



Red Hat 3scale API Management 2.6

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 2.6 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 © 2020 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[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

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

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] 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[®] 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 2.6.

Table of Contents

CHAPTER 1. DEVELOPER PORTAL	4
1.1. OVERVIEW OF THE DEVELOPER PORTAL	4
1.2. CONTENT	4
1.3. LAYOUTS AND PARTIALS	6
1.4. PORTLETS	7
1.4.1. Note	7
1.5. REDIRECTS AND CHANGES	8
CHAPTER 2. CUSTOM SIGNUP FORM FIELDS	9
CHAPTER 3. CONFIGURING SIGNUP FLOWS	15
3.1. REMOVING ALL APPROVAL STEPS	15
3.2. ENABLING ALL POSSIBLE DEFAULT PLANS	16
3.3. TESTING THE WORKFLOW	17
CHAPTER 4. MULTI-SERVICE SIGNUP	18
4.1. PREREQUISITES	18
4.2. INTRODUCTION	18
4.3. MULTI-SERVICE SIGNUP	18
4.3.1. Retrieving information about services	18
4.3.2. Configuring the signup columns	19
4.3.3. Configuring the subscription	19
4.3.4. Styling	19
CHAPTER 5. DEVELOPER PORTAL AUTHENTICATION	21
5.1. ENABLING AND DISABLING USERNAME/EMAIL AND PASSWORD	22
5.2. ENABLING AND DISABLING AUTHENTICATION VIA GITHUB	22
5.3. ENABLING AND DISABLING AUTHENTICATION VIA AUTH0	23
5.3.1. Note	24
5.4. ENABLING AND DISABLING AUTHENTICATION VIA RED HAT SINGLE SIGN-ON	24
5.4.1. Before You Begin	24
5.4.2. Configuring RH SSO	24
5.4.3. Configuring 3scale	28
CHAPTER 6. RED HAT SINGLE SIGN ON FOR DEVELOPER PORTAL	29
6.1. CREATING USERS IN THE 3SCALE PLATFORM	29
6.2. REQUESTING A LOGIN LINK	29
6.3. REDIRECTING USERS WITH AUTOMATIC LOGIN	29
CHAPTER 7. RESTRICTED CONTENT	31
7.1. RESTRICTED PAGES	31
7.2. RESTRICTED BLOCKS OF CONTENT	32
7.3. AUTOMATING THE CONFIGURATION OF EXTRA FIELDS	32
7.4. REQUIRING USER LOGIN	33
CHAPTER 8. EMAIL TEMPLATES	34
8.1. CUSTOMIZING EMAIL TEMPLATES	34
8.1.1. Define your workflows before email configuration	34
8.1.2. Test your workflow and identify active email templates	34
8.1.3. Edit and save your custom template	34
8.1.4. Repeat for all templates in your workflows	35
8.2. MORE INFORMATION	35

CHAPTER 9. LIQUIDS: DEVELOPER PORTAL	36
9.1. USING LIQUIDS IN THE DEVELOPER PORTAL	36
9.1.1. Enabling Liquids	36
9.1.2. Different use on pages, partials, and layouts	36
9.1.3. Use with CSS/JS	37
9.2. USAGE OF LIQUIDS IN EMAIL TEMPLATES	37
9.2.1. Differences from Developer Portal	37
9.3. TROUBLESHOOTING	37
9.3.1. Debugging	37
9.3.2. Typical errors and ways to solve them	37
9.3.3. Answers on the forum	38
CHAPTER 10. LIQUIDS: EMAIL TEMPLATES	39
10.1. ACCOUNT MANAGEMENT	39
10.2. CREDIT CARD NOTIFICATIONS	40
10.3. LIMIT ALERTS	40
10.4. APPLICATIONS	40
10.5. INVOICING	41
10.6. SERVICES	42
10.7. SIGNUP	43
CHAPTER 11. CUSTOMIZING THE DEVELOPER PORTAL LAYOUT	44
11.1. CREATING A NEW CSS FILE	44
11.2. LINKING THE STYLESHEET INTO YOUR PAGE LAYOUT	44
CHAPTER 12. CHANGE BUILT-IN PAGES	45
12.1. IDENTIFY THE ELEMENTS	45
12.2. MODIFY OR HIDE THE ELEMENTS	45
12.3. OPTION A: CSS	46
12.4. OPTION B: JQUERY	46
CHAPTER 13. WEBHOOKS	48
13.1. INTRODUCING WEBHOOKS	48
13.2. WEBHOOKS FORMAT	48
13.3. TROUBLESHOOTING	49
CHAPTER 14. SETTING TERMS AND CONDITIONS	50
14.1. TERMS AND CONDITIONS	50
14.2. CREDIT CARD POLICIES	51
CHAPTER 15. SUMMARY: FROM ZERO TO HERO DEVELOPER PORTAL	53
15.1. GOAL	53
15.2. PREREQUISITES	53
15.3. CONFIGURING YOUR DEVELOPER PORTAL	53
15.3.1. Planning your portal concept	53
15.3.2. Setting up the editing environment	53
15.3.3. Defining the page layout templates	54
15.3.4. Creating your page hierarchy	54
15.3.5. Editing your page headers	56
15.3.6. Adding images and other assets	56
15.3.7. Customizing with your branding	57
15.3.8. Going live	57

CHAPTER 1. DEVELOPER PORTAL

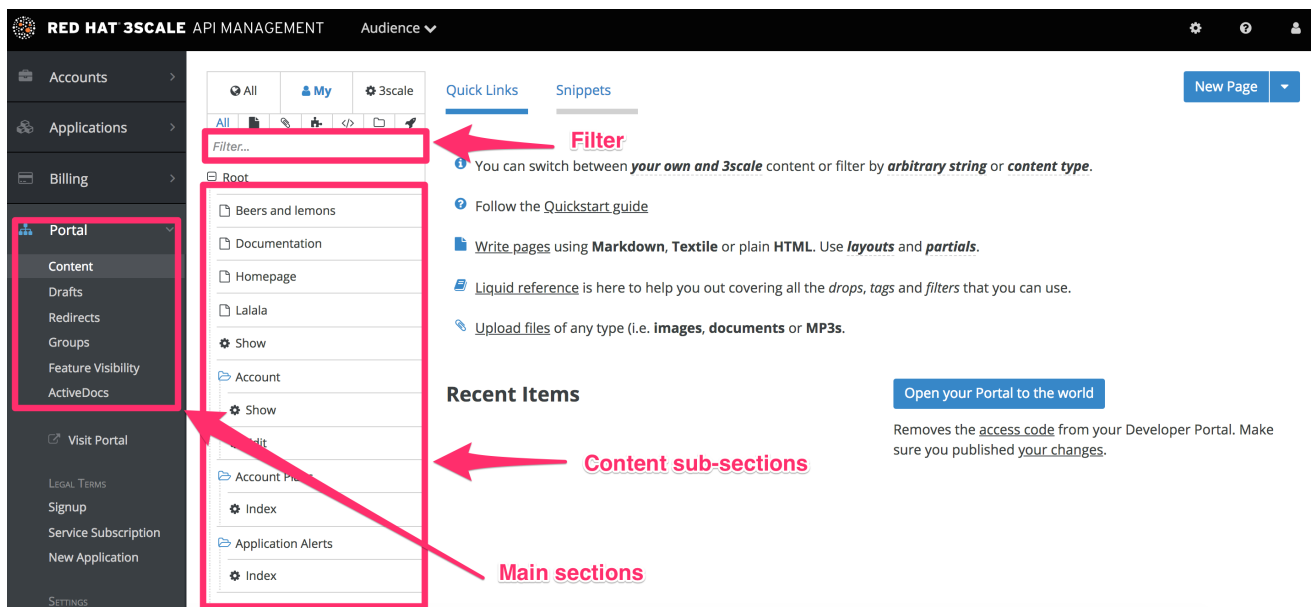
By the end of this section you should be familiar with the Developer Portal, including its structure, use, and functionality.

You can customize the look and feel of the entire Developer Portal to match your own branding. You have complete control over every element of the portal, so you can make it as easy as possible for developers to learn how to use your API. A successful API Developer Portal will help your developers turn concepts into working apps in no time at all.

1.1. OVERVIEW OF THE DEVELOPER PORTAL

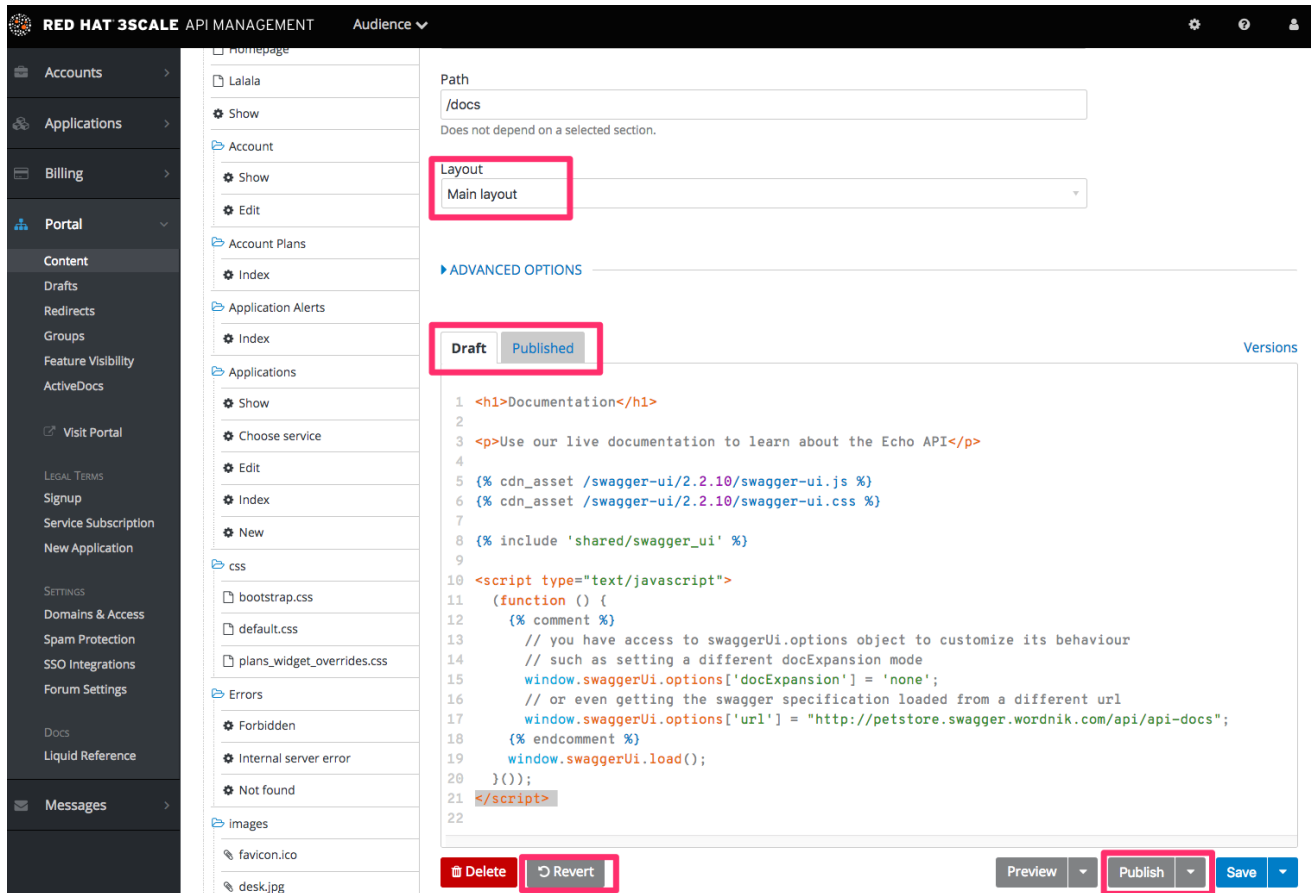
The Developer Portal consists of a few elements:

- Left menu with access to Content, Drafts, Redirects, Groups, Feature Visibility, ActiveDocs, Legal Terms, Settings and Liquid documentation.
- The main area containing details of the sections above.



1.2. CONTENT

This is the most important part of your view of the Developer Portal system. The content section shows the site structure and hierarchy and provides editing functionality within the same page. This means you can manage the site structure, the pages, and other assets stored in it. The portal's hierarchy is displayed in the form of a directory tree.

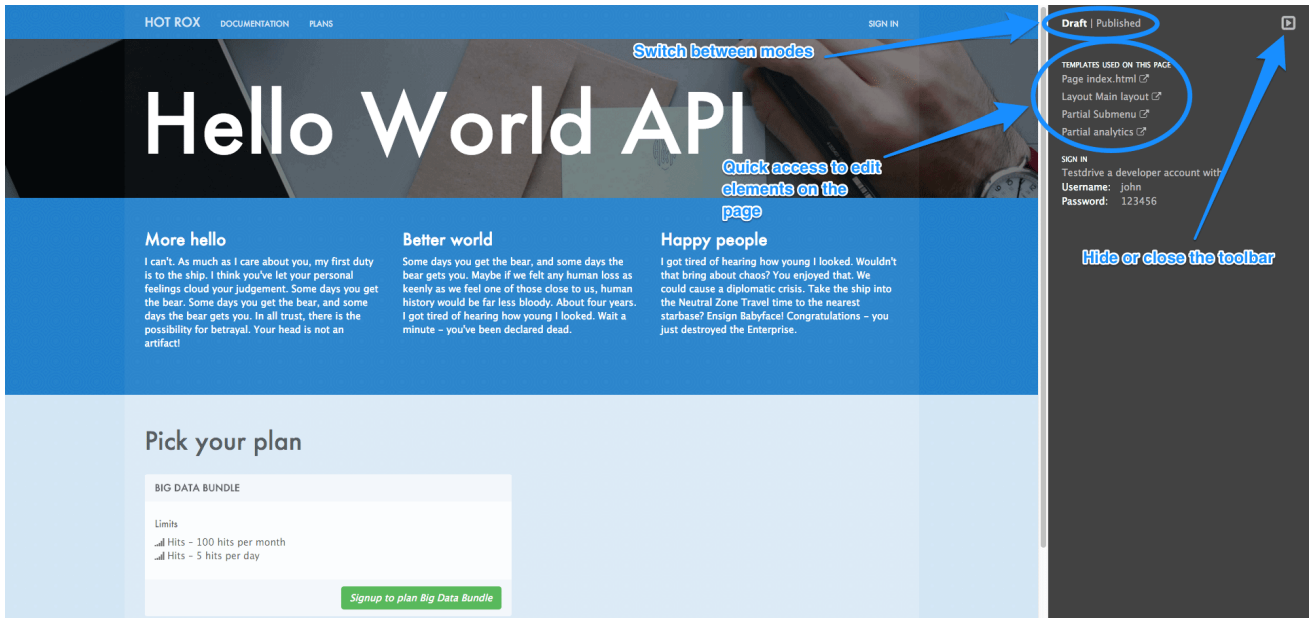


The image above shows a sample view of one of the pages inside the contents section. As you can see, it displays all the files (pages, images, stylesheets, JavaScript, etc.) preserving the site's path hierarchy. As before, sections are functionally equal to directories.

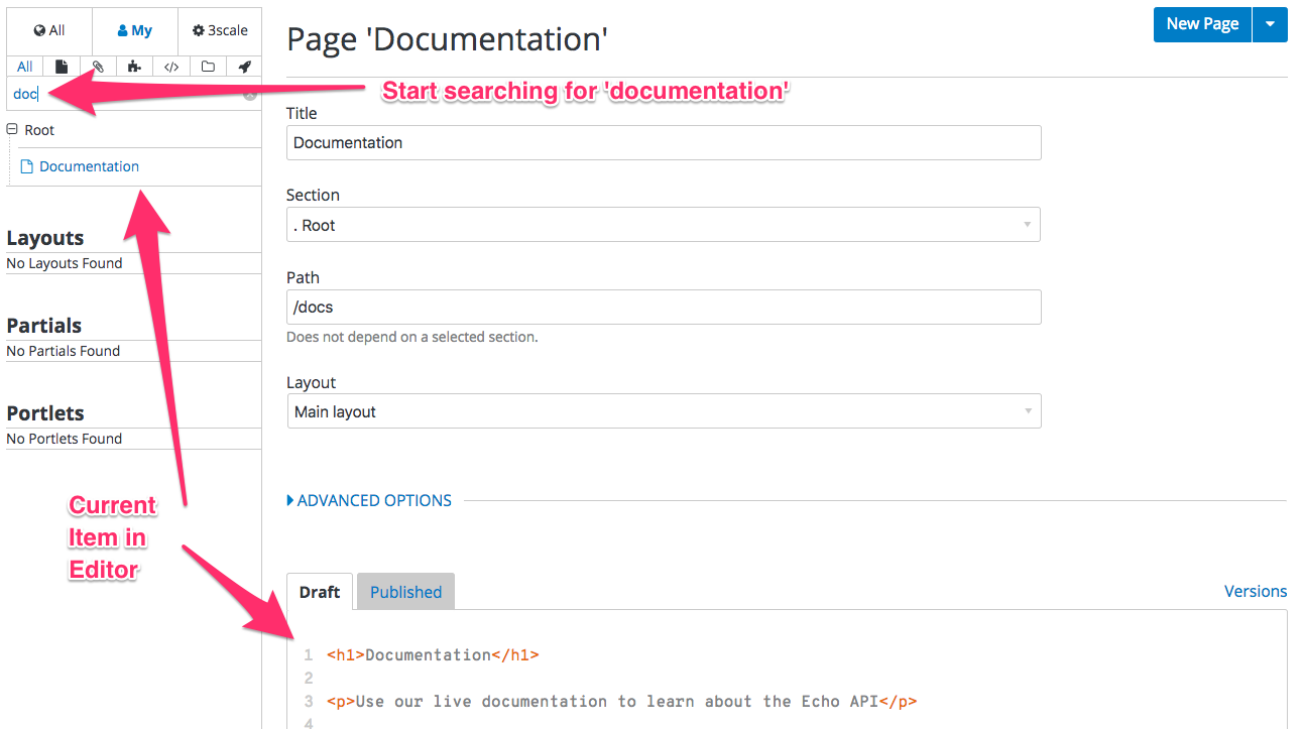
On the right-hand side, you can see the edit page view. Here you can see the page name (which also indicates whether it's a standard or built-in page) and a button to add a new element to the content (page, layout, partial, section, file, or portlet). Below, you can choose which layout the page will use and toggle the liquid tags functionality. The following part is the text editor, which supports code highlighting, tabulations, line numeration, and much more. The tab buttons Draft and Published switch between the draft and published versions of the edited document. The following two icons list the document's versions and open a pop-up edit window, respectively.

To edit page content, simply choose the desired layout, set a few additional options such as content type and URL path, and then input the code in HTML, Markdown, or Textile.

Another important feature in this view is the Preview button. You can choose whether you want to preview the published or draft version of the page. Clicking the button redirects you to Developer Portal mode, where you can see the live (or draft) rendered version of the page with a dark grey vertical bar on the right-hand side. This bar contains links to the page, layout, and partials edit views of the Developer Portal. It allows you to switch between draft and published views.



There's also a filter feature, which serves not only as a search field but allows you to limit the elements shown to only stylesheets, JavaScript, or any other types specified.



1.3. LAYOUTS AND PARTIALS

The layouts and partials sections manage the templates and the reusable parts of the page. Their functionality is similar to that of the content section.

The layouts section consists of definitions of the templates used by pages. Layout is the main structure of the page, and the contents of this template will be rendered on every page that uses it. The partials, portlets, and the actual content of the pages reside inside.

Partials are the reusable parts of code, which repeat in many places on different pages – for example, the footer is the same on every layout, and the sidebar is the same on a few pages with different layouts. To include a partial in a layout, partial, or email template or portlet, type: `{% include "partial_name" %}`. For full reference of liquid tags, check [here](#).

As with the other parts of the portal, layouts and partials also have draft and published states and offer a full version history.

1. Text editor for the layout template.
2. Save draft, publish current version, and revert to the last published state.
3. Switch the text editor between the draft and published versions, list the version history, and launch the pop-up editor.

1.4. PORTLETS

The last subsection of the content are the portlets. They give you some more advanced functionality without needing any advanced coding. Our Developer Portal provides three different portlets:

- External RSS feed - fetches the RSS feed from a given source
- Table of contents - generates the links list for the pages in a given section
- Latest forum posts - generates the list of the n latest forum posts

While creating your desired portlet, you have to input the requested data in the setup page such as title, system name, and the URL feed for the external RSS feed portlet.

1.4.1. Note

The editor will come pre-filled with standard portlet code using some custom liquid tags. You can try editing the generated structure, but be careful. When in doubt, you can refer to the **Liquid Reference** chapter.

1.5. REDIRECTS AND CHANGES

The last elements of the Developer Portal are the redirects and changes sections. They are much less complicated than the content section but are still important and provide some custom functionality.

Redirects help you set up redirects from one portal URL to another. This is useful, for example, when you deprecate an old page and don't want to change all the links. Redirects cannot be used for built-in Developer Portal pages – **they are only for pages created by you**

Last but not least is the changes section. It contains a list of all the newly edited and unpublished pages and gives you the choice to publish them individually or all at once.

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

org_name	"Organization/Group Name"	Required	Edit
-----------------	---------------------------	----------	------


User Create

username	"Username"	Required	Edit
email	"Email"	Required	Edit

Application Create

name	"Name"	Required	Edit
description	"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 \(show\)](#)

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.

[Create](#)

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 API. For example, "application create" on [3scale API](#).

CHAPTER 3. CONFIGURING SIGNUP FLOWS

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

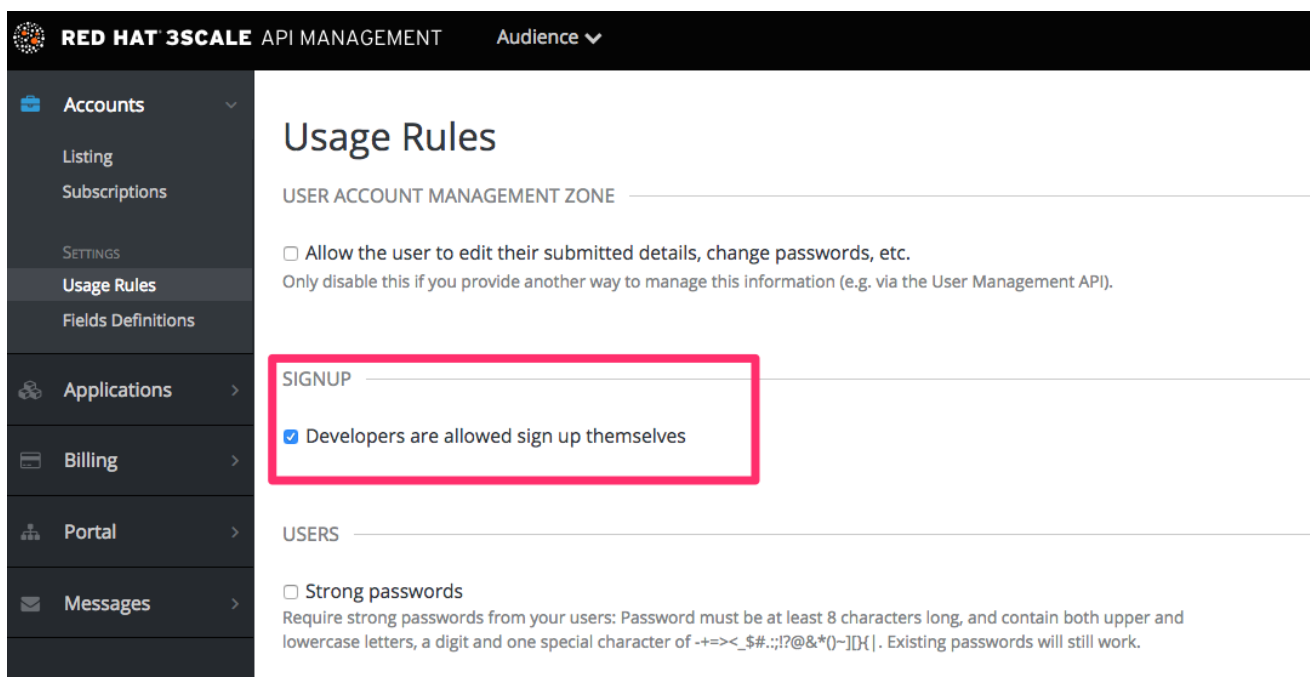
Signup workflows are a critical aspect of the developer experience you provide through your Developer Portal. The process can range from being completely automatic and self-service to the other extreme of requiring total control over who gains access to what, with various levels of granularity.

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.

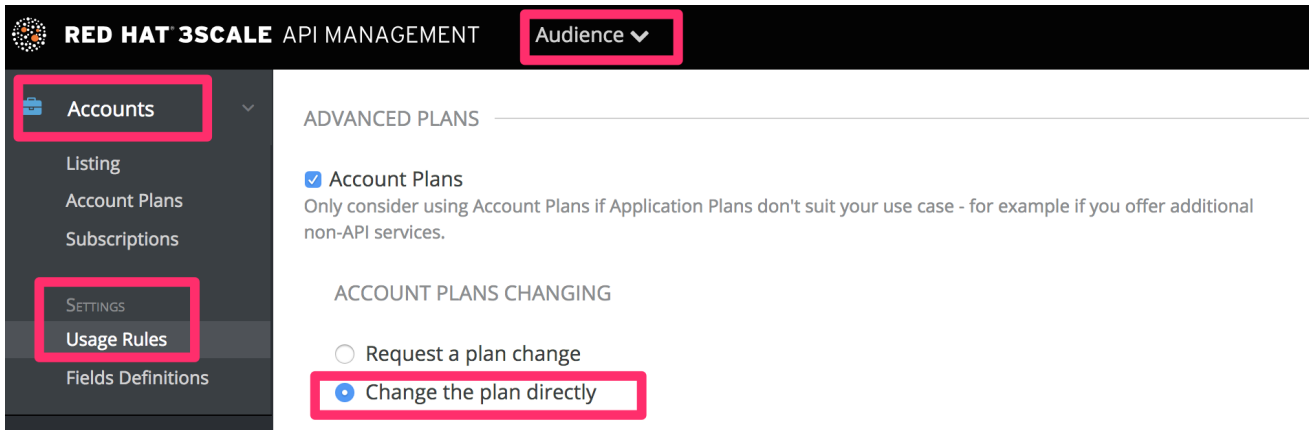
3.1. REMOVING ALL APPROVAL STEPS

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.



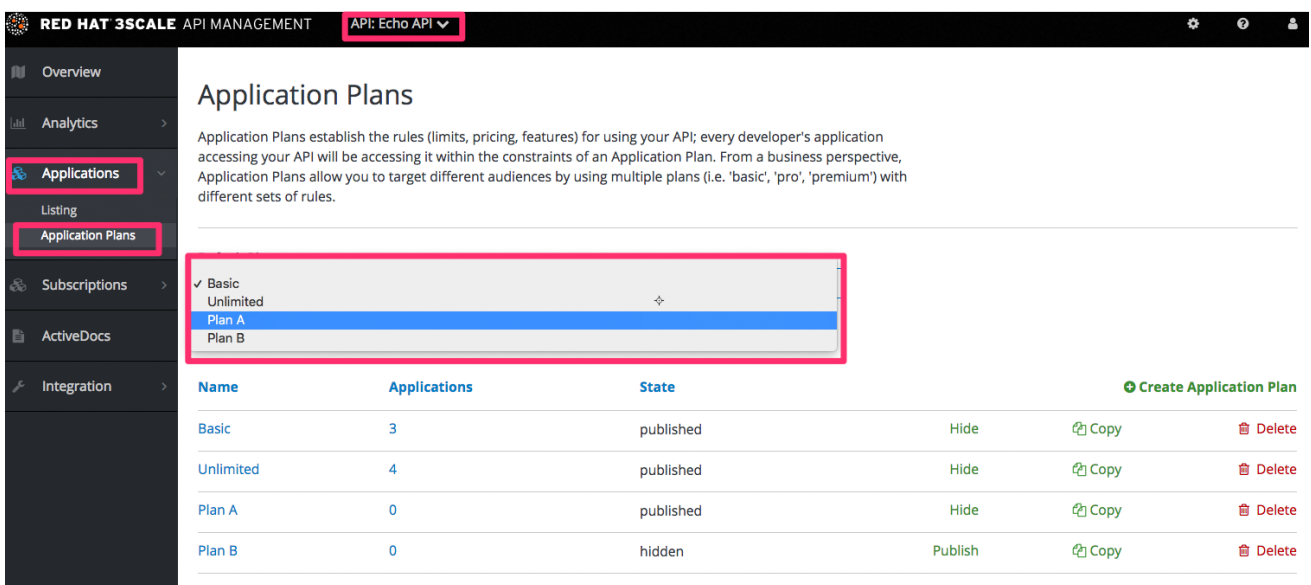
The screenshot shows the Red Hat 3Scale API Management interface. The top navigation bar includes the Red Hat 3Scale logo, 'API MANAGEMENT', and a dropdown menu for 'Audience'. The left sidebar contains a navigation menu with 'Accounts' (sub-items: Listing, Subscriptions), 'SETTINGS' (sub-items: Usage 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', 'SIGNUP', and 'USERS'. In the 'SIGNUP' section, the checkbox 'Developers are allowed sign up themselves' is checked and highlighted with a red rectangular box. Other settings include 'Allow the user to edit their submitted details, change passwords, etc.' (unchecked) and 'Strong passwords' (unchecked).

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:



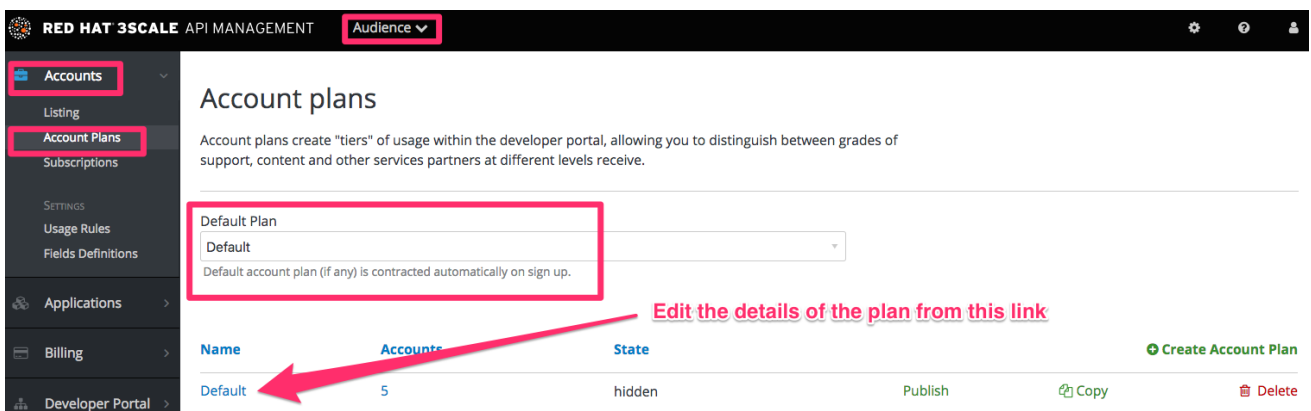
3.2. ENABLING ALL POSSIBLE DEFAULT PLANS

Application plans



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

Account plans (optional)



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			
Default	6	published	Hide	Copy	Delete

[Create Service Plan](#)

3.3. TESTING THE WORKFLOW

Once you've 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're happy with the workflow, it's 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'll be familiar with the procedure to create and customize a multiple-service signup page.

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

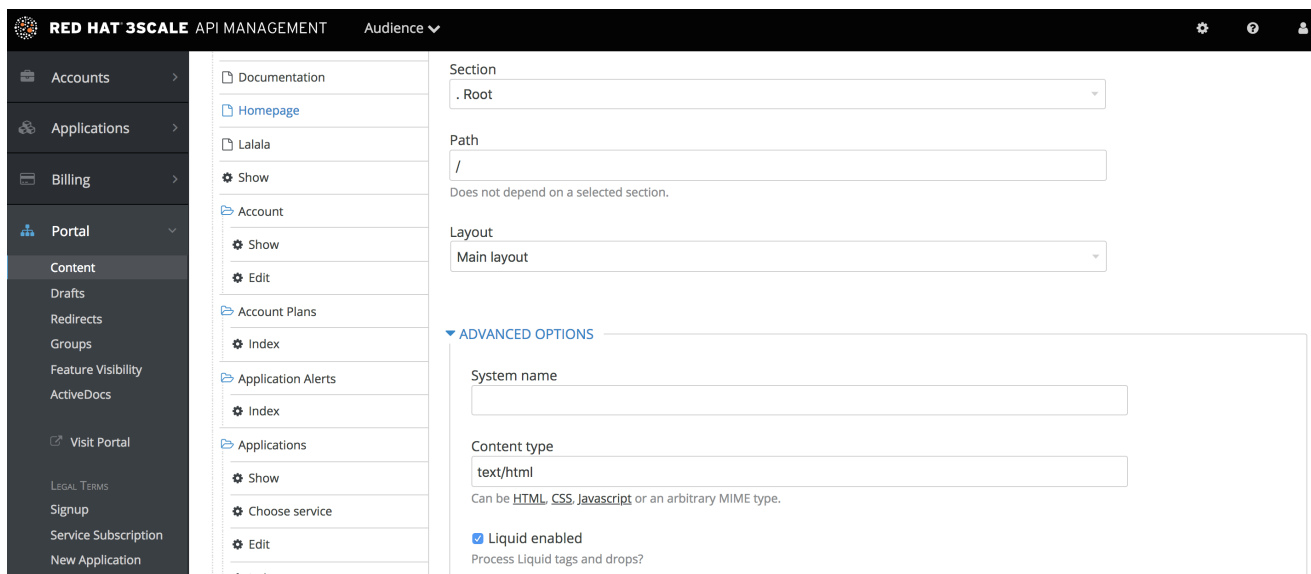
4.1. PREREQUISITES

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's strongly recommend that you read about [signup workflows](#), so you'll 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 don't 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's 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've used the percentage-based layout to dynamically assign the width of the column basic on the containing div's dimensions.

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

If you'd 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 three different ways 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            <input name="remember_me" type="hidden" value="1">
36          </fieldset>
37          <fieldset>
38            <div class="form-group">
39              <div class="col-md-10">
40                <input name="commit" type="submit" value="Sign in" class="btn btn-success btn-lg pull-
41                right">
42              </div>
43            </div>
44          </fieldset>
45
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
GitHub	Hidden
★ Auth0	Hidden
Red Hat Single Sign-On	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

5.3.1. Note

This feature is only available on the Enterprise plans.

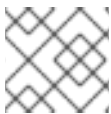
In order to have your developers authenticate using Auth0, you first need to have a valid Auth0 subscription.

Authentication via Auth0 won't be 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'll need to add the details of your Auth0 account. Once you've 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 VIA RED HAT SINGLE SIGN-ON



NOTE

This feature is only available on enterprise plans.

Red Hat Single Sign-On (RH-SSO) 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 RH-SSO 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 RH-SSO 7.2](#)



5.4.2. Configuring RH SSO

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/*.
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.

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 go this way, and in order for your developers to be able to skip both the Red Hat Single Sign-On and 3scale account creation steps, we recommend the following configuration. In the example provided, we're 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" value=""/>
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.
- In **Mapper Type** select *Attribute Importer*.

- d. In **Social Profile JSON Field Path** add company, which is the name of the attribute on GitHub.
- e. 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. E.g. In GitHub, if you set up your email address as private, it won't be shared.

5.4.3. Configuring 3scale

Authentication via Red Hat Single Sign-On won't be enabled by default. If you want to use your Red Hat Single Sign-On account in conjunction with 3scale to manage access to your developer portal, you can follow the steps below to configure it.

Go to your 3scale Admin Portal, in **Audience > Developer Portal > SSO Integrations** click on **Red Hat Single Sign-On**. (Remember: this is an enterprise only feature so you may have to ask your account manager to enable this for you).

On this configuration screen, you'll need to add the details of your Red Hat Single Sign-On client that you have configured in the previous step:

- **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

Once you've entered those, check the "Published" box and click **Create RH-SSO** to make it available on your developer portal.

The screenshot shows the 'New Red Hat Single Sign-On Authentication Provider' configuration page in the Red Hat 3scale Admin Portal. The page has a dark sidebar on the left with navigation options like 'Content', 'Drafts', 'Redirects', 'Groups', 'Feature Visibility', 'ActiveDocs', 'Visit Portal', 'LEGAL TERMS', 'Signup', 'Service Subscription', 'New Application', 'SETTINGS', 'Domains & Access', 'Spam Protection', 'SSO Integrations', and 'Forum Settings'. The main content area contains the following fields and options:

- Client ID:** A text input field.
- Client Secret:** A text input field.
- Realm:** A text input field with a placeholder example: `e.g.: https://rh-ssso.example.com/auth/realms/demo`.
- Do not verify SSL certificate
- Published
- Display on Developer Portal

At the bottom right of the page, there is a blue button labeled 'Create Red Hat Single Sign-On'.

CHAPTER 6. RED HAT SINGLE SIGN ON FOR DEVELOPER PORTAL

Red Hat Single Sign On (RH SSO) allows you to manage access control of multiple independent systems. By following this guide, you'll 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 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 RH SSO 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 bear in mind that the URL can expire before the user clicks it, so it's 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 your 3scale-powered Developer Portal.



NOTE

The URL needs to be unescaped. If you want to try it by hand in a browser or cut and paste, remember to replace the "&" for "&" in your browser. Also any "%" encodings in the token need to be replaced by their unescaped character.

CHAPTER 7. RESTRICTED CONTENT

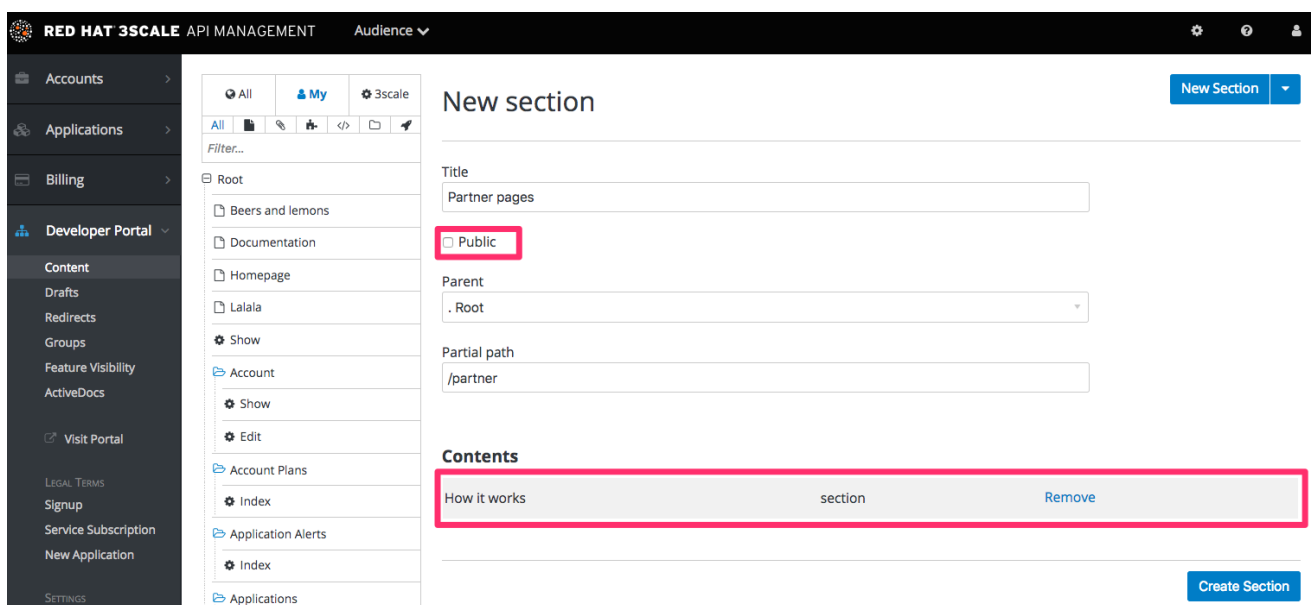
Here you can learn how to have content in your Developer Portal that is only visible for some users.

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.

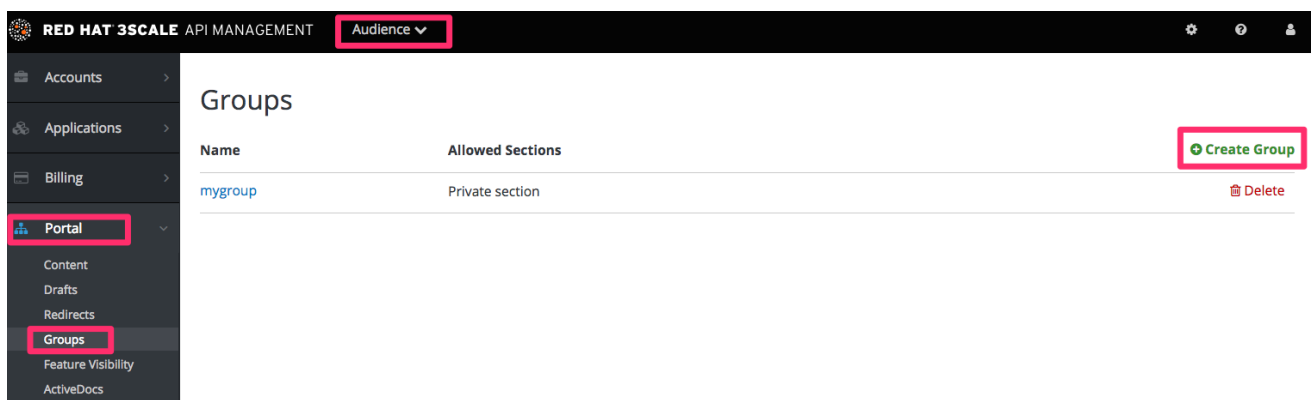
7.1. RESTRICTED PAGES

When creating restricted sections, it's 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 CMS 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	Create 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

In some cases, you'll want to provide access to restricted content to developers based on a change in state. For instance, when they upgrade application plan.

You can streamline the process by using [webhooks](#) together with the Account Management API. 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 Doc**section. Based on the developer's new plan – which you can know from the message sent by the webhook request – you can 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've 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'll actually "create" a custom template. Then in subsequent edits, you'll save your changes. Warning: there is no version control. We recommend 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's 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

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 simply checking the checkbox right under the `system_name` input field. However, to enable them on pages, you'll 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, which are single-use elements and partials/layouts, which are the reusable elements of your portal. This means that instead of using multiple layouts or partials with small changes for use on different pages, you can add some logic liquid tags inside 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 doesn't 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

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'll 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 anymore, as an email does not have a URL.

```
<!--samples-->
```

9.3. TROUBLESHOOTING

9.3.1. Debugging

If something is not working as intended (but saved correctly) check that:

- All the tags are closed correctly
- You're referring to variables available on the current page
- You're 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's usually because some tags or drops were not closed correctly. Check that all your `{% %}` and `{{ }}` tags were properly closed and that the logic expressions (if, for, etc.) are terminated correctly (with `endif`, `enfor`, etc.) 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 don't see any effect, check that you're not referring to an empty element and you're 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 instead of what you wanted to see only a # is displayed, it means that you've tried to display an element that is an array. Check the section on the liquid hierarchy in this article ([link](#)).

9.3.3. Answers on the forum

If you still have a problem, try looking for an answer on our [forum](#) or ask a question yourself.

CHAPTER 10. LIQUIDS: EMAIL TEMPLATES

You'll probably want 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 CMS, 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

The following email templates fall under this category:

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

For these, 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
have access to the following 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

They have 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**

Finally, the following email templates have an increasing number of available liquid drops, starting with the base for...

- 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.

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**. It is quite large and complex, so rather than extend it, it's 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 (just remember to choose an appropriate MIME content type in the advanced page settings).

It's important 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!

CHAPTER 12. CHANGE BUILT-IN PAGES

By the end of this section, you'll be able to modify and/or hide any elements on the system-generated pages.

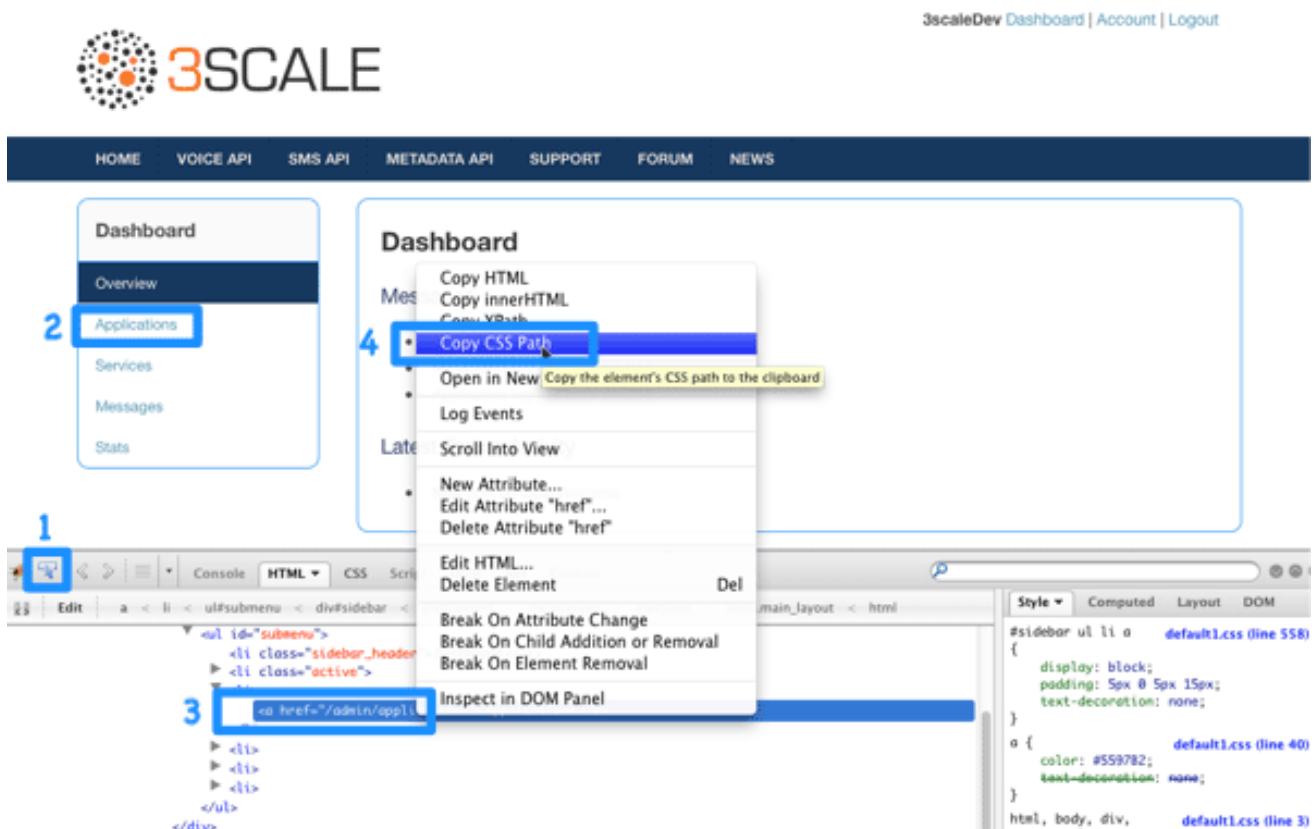
There are some elements generated by the system that are not possible to change from the CMS: the Signup, Dashboard, and Account pages. This guide shows how to customize the content on these pages with some simple CSS and JavaScript scripts.

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, please be sure that you'll be able to monitor for any disruptive changes and do the necessary maintenance work to keep your portal functioning correctly.

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'll 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 doesn't 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'll see both of these approaches.

12.3. OPTION A: CSS

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

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

Keep in mind that it's the second box with the same path, so you'll 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's better to use some jQuery. The CSS path of these elements is identical on the Dashboard and Account sections, and you don't 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, you'll use the `.text()` function. You'll also include the code inside the document's head and inside the ready function so it's executed after all the content has been generated.

The screenshot shows the 3scale Developer Portal interface. The top navigation bar includes 'Dashboard', 'Developers', 'Applications', 'Billing', 'Analytics', 'API', 'Developer Portal' (highlighted with a blue box), and 'Settings'. The left sidebar has 'Content' (highlighted with a blue box), 'Redirects', and 'Changes'. Under 'Content', there are sub-sections: 'main_layout', 'Content', 'Layouts', 'main_layout' (highlighted with a blue box), 'Partials', 'Emails +', 'Legal Terms +', and 'Portlets +'. The main content area is titled 'Layout 'Main layout'' and has a 'New' button. It contains a 'Title' field with 'Main layout', a 'System name' field with 'main_layout', and a checked 'Liquid enabled' checkbox with the text 'Process Liquid tags and drops?'. Below this are 'Save' and 'Publish' buttons, and 'Draft' and 'Published' status buttons. A code editor shows the following HTML and JavaScript code:

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8" />
5 <title>{{ page.title }}</title>
6 <link rel="stylesheet" href="/stylesheets/screen.css" />
7
8
9
10
11 <script>
12 $(document).ready(function() {
13 //the jQuery code for hiding elements
14 });
15 </script>
16
17
18 {% scale_essentials %}
19 </head>

```

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](#).

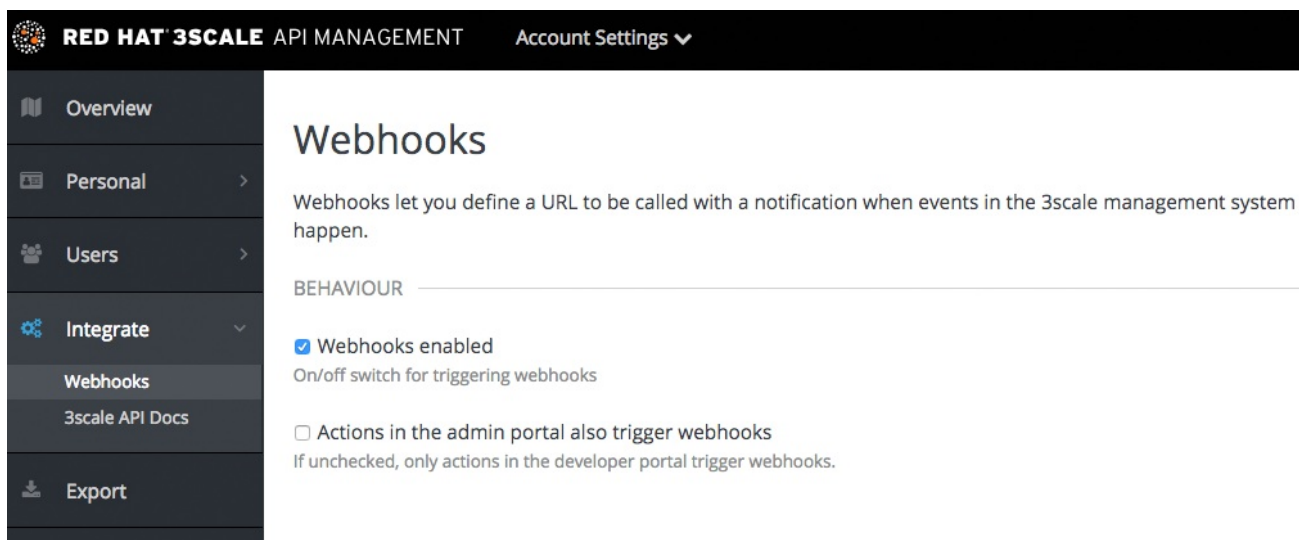
CHAPTER 13. WEBHOOKS

By the end of this section, you'll be able to configure and take action on the webhooks in **Developer Portal**.

The use of webhooks allows you to tightly integrate 3scale with your back-office workflow. When specified events happen within the 3scale system, your applications will be notified with a webhook message, and you can use the data such as from a new account signup to populate your CRM system.

13.1. INTRODUCING WEBHOOKS

A webhook is a custom HTTP callback triggered by an event. In the 3scale system, all the possible events are displayed in **Account Settings** (gear icon in the top right corner) > **Integrate** > **Webhooks**



When one of these events occurs, the 3scale system makes an HTTP (or HTTPS) request to the URI configured in the webhooks section. On your end, you can configure the listener to invoke some desired behavior such as event tracking.

The remaining two checkboxes on the screenshot turn on webhooks ("Webhooks are" switch) and allow webhooks to be fired by actions in the Admin Portal. The default behavior is to trigger webhooks only by actions triggered from within the Developer Portal. Bear in mind that this means not all events can be triggered.

13.2. WEBHOOKS FORMAT

The format of the webhook is always the same. It makes a post to the endpoint with an XML document of the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<event>
  <type>application</type>
  <action>updated</action>
  <object>
    THE APPLICATION OBJECT AS WOULD BE RETURNED BY A GET ON THE ACCOUNT
    MANAGEMENT
    API
  </object>
</event>
```

The <type> gives you the subject of the event such as "application", "account", etc. The <action> – what has been done such as "updated", "created", "deleted". Finally the <object> is the XML object itself in the same format that is returned by the Account Management API. To check this, you can use our interactive ActiveDocs, available in your Admin Portal, under the question mark (?) icon located in top right corner, and 3scale API Docs.

If you need to provide assurance that the webhook was fired by 3scale, expose an HTTPS webhook URL and add a custom parameter to your webhook declaration in 3scale. For example: <https://your-webhook-endpoint?someSecretParameterName=someSecretParameterValue>. Decide on the parameter name and value. Then, inside your webhook endpoint, check for the presence of this parameter value.

13.3. TROUBLESHOOTING

If you want to experiment with the webhooks or troubleshoot issues, you may find RequestBin a great (and free) service to view the results of the webhooks: <http://requestbin/>

If you experience an outage for your listening endpoint, you can recover failed deliveries. 3scale will consider a webhook delivered if your endpoint responds with a 200 code. Otherwise, it will retry 5 times with a 60 seconds gap. After any recovery from an outage, or periodically, you should run a check and if applicable clean up the queue. You can find more in the [ActiveDocs](#) for the two methods:

- Webhooks list failed deliveries
- Webhooks delete failed deliveries

CHAPTER 14. 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

14.1. 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.

The screenshot shows the Red Hat 3scale Admin Portal interface. At the top, the header reads "RED HAT 3SCALE API MANAGEMENT" and "Audience" with a dropdown arrow. The left sidebar contains a menu with items: Accounts, Applications, Billing, Portal (highlighted with a pink box), Content, Drafts, Redirects, Groups, Feature Visibility, ActiveDocs, Visit Portal, LEGAL TERMS (highlighted with a pink box), Signup (highlighted with a pink box), Service Subscription, and New Application. The main content area is titled "Legal Terms for Signup" and contains the following text:

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.

Below the text is a large, empty text area for editing the legal terms, with a small "1" at the top left corner.

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.

14.2. 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>
```

And that's it!

CHAPTER 15. SUMMARY: FROM ZERO TO HERO DEVELOPER PORTAL

Most best practice reviews of API deployments agree that a well structured developer portal and great documentation are key elements to assure adoption. Your developer portal is not only a source of documentation. It is also your main hub to manage interactions with developers and for developers to access to their API keys in a secure way.

15.1. GOAL

By the end of this tutorial, you'll have a developer portal up and running to promote your API, allow developers sign up for accounts, and access their API keys.

15.2. PREREQUISITES

There are a few other areas for you to set up that are interdependent with the portal. You can take care of them before or after:

- [Set up your application plans](#) – so you have access rights configured for the API keys that you'll issue.
- [Configure your developer signup flows](#) –, which can be anywhere on the spectrum from self-service, to approval required, to invite only (with signups disabled).
- [Set up a custom domain for your portal](#) (optional) – – This can have a lead time of 1-2 weeks and is normally done at the same time you create a custom outbound email address.

15.3. CONFIGURING YOUR DEVELOPER PORTAL

15.3.1. Planning your portal concept

Before you even open the 3scale Developer Portal, it's a good idea to plan out the concept for your portal. The better organized you are, the more efficiently you'll be able to complete these steps.

The most important elements to plan out are your:

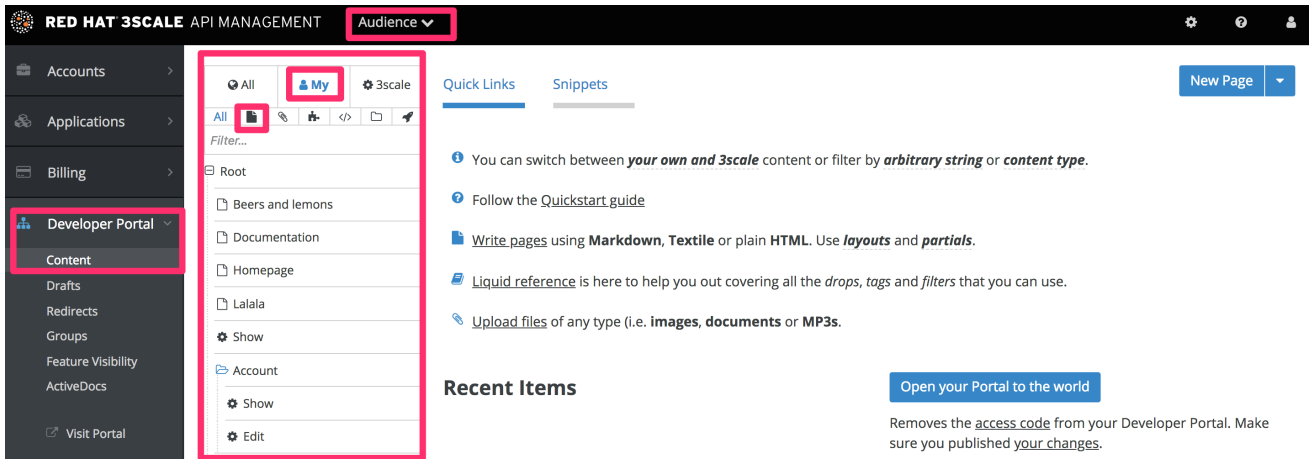
- Site map -- a skeleton of what the portal structure will be
- Top menu bar -- the navigation that will be repeated on every page
- Side menu bars -- for access to individual pages within each section
- Page layout guidelines -- to give your portal a consistent look and feel

15.3.2. Setting up the editing environment

The best setup of your editing environment has proven to be:

- A tab showing *yourcompany-admin.3scale.net/p/admin/cms* logged in with your admin credentials, which gives access to Developer Portal for the portal
- Second tab pointing to *yourcompany.3scale.net*, the public view of your portal (if you access this through the Site link, you don't have to worry about the access code)

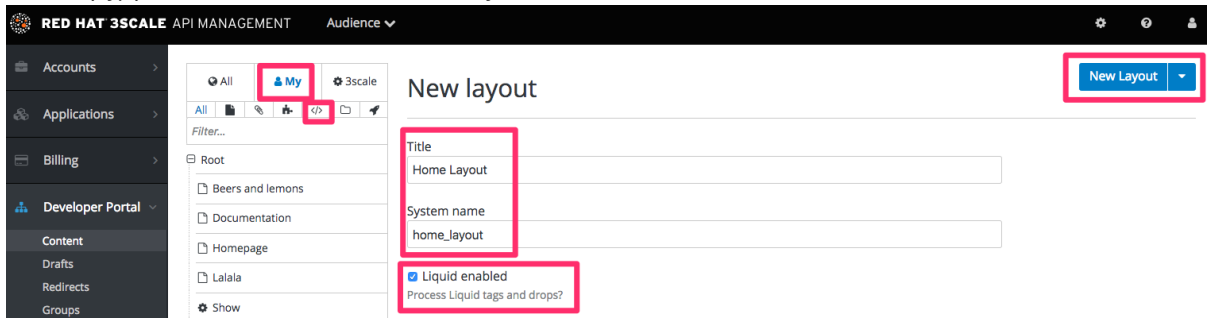
In the Admin Panel, you can see all your developer portal’s elements in the left sidebar:



15.3.3. Defining the 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. It’s best not to make any changes to this until you are quite expert at using the Developer Portal because this is used by all the system-generated pages. Usually you’ll want to have a unique style for the home page of your portal.

1. The main layout template will be a starting point for your customizations. Create a new layout, and copy/paste the content of main layout into it.



2. Remove the sidebar menu by deleting this line from your “home layout”:

`{% include 'submenu'%}`

Layouts

- <> Error layout
- <> Main layout

Partials

- analytics
- applications/form
- field
- login/ssso

```

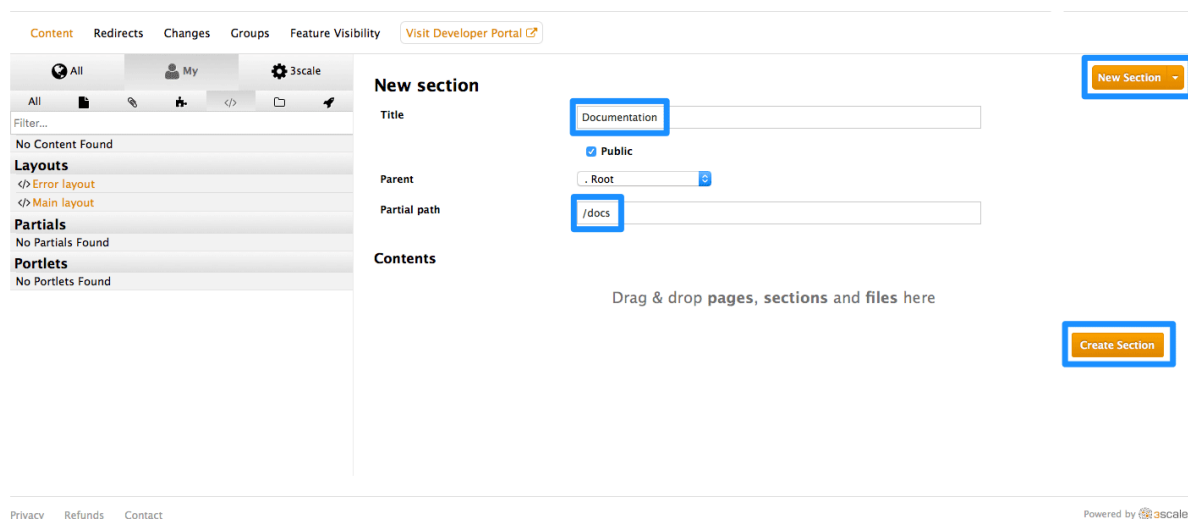
35 data-target="#navbar-1"
36 <span class="sr-only">Toggle navigation</span>
37 <span class="icon-bar"></span>
38 <span class="icon-bar"></span>
39 </button>
40 <a class="navbar-brand" href="/>{{ provider.name }}</a>
41 </div>
42 {% include 'submenu'%}
43 </div>
44 </nav>

```

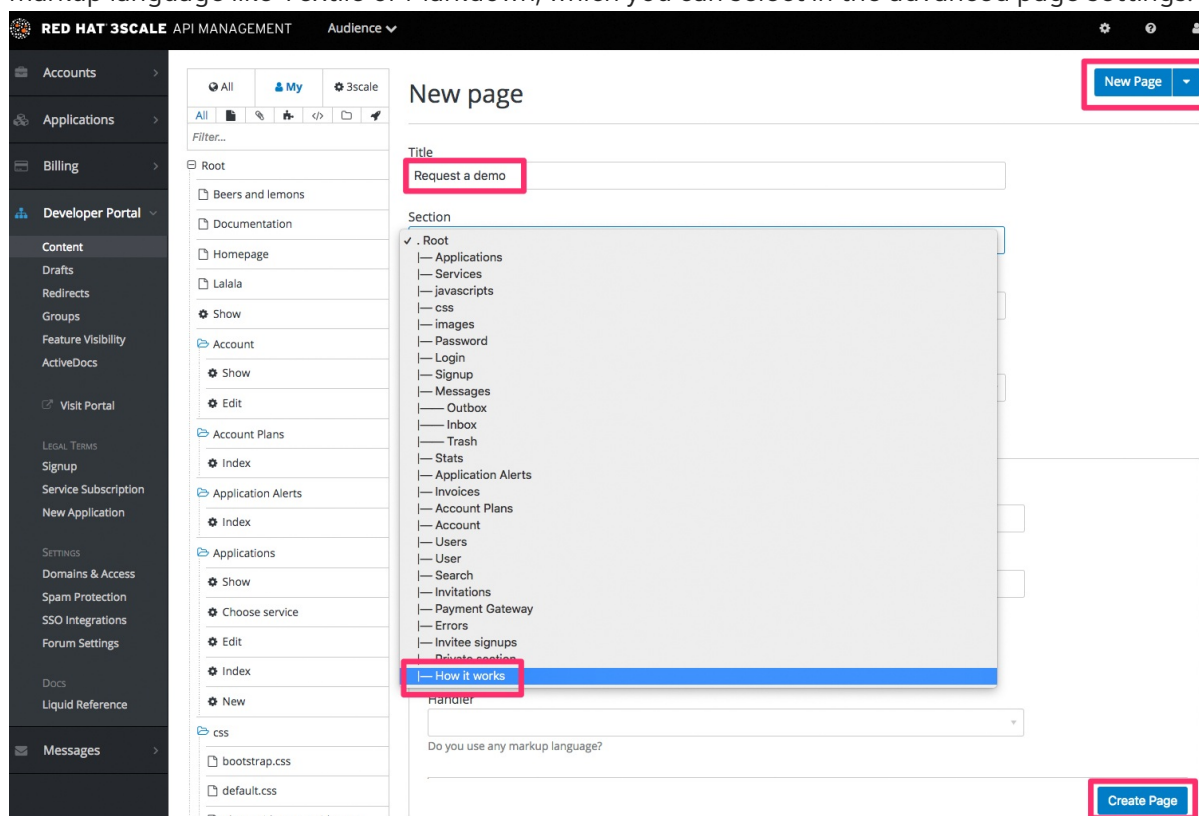
Content copied from the "Main Layout" template.

15.3.4. Creating your page hierarchy

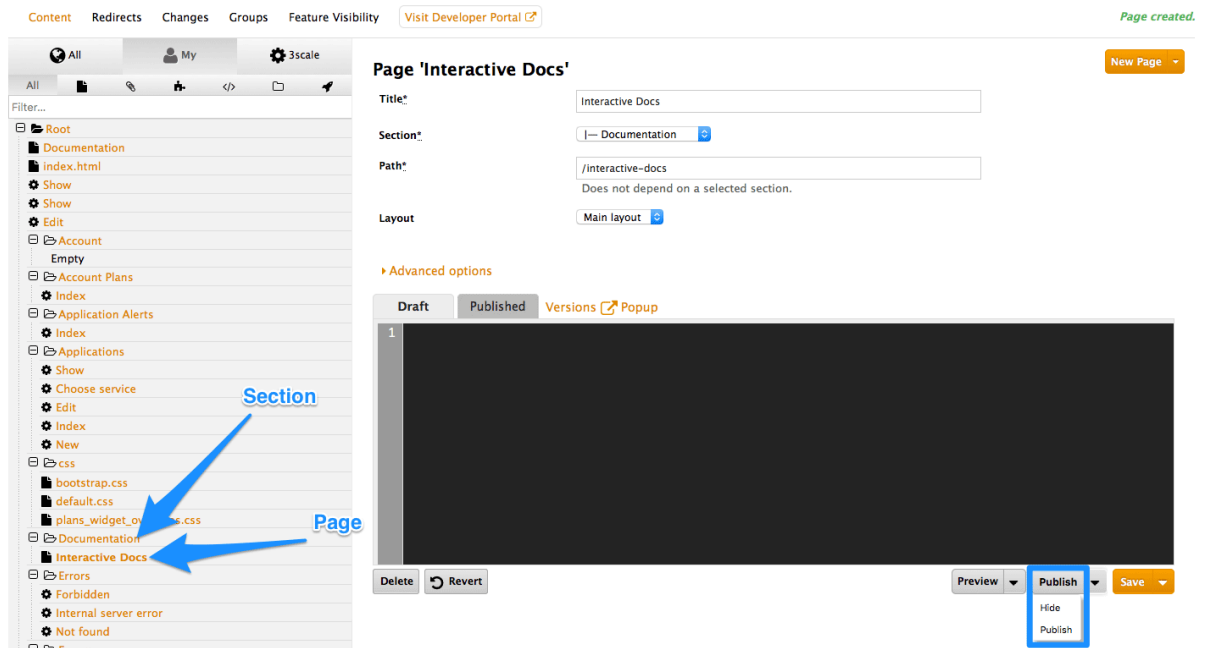
1. Begin at the root level in the site map, and add a new section for each of your top menu items (add sections by expanding the “new” button on the righthand side). Assign a title, parent section, and the path.



- Similar to adding a section, add a page. Choose the desired section in order to structure your URL paths consistently. Next, select the layout that the page will be using. After completing the page content, hit "create page". If you are writing a lot of content, you may prefer to use a markup language like Textile or Markdown, which you can select in the advanced page settings.



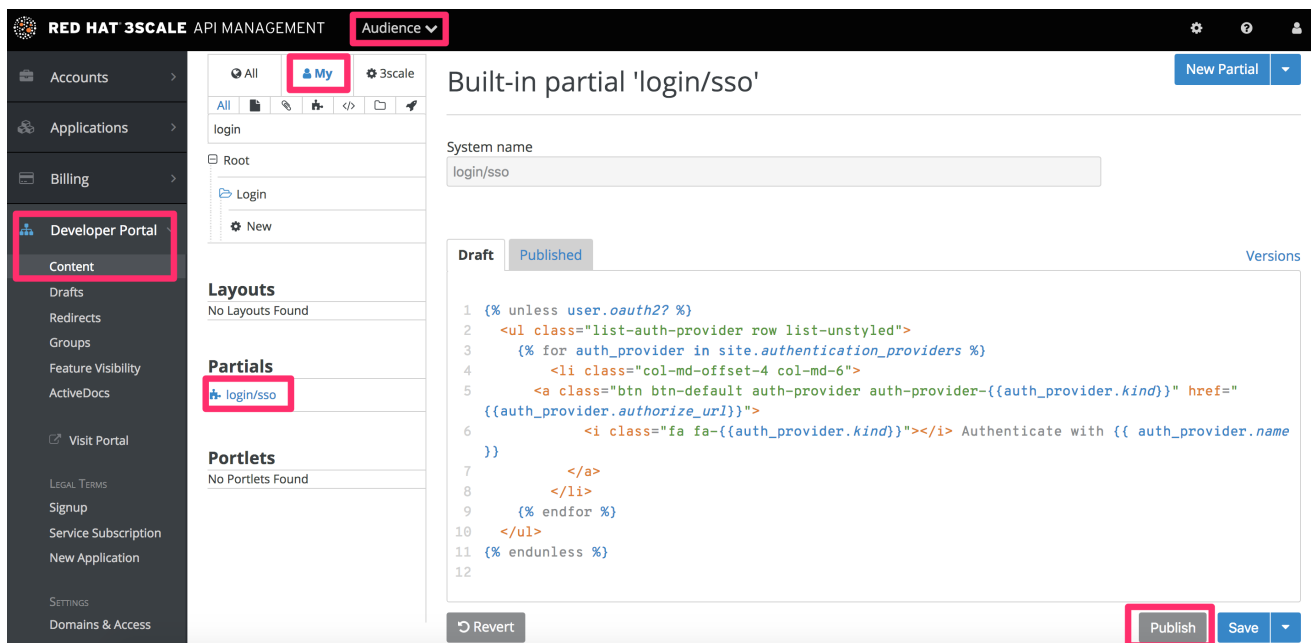
- View the draft preview and refine the page content until you are happy and ready to publish it.



4. Repeat for all pages in the section.
5. Then repeat for all sections in the site.

15.3.5. Editing your page headers

All repetitive page elements such as headers and footers are defined in the portal Developer Portal section called "partials". If you have only one layout, or very few of them, you can omit this step and include the header and footer inside the layouts code. However, remember to customize these elements in the layout. For example, the default "menu" partial should be edited to reflect your site map.

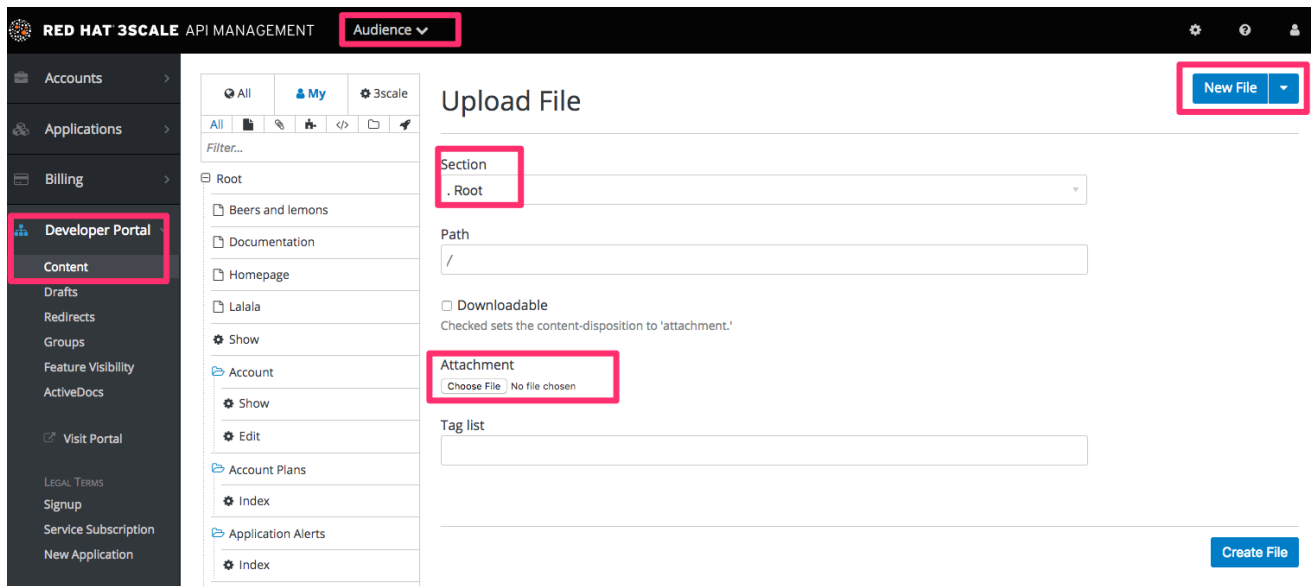


15.3.6. Adding images and other assets

For images or other files, first load the files into the content library, then insert a link into your text content.

1. Select New File in order to choose your file, and identify where you will save it on your site.

2. Copy the URL to the image
3. Now you can add your HTML or `<a>` tag, and paste in the URL for your image.



15.3.7. Customizing with your branding

There is a default stylesheet called `default.css`, which is quite large and complex. Rather than extend this, it's better to create your own stylesheet with your own customizations to overwrite the defaults.

You can create a new stylesheet the same way you create a page – just remember to choose an appropriate MIME content type in the advanced page settings. Then add the link to your custom CSS in your layout templates after the link to `default.css`, for example:

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

15.3.8. Going live

The final task is to view the entire portal site and check all the workflows. You can publish each page or all of the pages in the Changes section. Once you're happy with everything, make a final check that all pages have been published.

Now you're ready to remove any access code for the portal:

RED HAT 3SCALE API MANAGEMENT Audience

Domains & Access

ACCESS CONTROL

Developer Portal Access Code

<no access code>

The access code hides your site from the world, but allows you to share access to a select few users during setup. Add text to the field to enable the screen. **Delete the text to open the site to the world.**

DOMAINS

Developer Portal Site

xxxxxxxx

You can change the domain of your Developer Portal to your own domain, for instance https://developer.example.com. To do so, **add a route on OpenShift** and then indicate the new domain here.

Outgoing Email

xxxxxxxx

You can change the domain of the email address received by your users, for instance api@example.com

Update Account

Congratulations! Your developer portal is now live and ready to help build your developer community.