps and enable all possible default plans. This way, a key can be issued to provide access to your API immediately after signup.

### 3.1. REMOVING ALL APPROVAL STEPS

To remove approvals, go to **Audience > Accounts > Usage Rules** and in the **Signup** section, make sure the option of **Developers are allowed to sign up themselves** is checked.



The screenshot shows the Red Hat 3scale API Management interface. The top navigation bar includes the Red Hat 3scale logo, 'API MANAGEMENT', and 'Audience' with a dropdown arrow. The left sidebar contains a menu with 'Accounts' (sub-items: Listing, Subscriptions), 'SETTINGS' (sub-items: Usage Rules, Fields Definitions), 'Applications', 'Billing', 'Portal', and 'Messages'. The main content area is titled 'Usage Rules' and is divided into sections: 'USER ACCOUNT MANAGEMENT ZONE' with a checkbox for 'Allow the user to edit their submitted details, change passwords, etc.' (disabled), 'SIGNUP' (highlighted with a red box) with a checked checkbox for 'Developers are allowed sign up themselves', and 'USERS' with a checkbox for 'Strong passwords' (disabled).

Optionally, if you have account and service plans enabled, scroll down the page and make sure the option **Change plan directly** is enabled in both cases:

RED HAT 3SCALE API MANAGEMENT Audience ▾

Accounts ▾

- Listing
- Account Plans
- Subscriptions

SETTINGS

- Usage Rules
- Fields Definitions

ADVANCED PLANS

Account Plans  
Only consider using Account Plans if Application Plans don't suit your use case - for example if you offer additional non-API services.

ACCOUNT PLANS CHANGING

Request a plan change

Change the plan directly

## 3.2. ENABLING ALL POSSIBLE DEFAULT PLANS

### Application plans

RED HAT 3SCALE API MANAGEMENT API: Echo API ▾

Overview

Analytics

Applications ▾

- Listing
- Application Plans

Subscriptions

ActiveDocs

Integration

### Application Plans

Application Plans establish the rules (limits, pricing, features) for using your API; every developer's application accessing your API will be accessing it within the constraints of an Application Plan. From a business perspective, Application Plans allow you to target different audiences by using multiple plans (i.e. 'basic', 'pro', 'premium') with different sets of rules.

- Basic
- Unlimited
- Plan A
- Plan B

Name	Applications	State				
Basic	3	published	Hide	Copy	Delete	
Unlimited	4	published	Hide	Copy	Delete	
Plan A	0	published	Hide	Copy	Delete	
Plan B	0	hidden	Publish	Copy	Delete	

[Create Application Plan](#)

Optionally, if you have account and service plans enabled, choose default plans for those too.

### Account plans (optional)

RED HAT 3SCALE API MANAGEMENT Audience ▾

Accounts ▾

- Listing
- Account Plans
- Subscriptions

SETTINGS

- Usage Rules
- Fields Definitions

Applications

Billing

Developer Portal

### Account plans

Account plans create "tiers" of usage within the developer portal, allowing you to distinguish between grades of support, content and other services partners at different levels receive.

Default Plan  
Default

Default account plan (if any) is contracted automatically on sign up.

[Edit the details of the plan from this link](#)

Name	Accounts	State				
Default	5	hidden	Publish	Copy	Delete	

[Create Account Plan](#)

### Service plans (optional)

RED HAT 3SCALE API MANAGEMENT API: Echo API

## Service Plans

Service plans allow you to define grades of service for each of the services (APIs) available through your developer portal. The plans allow you to define pricing per service and features available.

Default Plan  
 Default  
 Default service plan (if any) is contracted automatically on sign up.

[Click on this link to edit features, etc.](#)

Name	Subscriptions	State			
<a href="#">Default</a>	6	published	Hide	Copy	Delete

[Create Service Plan](#)

### 3.3. TESTING THE WORKFLOW

Once you have made your desired settings changes, test out the results by going to your Developer Portal and attempting to sign up as a new developer. Experiment and make any necessary adjustments to get exactly the right workflow for your API. When you are happy with the workflow, it is a good time to check your email notifications to make sure they provide the right information for your developers.

RED HAT 3SCALE API MANAGEMENT Audience

## Email Templates

Name	Description	
Buyer Account approved	After provider approves sign up, notification for buyer	Edit
Buyer Account confirmed	Buyer Account confirmed	Override
Credit card expired notification for buyer	Credit card expired notification for buyer	Override
Buyer account rejected	Buyer account rejected	Override
Alert notification for buyer (< 100%)	Alert notification for buyer when near 100% threshold	Override
Alert messenger limit alert for provider of master	No description.	Override
Alert notification for buyer (>= 100%)	Alert notification for buyer when over 100% threshold	Override



## CHAPTER 4. MULTI-SERVICE SIGNUP

By the end of this section, you will be familiar with the procedure to create and customize a multiple-service signup page.

If you are using the multiple services functionality, you are able to customize the signup procedure to allow customers to subscribe to different services.

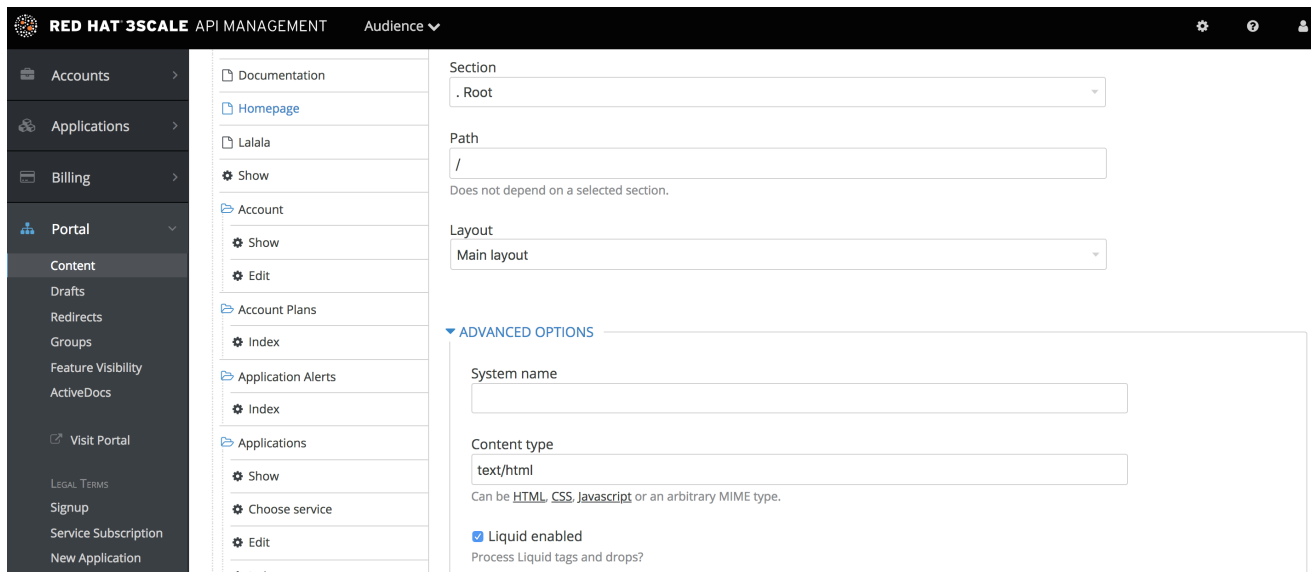
### 4.1. PREREQUISITES

You should be familiar with layout and page creation procedures as well as with the basics of Liquid formatting tags. For more details about liquid tags, see [Liquid reference](#). "Multiple Service" functionality must also be enabled on your account (available for Pro plan and up).

It is strongly recommend that you read about [signup workflows](#), so you will have the whole setup prepared and know how it works.

### 4.2. INTRODUCTION

Start the process by creating a new layout, which will serve as the template for your multi-service signup page. Go into the Layouts section of the CMS system, and create the new layout. You can call it *multipleservicesignup* to be able to easily distinguish it from the other layouts. In the editor, paste the general structure of your standard layout (such as home or main layout). Now delete everything you do not need – all the containers, sidebars, additional boxes, etc.



Having created the backbone of your layout, proceed to customizing the code for signup.

### 4.3. MULTI-SERVICE SIGNUP

#### 4.3.1. Retrieving information about services

In order to retrieve all the information about the services that you need to construct the proper signup link, you have to loop through the service objects. Services are a part of the model object.

```
{% for service in provider.services %}
```

```
.
```

```
.
.
{% endfor %}
```

### 4.3.2. Configuring the signup columns

You already have your layout and loop accessing the service objects. Now decide how you want to display information about the service and the signup link. For example, divide them into columns with a service description and a signup link at the bottom. Every column will be a div box with a `service-column` class to contain all the necessary information.

```
{% for service in provider.services %}
  <div class="service-column">
    <p>{{ service.name }}</p>
    <p>{{ service.description }}</p>
    .
    .
    .
  </div>
{% endfor %}
```

The container inside serves as a custom description field. `service.name` is the service name, which in this case will be the container's name.

### 4.3.3. Configuring the subscription

Now the main part of your custom service signup – to create the signup link, extract the signup URL and the service ID. Take the signup URL from `URL`'s object and the service ID from your service object on which you iterate in the loop. The final link code will look like this:

```
<a href="{{ urls.signup }}?{{ service | toparam }}">Signup to {{ service.name }}</a>
```

You also have to take into account that the user may already have signed up for some of your services. Create a conditional block to check.

```
{% unless service.subscribed? %}
  <a href="{{ urls.signup }}?{{ service | toparam }}">Signup to {{ service.name }}</a>
{% endunless %}
```

With this, you can generate the final code:

```
{% for service in provider.services %}
  <div class="service-column">
    <p>{{ service.name }}</p>
    <p>{{ service.description }}</p>
    {% unless service.subscribed? %}
      <a href="{{ urls.signup }}?{{ service | to_param }}">Signup to {{ service.name }}</a>
    {% endunless %}
  </div>
{% endfor %}
```

### 4.3.4. Styling

Add some final touches to the generated markup, depending on the number of services you have. In the case of this example it is two, so the CSS code for the service-column div will be:

```
.service-column {  
  float: left;  
  margin-left: 10%;  
  width: 45%;  
}  
.service-column:first-child {  
  margin-left: 0;  
}
```

In the example, we have used the percentage-based layout to dynamically assign the width of the column based on the containing div's dimensions.

Now you should have a properly working and good-looking multiple services subscription page. Congratulations!

If you would like to display the columns in a specific order, try using conditional expressions (if/else/case) conditioning the service name or another value you know.

## CHAPTER 5. DEVELOPER PORTAL AUTHENTICATION

Follow these steps to configure access to your developer portal.

This article shows how to enable and disable the different types of authentication that can be made available on your developer portal to allow your developers to sign up or sign in.

At the moment, 3scale supports several methods of authenticating to the Developer Portal, which are covered in the following sections:

1. [Username/email and password](#)
2. [Authentication via GitHub](#)
3. [Authentication via Auth0](#)
4. [Authentication via Red Hat Single Sign-On](#)

By default, only one type of authentication will be enabled on your developer portal, two if you signed up on 3scale.net:

- Username/email and password.
- Authentication via GitHub using the 3scale GitHub application - only enabled by default if you signed up on 3scale.net



### NOTE

Older 3scale accounts (created prior to December 14th, 2015) might need to follow an extra step in order to enable GitHub and Auth0 authentication.

If this applies to you, you will need to add the following code snippet to the login and sign up templates in order to enable this feature in both forms.

```
{% include 'login/sso' %}
```

## 5.1. ENABLING AND DISABLING USERNAME/EMAIL AND PASSWORD

By default, the username/email and password authentication is enabled on your developer portal. Usually there is no change to be made here, as this is a standard way for your developers to create an account and to login.

However, in some rare cases you might want to remove this authentication type. To do so, edit the **Login > New** template as in the screenshot below:

```

1 <div class="row">
2   <div class="col-md-9">
3     <div class="panel panel-default">
4       <div class="panel-heading">
5         <i class="fa fa-user"></i>
6         Sign in
7       </div>
8       <div class="panel-body">
9
10        {% include 'login/sso' %}
11
12        {% comment %}
13
14        {% form 'login_form', class: 'form-horizontal' %}
15          {% include 'login/cas' %}
16          {% include 'login/rainrain' %}
17
18          <fieldset>
19            <div class="form-group" id="session_username_input">
20              <label for="session_username" class="control-label col-md-4">Username or Email</label>
21              <div class="col-md-6">
22                <input id="session_username" name="username" tabindex="1" autofocus="autofocus"
23                  type="text"
24                  class="form-control">
25              </div>
26            </div>
27            <div class="form-group" id="session_password_input">
28              <label for="session_password" class="control-label col-md-4">Password</label>
29              <div class="col-md-6">
30                <input id="session_password" name="password" tabindex="2"
31                  type="password"
32                  class="form-control">
33              </div>
34            </div>
35
36            <input name="remember_me" type="hidden" value="1">
37          </fieldset>
38          <fieldset>
39            <div class="form-group">
40              <div class="col-md-10">
41                <input name="commit" type="submit" value="Sign in" class="btn btn-success btn-lg pull-
42right">
43              </div>
44            </div>
45          </fieldset>
46          {% endform %}
47          {% endcomment %}
48
49        </div>
50      </div>
51      <div class="panel-footer">
52        <a href="{{ urls.forgot_password }}">Forgot password?</a>
53
54        {% if provider.signups_enabled? %}
55          <a href="{{ urls.signup }}" class="link">Sign up</a>
56        {% endif %}
57      </div>
58    </div>
59  </div>
60 </div>
61

```

If you need to add back the username/email and password authentication to your developer portal, just remove the liquid comment tags added in the previous step.

## 5.2. ENABLING AND DISABLING AUTHENTICATION VIA GITHUB

In order to enable your own GitHub application, first you will need to create one and retrieve the corresponding credentials.

There are two different ways you can configure authentication via GitHub:

- Using the 3scale GitHub application (enabled by default for hosted 3scale accounts)
- Using your own GitHub application (for on-premises installations)

To make changes to this default configuration, you can go to your 3scale Admin Portal, in **Audience > Developer Portal > SSO Integrations** you will see the following screen:

Integration	State
<a href="#">GitHub</a>	Hidden
<a href="#">★ Auth0</a>	Hidden
<a href="#">Red Hat Single Sign-On</a>	Hidden

Click on **GitHub** to access the configuration screen:

## GitHub

[Edit](#)

**Published:**

**Branding:** 3scale branded

**Authentication Flow** [Test](#)

From this screen you can:

1. Make the GitHub authentication available or unavailable on your developer portal – to do so, simply check or uncheck the **Published** box.
2. Choose the 3scale branded GitHub application or add your own GitHub application – the 3scale GitHub application is enabled (published) by default. You can configure your own GitHub application by clicking on **Edit** and entering the details of the OAuth application created in GitHub ("Client" and "Client secret"). Please note that in order to make the integration work properly with your own GitHub application, you should configure the authorization callback URL of your GitHub application using the "Callback URL" that you should see after switching to the "custom branded" option (e.g. <https://yourdomain.3scale.net/auth/github/callback>).
3. Test that the configured authentication flow works as expected.

## 5.3. ENABLING AND DISABLING AUTHENTICATION VIA AUTH0



## NOTE

This feature is only available on the Enterprise plans.

In order to have your developers authenticate using Auth0, you first need to have a valid Auth0 subscription.

Authentication via Auth0 is not enabled by default. If you want to use your Auth0 account in conjunction with 3scale to manage the access to your developer portal, you can follow these steps to configure it:

Go to your 3scale Admin Portal, in **Audience > Developer Portal > SSO Integrations** click on **Auth0**.

On this configuration screen, you will need to add the details of your Auth0 account. Once you have entered the client ID, client secret, and site, check the **Published** box and click on **Create Auth0** to make it available on your developer portal.

## 5.4. ENABLING AND DISABLING AUTHENTICATION THROUGH RED HAT SINGLE SIGN-ON



## NOTE

This feature is only available on enterprise plans.

Red Hat single sign-on is an integrated sign-on solution (SSO) that, when used in conjunction with 3scale, allows you to authenticate your developers using any of the available Red Hat single sign-on identity brokering and user federation options.

Refer to the [supported configurations](#) page for information on which versions of Red Hat Single Sign-On are compatible with 3scale.


### 5.4.1. Before You Begin





Before you can integrate Red Hat Single Sign-On with 3scale, you must have a working Red Hat Single Sign-On instance. Refer to the Red Hat Single Sign-On documentation for installation instructions: [Installing Red Hat single sign-on 7.2](#)

### 5.4.2. Configuring RH SSO to authenticate the Developer Portal



Perform the following steps to configure Red Hat Single Sign-On:

1. Create a realm as described in the [Red Hat Single Sign-On documentation](#).
2. Add a client by going to **Clients** and clicking on **Create**.
3. Fill in the form considering the following fields and values:
  - **Client ID:** type the desired name for your client.
  - **Enabled:** switch to **ON**.
  - **Consent Required:** switch to **OFF**.
  - **Client Protocol:** select *openid-connect*.
  - **Access Type:** select *confidential*.
  - **Standard Flow Enabled:** switch to **ON**.
  - **Root URL:** type your 3scale admin portal URL. This should be the URL address that you use to log in into your developer portal, e.g.: <https://yourdomain.3scale.net> or your custom URL.
  - **Valid Redirect URLs:** type your developer portal again by /\* like this: [https://yourdomain.3scale.net/\\*](https://yourdomain.3scale.net/*).  
All the other parameters should be left empty or switched to **OFF**.
4. Get the client secret with the following steps:
  - Go to the Client you just created.
  - Click on **Credentials** tab.
  - Select *Client Id and Secret* in **Client Authenticator** field.

Account 


Settings **Credentials** Roles Mappers  Scope  Revocation Sessions  Offline Access 

---

Client Authenticator  Client Id and Secret 

Secret

---

Registration access token 

5. Configure the **email\_verified** mapper. 3scale requires that the **email\_verified** claim of the user data is set to **true**. In order to map the *"Email Verified"* user attribute to the **email\_verified** claim:
  - Go to the **Mappers** tab of the client.
  - Click **Add Builtin**.



Master > Clients > 3scale-dev-portal

3scale-dev-portal

Settings Roles **Mappers** Scope Revocation Sessions Offline Access Installation

Search... [Q] Create Add Builtin

Name	Category	Type	Actions	
full name	Token mapper	User's full name	Edit	Delete
given name	Token mapper	User Property	Edit	Delete
email	Token mapper	User Property	Edit	Delete
username	Token mapper	User Property	Edit	Delete
family name	Token mapper	User Property	Edit	Delete

- Select the *email verified* option, and click **Add selected** to save the changes.

Master > Clients > 3scale-dev-portal > Mappers > Add Builtin Protocol Mappers

Add Builtin Protocol Mapper

Search... [Q]

Name	Category	Type	Add
email verified	Token mapper	User Property	<input checked="" type="checkbox"/>
locale	Token mapper	User Attribute	<input type="checkbox"/>
address	Token mapper	User Address	<input type="checkbox"/>
gss delegation credential	Token mapper	User Session Note	<input type="checkbox"/>

Add selected

If you manage the users in the Red Hat Single Sign-On local database, make sure that the *Email Verified* attribute of the user is set to **ON**.

If you use [User Federation](#), in the client created previously for 3scale SSO integration, you can configure a hardcoded claim by setting the token name to **email\_verified** and the claim value to **true**.

6. Optionally, configure the **org\_name** mapper.

When a user signs up in 3scale, the user is requested to fill in the signup form with the Organization Name value. In order to make the signup via Red Hat Single Sign-On transparent for the user by not requiring to fill in the signup form on the developer portal, you need to configure an additional **org\_name** mapper:

- Go to the **Mappers** tab of the client.
- Click **Create**.
- Fill the mapper parameters as follows:
  - **Name:** type any desired name, e.g. **org\_name**.
  - **Consent Required:** switch to **OFF**.
  - **Mapper Type:** select *User Attribute*.
  - **User Attribute:** type *org\_name*.
  - **Token Claim Name:** type *org\_name*.
  - **Claim JSON Type:** select *String*.
  - **Add to ID token:** switch to **ON**.
  - **Add to access token:** switch to **ON**.
  - **Add to userinfo:** switch to **ON**.

- **Multivalued:** switch to **OFF**.
- Click **Save**.

#### Create Protocol Mapper

Protocol	<input type="text" value="openid-connect"/>
Name	<input type="text" value="org_name"/>
Consent Required	<input type="checkbox"/> OFF
Mapper Type	<input type="text" value="User Attribute"/>
User Attribute	<input type="text" value="org_name"/>
Token Claim Name	<input type="text" value="org_name"/>
Claim JSON Type	<input type="text" value="String"/>
Add to ID token	<input checked="" type="checkbox"/> ON
Add to access token	<input checked="" type="checkbox"/> ON
Multivalued	<input type="checkbox"/> OFF
	<input type="button" value="Save"/> <input type="button" value="Cancel"/>

If the users in Red Hat Single Sign-On have the attribute **org\_name**, 3scale will be able to create an account automatically. If not, then the user will be asked to indicate Organization Name before the account can be created. Alternatively, a mapper of type *Hardcoded claim* can be created to set the organization name to a hardcoded value for all users signing in with the Red Hat Single Sign-On account.

- To test the integration, you need to add a user. To achieve this, navigate to **Users**, click **Add user**, and fill the required fields. Note that when you create an User in Red Hat single sign-on the Email Verified attribute (**email\_verified**) should be set to **ON**, otherwise the user will not be activated in 3scale.

## Using Red Hat Single Sign-On as an identity broker

You can use Red Hat Single Sign-On as an identity broker or configure it to federate external databases. For more information about how to configure these, see the Red Hat Single Sign-On documentation for [identity brokering](#) and [user federation](#).

If you decide to use Red Hat Single Sign-On as an identity broker, and if you want your developers to be able to skip both the RH-SSO and 3scale account creation steps, we recommend the following configuration. In the example provided, we are using GitHub as our identity provider.

- In Red Hat Single Sign-On, after configuring GitHub in **Identity providers**, go to the tab called **Mappers** and click **Create**.

#### Organization Name

ID	<input type="text"/>
Name *	<input type="text" value="organization name"/>
Mapper Type	<input type="text" value="Attribute Importer"/>
Social Profile JSON Field Path	<input type="text" value="company"/>
User Attribute Name	<input type="text" value="org_name"/>
	<input type="button" value="Save"/> <input type="button" value="Cancel"/>

- Give it a name so you can identify it.

3. In **Mapper Type** select *Attribute Importer*.
4. In **Social Profile JSON Field Path** add `company`, which is the name of the attribute on GitHub.
5. In **User Attribute Name** add `org_name`, that is how we called the attribute in Red Hat Single Sign-On.



#### NOTE

Red Hat Single Sign-On requires first and last name as well as email as mandatory fields. 3scale requires email address, username, and organization name. So in addition to configuring a mapper for the organization name, and for your users to be able to skip both sign up forms, make sure that:

- In the IdP account, they have their first name and last name set.
- In the IdP account, their email address is accessible. For example, in GitHub, if you set up your email address as private, it is not shared.

### 5.4.3. Configuring 3scale API Management to authenticate the Developer Portal

As an API provider, configure 3scale to allow authentication for the Developer Portal using Red Hat single sign-on.



#### NOTE

Authentication through Red Hat single sign-on is not enabled by default. Red Hat single sign-on is available for only enterprise 3scale accounts, so you need to ask your account manager to enable the authentication via Red Hat single sign-on.

#### Prerequisites

- Your enterprise 3scale account is set up to enable Red Hat single sign-on.
- You know the following details after [Configuring Red Hat single sign-on to authenticate the Developer Portal](#):
  - **Client**  
Name of your client in Red Hat single sign-on.
  - **Client secret**  
Client secret in Red Hat single sign-on.
  - **Realm**  
Realm name and URL address to your Red Hat single sign-on account.

#### Procedure

1. In the 3scale Admin Portal, select **Audience > Developer Portal > SSO Integrations**
2. Click **Red Hat Single Sign-On**
3. Specify the details of the Red Hat Single Sign-On client that you have configured in [Configuring Red Hat Single Sign-On to authenticate the Developer Portal](#): client, client secret and realm.

4. To save your changes, click **Create Red Hat Single Sign-On**

## CHAPTER 6. RED HAT SINGLE SIGN ON FOR DEVELOPER PORTAL

Red Hat Single Sign On allows you to manage access control of multiple independent systems. By following this guide, you will be able to allow users that are logged in to your system to log in automatically to your 3scale-powered Developer Portal without being prompted to log in again.

This article shows how existing user credentials of your website can be used to automatically log in to your 3scale-powered Developer Portal.

This feature is meant for API providers that already own the identity of their API consumers (username and password) – such as when the API provider is also the identity provider.

### 6.1. CREATING USERS IN THE 3SCALE API MANAGEMENT PLATFORM

First of all, the API consumer must have an account in your Developer Portal. You can import your users to 3scale using the Account Management API or create them manually. Find the Account Management API in the 3scale ActiveDocs, available in your Admin Portal, under the **Documentation (question mark icon (?) in the top right corner) → 3scale API Docs** section.

### 6.2. REQUESTING A LOGIN LINK

Once the user exists, you can use an API request call to generate a URL with a built-in SSO token:

```
curl -X POST -d "provider_key=YOUR_PROVIDER_KEY&username=USERNAME&expires_in=60"
https://YOUR_ADMIN_PORTAL.3scale.net/admin/api/sso_tokens.xml
```

There are 2 parameters in this call: `username` to specify who you are requesting the token for and `expires_in` which is the number of seconds that the token will be valid for (it defaults to 10 minutes).

You can also pass an additional parameter `redirect_url` with a location to redirect the user after a successful login. This parameter should be [percent encoded](#). The XML response will contain a URL with a secret token included:

```
<?xml version="1.0" encoding="UTF-8"?>
<sso_url>
https://YOUR_DEVELOPER_PORTAL/session/create?
expires_at=1365087501&token=Q0dNWGtjL2h2MnloR11yWmNwazVZY0NhenlabnBoRUNaNUlyWjZa
VG8wMnBGdVNHt0VGN1NUb3FRc1pwSnRrcIBZSTlwOUFwRkVTc3NuK1JTbjUrMEE9PS0tY1ZrOG
FldzFJNkxna1hrQzQyZ0NGQT09--712f2990ac9248ab4b8962be6467fb149b346000
</sso_url>
```



#### NOTE

You can pass either `user_id` or `username` to identify the 3scale user. Typically, the `username` will be the same for your system and 3scale portal. In that case, using the `username` should be easy since it does not require any additional information to be stored on your side. However, if you need to do some pairing and machine processes to the URLs anyway, you might be better off with `user_id`.

### 6.3. REDIRECTING USERS WITH AUTOMATIC LOGIN

The response contains an Red Hat single sign-on login URL with a token:

```
https://YOUR_DEVELOPER_PORTAL/session/create?  
expires_at=1365087501&token=Q0dNWGtjL2h2MnloR11yWmNwazVZY0NhenlabnBoRUNaNUlyWjZa  
VG8wMnBGdVNHt0VGN1NUb3FRc1pwSnRrcIBZSTlwOUFwRkVTc3NuK1JTbjUrMEE9PS0tY1ZrOG  
FldzFJNkxna1hrQzQyZ0NGQT09--712f2990ac9248ab4b8962be6467fb149b346000
```

The URL contains all the required information for the 3scale Developer Portal SSO to log you in. You can embed it directly into web. However, the URL can expire before the user clicks it, so it is recommended to have a generic link on your page that will dynamically request a fresh SSO URL and redirect to it. This way, the user will be seamlessly logged in to the Developer Portal.



#### NOTE

The URL address needs to be unescaped. If you want to try it manually in a browser, remember to replace the **&amp;** with **&** in your browser. Also any **%** encodings in the token need to be replaced by their unescaped character.

## CHAPTER 7. RESTRICTED CONTENT

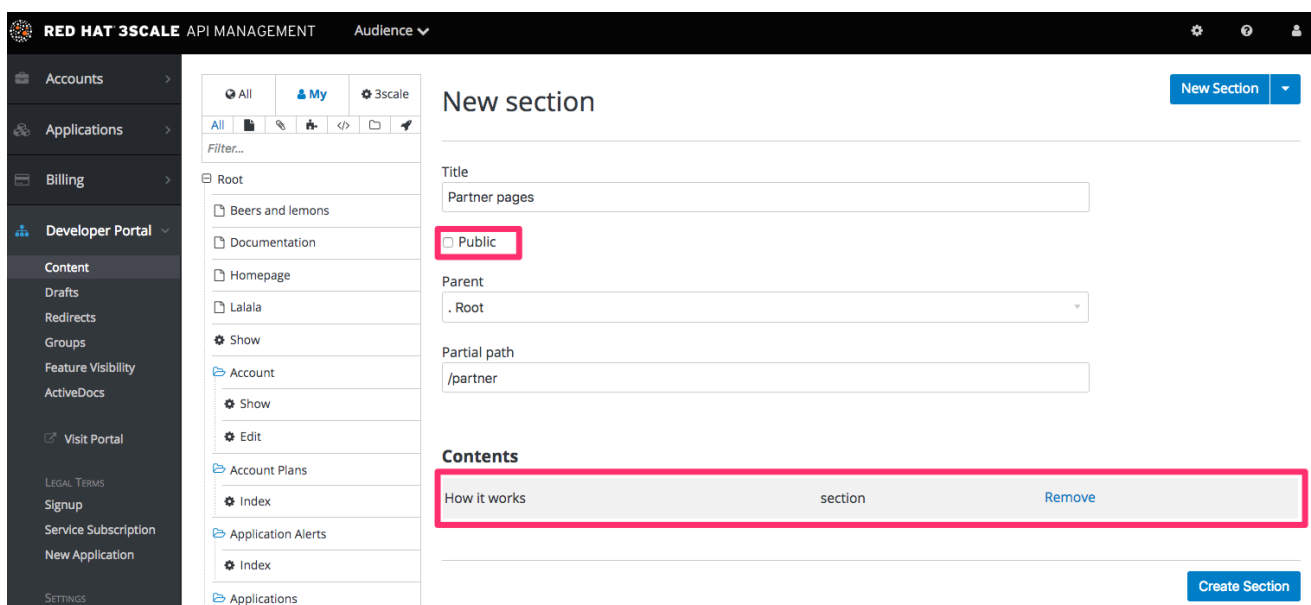
Here you can learn how to have content in your Developer Portal that is only visible for some users.

You may need to have some pages of your Developer Portal that are only accessible for a specific group of developers, either part of a page or items in a certain menu. Both goals are achievable through the two techniques introduced below.

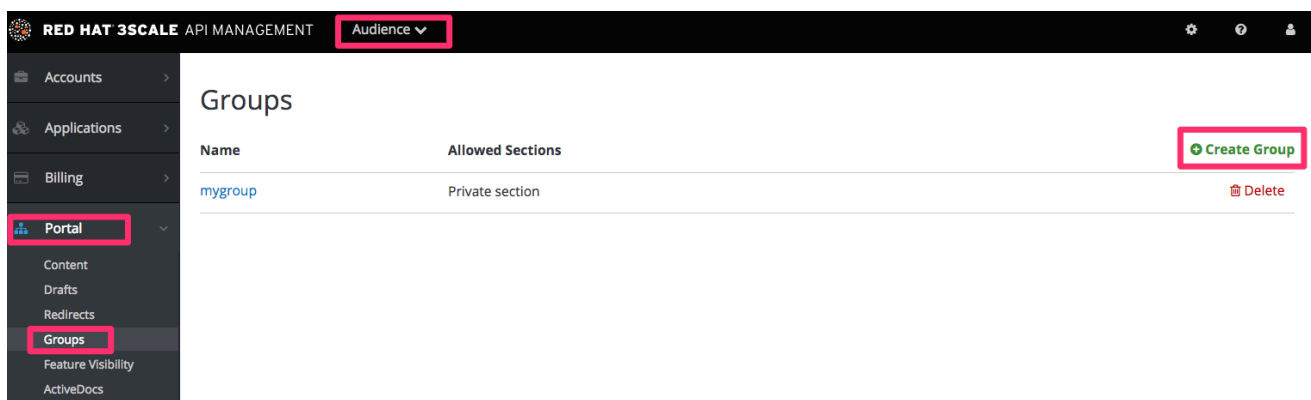
### 7.1. RESTRICTED PAGES

When creating restricted sections, it is useful to do it so that each section maps to a logical group of users. For this example, assume that there is a group of developers called "partners".

Create a new section in the Developer Portal for every page or group of pages that you want to restrict access to. Uncheck the "public" status field. Then drag and drop any pages you want inside this section.



Create a group and give it access to the section you created.



Now every time you have to grant one of your users access to this section, all you have to do is assign them to this group. To do this, go into the corresponding account detail page, then to "Group Permissions." Once there, check the boxes for the sections you want to allow.

RED HAT 3SCALE API MANAGEMENT Audience ▾

Accounts ▾  
Listing  
Account Plans  
Subscriptions

Account 'Metro' | 1 Application | 1 User | 0 Invitations | 0 Group Memberships | 0 Invoices | 1 Service Subscription

## Groups of 'Metro'

Groups

mygroup (Private section)

## 7.2. RESTRICTED BLOCKS OF CONTENT

Liquid tags are a very powerful way to customize your Developer Portal. Use them here to hide or display parts of a page based on a condition. 3scale allows you to create custom fields for accounts, applications, and users. You can leverage this to store information that is useful for you as the API provider. Here you'll create a custom field attached to all accounts and use it to indicate whether a given account is a partner or not. You can create this field by going to **Audience > Account > Field Definitions**. Add a field to the Account section, and mark it as hidden so it will not be displayed on the signup page or anywhere else on the portal.

RED HAT 3SCALE API MANAGEMENT Audience ▾

## Fields Definitions

Here you can manage all the information you gather from your users. You can add new fields and change the existing ones; making them Hidden, Read Only, or Required. You can change the text your users see when viewing or entering data (shown here between quotes). Drag and drop the fields to set the order in which they will be shown.

Account

org_name	"Organization/Group Name"	Required	Create Edit
Partner	"partner"	Hidden	Edit

With the custom field in place, you are now able to show special content to partners by wrapping it in a conditional like in the following snippet:

```

{{ if current_account.extra_fields.partner == 'true' }}
  // content only accessible to partners
{{ endif }}

```

Or use the inverse logic if it suits your case better:

```

{{ unless current_account.extra_fields.partner == 'true' }}
  // content forbidden for partners
{{ endunless }}

```

From here on, whenever you want to show these pieces of hidden content to a user, all you need to do is type in 'true' in the partner field of their account detail page.

## 7.3. AUTOMATING THE CONFIGURATION OF EXTRA FIELDS

You can provide access to restricted content to developers based on a change in state. For instance, when they upgrade their application plan.



Streamline the process of access provision by using [webhooks](#) together with the Account Management API. The Account Management API is in the 3scale ActiveDocs, available in your Admin Portal:

1. Click on **Documentation**, a question mark icon (?) located in the upper right side of the window.
2. Choose **3scale API Docs**.
3. Get the new plan of the developer who will access the restricted content, by checking the message sent by the webhook request.
4. Based on the developer's new plan, grant access to the private content by calling the API to update the *partner* field.

## 7.4. REQUIRING USER LOGIN

In addition to the two ways to restrict access to content described above, there is another technique that can be useful: requiring a logged-in user.

This is very easy to achieve using Liquid tags. All you have to do is wrap the content that will be available only for logged-in users inside the following conditional:

```

{{ if current_user }}
  // only visible if the user is logged in
{{ endif }}
```

## CHAPTER 8. EMAIL TEMPLATES

By the end of this section, you will have edited and saved a custom email template.

You can completely customize the content of all standard email communication with developers, allowing you to closely match the workflows you have set up for your Developer Portal.

### 8.1. CUSTOMIZING EMAIL TEMPLATES

#### 8.1.1. Define your workflows before email configuration

There are a lot of email template options, only a subset of which will be relevant for your workflows. Save yourself time by making sure you're happy with your workflows before beginning to edit the email templates. This way, you'll only edit the templates that you'll actually use.

#### 8.1.2. Test your workflow and identify active email templates

Perform a dry run of your finalized workflows, making sure to test all the possible branches, such as approval and rejection. Then, identify each email notification that your test developer account receives to determine what to edit in the next step.

#### 8.1.3. Edit and save your custom template

The first time you edit a template, you will actually create a custom template. Then in subsequent edits, you will save your changes.



#### WARNING

There is no version control. Red Hat recommends you make a local copy if you want to be able to revert changes.

You can use liquid tags for dynamic content in your email. We especially recommend you make backups when you make changes to the liquid tags.

RED HAT 3SCALE API MANAGEMENT Audience

## Edit template "Buyer Account approved"

You can use Liquid tags to set the email headers or disable sending. Read more in the [liquid documentation](#).

Subject  
use 3scale default

Bcc

Cc

Reply to

From  
use 3scale default

**Liquid tags for dynamic email customisation**

```

1 {% email %}{% do_not_send %}{% endemail %}
2
3 Dear {{ user.display_name }},
4
5 {{ provider.name }} has approved your signup for the {{ provider.name }} API.
6
7 You may now view and manage your app/API key at https://{{ provider.domain }}/admin/
8
9 If you have problems logging into the account please contact {{ provider.support_email }}.
10
11 Sincerely,
12 The {{ provider.name }} API Team
13

```

Disable Sending Snippet Save

### 8.1.4. Repeat for all templates in your workflows

Complete these same steps until you've covered all possible branches for your workflows.

## 8.2. MORE INFORMATION

- Before customizing your email templates, it is best to have the [signup flows](#) fully finalized and tested.
- If you intend to change any of the liquid tags within the email templates, be sure to read up on the [liquid reference documentation](#).

## CHAPTER 9. LIQUIDS: DEVELOPER PORTAL

This section contains information about Liquid formatting tags and how they work in the 3scale system, including the different elements of the markup, the connections between them, and short examples of how to use them in your Developer Portal.

To learn the basics about Liquids, see the [Liquid reference](#).

### 9.1. USING LIQUIDS IN THE DEVELOPER PORTAL

This section explains how to enable liquid markup processing in layouts and pages.

#### 9.1.1. Enabling Liquids

Liquid markup processing is enabled by default for all partials and email templates. Enabling them on layouts is done by ticking the checkbox right under the system\_name input field. However, to enable them on pages, you will have to go to the advanced options section of the page.

##### ▼ ADVANCED OPTIONS

System name

Content type

text/html

Can be [HTML](#), [CSS](#), [Javascript](#) or an arbitrary MIME type.

Liquid enabled

Process Liquid tags and drops?

Handler

Do you use any markup language?

Tag list

Just expand the **Advanced options** section and mark the Liquid enabled checkbox. From now on, all the liquid markup will be processed by the internal engine, and the Developer Portal built-in editor will also add code highlighting for liquid.

#### 9.1.2. Different use on pages, partials, and layouts

The use of liquids usually differs slightly between pages, partials and layouts. Within pages, liquids are single-use elements; while liquids with partials and layouts are the reusable elements of the Developer Portal. This means that instead of applying multiple layouts or partials with small changes to different pages, you can add some logic liquid tags, and alter the layout depending on the page the user is on.

```
<!-- if we are inside '/documentation' URL -->
<li class="{% if request.request_uri contains '/documentation' %}active{% endif %}"><!-- add the
active class to the menu item -->
```

```
<a href="/documentation">Documentation</a>
</li>
```

### 9.1.3. Use with CSS/JS

Liquid markup does not just work with HTML, you can easily combine it with CSS and/or JavaScript code for even more control. To enable liquid in a stylesheet or JS, create them as a page and follow the same steps as if you were enabling it for a normal page. Having done that, you'll be able to add some conditional markup in CSS or use the server-side data in JavaScript. Just remember to set the content type of the page as CSS or JS.

## 9.2. USAGE OF LIQUIDS IN EMAIL TEMPLATES

This section explains how you can use liquid tags to customize email templates.

### 9.2.1. Differences from Developer Portal

As previously mentioned, liquid tags can also be used to customize the email templates sent to your users. All the general rules for writing liquid mentioned before also apply to the email templates, with some exceptions:

- There is no commonly shared list of variables that are available on every template. Instead, you will have to do some testing using the previously mentioned **{% debug:help %}** tag.
- Since emails are by nature different from web pages, you will have limited or no access to some tags. For example, **{{ request.request\_uri }}** will not make sense, as an email does not have a URL.

## 9.3. TROUBLESHOOTING

This troubleshooting section will help you debug and fix typical errors that might occur.

### 9.3.1. Debugging

If something is not working as intended, but is saved correctly, check the following:

- All the tags are closed correctly.
- You are referring to variables available on the current page.
- You are not trying to access an array – for example `current_account.applications` is an array of applications.
- The logic is correct.

### 9.3.2. Typical errors and ways to solve them

- If the document cannot be saved due to a liquid error, it is usually because some tags or drops were not closed correctly. Check that all your **{% %}** and **{{ }}** tags were properly closed and that the logic expressions, for example, *if*, *for* and so on, are terminated correctly with *endif*, *enfor*. Normally if this is the case, an error will be displayed at the top of the page above the editor with a descriptive error message.
- If everything saved correctly and you do not see any effect, check that you are not referring to

an empty element and you are not using a logic tag to display content. `{% %}` will never render any content, besides usage in tags which is already an alias of a more complex set of tags and drops.

- If only a `#` symbol is displayed, it means that you have tried to display an element that is an array. Check the section on the [liquid hierarchy](#).

### 9.3.3. Contacting support

If you still have a problem, you can open a new case via the [Red Hat Customer Portal](#).

## CHAPTER 10. LIQUIDS: EMAIL TEMPLATES

3scale offers features to customize the email templates with your organization's own messaging and terminology. You can also take advantage of liquid drops to display personalized information for each of your customers.

Similar to how liquid drops are used in the Developer Portal, every email template has its own context. This means that liquid drops available in one email template may not necessarily be available for other email templates.

This reference outlines which liquid drops are available where, with email templates grouped together by subject matter and the set of liquid drops that they support.

### 10.1. ACCOUNT MANAGEMENT

These email templates fall under the account management category:

- Buyer Account confirmed.
- Buyer Account approved.
- Buyer account rejected.

For these templates, you can use the following liquid drops:

- **user** ⇒ **User**
- **domain** ⇒ **String**
- **account** ⇒ **Account**
- **provider** ⇒ **Provider**
- **support\_email** ⇒ **String**

Additionally, the following template:

- Password recovery for buyer has access to the these liquid drops:
- **user** ⇒ **User**
- **provider** ⇒ **Provider**
- **url** ⇒ **url**

The email to invite additional users to an account:

- Invitation has access to:
- **account** ⇒ **Account**
- **provider** ⇒ **Provider**
- **url** ⇒ **url**

### 10.2. CREDIT CARD NOTIFICATIONS

- Credit card expired notification for provider.
- Credit Card expired notification for buyer.

You can use the following liquid drops:

- **user\_account** ⇒ **Account**
- **account** ⇒ **Account**
- **provider\_account** ⇒ **Provider**
- **provider** ⇒ **Provider**

### 10.3. LIMIT ALERTS

- Alert notification for provider ( $\geq 100\%$ )
- Alert notification for buyer ( $\geq 100\%$ )
- Alert notification for provider ( $< 100\%$ )
- Alert notification for buyer ( $< 100\%$ )

have access to:

- **application** ⇒ **Application**
- **account** ⇒ **Account**
- **provider** ⇒ **Provider**
- **service** ⇒ **Service**
- **alert** ⇒ **Alert**

### 10.4. APPLICATIONS

The following email templates all deal with application and application plan notifications.

- Application created for provider has access to:
- **url** ⇒ **url**

Application plan change request notification email templates:

- Plan change request for buyer.
- Plan change request for provider.

They have access to:

- **application** ⇒ **Application**
- **provider** ⇒ **Provider**
- **account** ⇒ **Account**



- **user** ⇒ **User**
- **plan** ⇒ **Plan**
- **credit\_card\_url** ⇒ **credit\_card\_url**

The following email templates contain a number of available drops, such as:

- Application plan changed for buyer.
- Application plan changed for provider.
- Application trial period expired for buyer.

They have access to:

- **provider** ⇒ **Provider**
- **account** ⇒ **Account**
- **user** ⇒ **User**
- **plan** ⇒ **Plan**

As well as all of the above liquid drops, the following application plan messages:

- Application suspended for buyer.
- Application accepted for buyer.
- Application rejected for buyer.
- Application contract cancelled for provider.

have the additional liquid drops listed:

- **application** ⇒ **Application**
- **service** ⇒ **Service**

More liquid drops accumulate for the following email templates for application keys:

- Application key created for buyer.
- Application key deleted for buyer.
- **key** ⇒ **key**

## 10.5. INVOICING

The following email template:

- Review invoices prior to charging for provider has access to:
- **provider** ⇒ **Provider**
- **url** ⇒ **String**

Additionally, the following templates:

- Invoice charge failure for provider without retry.
- Invoice upcoming charge for buyer.
- Invoice charge failure for provider with retry.
- Invoice charge failure for buyer without retry.
- Invoice charged successfully for buyer.
- Invoice charge failure for buyer with retry.

share the following liquids:

- **account** ⇒ **Account**
- **provider** ⇒ **Provider**
- **cost** ⇒ **cost**
- **invoice\_url** ⇒ **invoice\_url**
- **payment\_url** ⇒ **payment\_url**

## 10.6. SERVICES

The following email templates:

- Service contract cancelled for provider.
- Service trial period expired for buyer.
- Service plan changed for provider.
- Service contract suspended for buyer.

have access to:

- **provider** ⇒ **Provider**
- **account** ⇒ **Account**
- **user** ⇒ **User**
- **plan** ⇒ **Plan**

As well as the above liquid drops, the following service templates:

- Service created for provider.
- Service accepted for buyer.
- Service rejected for buyer.

have the additional liquid drops listed:

- **service** ⇒ **Service**
- **service\_contract** ⇒ **Contract**
- **subscription** ⇒ **Contract**

## 10.7. SIGNUP

The following email templates:

- Sign-up notification for provider.
- Sign-up notification for buyer.

have access to:

- **user** ⇒ **User**
- **provider** ⇒ **Provider**
- **url** ⇒ **activate\_url**

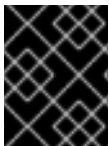
# CHAPTER 11. CUSTOMIZING THE DEVELOPER PORTAL LAYOUT

You can customize the look and feel of the entire Developer Portal to match your own branding. A standard CSS stylesheet is available to provide an easy starting point for your customizations. To create layout templates, use the code for **Main layout** as your starting point.

In this tutorial, you'll add your own CSS customizations to your Developer Portal and reload it to put your new styling changes live.

## 11.1. CREATING A NEW CSS FILE

There is a default stylesheet, **default.css**. This is quite large and complex, so rather than extend it, it is better to create your own stylesheet for any of your own customizations to overwrite the defaults. You create a new stylesheet the same way you create a page. Remember to choose an appropriate MIME content type in the advanced page settings.



### IMPORTANT

Make sure that the selected layout is blank. Otherwise the page layout HTML will obscure the CSS rules.

## 11.2. LINKING THE STYLE SHEET INTO YOUR PAGE LAYOUT

Add the link to your custom CSS in each of your layout templates (or in a partial if you have a common HEAD section) after the link to bootstrap.css. For example:

```
<link rel="stylesheet" href="/stylesheets/custom.css">
```

Now enjoy the beauty of your own unique branding!

## 11.3. DEFINING PAGE LAYOUT TEMPLATES

The general idea is to define a separate layout for each of the different page styles in your portal. There is one standard layout called **Main layout** when you start. Do not make any changes to this layout until you are an expert at using the Developer Portal because this layout is used by all system-generated pages.

Typically, you want a unique style for the home page of your portal. The **Main layout** template is a starting point for your customizations. To create a page layout template:

1. Open **Main layout** and copy its code to the clipboard.
2. Create a new layout, give it a title, a system name, and select **Liquid enabled**.
3. Paste the **Main layout** code into your new layout.
4. Remove the sidebar menu by deleting this line from your new layout:

```
{% include 'submenu'%}
```

5. Customize the code to create your layout template.

## CHAPTER 12. CHANGE BUILT-IN PAGES

By the end of this section, you will be able to modify, and configure the visibility of any element in the system-generated pages.

Some elements generated by the system are not possible to modify from the Developer Portal, such as the Signup, Dashboard, and Account pages. This guide shows how to customize with CSS and JavaScript the content on these pages.

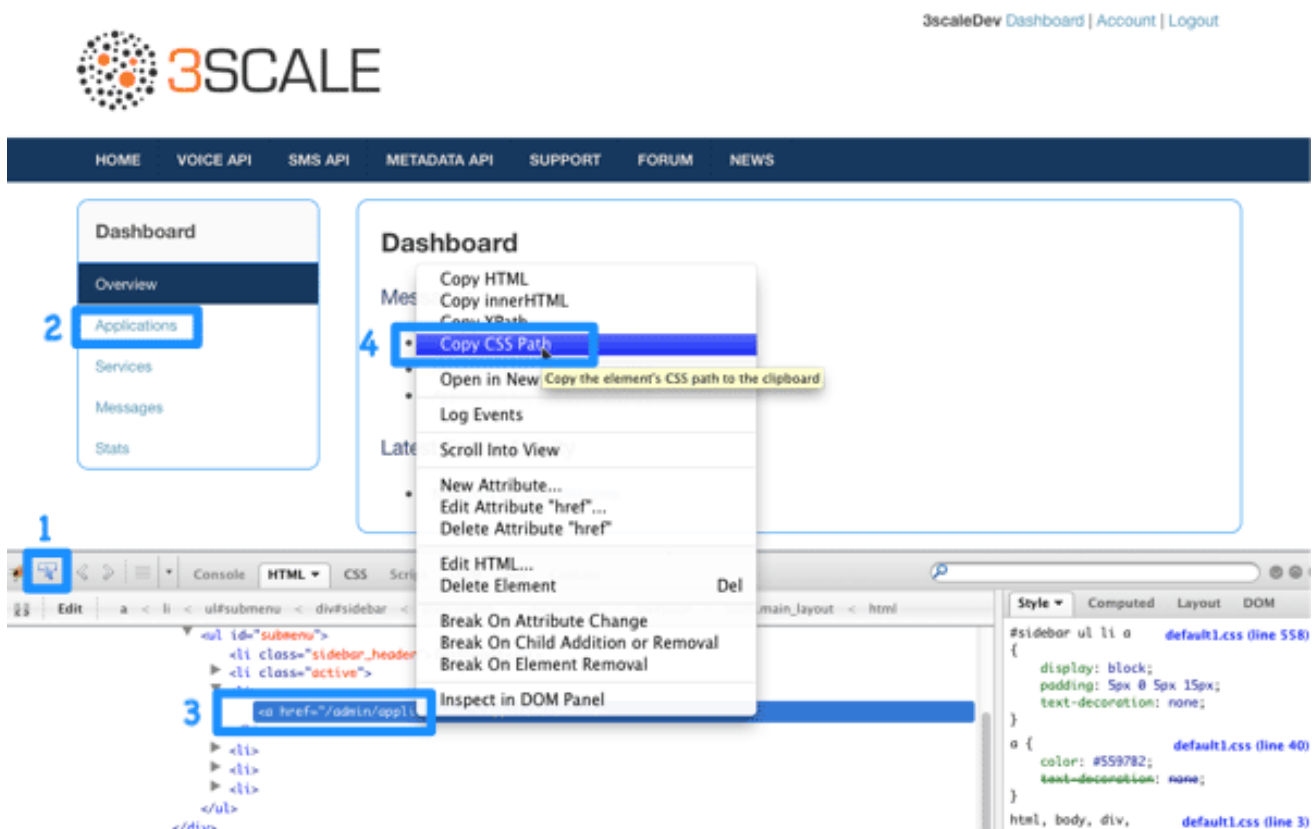
The system-generated pages follow backend rules for access and visibility so their URLs must be specific and hard coded values. A Knowledge Base article in the customer portal provides a [list of system-generated pages and their URLs](#).

### CAUTION

The 3scale system-generated pages are subject to change, although infrequently. These changes may break any customizations that you implement following this guide. If you can avoid using these hacks, please do so. Before you continue, be sure that you can monitor for any disruptive changes, and do the necessary maintenance work to keep your portal functioning.

### 12.1. IDENTIFY THE ELEMENTS

The first and most important thing to do is identify what you want to hide. To do that, use Firebug (or any other developer tools such as Chrome Developer tools or Opera Dragonfly). Choose the desired element, and in the console, right click on it and select Copy CSS path. This way you save the exact CSS path to make it easy to manipulate. Remember, if the element is a part of the sidebar navigation widget, you will also have to specify which position in the list. For this, you can use either the "+" selector (for example, to choose 3rd li element: `ul + li + li + li`) or the `:nth-child(n)` CSS3 pseudoclass.



### 12.2. MODIFY OR HIDE THE ELEMENTS

Now, having identified the elements, you can change their display settings. Depending on the type of element, you can choose from two possible methods: CSS manipulation or jQuery script. CSS manipulation is more lightweight and reliable, but does not work well for some kinds of elements that exist on a number of pages, for example, the 3rd element in the sidebar of the Admin Portal's Dashboard also exists in the Account section but has a different value. Some trickier implementations require use of CSS3 which is not supported by old browsers. In the next two steps, you will see both of these approaches.

### 12.3. OPTION A: CSS

As an example, try to hide the latest element from the Dashboard page. Following the first step, you have identified its CSS path as:

```
#three-scale .dashboard_bubble
```

Keep in mind that it is the second box with the same path, so you will use the "+" selector. Your path will now look like this:

```
.main_layout #three-scale .dashboard_bubble + .dashboard_bubble  
/* or */  
.main_layout #three-scale .dashboard_bubble:nth-child(1)
```

Changing display property to none makes that box invisible:

```
.main_layout #three-scale .dashboard_bubble:nth-child(1) {  
  display: none;  
}
```

### 12.4. OPTION B: JQUERY

If you have a trickier element to hide such as a sidebar menu element, it is better to use some jQuery. The CSS path of these elements is identical on the Dashboard and Account sections, and you do not want to hide elements in both sections. So choose the element based on the CSS path and the content. In this example, assume you want to hide the messages section from the Dashboard's sidebar. Your CSS path is:

```
#three-scale #submenu li a
```

In order to match the content, use the `.text()` function. Also, include the code inside the document's head and inside the ready function, so it is run after all the content has been generated.

The resulting code snippet will look like this:

```
$(function() {
  $('#three-scale #submenu li a').each(function() {
    if ($(this).text() == "Messages")
      $(this).parent().css('display', 'none');
  });
});
```

This is not the only solution. It just shows one possible way of doing it. The same example could be done using pure CSS with CSS3 selectors basing on the attributes values. For the complete CSS3 selectors specification, take a look [here](#).

## CHAPTER 13. SETTING TERMS AND CONDITIONS

When you allow developers to sign up for your API, you will probably want to get them to agree to your Terms and Conditions to make some of your policies clear before you grant them access.

There may be different versions of your Terms and Conditions you want developers to abide by. These are easy to set up at different points throughout the registration process. For example:

1. Signup Terms and Conditions.
2. Application Terms and Conditions.
3. Service/subscription Terms and Conditions.
  - Only available when you have multiple services.

Additionally, if you are charging for use of your API, you may want to make your credit card policies explicit. 3scale provides an easy way to set up the following kinds of credit card policy URLs:

1. Legal Terms
2. Privacy
3. Refunds

### 13.1. TERMS AND CONDITIONS

This part of the workflow is easy to set up in the Admin Portal by following the steps below.

Go to **Audience > Developer Portal > Signup** where you will be presented with a blank page to populate with your signup legal terms. You can use any combination of HTML, JavaScript, and CSS. There is also some toggling code provided by clicking Insert toggling code. The content you write in this box will appear just above the **Sign Up** button on the Signup page of your Developer Portal.

**RED HAT 3SCALE** API MANAGEMENT Audience ▾

- Accounts >
- Applications >
- Billing >
- Portal** ▾
  - Content
  - Drafts
  - Redirects
  - Groups
  - Feature Visibility
  - ActiveDocs
  - Visit Portal
  - LEGAL TERMS**
  - Signup**
  - Service Subscription
  - New Application

### Legal Terms for Signup

When signing up, your developers have to accept these terms.

Use any combination of custom HTML, Javascript and CSS to craft your legal terms. You can also insert the most common one-line warning with Show/Hide toggle using the button below.

**Expert Note:** Legal terms are just partials included by default next to the submit button of the form. You can edit them [using the CMS](#) too. If you remove the `include` statement from the pages that use those snippets, these settings will no longer have any effect.

1



Once you have filled out your Terms and Conditions, save them by clicking Update.

If you have used the toggling code, it will display "By signing up you agree to the following Legal Terms and Conditions" followed by a link that toggles between showing and hiding the Terms and Conditions you specified.

The screenshot shows a sign-up form titled "SIGN UP". At the top, a message states "You are signing up to plan Big Data Bundle." Below this are five input fields: "ORGANIZATION/GROUP NAME", "USERNAME", "EMAIL", "PASSWORD", and "PASSWORD CONFIRMATION". A blue-bordered box highlights the text "By signing up you agree to the following Legal Terms and Conditions (show)". At the bottom right, there is a green "Sign up" button.

This is placed on the Signup page by default, but it is a partial (`signup_licence`) that can be included anywhere on your Developer Portal. To remove this from the Signup page, simply remove the `{% include 'signup_licence' %}` line from the page. Similarly, if you want to include it somewhere else, you can use the same partial by means of the snippet, which can be placed anywhere on your Developer Portal.

You might also want your users to accept another set of Terms and Conditions when they create a new application (`new_application_licence` partial) and/or when they subscribe to a new service (`service_subscription_licence` partial). To set these up, you can follow the same procedure outlined above.

## 13.2. CREDIT CARD POLICIES

You can also define other URLs where different policies reside. Set them up by going to **Audience > Billing > Credit Card Policies** and setting the path where your policy pages will be located.

RED HAT 3SCALE API MANAGEMENT Audience ▾

Accounts >

Applications >

Billing ▾

Earnings by Month

Invoices

SETTINGS

Charging & Gateway

Credit Card Policies

Portal >

Messages >

## Credit Card Policies

POLICY URLS FOR CREDIT CARD DETAILS

Path to Legal Terms page  
/termsofservice

Path to Privacy page  
/privacypolicy

Path to Refund page  
/refundpolicy

In order for these links to work, you will then need to create new pages in the Developer Portal.

RED HAT 3SCALE API MANAGEMENT Audience ▾

Accounts >

Applications >

Billing >

Developer Portal ▾

Content

Drafts

Redirects

Groups

Feature Visibility

ActiveDocs

Visit Portal

LEGAL TERMS

Signup

Service Subscription

New Application

## Legal Terms for Signup

When signing up, your developers have to accept these terms.

Use any combination of custom HTML, Javascript and CSS to craft your legal terms. You can also insert the most common one-line warning with Show/Hide toggle using the button below.

**Expert Note:** Legal terms are just partials included by default next to the submit button of the form. You can edit them using the CMS too. If you remove the `include` statement from the pages that use those snippets, these settings will no longer have any effect.

1

Once that is done, you can reference them using the URL's liquid drop. For example:

```
<a href="{{ urls.credit_card_terms }}">Legal Terms</a>
<a href="{{ urls.credit_card_privacy }}">Privacy</a>
<a href="{{ urls.credit_card_refunds }}">Refunds</a>
```

## CHAPTER 14. EXPORTING AND IMPORTING THE DEVELOPER PORTAL

As a 3scale API provider, you can export and import the Developer Portal for the following purposes:

- Creating backups.
- Keeping the Developer Portal in an external repository, for example, GitHub.
- Integrating the Developer Portal with other applications.

Use the Developer Portal API as a Content Management System (CMS) to import and export the Development Portal content. For this, perform the following steps to generate a key with enough permissions to use the Developer Portal API.

### Procedure

1. Navigate to **Account settings > Personal > Tokens** and click **Add Access Token**.
2. Name the access token and check **Developer Portal API**.
3. Choose permissions:
  - a. **Read only** allows retrieving Developer Content portal contents.
  - b. **Read and Write** allows retrieving and restoring Developer Portal contents.
4. Click **Create Access token**.
5. Copy and store the token information displayed.

Check the endpoints list navigating from the left panel to **Integrate > 3scale API Docs**. Then, scroll down to **Developer Portal API**. Use the token generated to call each endpoint and fill in the fields according to your needs.

### Developer Portal API endpoints considerations

The Developer Portal API available from 3scale 2.14 is not compatible with previous versions. Also, from 3scale 2.14, JSON is the only data format compatible for all requests and responses.

For each endpoint, you can perform the following actions:

- **GET** to read and list resources.
- **POST** to create and add resources.
- **PUT** to modify resources.
- **DELETE** to delete resources.



## NOTE

- Built-in objects cannot be deleted. Call the **GET /admin/api/cms/templates** endpoint with the **type=builtin\_page** parameter to get a list of builtin pages and the **type=builtin\_partial** parameter to get a list of builtin partials.
- To do a complete backup, you must call each content. There is no API endpoint that downloads a complete archive containing all files.
- If **content** is not sent, it does not return the published or draft content. Instead, it returns a summary with information like template name and section because the content is too long.

Use the details listed under each endpoint to refine its output after executing it. Consider the following for the listed parameters:

- All endpoints reject unsupported parameters; if unsupported parameters are sent, the request is canceled.
- **GET /admin/api/cms/templates** endpoint accepts a **content** parameter. By default, it returns a list of Developer Portal templates. To also get published and draft content, use **content=true** parameter.
- **GET /admin/api/cms/templates** endpoint accepts **type** and **section\_id** parameters to filter results.
- **GET /admin/api/cms/sections** endpoint accepts **parent\_id** parameter to filter results.
- **GET /admin/api/cms/files** endpoint accepts **section\_id** parameter to filter results.