Red Hat 3scale API Management 2.1

API Authentication

Choose your authentication pattern: API keys, app identifier and keys, or OAuth 2.0.

Last Updated: 2019-05-08

Red Hat 3scale API Management 2.1 API Authentication

Choose your authentication pattern: API keys, app identifier and keys, or OAuth 2.0.

Legal Notice

Abstract

This guide documents methods for api authentication with Red Hat 3scale API Management 2.1.

Table of Contents

CHAPTER 1. AUTHENTICATION PATTERNS

By the time you complete this tutorial you'll know how to set the authentication pattern on your API and the effect this has on applications communicating with your API.

Depending on your API you may need to use different authentication patterns to issue credentials for access to your API. These can range from API keys to openAuth tokens and custom configurations. This tutorial covers how to select between the standard Authentication Patterns Available.

1.1. AUTHENTICATION PATTERNS

3scale supports the following authentication patterns out of the box:

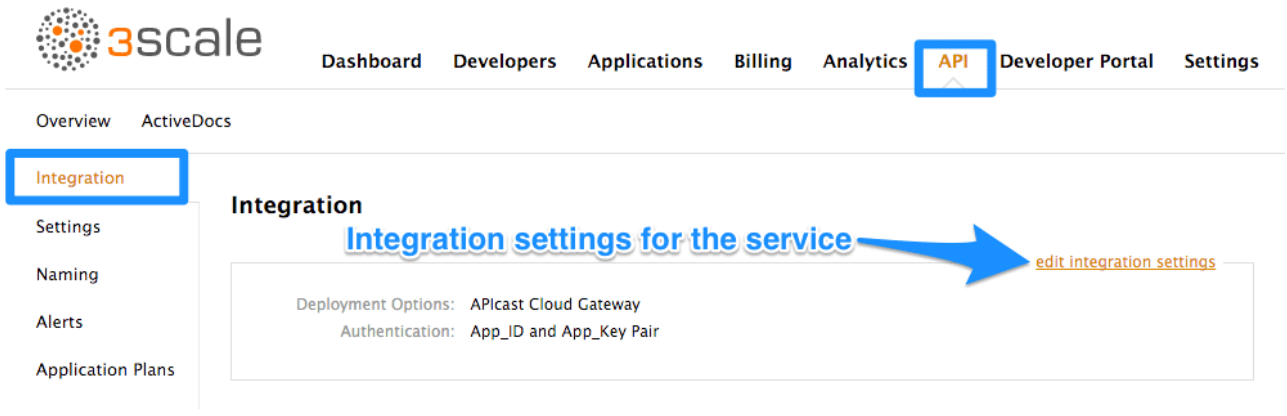
- **Standard API Keys**: single randomized strings or hashes acting as an identifier and a secret token.
- **Application Identifier and Key pairs**: immutable identifier and mutable secret key strings.
- **OAuth 2.0**: OAuth 2.0 client id, client secret and access token combinations.

1.2. EXTENSIONS

3scale also offers commercially reasonable support for coupling issues credentials with IP address filtering or referrer domain filtering – see the extra section at the end of this tutorial.

1.3. STEP 1: SELECT THE AUTHENTICATION MODE FOR YOUR SERVICE.

Once in your Administration Dashboard, navigate to the API tab and select the service you wish to work on (there may be only one service named API in which case select this). Click on the Integration Link.

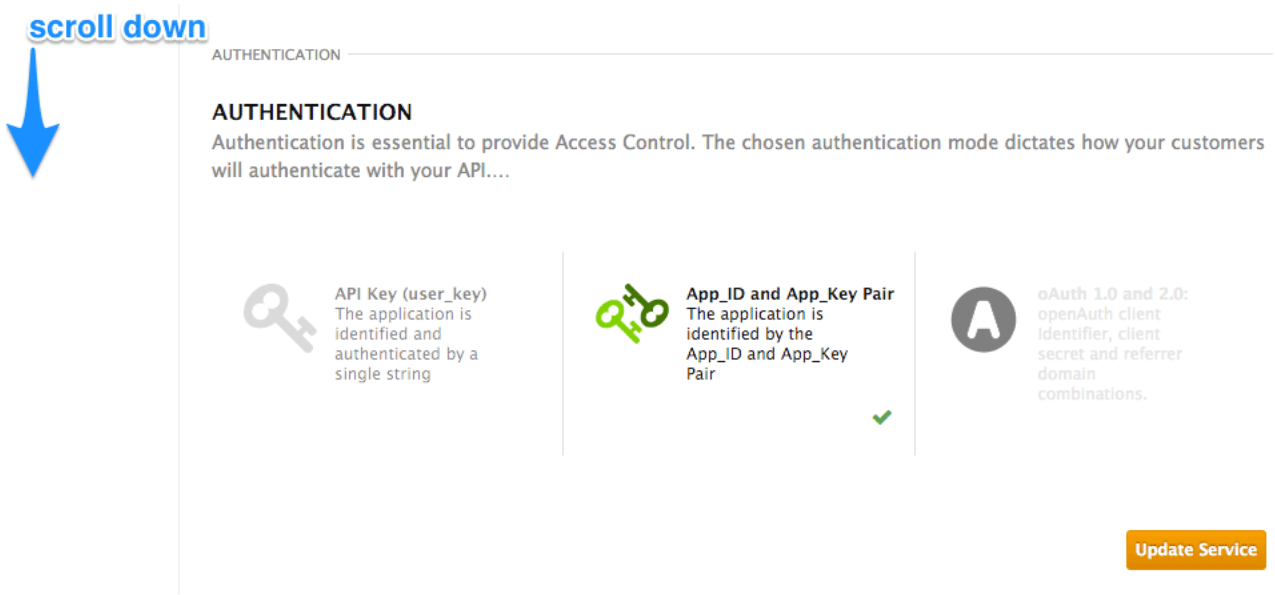


Note that each service you operate could use a different authentication pattern, but only one pattern can be used per service.

It is **not recommended** you change the authentication pattern once credentials have already been registered since behavior may be unpredictable. To change authentication patterns we recommend creating a new service and migrating customers.

1.4. STEP 2: SELECT THE AUTH MODE YOU WANT TO USE

Scroll down to the Authentication section an choose the required Authentication mode.



1.5. STEP 3: ENSURE YOUR API ACCEPTS THE RIGHT TYPES OF CREDENTIALS

Depending on the credential type chosen, you will likely need to accept different parameters in your API calls (key fields, IDs etc.). The names of these parameters need not be the same as those used internally at 3scale, your 3scale authentication will function correctly just as long as the correct parameter names are used in calls to the 3scale backend.

1.6. STEP 4: CREATE AN APPLICATION TO TEST CREDENTIALS

In order to test the credential sets are working you can create a new application to issue credentials to use the API. Navigate to the Accounts area of your dashboard, click on the account you wish to use and click the **new application** button.

Filling out the form and clicking save will create a new application with credentials to use the API. You can now use these credentials to make calls to your API and records will be checked against 3scale list of registered applications.

1.7. STANDARD AUTHENTICATION PATTERNS

3scale supports the following authentication patterns out of the box.

1.7.1. API Key

The simplest form of credentials supported is the single API model. Here each application with permissions on the API has a single (unique) long character string something like this:

```
API-key = 853a76f7c8d5f4a1ee8bf10a4e0d1f13
```

By default the name of the key parameter is **user_key**. You can use this same label or choose another such as **API-key** in which case you simply need to map the value over before you make the authorization calls to 3scale. The string acts as both an identifier and a secret token for use of the API. It is recommended such patterns are only used either in environments with low security requirements or with SSL security on API calls. The operations which can be carried out on the token and application are:

- Application Suspend: this suspends the applications access to the API and in effect all calls to the API with the relevant key will be suspended
- Application Resume: undoes the effect of an application suspend action
- Key Regenerate: this action generates a new random string key for the application and associates it with the application. Immediately after this action is taken calls with the previous token will cease to be accepted.

The latter action can be triggered from the API Administration dashboard and (if permitted) from the API Developers User console.

1.7.2. App_ID and App_Key Pair

Whereas the API Key Pattern combines the identity of the application and the secret usage token in one token, this pattern separates the two. Each application using the API issues an immutable initial identifier known as the **Application ID** (App ID). The App ID is constant and may or may not be secret. In addition each application may have 1-n **Application Keys** (App_Keys). Each Key is associated directly with the App_ID and should be treated as secret.

```
app_id = 80a4e03 app_key = a1ee8bf10a4e0d1f13853a76f7c8d5f4
```

In the default setting, developers are able to create up to 5 keys per application. This allow a developer to create a new key, add it to their code redeploy their application and then disable old keys – causing no application downtime as an API Key Regeneration would.

Note that statistics and rate limits are always kept at the application ID level of granularity and not per API Key. If a developer wishes to track two sets of statistics, they should create two applications rather than two keys.

It is also possible to change mode in the system and allow applications to be created without applications keys present. In this case the 3scale system will authenticate access based on the App ID only (and no key checks are made). This mode is useful for (for example) widget type scenarios or where rate limits are applied to users rather than applications. Note that in most cases you will want your API to enforce there to be at least one application key per application present (this setting can be found on the **Settings** menu item under usage rules).

1.7.3. openAuth Support

openAuth is a set of specifications which enable a variety of different authentication patterns for APIs. The two major versions released so far (v1 and v2) are significantly different and version 2 includes a number of different supported authentication flows.

1.7.4. More Information

If you have authentication needs which are not covered by this how to, let us know by contacting support.

In addition, 3scale supports coupling credentials with authorized IP ranges per key or referrer domains per key to restrict where specific API credentials can be used. You can enable this additional filtering by navigating to the API Tab and accessing the Settings screen on the API.

CHAPTER 2. 3SCALE INTEGRATION WITH RED HAT SINGLE SIGN-ON USING OPENID CONNECT

3scale can synchronize client credentials between 3scale (Application credentials) and a Red Hat Single Sign-On server using OpenID Connect (OIDC). 3scale utilizes a service called Zync to synchronize calls to the Red Hat Single Sign-On server.

2.1. PREREQUISITES:

- 3scale 2.1
- A Red Hat Single Sign-On version as specified on the Supported Configurations page
- SSL connection between Zync and RHSSO.

In order to connect 3scale and Red Hat Single Sign-On, perform the following configurations in Zync, your Red Hat Single Sign-On server and 3scale:

2.2. CONFIGURE ZYNC

In 3scale API Management Platform 2.1, self-signed and custom certificates are not supported and will be rejected by Zync. To establish an SSL connection between Zync and RHSSO, configure Zync to support custom certificates with the following workaround:

1. Validate the new certificate with the following cURL command. The expected response is a JSON configuration of the realm. If validation fails it is an indicator that your certificate may not be correct.

```
curl -v https://<secure-sso-host>/auth/realms/master --cacert customCA.pem
```

2. Add the certificate bundle to the Zync pod:

   a. Gather the existing content of the **/etc/pki/tls/cert.pem** file on the Zync pod. Run:

```
oc exec <zync-pod-id> cat /etc/pki/tls/cert.pem | tee -a cacert.pem cert.pem
```

   b. Append the contents of the custom CA certificate file to both **cacert.pem** and **cert.pem**:

```
cat customCA.pem | tee -a cacert.pem cert.pem
```

3. Log in to the 3scale AMP project in OpenShift and run the following commands. This updates the CA certificate used by the libraries that Zync depends on.

```
oc create configmap cacertpem --from-file=./cacert.pem
```

```
oc create configmap certpem --from-file=./cert.pem
```

```
oc set volume dc/zync --add --name=cacertpem --mount-path
/opt/zync/vendor/bundle/ruby/2.3.0/gems/httpclient-2.8.3/lib/httpclient/cacert.pem --
source='{"configMap":{"name":"cacertpem","items":
[{"key":"cacert.pem","path":"cacert.pem"}]}}'
```

```
oc set volume dc/zync --add --name=certpem --mount-path /etc/pki/tls/cert.pem --
source='{"configMap":{"name":"certpem","items":[{"key":"cert.pem","path":"cert.pem"}]}}'
```

```
oc patch dc/zync --type=json -p '[{"op": "add", "path":
"/spec/template/spec/containers/0/volumeMounts/0/subPath", "value":"cacert.pem"},{"op":
"add", "path": "/spec/template/spec/containers/0/volumeMounts/1/subPath",
"value":"cert.pem"}]'
```

2.3. CONFIGURE RED HAT SINGLE SIGN-ON

1. Create a realm
2. Create a client:
   a. Specify a client ID
   b. Select the **openid-connect** client protocol
3. Configure client permissions, setting the following:
   a. **Access Type** to confidential
   b. **Standard Flow Enabled** to OFF
   c. **Direct Access Grants Enabled** to OFF
   d. **Service Accounts Enabled** to ON
4. Set service account roles for your client
   a. Navigate to the **service account roles** tab of your client
   b. In the **client roles** dropdown, select **realm management**
   c. In the **available roles** pane, select the **manage-clients** list item and assign the role by clicking **Add Roles**
5. Note client credentials
   a. Take note of the client ID
   b. Navigate to the **Credentials** tab of your client and take note of the secret

2.4. CONFIGURE 3SCALE

Once you have created and configured a server and client in Red Hat Single Sign-On, you must configure 3scale to work with Red Hat Single Sign-On:

1. Enable OIDC
   a. Select the service on which you want to enable Red Hat Single Sign-On, navigate to the **APIs → <your_service_name> → integration** page
   b. On that page, select **edit integration settings**
   c. Under the **Authentication** deployment options, select **OpenID Connect**

d. Update the service

2. Edit Your APIcast Configuration

a. navigate to the **APIs → <your_service_name> → integration**page

b. On that page, select **edit APIcast configuration**

c. Under the **Authentication Settings** expandable heading, in the OpenID Connect Issuer field, enter your previously noted client credentials with the URL of your Red Hat Single Sign-On server:

> https://<CLIENT_ID>:<CLIENT_SECRET>@<HOST>:
> <PORT>/auth/realms/<REALM_NAME>

d. Save your configuration

2.5. TEST INTEGRATION

Peform the following procedures in 3scale and verify the results in Red Hat Single Sign-On to test your integration:

3scale Procedure	Red Hat Single Sign-On Result
Create an application for the service where you configured the OpenID Connect Issuer. Note the Client ID and client secret generated for the application	The new client appears in the Red Hat Single Sign-On realm you are using
Change the Redirect URL on the application in 3Scale	The Redirect URL was updated
Delete the application	The client is deleted from the realm