



# **Red Hat 3scale 2-saas**

## **Use Cases**

For Use with Red Hat 3scale 2-saas



# Red Hat 3scale 2-saas Use Cases

---

For Use with Red Hat 3scale 2-saas

## Legal Notice

Copyright © 2018 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, 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 Software Collections 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 discusses various use cases for Red Hat 3scale 2-saas.

## Table of Contents

<b>CHAPTER 1. APIS AS A BUSINESS .....</b>	<b>3</b>
1.1. API REQUIREMENTS	3
1.1.1. Purpose	3
1.1.2. Audience	3
1.1.3. Functionality	4
1.1.4. Access control and security	4
1.1.5. Visibility	4
1.1.6. APIUX	5
1.2. DEFINING THE API IN 3SCALE	5
1.3. INTEGRATING THE API WITH 3SCALE	6
1.4. MANAGE API ACCESS	8
1.4.1. Application plans	8
1.4.2. Limits and pricing rules	9
1.5. ENGAGE DEVELOPERS	11
1.6. EXPOSING THE API	13
<b>CHAPTER 2. CUSTOMER INTEGRATION .....</b>	<b>15</b>
2.1. API REQUIREMENTS	15
2.1.1. Purpose	15
2.1.2. Audience	15
2.1.3. Functionality	15
2.1.4. Access Control and Security	16
2.1.5. Visibility	16
2.1.6. APIUX	16
2.2. DEFINING THE API IN 3SCALE	17
2.3. INTEGRATING THE API WITH 3SCALE	18
2.4. MANAGE API ACCESS	20
2.4.1. Application plans	20
2.4.2. Limits	21
2.5. ENGAGE DEVELOPERS	21
2.6. EXPOSING THE API	23
<b>CHAPTER 3. PARTNER INTEGRATION .....</b>	<b>25</b>
3.1. API REQUIREMENTS	25
3.1.1. Purpose	25
3.1.2. Audience	25
3.1.3. Functionality	26
3.1.4. Access Control and Security	26
3.1.5. Visibility	26
3.1.6. APIUX	26
3.2. DEFINING THE API IN 3SCALE	27
3.3. INTEGRATING THE API WITH 3SCALE	28
3.4. MANAGE API ACCESS	30
3.4.1. Application plans	30
3.4.2. Limits and access	32
3.5. ENGAGE DEVELOPERS	34
3.6. EXPOSING THE API	38



# CHAPTER 1. APIS AS A BUSINESS

In the "APIs as a business" use case, your APIs themselves are your product, either creating a new business model based solely on your API or creating a new channel for an existing model.

Twilio is one of the best examples of a company that successfully sells their APIs directly as their core product. They provide APIs for text messaging, VoIP and voice in the cloud, allowing customers to easily build communication-enabled apps.

What does it take for a company to successfully expose an API as product in its own right?

## 1.1. API REQUIREMENTS

The first thing to do is review your requirements for the API before even thinking about the setup:

- Purpose
- Audience
- Functionality
- Access control and security
- Visibility
- APIUX

### 1.1.1. Purpose

When looking to expose an API as a product in its own right, your intention will most likely be to turn it into a primary revenue channel. An API-driven business model may not always be apparent from the get-go, but it may become an obvious next step to open up new business opportunities. As such, you'll need a business model for your API. This will most likely be based on number of transactions made, amount of data returned, functionality offered, and any other measures related to API traffic.

The FlightStats API is a prime example of this. While FlightStats initially offered only a set of highly popular web and mobile applications, the data powering them was so valuable that they enabled FlightStats to build new opportunities. They then spun the API off as a separate product.

Another example is the Google Maps API. Although the maps are a product in themselves and the API is free for end users, it's widely used to add maps to a large number of third-party websites and applications.

### 1.1.2. Audience

When your API is your product, your audience is your potential and existing customers. They could be hobbyists, independent application developers, or even large organizations looking to use your API. These different sets of consumers will have different data volume needs and SLAs, which you should try to cater to.

Someone that is playing around with your API for their own personal use will probably not be prepared to pay for access. However, they won't be a very heavy use of your API either, and they may build a cool product that showcases the potential uses for your API. You should not necessarily exclude them from experimenting with your API. On the other hand, someone who uses your API to sell an application should probably be charged for access, but they will also be a heavier user of it too.

The Google Maps API provides different "editions" based on whether the end application is publicly and freely accessible to end users or whether the application charges for usage. Other APIs will have a free or trial plan to allow potential customers to try out a limited number of calls on the API before they commit to a paid subscription with larger usage volumes.

### 1.1.3. Functionality

Once you're clear on the business model for your API and the audiences you are targeting, the next important thing to think about is the actual functionality your API is going to provide for consumers. Since it will be a product in its own right, you'll want to make sure that the offering is as complete and attractive as possible, especially since you'll likely be charging for access.

A good way to think about this is in terms of the resources you want to expose and the operations you want to allow on those resources. It could be anything from airports as in the FlightStats API or nutritional data as in the Nutritionix API. Whatever the resources or functionality you're exposing, you should group it in a way that makes sense for your business. This could be by having an API per resource type as FullContact does it, or a single API to encompass all resources offered.

### 1.1.4. Access control and security

The next step after you've decided which resources and methods to expose will be to think about the conditions under which you'll make them available. This includes implementing some usage limits as well as potentially limiting access to some of the resources you expose, based on the customer type and their subscription plan to your API.

Yummly, the recipe search site, provides access to their recipe API under different contracts, each including a number of free calls and different functionality. For example, local caching of recipes is only available on the Prix Fixe menu.

Another example of a business model based on transaction volumes is the FullContact API, which provides contact information for people based on their email addresses or social media handles. Once customers use up the free calls included in their plan, there is an extra cost per call. Their business model is based mainly on call volume but includes a mixture of both a pre-pay model and pay-per-use.

It's also important to think about the authentication mode you will implement for your API. Developer experience is extremely important, especially if your API is a product. As such, you'll want to make it as secure as possible, yet painless for your developers to use. In this case, we recommend a combination of an application identifier and a secret key to secure access to your API. Ideally, applications should be able to keep a number of valid secret keys in rotation, which can be revoked easily if any of them become compromised.

### 1.1.5. Visibility

As your API is your product, you'll want to publicize it as much as possible to ensure people can find out about it, are encouraged to play around with, and eventually use it. As such, a really excellent Developer Portal is a must to showcase your API.

Another important factor is providing clear, useful documentation with plenty of examples of how to use your API. Interactive documentation is a very important part of this, so your customers can try out the API without needing to write a single line of code.

You'll also want to make signup as easy and painless as possible for any potential customers, so they can get started using your API right away. When exposing an API as a product, reducing any barriers to entry should be your primary goal.



### 1.1.6. APIUX

These are all conscious decisions, which should be made up front when deciding to expose your API. That doesn't mean you can't change them later, but it's good to have a clear idea of the factors mentioned above and how you're going to implement them.

The [Algolia API](#) will serve as an example of the best way to achieve these goals with 3scale. It's a hosted search API, which allows anyone to easily integrate search into their website or application. In this example, you'll see how to integrate their REST API with 3scale. The API uses a combination of an application identifier with an API key to authorize requests. The response format for all requests is a JSON object. For the purpose of this example, pretend that this API currently has no authentication mechanisms in place and is completely unsecured and open for anyone to use.

## 1.2. DEFINING THE API IN 3SCALE

The first thing to do, after creating a 3scale account of course, is to define all of the existing endpoints for your API and set these up within 3scale. Set this up within the API service which is created by default with your 3scale account by going to the **Integration > Methods & Metrics** view in your API space.

The screenshot shows the Red Hat 3Scale API Management interface. The left sidebar contains a navigation menu with the following items: Overview, Analytics, Applications, Subscriptions, ActiveDocs, Integration (highlighted with a red box), Configuration, Methods & Metrics (highlighted with a red box), and Settings. The main content area is titled 'Methods & Metrics' and is for the 'API: Echo API'. It is divided into two sections: 'Methods' and 'Metrics'.

**Methods Section:** Includes a description: 'Add the methods of this API to get data on their individual usage. Method calls trigger the built-in Hits-metric. Usage limits and pricing rules for individual methods are defined from within each [Application Plan](#). A method needs to be mapped to one or more URL patterns in the [Mapping Rules](#) section of the integration page so specific calls to your API up the count of specific methods.' Below this is a table with columns: Method, System Name, Unit, Description, and Mapped. The table is currently empty, with the text 'You have no methods' displayed. A red arrow points to a '+ New method' button, which is also labeled 'Create new method' in red text.

**Metrics Section:** Includes a description: 'Hits is the built-in metric to which all methods report. Additional top-level metrics can be added here in order to track other usage that shouldn't increase the hit count. A metric needs to be mapped to one or more URL patterns in the [Mapping Rules](#) section of the integration page so specific calls to your API up the count of specific metrics.' Below this is a table with columns: Metric, System Name, Unit, Description, and Mapped. The table contains one row: 'Hits' with System Name 'hits', Unit 'hit', Description 'Number of API hits', and Mapped status '✓'. A red arrow points to a '+ New metric' button, which is also labeled 'Create new metric' in red text.

You'll now create all of the endpoints that you want to expose and track as methods under an application plan. Methods are the means by which 3scale tracks usage of your APIs. You can set up a method for each of the HTTP methods available on the API endpoints for your API. On the Algolia API, these are known as operations, so you can rename the hits metric to reflect this.



## Application Plans

Application plans define access policies for your API itself, allowing you to differentiate between restricted / limited use (e.g. a testing sandbox) and production or premium usage.

### Default plan

Default application plan (if any) is contracted automatically on service subscription.

Name	State	Create Application Plan		
Starter	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Growth	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Pro	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Enterprise	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>

You also want to track how many records your API users have added to their index, as these are included and charged for as part of their plan. You'll add a new metric called "records" to track these.

Setup fee	<input type="text" value="0.00"/>	USD
Cost per month	<input type="text" value="49.00"/>	USD

[Update Application plan](#)

## Metrics & Limits

Name	Enabled ?	Visible ?	Text only ?	New metric
Operations	<a href="#">Pricing (0)</a>	<a href="#">Limits (0)</a>	<a href="#">New method ?</a>	<a href="#">Edit</a>
Update Object	<a href="#">Pricing (0)</a>	<a href="#">Limits (0)</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
Add Object	<a href="#">Pricing (0)</a>	<a href="#">Limits (0)</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
Delete Index	<a href="#">Pricing (0)</a>	<a href="#">Limits (0)</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
Query Index	<a href="#">Pricing (0)</a>	<a href="#">Limits (0)</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
List Indexes	<a href="#">Pricing (0)</a>	<a href="#">Limits (0)</a>	<a href="#">Edit</a>	<a href="#">Delete</a>

## Features

Name	Description	Enabled?	New feature
Unlimited Greetings		✗	<a href="#">Edit</a> <a href="#">Delete</a>
24/7 support		✗	<a href="#">Edit</a> <a href="#">Delete</a>
Unlimited calls		✗	<a href="#">Edit</a> <a href="#">Delete</a>

## 1.3. INTEGRATING THE API WITH 3SCALE

Now that you've created all of the endpoints you want to expose and track in 3scale and have defined

how your customers will access your API, you need to set up the API gateway to easily integrate with 3scale. For this, you're first going to use the APIcast Cloud Gateway. It's a hosted API gateway based on NGINX that 3scale provides to test-run API integrations with 3scale in a staging environment. It will allow you to confirm that your setup is correct so far and that the endpoints you've entered will be tracked correctly in 3scale.

Go ahead and set up the staging cloud gateway by navigating to the Integration page. The first thing to do there is to enter your API base URL, including the port number such as 80 for HTTP and 443 for HTTPS. This will be your private API backend, which will only be accessible through the API gateway. This would typically be an internal address for the API backend. The next field specifies the APIcast Cloud Gateway location. Your calls to the API backend will go to through this host, and this will be what you hit during your tests instead of calling the private API backend directly.

The next step is to map your actual API endpoints or paths to the metrics you just defined in the application plan. Since there are many actions available on these API endpoints, you'll create a rule for all the HTTP methods available on the API paths.

Once the setup is done, you can test the integration with the staging APIcast Cloud Gateway to ensure the setup will work in production. If all is well, you should get a green line showing a correct test integration between 3scale and your API backend.

The screenshot shows the 3scale dashboard interface. At the top, there's a navigation bar with links for Documentation, Dashboard, Account, Site, and Logout. The main header includes the 3scale logo and a secondary navigation bar with links for Dashboard, Developers, Applications, Billing, Analytics, API, Developer Portal, and Settings. Below this, a sub-navigation bar shows Dashboard, Messages, and Forum. The main content area is divided into three sections: 'Latest Signups' with a table, 'Latest Messages' with a list, and a '5-minute integration guide' with a checklist.

Group/Org.	Admin	Apps
3scale support	buyer	1

Subject	From
✉ API System: New Application submission	3scale support
✉ API System: New Service subscription	3scale support

**Integrate your API in 5 minutes**

This short to-do list is the fastest way to integrate your API with 3scale. Check the [Fast Track Guide](#) for a more in-depth view and to learn more about the [different deployment options](#) to integrate your API with 3scale.

- ☐ Add the Base URL of your API and hit save & test
- ☐ Verify the test call was recorded
- ☐ Deploy your API Gateway to production on APIcast

Now that you've tested the integration, the final step is to set up the authentication mode that your customers will use to access the API. The Algolia REST API uses a pair of credentials, application ID and API key, to authenticate an API call which maps to the app\_id/app\_key authentication mode in 3scale. These parameters are sent as the X-Algolia-API-Key and X-Algolia-Application-Id headers. You can change to this auth mode by changing the integration settings and the name of the headers in the API gateway setup.

The screenshot shows the 3scale API management dashboard. At the top, there's a navigation bar with links: Documentation, Dashboard, Account, Site, Logout, and 3scaleadmin. Below this is a secondary navigation bar with links: Dashboard, Developers, Applications, Billing, Analytics, API (highlighted), Developer Portal, and Settings. The main content area is divided into two sections: Integration and Staging.

**Integration Section:**

- Left sidebar: Overview, ActiveDocs, Integration (selected), Settings, Naming, Alerts, Application plans.
- Integration settings box: Deployment option: APICast Cloud Gateway, Authentication: API Key (user\_key). A link "edit integration settings" is in the top right.
- Text: "Configure your API gateway in the staging environment. Once your staging environment is green you can deploy the gateway to the 3scale production environment."

**Staging Section:**

- Header: "Staging – configure & test your integration" with status "deployed" and a link "deploy history".
- API configuration area: A red puzzle piece icon labeled "API" is on the left. A text input field "Private Base URL\*" contains "https://api.algolia.net:443". A checkbox "Use Hello World API" is checked. Below the input field is the text: "Private address of your API that will be called by the API gateway."

## 1.4. MANAGE API ACCESS

Now that you have the gateway set up, you can start creating application plans to allow different types of access to your API. You'll want to look at any limits you want to enforce on the different application plans, as well as any additional pricing rules for when developers go above their plan limits.


### 1.4.1. Application plans

Application plans can determine rate limits, which methods or resources are accessible, and which features are enabled. Every application accessing your API will be associated with an application plan.

There are already two application plans created by default in the 3scale account, Basic and Unlimited. Delete these and create new ones to fit in with your plans for exposing your APIs to different types of users, depending on their predicted usage volumes. Algolia provides four different pricing plans – starter, growth, pro and enterprise – each with different usage limits. You'll create matching application plans in 3scale to differentiate between them.

Documentation ▾

Dashboard Account Site Logout 3scaleadmin



Dashboard Developers Applications Billing Analytics API Developer Portal Settings

Dashboard Messages Forum

### Latest Signups

You have no subscribed accounts.

### Latest Messages


You have no messages.

### Integrate your API in 5 minutes

This short to-do list is the fastest way to integrate your API with 3scale. Check the [Fast Track Guide](#) for a more in-depth view and to learn more about the [different deployment options](#) to integrate your API with 3scale.

- Add the Base URL of your API and hit save & test ?
- Verify the test call was recorded
- Deploy your API Gateway to production on APIcast

Privacy Refunds Contact

Powered by  3scale

You want to make getting started with the API as quick and smooth as possible, so leave the Applications require approval checkbox unticked in every case. Additionally, although all of the plans charge for access, you'll offer a 14-day free trial period, which you can configure under each plan.

### 1.4.2. Limits and pricing rules

To begin with, you'll implement a limit of 1,000,000 operations per month on the starter plan and a limit of 100,000 records per month.



## Application Plans

Application plans define access policies for your API itself, allowing you to differentiate between restricted / limited use (e.g. a testing sandbox) and production or premium usage.

### Default plan

Default application plan (if any) is contracted automatically on service subscription.


Name	State	<a href="#">+ Create Application Plan</a>		
Starter	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Growth	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Pro	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Enterprise	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>

Additionally, for every set of 1,000 extra operations, you will charge 0.08 USD. For each 1,000 extra records, you will charge 0.80 USD, as per the current Algolia pricing plans. You'll implement this by adding multiple pricing rules, so once you go over the call limit, call 1,000,001 will be charged at 0.08USD, while the next 1,000 calls will be free, and so on.

1. From 1,000,001 - 1,000,001 operations: 0.08 USD
2. From 1,000,002 - 1,001,000 operations: 0 USD
3. From 1,001,001 - 1,001,000 operations: 0.08 USD

Similarly, you'll implement similar pricing rules for the records metric.

Documentation ▾
Dashboard Account Site Logout 3scaleadmin


Dashboard Developers Applications Billing Analytics **API** Developer Portal Settings

Overview ActiveDocs

Integration
Settings
Naming
Alerts
Application plans

### Application Plans


Application plans define access policies for your API itself, allowing you to differentiate between restricted / limited use (e.g. a testing sandbox) and production or premium usage.

**Default plan** Starter ▾  
Default application plan (if any) is contracted automatically on service subscription.

Name	State			
Starter	hidden	Publish	Copy	Delete
Growth	hidden	Publish	Copy	Delete
Pro	hidden	Publish	Copy	Delete
Enterprise	hidden	Publish	Copy	Delete

Create Application Plan

Privacy Refunds Contact

Powered by  3scale

That's the initial setup you need to do. You can always come back to your application plans to add any new methods as well as change or refine any limits later on.

## 1.5. ENGAGE DEVELOPERS

Now that you've set up some limits, the next step is to set up your Developer Portal, so customers can find out more about the API, get their API credentials, and monitor their usage.

The final things to set up on your 3scale account are all around customizing the Developer Portal to fit your particular needs and so that it conforms to the look and feel of your product branding.

You can do this through:

1. [Setting up a custom domain for your Developer Portal](#) such as `developer.algolia.com`.
2. [Setting up an API contact email](#) such as `api@algolia.com` for emails from your 3scale account to your API consumers.
3. [Customizing the Developer Portal](#) so that it matches the look and feel of your brand.
4. [Edit the email templates](#) to tailor the messages you want to send to your customers.
5. [Using liquids](#) to control the information that gets shown to different customers.
6. Adding a favicon

For the favicon, the path has to be different than the standard `/favicon.ico`, as this is already taken by the default 3scale one.

You need to upload your favicon into the 3scale CMS by creating a new file. In this, place it in the images folder and call it icon.ico. You'll then refer to it from the main layout with the following snippet placed in the head element, as below:

```
<!-- Favicon -->
<link rel="shortcut icon" href="/images/icon.ico" type="image/x-icon"
/>
```

The screenshot shows the 3scale dashboard interface. At the top, there's a navigation bar with links: Documentation, Dashboard, Account, Site, Logout, and 3scaleadmin. Below this is a secondary navigation bar with links: Dashboard, Developers, Applications, Billing, Analytics, API, Developer Portal, and Settings. The main content area is divided into several sections:


- DEVELOPERS**: Contains a card for "0 Signups" (last 30 days) and a card for "Potential Upgrades" (0 today). The "Potential Upgrades" card includes text about accounts hitting usage limits and a link to "Web Alerts for Admins of this Account of 100% (and up)".
- 1 ACCOUNT**: A section header.
- MESSAGES (11)**: A section header.
- TODAY**: A section header with a notification "Blue skies ahead...".
- BEFORE TODAY**: A section header with a list of notifications, all starting with "API System: Invoices to review".
- IMPORTANT: Planned Maintenance Saturday 24th ...**: A notification.
- API**: A section header.
- 1 APPLICATION (4 PLANS)**: A section header.
- 0 Hits** (last 30 days): A card showing API hits.
- Top Applications** (0 today): A card showing top applications. It includes text about showing top applications and a link to "making some test calls".

You'll also want to set up some interactive documentation (ActiveDocs) so your customers can explore the API without needing to write any code. If you already have a Swagger spec for your API, navigate to the API service you want to work on and go to the ActiveDocs section, create a new spec and paste your Swagger spec in.



Documentation ▾ Dashboard Account Site Logout 3scaleadmin

---

 Dashboard Developers Applications Billing Analytics API Developer Portal Settings

---

DEVELOPERS

**0 Signups**  
*last 30 days*

**0**  
*today*

**Potential Upgrades**  
*Accounts that hit their Usage Limits in the last 30 days*

In order to show Potential Upgrades, add 1 or more usage limits to your [Application Plans](#).

Furthermore, [Web Alerts for Admins of this Account of 100% \(and up\)](#) should be enabled for service(s) with usage limits.

1 ACCOUNT

MESSAGES (11)

TODAY

*Blue skies ahead...*

BEFORE TODAY

API System: Invoices to review

API System: Invoices to review

API System: Invoices to review

API System: Invoices to review

IMPORTANT: Planned Maintenance Saturday 24th ...

API System: Invoices to review

API System: Invoices to review

API System: Invoices to review

API System: Invoices to review

API System: New Application submission

API System: New Service subscription

API

1 APPLICATION (4 PLANS)

**0 Hits**  
*last 30 days*

**0**  
*today*

**Top Applications**  
*Apps with consistently high traffic in the last 30 days*


In order to show Top Applications you need to have at least one application sending traffic to the API.

Consider [making some test calls](#) from the Integration page to get a feel for what you'd see here.


## 1.6. EXPOSING THE API

Now that you have integrated the API and customized the Developer Portal, start thinking about attracting users to the API. Organizing a hackathon can be a good idea to create some buzz around your API and potentially surface some interesting use cases that you may not have thought of. It can also give you an opportunity to gather direct feedback from potential users of your API about its functionality and usability.

Other than that, from a 3scale setup point of view, you just need to ensure that users are able to sign up to your API freely and can get up and running right away. As such, you'll need to ensure developers are allowed to sign themselves up and that there is no account approval required (the default settings in 3scale).

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

---

 Accounts ▾

Listing


Account Plans


Subscriptions

SETTINGS

**Usage Rules**

Fields Definitions

 Applications >

 Billing >

## Usage Rules

USER ACCOUNT MANAGEMENT ZONE

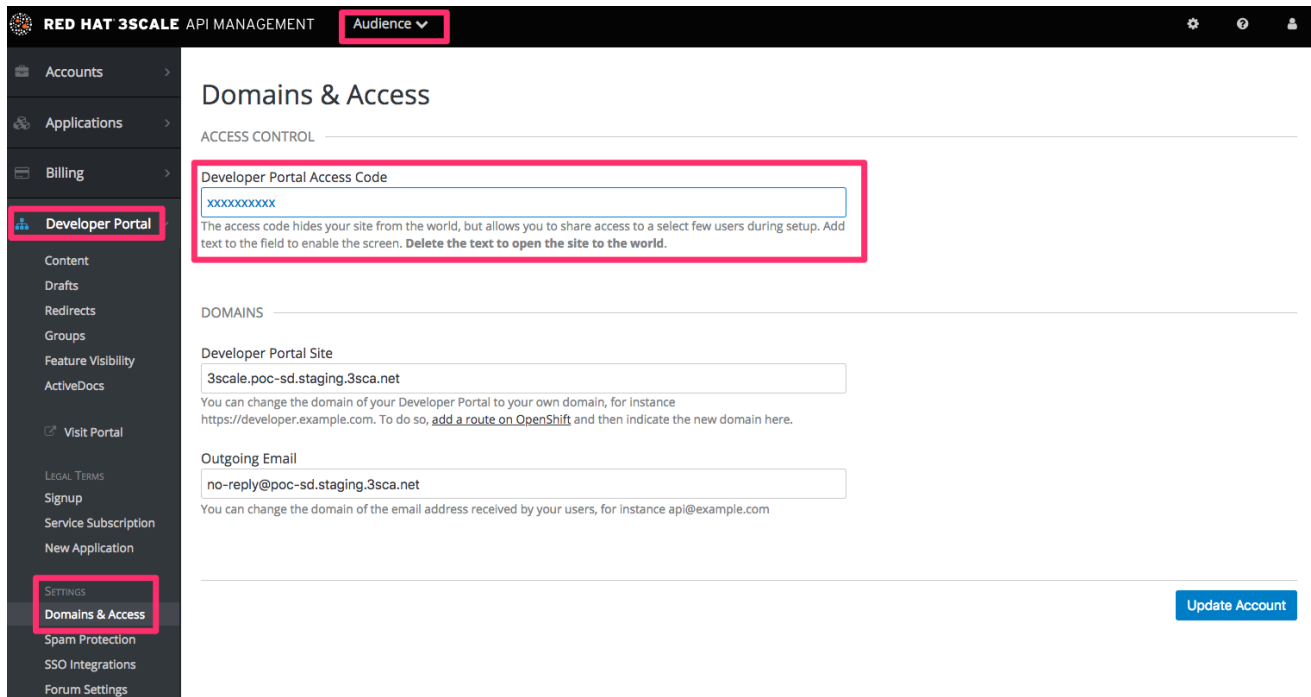
☒ Allow the user to edit their submitted details, change passwords, etc.  
Only disable this if you provide another way to manage this information (e.g. via the User Management API).

SIGNUP

☒ Developers are allowed sign up themselves

☐ Account approval required  
Set per account plan from [Account Plans](#).

You'll also need to make your Developer Portal public by removing the access code.



That's all regarding the integration and setup of your 3scale account. However that's by no means all that 3scale provides. Once your API has been up and running for a while, the analytics module can help give some insight into the usage patterns for your API, which of our customers are most engaged, as well as which are the busiest hours and days for the API. You can read all about how to make the most of the 3scale Analytics [here](#) and [here](#).

What are you waiting for? If you haven't already done so, create your own free account now and give it a go.

## CHAPTER 2. CUSTOMER INTEGRATION

The "customer ecosystem" use case involves using APIs to allow existing customers to automate the processes they run on your system, to use bulk operations or just integrate more tightly with your system. You may also choose to expose your API to partners, in which case you might find the [partner integrations](#) section useful too.

Some examples of successful companies that use APIs to enable customers to tightly integrate with their platform through APIs include Zendesk, a customer support platform that exposes an API to allow their customers to automate and enhance their customer support, and Dropbox, which has both a Core API and Dropbox for Business API that allows business customers to easily administer team accounts and monitor usage and activity.

Although APIs to enable customer integrations are generally publicly documented and free to use, they're often restricted for use by existing customers only. As such, you may want to provide API access automatically for all user accounts, or only to customers on certain contracts. You probably have a pre-existing relationship with your API consumers before they start using your API.

### 2.1. API REQUIREMENTS

Before diving in to the setup for a customer API, review your requirements for the API:

- [Purpose](#)
- [Audience](#)
- [Functionality](#)
- [Access control and security](#)
- [Visibility](#)
- [APIUX](#)

#### 2.1.1. Purpose

The main driver behind exposing an API to your customers would be to allow them to customize their experience, for your core product to closely suit their needs by allowing them to automate processes and integrate with other systems. This level of integration encourages customers to use your product more as well as increasing loyalty due to the time and effort spent integrating.

A final reason could be as an upsell driver such as how Salesforce only allows API access for Enterprise level contracts.

#### 2.1.2. Audience

The audience here is clear – you're targeting existing (as well as potential) customers. If your existing product currently provides distinct contracts or plans, you might choose to differentiate between levels of API access based on these plans. This is especially important if these contracts allow for access to different sets of functionality in your core product.

Additionally, if you're planning to use your API as an upsell driver, you'll want to provide trial access to all customers either with a reduced usage allowance and/or functionality.

#### 2.1.3. Functionality

Since you're offering an API to your existing customers, you'll want to ensure your API offering covers most (if not all) of your core product's functionality as well as ways to import/export data via API and automate configuration changes. You may decide to split these different sets of functionality into multiple APIs or services to be bundled and exposed separately, or you could just offer them within a single API offering.

#### **2.1.4. Access Control and Security**

Now that you have a clear idea of the sort of functionality you want to make available and your audience, you'll want to think about exactly which resources you want to expose as well as the operations you want to expose on them. As previously mentioned, this will probably be tied to any existing contracts you have with your customers.

You should also think about any usage limits that you want to enforce for your customers. Again this may be based on customer type, with higher paying customers receiving a higher usage allowance. You might also want to assess the performance of the different API calls at this point with a view to implementing different usage limits and access rights per method. Are there any methods that are computationally more expensive than others? And if so, is this something that you want to offer with the same usage limits as for other "cheaper" calls? Another alternative would be to only offer such methods to customers on higher plans.

The next thing to think about is the authentication method that you want for your API. API security tends to be a trade off between API credential security and usability of your API. Typically, more secure authentication methods tend to be a hurdle to API integration for your customers. For a customer integration use case, a good compromise between the two would be the combination of an application identifier and a secret key to secure access to your API. Ideally, applications should be able to keep a number of valid secret keys in rotation that can be easily revoked if any of them become compromised.

#### **2.1.5. Visibility**

How you want to publicize and document your API is an important next step to think about. Even though API access will only be available to existing customers, you should still provide a public documentation portal. This will help differentiate you from your competitors, especially if they do not provide an API to enable automation of workflows.

To reduce any friction when provisioning API access for your customers, you should automatically provide API access to any new customers on signup. This will mean that API account creation and management will be maintained on your side.

As such, any customer-specific areas such as API credentials and usage statistics should be available directly from your core product. This can be done by providing single sign-on to the provided Developer Portal or by pulling in the relevant data using 3scale's Account Management API.

#### **2.1.6. APIUX**

Once you're clear on all of the above, it's important to think about what the experience is going to be like for customers integrating with your API. Part of this will be relevant when designing your APIs and choosing an authentication method, but it's also important to document your API in a way that allows for quick and easy testing and integration by your customers.

Some of this will be covered by providing API access credentials to your customers by default as well as providing interactive documentation for customers to test out your API. If you want to encourage customers to use your API, it's a good idea to showcase innovative integrations in a "Customer Success Stories" section on your Developer Portal.

These are all conscious decisions that should be taken upfront when deciding to expose your API, it doesn't mean that you can't change them later, but it's good to have a clear idea of the above and how you are going to implement them.

SendGrid will serve as an example to see the best way to achieve this with 3scale. SendGrid provides a number of APIs that allow their customers and resellers to automate processes and integrate with their own platform and tools. You can find out more about what their APIs look like [here](#). SendGrid maintains multiple versions of their API. The earliest version (v1) is currently deprecated, so this example will mainly be focusing on their v2 and v3 APIs. These use basic authentication for access and return JSON data responses.

## 2.2. DEFINING THE API IN 3SCALE

The first thing to do, after creating a 3scale account of course, is to define all of the existing endpoints for your API and set these up within 3scale. Set this up within the API service which is created by default with your 3scale account by going the **Integration > Methods & Metrics** view in your API space.

**RED HAT 3SCALE** API MANAGEMENT API: Echo API

**Methods & Metrics**

**Methods**

Add the methods of this API to get data on their individual usage. Method calls trigger the built-in Hits-metric. Usage limits and pricing rules for individual methods are defined from within each [Application Plan](#). A method needs to be mapped to one or more URL patterns in the [Mapping Rules](#) section of the integration page so specific calls to your API up the count of specific methods.

Method	System Name	Unit	Description	Mapped
You have no methods				

[Create new method](#)

**Metrics**

Hits is the built-in metric to which all methods report. Additional top-level metrics can be added here in order to track other usage that shouldn't increase the hit count. A metric needs to be mapped to one or more URL patterns in the [Mapping Rules](#) section of the integration page so specific calls to your API up the count of specific metrics.

Metric	System Name	Unit	Description	Mapped
Hits	hits	hit	Number of API hits	✓

[Create new metric](#)

Now create all of the endpoints that you want to expose and track as methods. Methods are the means by which 3scale tracks usage of the APIs. You can set up a method for each of the API endpoints (and potentially also operations on those API endpoints) available under the APIs. You can be as specific or as broad as you like when defining your methods. For now, you're only interested in tracking usage at the API level, so begin by creating a metric for each SendGrid API.

Setup fee

0.00

USD

Cost per month

49.00

USD

Update Application plan

### Metrics & Limits

Name		Enabled ?	Visible ?	Text only ?	New metric
Operations	<a href="#">Pricing (0)</a> <a href="#">Limits (0)</a> <a href="#">New method ?</a>	✓	✓	✓	<a href="#">Edit</a>
Update Object	<a href="#">Pricing (0)</a> <a href="#">Limits (0)</a>	✓	✓	✓	<a href="#">Edit</a> <a href="#">Delete</a>
Add Object	<a href="#">Pricing (0)</a> <a href="#">Limits (0)</a>	✓	✓	✓	<a href="#">Edit</a> <a href="#">Delete</a>
Delete Index	<a href="#">Pricing (0)</a> <a href="#">Limits (0)</a>	✓	✓	✓	<a href="#">Edit</a> <a href="#">Delete</a>
Query Index	<a href="#">Pricing (0)</a> <a href="#">Limits (0)</a>	✓	✓	✓	<a href="#">Edit</a> <a href="#">Delete</a>
List Indexes	<a href="#">Pricing (0)</a> <a href="#">Limits (0)</a>	✓	✓	✓	<a href="#">Edit</a> <a href="#">Delete</a>

### Features

Name	Description	Enabled?	New feature
Unlimited Greetings		✗	<a href="#">Edit</a> <a href="#">Delete</a>
24/7 support		✗	<a href="#">Edit</a> <a href="#">Delete</a>
Unlimited calls		✗	<a href="#">Edit</a> <a href="#">Delete</a>

## 2.3. INTEGRATING THE API WITH 3SCALE

Now that you've created all of the endpoints that you want to expose and track in 3scale, and you have defined how customers will access the API, you need to set up the API gateway to easily integrate with 3scale. For this, you're first going to use the APIcast Cloud Gateway. This is a hosted API gateway based on NGINX that 3scale provides to test-run your API integration with 3scale in a staging environment. This will allow you to confirm that your setup is correct so far and that the endpoints you've entered will be tracked correctly in 3scale.

Go ahead and set up the staging cloud gateway by navigating to the Integration page. The first thing to do here is to enter in your API base URL, including the port number such as 80 for HTTP and 443 for HTTPS. This will be your private API backend, which will only be accessible through the API gateway. This would typically be some internal address for the API backend. The next field specifies the APIcast cloud gateway location. Your calls to the API backend will go to through this host, and this will be what you hit during your tests instead of calling the private API backend directly.

The next step is to map your actual API endpoints or paths to the metrics you just defined in the application plan. Since there are many actions available on these API endpoints, you'll create a rule for all the HTTP methods available on the API paths.

Once the setup is done, you can test the integration with the staging APIcast Cloud Gateway to ensure the setup will work in production. If all is well, you should get a green line showing a correct test integration between 3scale and your API backend.

The screenshot shows the 3scale API dashboard. At the top, there's a navigation bar with 'Help', 'Dashboard', 'Account', 'Site', 'Logout', and '3scaleadmin'. Below this, the 3scale logo is on the left, and a menu with 'Dashboard', 'Developers', 'Applications', 'Billing', 'Analytics', 'API' (highlighted), 'Developer Portal', and 'Settings' is on the right. Under the 'API' menu, there are links for 'Overview' and 'ActiveDocs'. The main content area is titled 'API' and has a '+ Create Service' button. It is divided into two columns. The left column is 'Integration and Settings', which contains instructions on how to integrate the API and a button to 'Configure the APICast Cloud Gateway'. Below this, it shows authentication details: 'Authenticated by API key', 'ID for API calls is 2555417726753 and system name is api', and a list of permissions. The right column is 'Stats', which shows usage for four different APIs: Web API, Sub-User API, Parse API, and Marketing Email API, each with a bar chart and '0 hits 0%' status. At the bottom, there are links for 'Published Application plans' and 'Latest alerts'.

Now that you've tested the integration, the final step is to set up the authentication mode that your customers will use to access the API. The SendGrid Web API uses a pair of credentials: `api_user` and `api_key` to authenticate an API call. These are sent as query parameters. This maps to the `app_id/app_key` authentication mode in 3scale. Change to this auth mode by changing the integration settings.

The screenshot shows the 3scale Integration settings page. The navigation bar is the same as the previous screenshot. The main content area is titled 'Integration' and has a sidebar on the left with links for 'Integration', 'Settings', 'Naming', 'Alerts', and 'Application plans'. The 'Integration' section shows the 'Deployment option' as 'APICast Cloud Gateway' and 'Authentication' as 'API Key (user\_key)'. There is a button to 'edit integration settings'. Below this, there is a section titled 'Staging - configure & test your integration' with a 'deployed' status and a link to 'deploy history'. The 'Private Base URL\*' is set to 'https://hello-world-api.3scale.net:443'. A note below the URL states: 'Private address of your API that will be called by the API gateway.'

Once that is done, also modify the gateway authentication settings. In order to match the SendGrid API as closely as possible, change the 3scale credential name to the equivalent SendGrid credential name. Leave the setting for the credentials location as is, as the SendGrid credentials are also sent as query parameters.

Finally, you also want to restrict calls to your API backend to come from your gateway only. You can do this by specifying a secret token that the gateway should send with every request. That way, you can ensure any calls you get are from the gateway and your API backend can reject any calls not containing this token. For now, leave this at its default value.

See [Deployment Options](#) documentation to learn more about how to deploy your API gateway for production use.

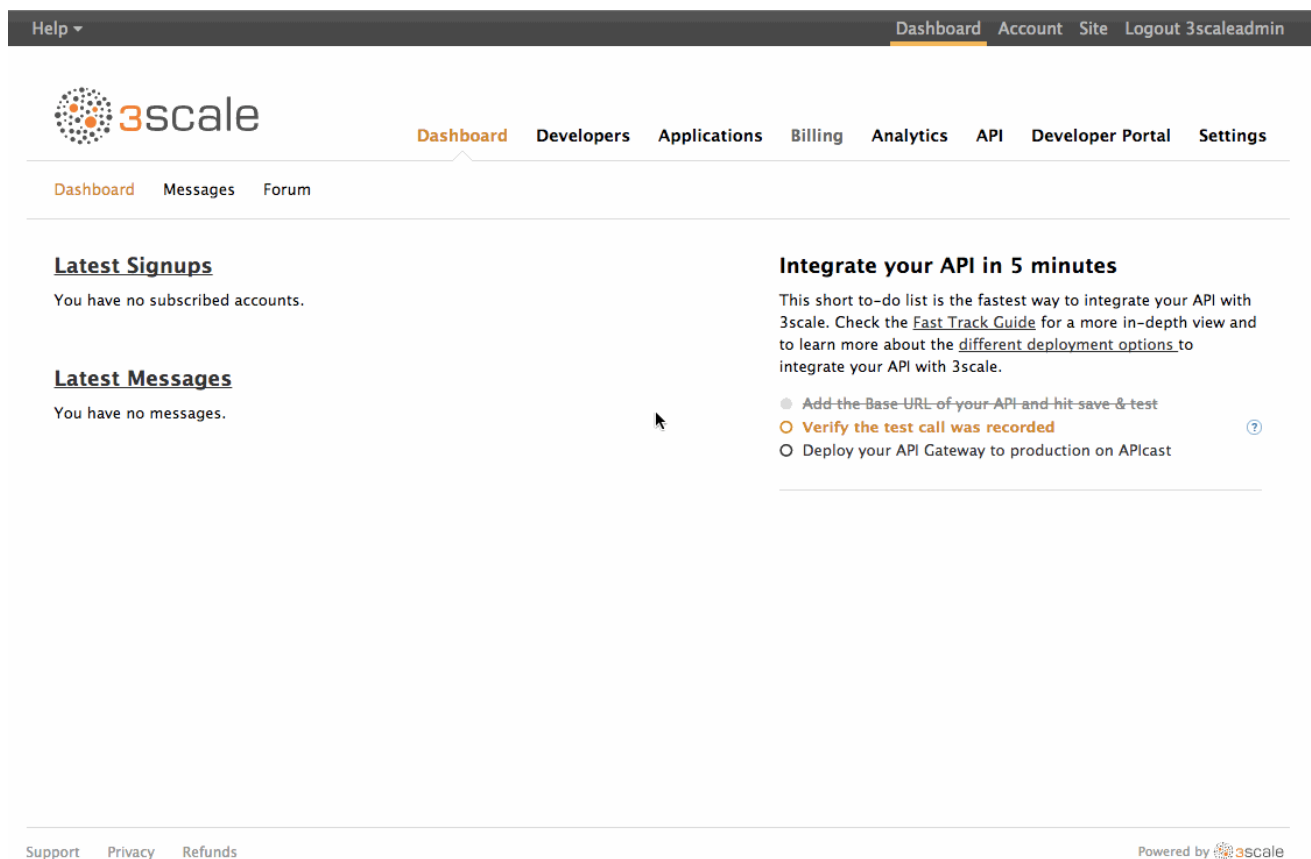
## 2.4. MANAGE API ACCESS

Now that you have the gateway set up, you can start creating application plans to allow different types of access to your API. You'll want to look at which additional metrics you want to expose and track for the different application plans, as well as what usage limits you want to enforce for their applications using your API.

### 2.4.1. Application plans

Application plans can determine can determine rate limits, which methods or resources are accessible, and which features are enabled. Every application accessing your API will be associated with an application plan.

You'll already have two application plans created by default with your 3scale account: Basic and Unlimited. Delete these and create new ones to fit in with your plans for exposing your APIs to different types of customers, depending on their existing pricing plan with SendGrid. SendGrid provides four different pricing plans – Bronze, Silver, Gold and Platinum – each with access to a different set of APIs. As such, you'll create matching application plans in 3scale to differentiate between them.



The screenshot shows the 3scale dashboard interface. At the top, there's a navigation bar with 'Help' and links for 'Dashboard', 'Account', 'Site', and 'Logout 3scaleadmin'. Below this is a secondary navigation bar with '3scale' logo and links for 'Dashboard', 'Developers', 'Applications', 'Billing', 'Analytics', 'API', 'Developer Portal', and 'Settings'. The 'Dashboard' link is highlighted. Under the 'Dashboard' link, there are sub-links for 'Dashboard', 'Messages', and 'Forum'. The main content area is divided into two columns. The left column has two sections: 'Latest Signups' with the text 'You have no subscribed accounts.' and 'Latest Messages' with the text 'You have no messages.'. The right column has a section titled 'Integrate your API in 5 minutes' with a paragraph of text and a list of three steps: 'Add the Base-URL of your API and hit save & test', 'Verify the test call was recorded' (highlighted in orange), and 'Deploy your API Gateway to production on APIcast'. A mouse cursor is visible over the 'Verify the test call was recorded' step. At the bottom of the dashboard, there are links for 'Support', 'Privacy', and 'Refunds', and a footer that says 'Powered by 3scale'.



You'll want to give your customers access to all of the APIs available under their SendGrid plan by default when they create an account, so leave the Applications require approval checkbox unticked in every case.

In each plan, disable any metrics that are not available under that plan. Later on, you might also choose to track usage at the individual endpoint level and create methods for that.

As for the SendGrid monthly plans, you'll only allow access to the Web APIs for the Bronze plan. So disable all other APIs in the Bronze application plan.

## 2.4.2. Limits

To begin with, you don't really want to place any limits on customers using your API. However, you will enforce an overall throttling limit on the hits metric to ensure your API is not brought down by large numbers of requests from a rogue app or script.

Help ▾ Dashboard Account Site Logout 3scaleadmin

3scale Dashboard Developers Applications Billing Analytics API Developer Portal Settings

Overview ActiveDocs

Integration

Settings

Naming

Alerts

Application plans

### Application Plans

Application plans define access policies for your API itself, allowing you to differentiate between restricted / limited use (e.g. a testing sandbox) and production or premium usage.

**Default plan**

Default application plan (if any) is contracted automatically on service subscription.

Name	State			
Bronze	hidden	Publish	Copy	Delete
Silver	hidden	Publish	Copy	Delete
Gold	hidden	Publish	Copy	Delete
Platinum	hidden	Publish	Copy	Delete

[Create Application Plan](#)

Support Privacy Refunds

Powered by 3scale

That's the initial setup you need to do. You can always come back to your application plans to add any new methods as well as to change or refine any limits later on, once your API is running in production and you have a better understanding of your customers' usage patterns. As your API program matures, you can look for ways to use your API for upsell opportunities based on your customers' usage.

## 2.5. ENGAGE DEVELOPERS

Now that you've set some limits, the next step is to set up the Developer Portal so your customers can find out more about the API, get their API credentials, and monitor their usage.

The final things to set up on your 3scale account are all around customizing the Developer Portal to fit your particular needs and so that it conforms to the look and feel and of your product branding.

You can do this through:

1. [Setting up a custom domain for your Developer Portal](#) such as `developer.sendgrid.com`.
2. [Setting up an API contact email](#) such as `api@sendgrid.com` for emails from your 3scale account to your API consumers.
3. [Customizing the Developer Portal](#) so it matches the look and feel of your brand.
4. [Editing the email templates](#) to tailor the messages you want to send to your customers.
5. [Using liquids](#) to control the information that gets shown to different customers.
6. Adding a favicon

For the favicon, the path has to be different from the standard `/favicon.ico`, as this is already taken by the default 3scale one.


You need to upload your favicon into the 3scale CMS by creating a new file. In this, place it in the images folder and call it `icon.ico`. You'll then refer to it from the main layout with the following snippet placed in the head element, as below:

```
<!-- Favicon -->
  <link rel="shortcut icon" href="/images/icon.ico" type="image/x-icon"
/>
```

The screenshot shows the 3scale Developer Portal dashboard. The top navigation bar includes 'Help', 'Dashboard', 'Account', 'Site', and 'Logout 3scaleadmin'. The main navigation bar includes 'Dashboard', 'Developers', 'Applications', 'Billing', 'Analytics', 'API', 'Developer Portal', and 'Settings'. The 'Dashboard' section is active, showing 'Latest Signups' (no subscribed accounts), 'Latest Messages' (a table with one message about planned maintenance), and 'Integrate your API in 5 minutes' (a list of steps to integrate the API).

You'll also want to set up some interactive documentation (ActiveDocs) so your customers can explore the API without needing to write any code. If you already have a Swagger spec for your API, navigate to the API service you want to work on and go to the ActiveDocs section, create a new spec and paste your Swagger spec in.

Help ▾
Dashboard
Account
Site
Logout 3scaleadmin


Dashboard
Developers
Applications
Billing
Analytics
API
Developer Portal
Settings

Dashboard
Messages
Forum

### Latest Signups

You have no subscribed accounts.

### Latest Messages


Subject	From
✉ IMPORTANT: Planned Maintenance Monday 11th May	3scale Inc.

### Integrate your API in 5 minutes

This short to-do list is the fastest way to integrate your API with 3scale. Check the [Fast Track Guide](#) for a more in-depth view and to learn more about the [different deployment options](#) to integrate your API with 3scale.

- Add the Base URL of your API and hit save & test ?
- Verify the test call was recorded
- Deploy your API Gateway to production on APIcast ?

Support
Privacy
Refunds

Powered by  3scale

## 2.6. EXPOSING THE API

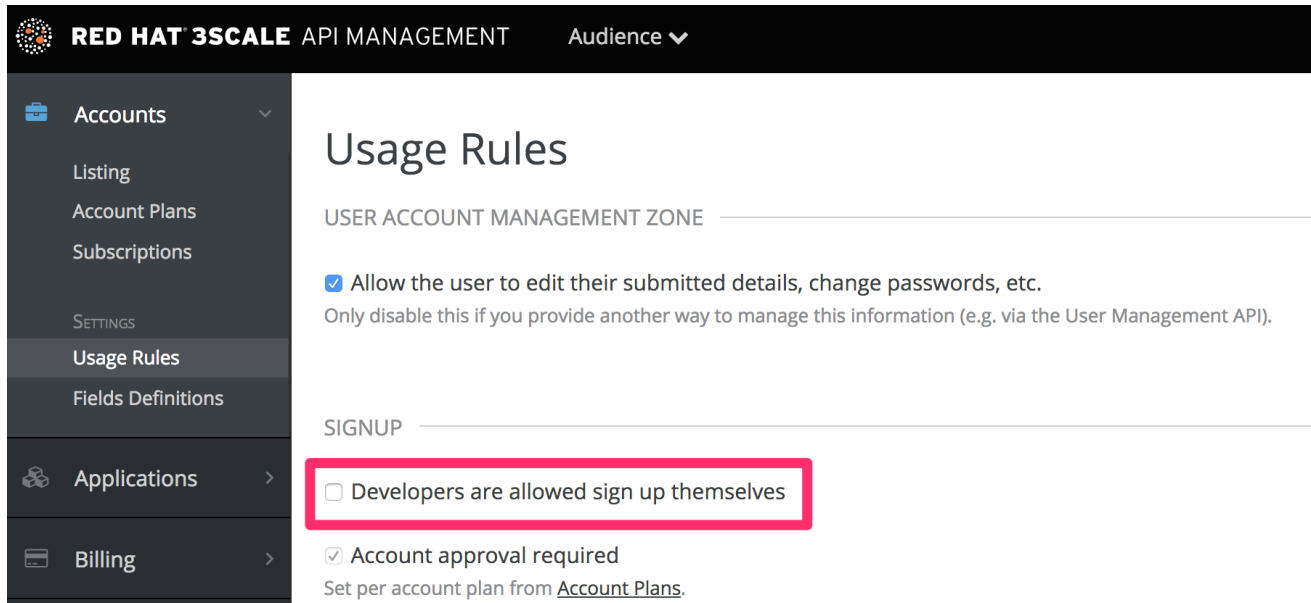
Now that you have the API integrated and have customized your Developer Portal to fit your needs, you can start thinking about giving customers access to the APIs.

Since you want to allow customers to get access to the API by default, you'll want to create an account for them in 3scale when they sign up to any monthly plan in SendGrid. You can use the signup express method in the 3scale Account Management API to do this. The idea is that you'll call this method from your own account creation code, as in the snippet below.

```
curl -v -X POST "https://sendgrid-admin.3scale.net/admin/api/signup.xml"
-d
'provider_key=PROVIDER_KEY&org_name=COMPANY_NAME&username=USERNAME&email=E
MAIL&password=PASSWORD&application_plan_id=MONTHLY_PLAN_ID'
```

You'll need to make sure you pass in the correct application plan ID in 3scale to match the equivalent monthly plan in SendGrid so customers get access to the relevant APIs.

You'll also disable signups directly on the 3scale Developer Portal to prevent anyone who's not a customer from getting access credentials to your APIs.



The screenshot shows the Red Hat 3scale API Management console. The top navigation bar includes the Red Hat 3scale logo, 'API MANAGEMENT', and a dropdown menu for 'Audience'. The left sidebar contains a menu with 'Accounts' (expanded), 'Listing', 'Account Plans', 'Subscriptions', 'SETTINGS', 'Usage Rules' (selected), 'Fields Definitions', 'Applications', and 'Billing'. The main content area is titled 'Usage Rules' and is divided into two sections: 'USER ACCOUNT MANAGEMENT ZONE' and 'SIGNUP'. In the 'USER ACCOUNT MANAGEMENT ZONE', there is a checked checkbox for 'Allow the user to edit their submitted details, change passwords, etc.' with a note: 'Only disable this if you provide another way to manage this information (e.g. via the User Management API)'. In the 'SIGNUP' section, there is an unchecked checkbox for 'Developers are allowed sign up themselves' which is highlighted with a red rectangle, and a checked checkbox for 'Account approval required' with a note: 'Set per account plan from [Account Plans](#)'.

Finally, to make the experience completely seamless, you'll use 3scale's SSO capabilities to automatically login your customers to the 3scale Developer Portal when they want to get their access credentials. This will require you to previously authenticate them on your side. You can read more about how to use 3scale's SSO [here](#).

That's all regarding the integration and setup of your 3scale account. However that's by no means all that 3scale provides. Once your API has been up and running for a while, the analytics module can help give some insight into the usage patterns for your API, which of your customers are most engaged, as well as which are the busiest hours and days for your API. You can read all about how to make the most of the 3scale Analytics [here](#) and [here](#).

What are you waiting for? If you haven't already done so, start your free trial now and give it a go.

## CHAPTER 3. PARTNER INTEGRATION

The "partner ecosystem" use case involves allowing third parties to provide added value to your existing service, either for your own platform or customers using your platform. It can also be used to create a marketplace of products built on top of your API.

Salesforce and New Relic are great examples of successful companies that expose their APIs to enable their ecosystems to flourish. They're service providers that have taken advantage of partner programs and extended their APIs to meet larger requirements.

What does it take for a company to provide a successful partner API?

### 3.1. API REQUIREMENTS

There are some main areas that are important to think about and cover before jumping in and exposing an API:

- [Purpose](#)
- [Audience](#)
- [Functionality](#)
- [Access Control and Security](#)
- [Visibility](#)
- [APIUX](#)

#### 3.1.1. Purpose

A typical reason to open up your API to partners is to allow third parties to provide a specific service to your customers that enhances or complements your existing offering. This has the dual benefit of driving extra traffic to your business as well as to your chosen partner.

If you have many customers, you'll find that they each have very specific and varied needs that are not always exactly met by your service. One alternative to providing bespoke integrations on a case-by-case basis (or losing potential customers to in-house builds) is to provide an API that others can use to provide these integrations. For example, Twitter's API has created a sandbox for third-party apps, which leverages the reach and user base of the social network.

#### 3.1.2. Audience

You'll probably want to expose your API to individual developers as well as to potential partners, so it's important to have a clear vision of how you're going to differentiate between these two groups up front.

Typically when exposing an API to private partners, it's to enable them to provide connectors to their own services. This increases customer traffic to both your service and theirs. For example, the Heroku addons ecosystem (<https://addons.heroku.com/>) allows Heroku to offer functionality and services beyond what their own platform can provide. Similarly, New Relic Connect (<http://newrelic.com/connect>) allows partners to provide New Relic integrations within their apps.

With public developers, the idea is more to provide a public marketplace of applications to augment the functionality of your product – just as Facebook did with its API, which enabled a whole marketplace of social gaming. They managed to diversify their revenue and find new ways to engage with their users.

### 3.1.3. Functionality

Once you've thought about your use cases and audience, you'll probably want to think about how to enable these through your API. If you're aiming to become a platform, you might want to think about exposing multiple APIs. This gives you more fine-grained control over the services that you expose as well as the ability to track access to each API separately.

One way that Twitter does this is by providing different APIs for different audiences: a public REST API, beta access to the Collections API on request, as well as both restricted public access to their Streams API.

### 3.1.4. Access Control and Security

Once you've defined the purpose, audience, and functionality for your API, think about which resources you want to expose as well as the operations you'll allow on them. For example, you may wish only to expose read-only methods on your API. Alternatively, you might choose to expose write methods to a certain segment of your API consumers.

You'll also want to think about usage limits, if any, you want to impose on the methods available through your API. These will probably be based on customer segment, with private partners receiving a higher allowance than public developers.

Choosing the correct authentication method for your API is also important. You'll need to think about the trade-off between API credential security and the usability of your API. If you want to expose any APIs to enhance the offering to your customers, you'll probably want to implement an OAuth 2.0 flow. This way your users can have complete control over which of the applications that are using your API can access their data and under which conditions, just as you see every day when you use social network credentials from Google, Facebook, or Twitter to sign up to new applications.

For other APIs not requiring access to your users' data, you will probably want to use the combination of an application identifier and a secret key to secure access to your API. Ideally, applications should be able to keep a number of valid secret keys in rotation that can be easily revoked if any of them become compromised.

### 3.1.5. Visibility

An important next step is to think about the signup flow you want potential partners to follow in order to get access to your API. Although APIs to enable partner integrations are generally publicly documented and might be free to use, they're often restricted for use by strategic partners and access is only available on-demand after following some sort of approval process.

You'll probably want to provide a public Developer Portal to allow people to learn more about your API and potentially provision some sort of trial access with restricted functionality, so they can do a proof of concept integration against your API. This sort of self-service approach can leave you free to spend more time helping larger partners to integrate.

Additionally, you probably want to have some areas and documentation on your Developer Portal that are only accessible to fully qualified partners.

### 3.1.6. APIUX

Once you're clear on all of the above, it's important to think about what the experience is going to be like for partners discovering and integrating with your API. Part of this will be relevant when designing your APIs and choosing an authentication method, but it's also important to document your API in a way that allows for quick and easy testing and integration by your partners.

A forum is another way to build community around your API and to share knowledge about how to use the platform. The more usable and useful your API is, the more partners which will flock to your platform and grow your market share. Both Salesforce and Twitter, as well as many other companies with successful partner API programs, provide forums for their developers to share. As well as building a community, a forum can also be another channel through which to reach your API consumers by answering any questions they might have, making announcements about your API, and providing additional documentation.

Finally, you want to make sure that your partners are successful by making sure they are visible to your customers. A marketplace or app gallery is a good way to achieve this. A lot of successful platforms have dedicated subdomains devoted to showcasing applications or integrations that complement their service. Heroku has their addons "app store for developers" at [addons.heroku.com](https://addons.heroku.com), whereas Salesforce opts for the aspirational [success.salesforce.com](https://success.salesforce.com).

These are all conscious decisions that should be taken up front when deciding to expose your API. It doesn't mean you can't change them later, but it's good to have a clear idea of the above and how you're going to implement them.

Dive in and see the best way to achieve this with 3scale. Twitter and their APIs will serve as an example. You can see an overview of their offering on the Twitter platform [here](#).

This example will integrate their [public REST API](#) with 3scale. This API provides access to read and write Twitter data programmatically. The API uses OAuth for identifying both users and applications and returns JSON data responses. For the purpose of this example, pretend that this API currently has no authentication mechanisms in place and is completely unsecured and open for anyone to use.

## 3.2. DEFINING THE API IN 3SCALE

The first thing to do, after creating a 3scale account of course, is to define all of the existing endpoints for your API and set these up within 3scale. Set this up within the API service which is created by default with your 3scale account by going to the **[your\_API\_name] > Integration > Methods & Metrics** view.

**RED HAT 3SCALE** API MANAGEMENT API: Echo API

**Methods & Metrics**

**Methods**

Add the methods of this API to get data on their individual usage. Method calls trigger the built-in Hits-metric. Usage limits and pricing rules for individual methods are defined from within each [Application Plan](#). A method needs to be mapped to one or more URL patterns in the [Mapping Rules](#) section of the integration page so specific calls to your API up the count of specific methods.

Method	System Name	Unit	Description	Mapped
You have no methods				

[Create new method](#)

**Metrics**

Hits is the built-in metric to which all methods report. Additional top-level metrics can be added here in order to track other usage that shouldn't increase the hit count. A metric needs to be mapped to one or more URL patterns in the [Mapping Rules](#) section of the integration page so specific calls to your API up the count of specific metrics.

Metric	System Name	Unit	Description	Mapped
Hits	hits	hit	Number of API hits	✓

[Create new metric](#)

Now create all of the endpoints that you want to expose and track as methods. Methods are the means by which 3scale tracks usage of your API endpoints. You need to set up a method for each of the API endpoints (and potentially also operations on those API endpoints) that you want to track. These methods will then be shared across all application plans and can be enabled/disabled accordingly. You can be as specific or as broad as you like when defining your methods. In this case, you'll create a method for each method defined in the Twitter Public REST API.

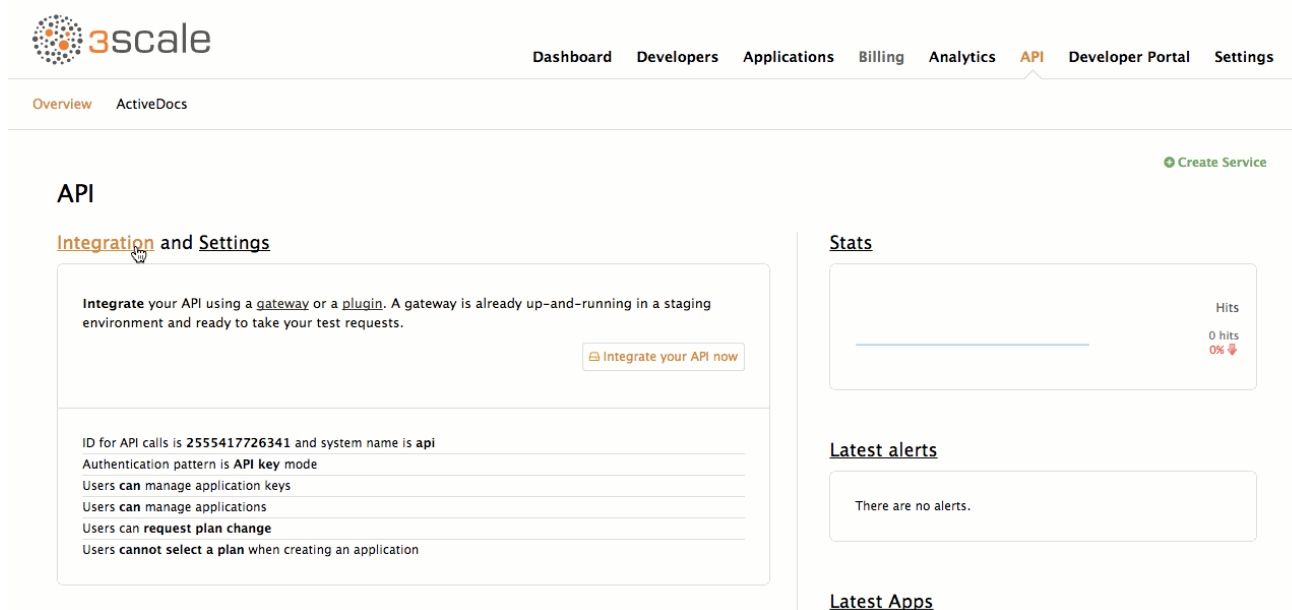
### 3.3. INTEGRATING THE API WITH 3SCALE

Now that you've created all of the endpoints that you want to expose and you've defined how your partners will access the API, you need to set up the API gateway to easily integrate with 3scale. For this, you're first going to use the APIcast Cloud Gateway, a hosted API gateway based on NGINX that 3scale provides to test-run your API integration with 3scale in a staging environment. This will allow you to confirm that the setup is correct so far and that the endpoints you've entered will be tracked correctly in 3scale.

Go ahead and set up the staging cloud gateway by navigating to the Integration page. The first thing to do here is to enter in your API base URL, including the port number such as 80 for HTTP and 443 for HTTPS. This will be your private API backend, which will only be accessible through the API gateway. This would typically be some internal address for the API backend. The next field specifies the APIcast Cloud Gateway location. Your calls to your API backend will go to through this host, and this will be what you'll hit during your tests instead of calling the private API backend directly.

The next step is to map your actual API endpoints or paths to the methods you just defined in the application plan. You'll choose the HTTP method that's available on the specific endpoint path as well and select the equivalent method to map against. Different operations (GET, PUT, POST, DELETE, etc...) on the same endpoint can be tracked separately.

Once the setup is done, you can test the integration with the staging APIcast Cloud Gateway to ensure your setup will work in production. If all is well, you should get a green line showing a correct test integration between 3scale and your API backend.



**3scale**

Dashboard Developers Applications Billing Analytics **API** Developer Portal Settings

Overview ActiveDocs

**API**

Integration and Settings

Integrate your API using a gateway or a plugin. A gateway is already up-and-running in a staging environment and ready to take your test requests.

[Integrate your API now](#)

ID for API calls is 2555417726341 and system name is api

Authentication pattern is API key mode

Users can manage application keys

Users can manage applications

Users can request plan change

Users cannot select a plan when creating an application

**Stats**

Hits

0 hits

0%

**Latest alerts**

There are no alerts.

**Latest Apps**

Now that you've tested the integration, the final step is to set up the authentication mode that your customers will use to access your API. Since you're interested in building a secure API, and one where your customers have complete control over what data they expose to applications consuming your API, select OAuth mode.



The screenshot displays the 3scale API Gateway configuration page. At the top, there are three mapping rules listed:

- GET /1.1/statuses/mentions\_timeline.json (1) get\_status: [edit] [delete]
- POST /1.1/statuses/retweets/{id}.json (1) post\_stat: [edit] [delete]
- GET /1.1/statuses/retweets/{id}.json (1) get\_status: [edit] [delete]

A green button labeled "Add Mapping Rule" is located below the list. Below the mapping rules is a section titled "AUTHENTICATION SETTINGS". Under this section, there is a "CLIENT" field with a question mark icon. Below the client field, there is a section titled "API test GET request" with a text input field containing the URL: `/1.1/statuses/retweets/580886189592444928.json`. Below the input field, there is a text block explaining the purpose of the request and providing a curl command:

```
Optional GET request to a API gateway endpoint. We will use this call to validate
your API gateway setup using credentials of the first live application. You can try it
yourself by copying the following command into your shell:

curl "https://api-
2445581227119.staging.apicast.io:443/1.1/statuses/retweets/58088
user_key=04f2873f91f02902ba3edfdb9d2cf5ae"
```

At the bottom left, there is a green message: "Connection between client, gateway & API is working correctly as reflected on the stats page." At the bottom right, there is a blue "Test" button.

Changing the OAuth mode will require some extra data to be entered in the Integration page. If you navigate there now, you'll see there's an additional field to fill in, the OAuth login URL. This is an authorization page that you'll host where your (Twitter's) users can log in to authenticate themselves and authorize access to their data on the Twitter platform.

At this point, it's worth giving a brief overview of how OAuth flows work at 3scale. Since 3scale does not hold any details on our customers, we need to have some way for them to authorize access to their data and for us to let the API gateway know that this has happened so that the access token can be issued and generated. This is typically done by providing an authorization page, which is hidden behind a login page. When a user authorizes an application to access their data, the authorization page should then call back the API gateway so that an access token for that user can be issued to the application.

Once all of this is done, the API gateway can inform 3scale of the `access_token` that has been issued for the application and user, so they can then be used to check for authorization and report usage against the correct application. This authentication mode requires a bit more integration between your application backend and our gateway.

You'll also modify the gateway authentication settings. In order to match the Twitter API as closely as possible, you'll set the authentication parameters to be sent in the headers.

You also want to restrict calls to your API backend to come from the gateway only. You can do this by specifying a secret token that the gateway should send with every request. That way, you can ensure any calls you get are from the gateway, and your API backend can reject any calls not containing this token.

▼ AUTHENTICATION SETTINGS

**Host Header**

Lets you define a custom Host request header. This is needed if your API backend only accepts traffic from a specific host.

**Secret Token**

Enables you to block any direct developer requests to your API backend; each 3scale's API gateway call to your API backend contains a request header called `x-3scale-proxy-secret-token`. The value of this header can be set by you here. It's up to you ensure your backend only allows calls with this secret header.

**CREDENTIALS LOCATION\***

☒ As HTTP Headers

☐ As query parameters (GET) or body parameters (POST/PUT/DELETE)

See [Deployment Options](#) documentation to learn more about how to deploy your API gateway for production use.

## 3.4. MANAGE API ACCESS


Now that you have the gateway set up, you can start creating application plans to allow different types of access to your API. You'll want to look which methods you want to expose for the different application plans, as well as what usage limits you want to enforce for the applications using your API.

### 3.4.1. Application plans

Application plans can determine rate limits, which methods or resources are accessible, and which features are enabled. Every application accessing your API will be associated with an application plan.

You'll already have two application plans created by default with your 3scale account: Basic and Unlimited. Delete these and create new ones to fit in with your plans for exposing your API to different types of partners.

To begin with, you'll want to allow potential new partners to test-run the API so they can see if it fits their needs, but you'll also want some way to upsell them to a more powerful plan later on if there's a good fit. For this, you'll create a Trial plan to give anyone who's interested a limited number of calls to play around with the API and get a taste for what's possible. You want this to be the default plan when people sign up to register their interest in becoming a partner and use your API.



Dashboard Developers Applications Billing Analytics **API** Developer Portal Settings

Overview ActiveDocs

[+ Create Service](#)

## API

### Integration and Settings

Integrate your API using a [gateway](#) or a [plugin](#). A gateway is already up-and-running in a staging environment and ready to take your test requests.

[Integrate your API now](#)

ID for API calls is **2555417726274** and system name is **api**

Authentication pattern is **API key mode**

Users **can** manage application keys

Users **can** manage applications

Users can **request plan change**

Users **cannot select a plan** when creating an application

### Stats

Hits


### Latest alerts

There are no alerts.

### Latest Apps

There are no latest applications.

The next plan you'll set up will be for public developers. Make this plan visible to everyone who signs up, but make developers request to be put on this plan so you can qualify them.



Dashboard Developers Applications Billing Analytics **API** Developer Portal Settings

Overview ActiveDocs

Integration

Settings

Naming

Alerts

**Application plans**

## Create Application Plan

Name\*

System name\*


Only ASCII letters, numbers, dashes and underscores are allowed.

☐ Applications require approval?

Set whether or not applications can be created on demand or if approval is required from you before they are activated.

[Create Application plan](#)

You'll also add a hidden partner plan for when you've weeded out some meatier integration prospects. You don't want any signups to see this plan, so you'll keep it hidden.



Dashboard Developers Applications Billing Analytics **API** Developer Portal Settings

Overview ActiveDocs

Integration  
Settings  
Naming  
Alerts

**Application plans**

### Application Plans

Application plans define access policies for your API itself, allowing you to differentiate between restricted / limited use (e.g. a testing sandbox) and production or premium usage.


**Default plan** Trial  
Default application plan (if any) is contracted automatically on service subscription.

Name	State			<a href="#">Create Application plan</a>
Trial	published	<a href="#">Hide</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Public	published	<a href="#">Hide</a>	<a href="#">Copy</a>	<a href="#">Delete</a>

### 3.4.2. Limits and access

The Twitter API enforces two types of limits, per user (user auth) and per application (app auth). Concentrate on the application rate limits first. In 3scale, these are specified for each method under the application plan.

Go to **[your\_API\_name] > Applications > Application Plans** and click on the Public application plan to get started applying limits. You can start with the GET statuses/mentions\_timeline method. Since Twitter only allows 15 calls in a 15 minute period to this method, set this up in 3scale as a limit of 1 call per minute as well as a limit of 60 calls per hour. Repeat the same process with the rest of the methods, applying any limits specified for the each method.



Dashboard Developers Applications Billing Analytics **API** Developer Portal Settings

Overview ActiveDocs

Integration  
Settings  
Naming  
Alerts

**Application plans**

### Application Plans

Application plans define access policies for your API itself, allowing you to differentiate between restricted / limited use (e.g. a testing sandbox) and production or premium usage.

**Default plan** Trial  
Default application plan (if any) is contracted automatically on service subscription.

Name	State			<a href="#">Create Application plan</a>
Trial	published	<a href="#">Hide</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Public	published	<a href="#">Hide</a>	<a href="#">Copy</a>	<a href="#">Delete</a>
Partner	hidden	<a href="#">Publish</a>	<a href="#">Copy</a>	<a href="#">Delete</a>

For the Trial plan, you'll only allow read operations, so you'll disable any methods that are not GETs. Additionally, you'll only allow a very limited allowance of 100 calls per month, enough for people to play with and evaluate the API, but not enough for full production use.

**3scale** Dashboard Developers Applications Billing Analytics **APIs** Developer Portal Settings

Overview ActiveDocs

[Create Service](#)

## Public API

### Integration and Settings

- ID for API calls is **2555417726274** and system name is **api**
- Authentication pattern is **API key** mode
- Users **can** manage application keys
- Users **can** manage applications
- Users can **request plan change**
- Users **cannot select a plan** when creating an application
- Users **cannot** see API log requests

### Published Application plans [?](#) [Create Application Plan](#)

Plan Type	Count
Trial	1 application
Public	0 applications

You have 3 application plans (2 published) with a total of 1 live application.

### Stats

Hits

10 hits  
0%

### Latest alerts

There are no alerts.

### Latest Apps

No apps found for this API.

For the Partner plan, you want to encourage your partners to use your API, so you want to be generous with the daily allowance you give them so they can get the data they need as often as they need it. At the same time, you want to enforce some throttling or burst limits at the minute level so your API backend is not overwhelmed by high numbers of requests in a short period of time.

To set up the per user rate limits, you want to set up end user plans in 3scale. As for the Twitter user rate limits, end user plans rate limit each individual user of an application, not the application itself. Since this is an advanced feature in 3scale, you need to enable it first in the overall account settings.

Once it's enabled, you'll see the End User Plans section in the API menu. You'll want to go in here and create a new end user plan. Call this plan "User Limits". It will apply to all Twitter users accessing Twitter through third-party applications that use the API.

Once the End User plan is created, you can add the required limits to all of the methods that you created previously.



Dashboard Developers Applications Billing Analytics API Developer Portal **Settings**

Dashboard Messages Forum

### Latest Signups

Group/Org.	Admin	Apps
3scale support	buyer	1

### Latest Messages

Subject	From
✉ API System: New Application submission	3scale support
✉ API System: New Service subscription	3scale support

### Integrate your API in 5 minutes

This short to-do list is the fastest way to integrate your API with 3scale. Check the [Fast Track Guide](#) for a more in-depth view and to learn more about the [different deployment options](#) to integrate your API with 3scale.

- Add the Base URL of your API and hit save & test
- **Verify the test call was recorded** ?
- Deploy your API Gateway to production on APICast

One final thing you want to do in the API settings is to disable Manual End User Registration so that end users are created in 3scale automatically the first time they are seen.

## Settings

### Default Service Plan

**Default plan** Default ▾  
Default service plan (if any) is contracted automatically on signup.

### Signup & Use

- ☐ Signup requires a description of intended use
- ☒ **Developers can manage Applications**  
Allows developers to manage Applications and Keys on their dashboard.
- ☐ **Require referrer filtering**  
Obliges developers to specify allowed domain/ip referrers.
- ☒ **Enable custom keys**  
Allows you to create custom keys for developers

### End Users Requirements

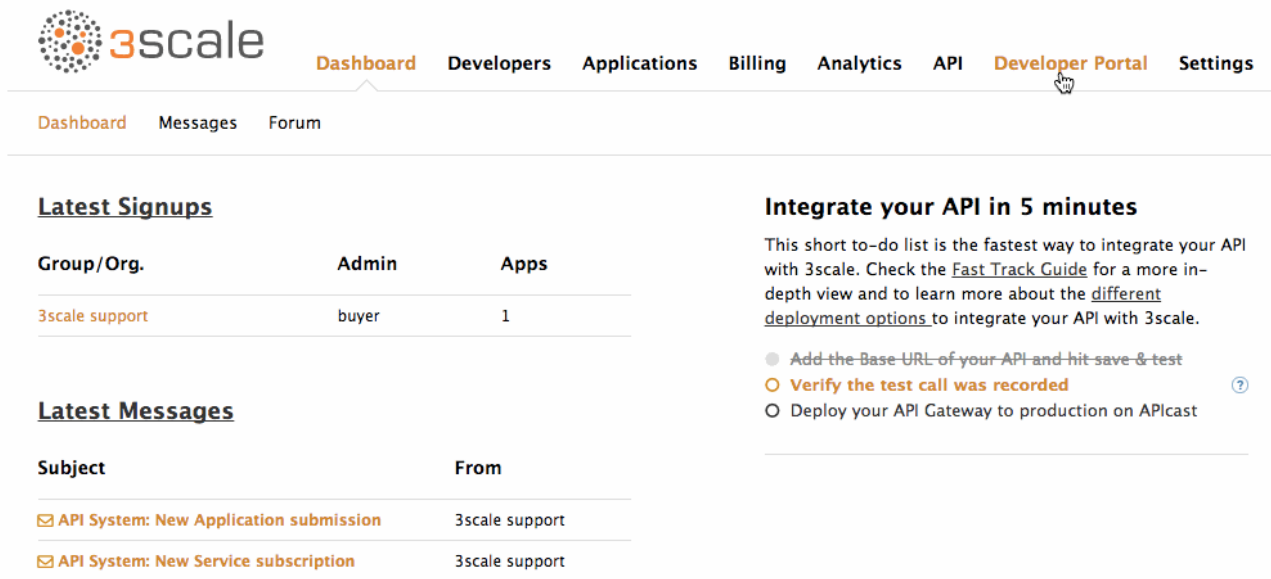
- ☐ **Manual End User registration**  
Requires End Users to be registered beforehand versus being created automatically

That's the initial setup you need to do. You can always come back to your application plans to add any new methods as well as change or refine any limits later on, once your API is running in production and you have a better understanding of your customers' usage patterns.

## 3.5. ENGAGE DEVELOPERS

Now that you've set some limits, the next step is to set up your Developer Portal so potential partners can find out more about the API and sign up for a trial.

It's likely that your partners will be large companies with a number of stakeholders in the integration project, so you'll want to allow them to create or invite multiple users to their developer account. You'll need to enable this in 3scale by going to the **Audience > Developer Portal > Feature Visibility** section and enabling the ability for your partners to invite multiple users to their account.



**3scale**

Dashboard Developers Applications Billing Analytics API **Developer Portal** Settings

Dashboard Messages Forum

### Latest Signups

Group/Org.	Admin	Apps
3scale support	buyer	1

### Latest Messages

Subject	From
✉ API System: New Application submission	3scale support
✉ API System: New Service subscription	3scale support

### Integrate your API in 5 minutes

This short to-do list is the fastest way to integrate your API with 3scale. Check the [Fast Track Guide](#) for a more in-depth view and to learn more about the [different deployment options](#) to integrate your API with 3scale.

- Add the Base URL of your API and hit save & test
- **Verify the test call was recorded** ?
- Deploy your API Gateway to production on APIcast

In this view, you can see other features that you'll probably want to enable such as multiple services so you can set up multiple, separate APIs as well as multiple applications so your partners can register more than one application using your API.

That's pretty much everything you need to do to integrate your API with 3scale.

Now that you have your first API set up in 3scale, you can easily create new services for your other APIs from the API overview page and set them up as you did for your Public REST API. You want to create a new service for each of your different APIs. You could also create a new separate service for each version of your API, or even separate services for each of the different environments your API might be available on, such as if you want to expose your API's QA environment for partner testing.

To create a new service, simply navigate to the API view and select Create Service. This will prompt you to choose a name for your API and enter a description. You can also start specifying some settings for the API such as the authentication mode. In this case, you'll choose OAuth mode from the start.

At this point, you can also rename your original API to be more descriptive by going to **[your\_API\_name] > Overview** and click 'edit'. There, you can change the name and description of your API.

**3scale** Dashboard Developers Applications Billing Analytics **APIs** Developer Portal Settings

Overview ActiveDocs

[Create Service](#)

## API

### Integration and Settings

- ID for API calls is **2555417726274** and system name is **api**
- Authentication pattern is **API key mode**
- Users **can** manage application keys
- Users **can** manage applications
- Users **can request plan change**
- Users **cannot select a plan** when creating an application
- Users **cannot** see API log requests

### Published Application plans

[Create Application Plan](#)

- Trial** - 1 application
- Public** - 0 applications

You have **3 application plans** (2 published) with a total of **1 live application**.

### Stats

Hits  
4 hits  
0% ↓

### Latest alerts

There are no alerts.

### Latest Apps

3scale support's App from 3scale support

The final things that you want to set up on your 3scale account are all around customizing the Developer Portal to fit your particular needs and so that it conforms to the look and feel of your product branding.

You can do this through:

1. [Setting up a custom domain for you Developer Portal](#) such as `developer.twitter.com`.
2. [Setting up an API contact email](#) such as `api@twitter.com` for emails from your 3scale account to your API consumers.
3. [Customizing the Developer Portal](#) so that it matches the look and feel of your brand.
4. [Editing the email templates](#) to tailor the messages you want to send to your partners.
5. [Using liquids](#) to control the information that gets shown to different customers.
6. Adding a favicon.

For the favicon, the path has to be different than the standard `/favicon.ico` as this is already taken by the default 3scale one.

You need to upload your favicon into the 3scale CMS by creating a new file. In this case, place it in the images folder and call it `icon.ico`. You'll then refer to it from the main layout with the following snippet placed in the head element, as below:

```
<!-- Favicon -->
<link rel="shortcut icon" href="/images/icon.ico" type="image/x-icon" />
```





Dashboard Developers Applications Billing Analytics APIs **Developer Portal** Settings

Dashboard Messages Forum

### Latest Signups

Group/Org.	Admin	Apps
3scale support	buyer	1

### Latest Messages

Subject	From
✉ API System: New Application submission	3scale support
✉ API System: New Service subscription	3scale support

### Integrate your API in 5 minutes

This short to-do list is the fastest way to integrate your API with 3scale. Check the [Fast Track Guide](#) for a more in-depth view and to learn more about the [different deployment options](#) to integrate your API with 3scale.

- Add the Base URL of your API and hit save & test
- **Verify the test call was recorded** ?
- Deploy your API Gateway to production on APIcast

Support Privacy Refunds

Powered by 3scale

You also want to set up some interactive documentation (ActiveDocs) so your customers can explore the API without needing to write any code. If you already have a Swagger spec for your API, navigate to the API service you want to work on and go to the ActiveDocs section, navigate to the API service you want to work on and go to the ActiveDocs section



Dashboard Developers Applications Billing Analytics **APIs** Developer Portal Settings

Overview **ActiveDocs**

[+ Create Service](#)

## Public API

### Integration and Settings

ID for API calls is 2555417726274 and system name is api  
 Authentication pattern is API key mode  
 Users can manage application keys  
 Users can manage applications  
 Users can request plan change  
 Users cannot select a plan when creating an application  
 Users cannot see API log requests

### Published Application plans ?

[+ Create Application Plan](#)

Trial - 1 application  
 Public - 0 applications

You have [3 application plans](#) (2 published) with a total of [1 live application](#).

### Stats



### Latest alerts

There are no alerts.

### Latest Apps

3scale support's App from 3scale support

## 3.6. EXPOSING THE API

Now that you have your API integrated and have customized your Developer Portal to fit your needs, you can now start thinking about getting some users for your API and what the signup process is going to look like for them.

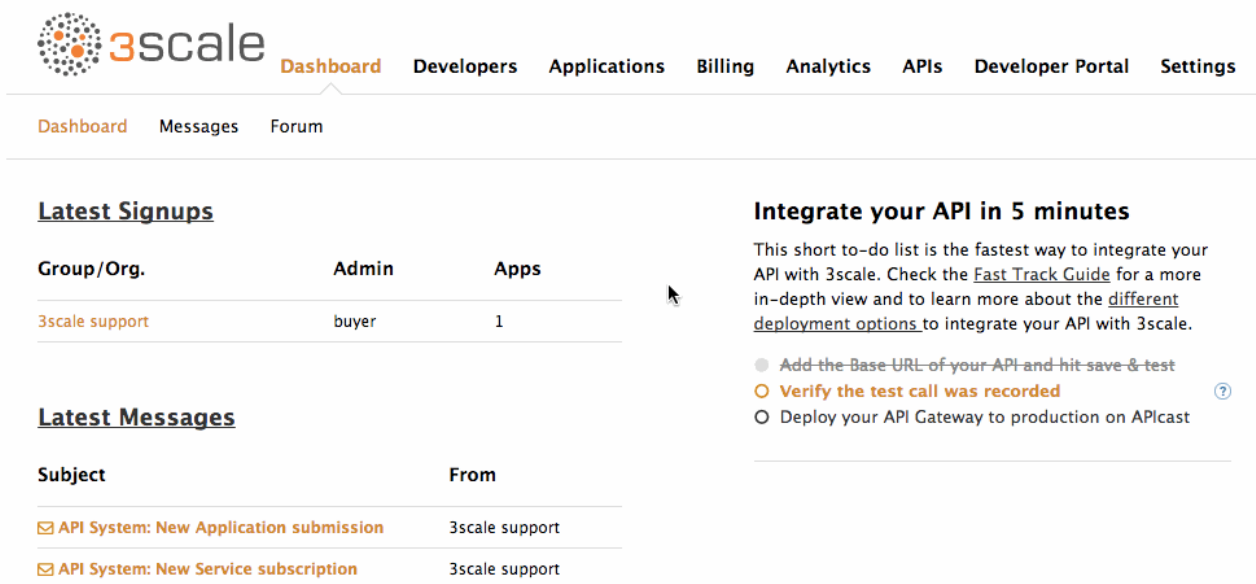
You've already done a lot of the groundwork by enabling signups, so anyone can create a developer account on your API. By setting the Trial plan as the default application plan, they'll have access to the API on a trial basis from the beginning. They can also see and request access to the Public Application plan if they want to take their API usage further.

One last thing to do now is to collect some extra data on your API users once they sign up. For this, you can set up some extra fields to be collected by 3scale on signup as well as application creation. Extra fields can be created for:

- Accounts
- Users of an account
- Applications

There are two types of extra fields, 3scale's built-in ones and custom ones that you can define. You can make most fields optional, although there are a few that are required by 3scale such as username for a user.

First of all, you want to collect the actual names of your users, so you'll add the "First Name" and "Last Name" extra fields to the user object. To make these mandatory for developers to fill in, make them required. You also want these to appear as the first and second fields to be filled out in the signup form, so drag them up to the relevant positions.



The screenshot shows the 3scale Developer Portal interface. At the top is the 3scale logo and a navigation bar with links: Dashboard, Developers, Applications, Billing, Analytics, APIs, Developer Portal, and Settings. Below this is a sub-navigation bar with Dashboard, Messages, and Forum. The main content area is divided into three sections:

- Latest Signups:** A table with columns Group/Org., Admin, and Apps. It shows one entry for '3scale support' with Admin 'buyer' and Apps '1'.
- Latest Messages:** A table with columns Subject and From. It shows two messages from '3scale support': 'API System: New Application submission' and 'API System: New Service subscription'.
- Integrate your API in 5 minutes:** A section with a heading and a paragraph explaining the fastest way to integrate. It includes a list of steps: 'Add the Base URL of your API and hit save & test', 'Verify the test call was recorded' (highlighted in orange), and 'Deploy your API Gateway to production on APIcast'.

For the application, you want to collect some information about the sort of application it is: mobile, web, or desktop. Create a custom field under "Application" with these options so that users can choose between them. This field will be optional.

## Listing fields definitions

Here you can manage all the information you gather from your partners. You can add new fields, and change the existing ones, e.g. make them Hidden, Read only, or Required. You can change the text your partners will 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.

### User

[+ Create](#)

↕ first_name	"First Name"	Required	<a href="#">Edit</a>	<a href="#">Delete</a>
↕ last_name	"Last Name"	Required	<a href="#">Edit</a>	<a href="#">Delete</a>
↕ username	"Username"	Required	<a href="#">Edit</a>	
↕ email	"Email"	Required	<a href="#">Edit</a>	

### Account

[+ Create](#)

↕ org_name	"Organization/Group Name"	Required	<a href="#">Edit</a>	
------------	---------------------------	----------	----------------------	--

### Application

[+ Create](#)

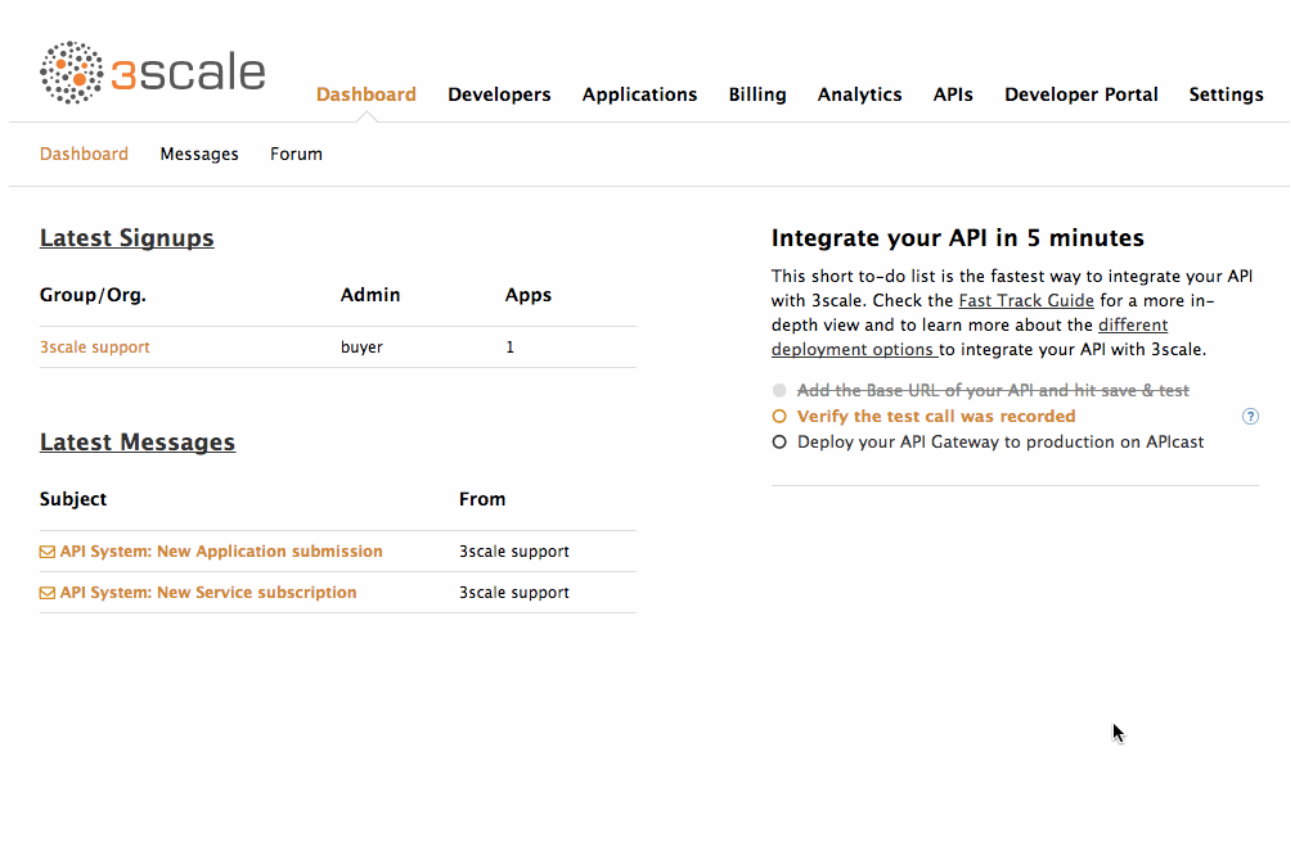
↕ name	"Name"	Required	<a href="#">Edit</a>	
↕ description	"Description"	Required	<a href="#">Edit</a>	

[Support](#) [Privacy](#) [Refunds](#)

 Powered by  3scale

Now that you have some more data, you'll also want to import a lot of it into other services that you use, such as Salesforce, to enable tracking for any interesting leads for upsell to your Partner Application plan and other services that you provide outside of your API. For this, you'll use webhooks to send a message with the account details whenever a new account is created on your 3scale Developer Portal.

For this, you just need to provide an endpoint that's listening for these messages and knows how to process them. Then it's just a case of entering in the URL for this endpoint, turning webhooks on, and selecting the events you want to trigger messages – in this case, account creation. Once that's done, just click save. Now, every time a new account is created in 3scale, you'll receive a copy of all the details ready for processing.



The screenshot shows the 3scale dashboard. At the top is the 3scale logo and a navigation bar with links: Dashboard (highlighted), Developers, Applications, Billing, Analytics, APIs, Developer Portal, and Settings. Below the navigation bar are sub-links: Dashboard, Messages, and Forum. The main content area is divided into two columns. The left column contains two sections: 'Latest Signups' and 'Latest Messages'. 'Latest Signups' is a table with columns 'Group/Org.', 'Admin', and 'Apps'. It shows one entry for '3scale support' with 'buyer' as the admin and '1' app. 'Latest Messages' is a table with columns 'Subject' and 'From'. It shows two messages from '3scale support' with subjects 'API System: New Application submission' and 'API System: New Service subscription'. The right column contains a section titled 'Integrate your API in 5 minutes'. It includes a paragraph of text and a list of three steps: 'Add the Base URL of your API and hit save & test', 'Verify the test call was recorded' (highlighted in orange), and 'Deploy your API Gateway to production on APIcast'. A question mark icon is next to the second step.

**Latest Signups**

Group/Org.	Admin	Apps
3scale support	buyer	1

**Latest Messages**

Subject	From
✉ API System: New Application submission	3scale support
✉ API System: New Service subscription	3scale support

**Integrate your API in 5 minutes**

This short to-do list is the fastest way to integrate your API with 3scale. Check the [Fast Track Guide](#) for a more in-depth view and to learn more about the [different deployment options](#) to integrate your API with 3scale.

- Add the Base URL of your API and hit save & test
- **Verify the test call was recorded** ?
- Deploy your API Gateway to production on APIcast

You can read more about webhooks [here](#).

That's all regarding the integration and setup of your 3scale account. However that's by no means all that 3scale provides. Once your API has been up and running for a while, the analytics module can help give some insight into the usage patterns for your API, which of your partners are most engaged, as well as which are the busiest hours and days for your API. You can read all about how to make the most of the 3scale Analytics [here](#) and [here](#).

What are you waiting for? If you haven't already done so, start your free trial of 3scale and give it a go.