Make API Requests with OAuth

Last updated March 31, 2020

Theming Module Development 8.9.x/9.0.x

When you create a fully decoupled application, the code in your application can't rely on things like the `fetch()` function's `same-origin` policy and the browser's use of cookies to authenticate requests. Instead, you need to use alternative methods like OAuth or JSON Web Tokens (JWTs).

We'll focus on setting up and using Drupal as an OAuth provider, and allowing a decoupled application to authenticate users via OAuth. This same technique applies just as well if you want to use JWTs.

In this tutorial we'll:

- Install the Simple OAuth Drupal module, and configure it to work with a password grant flow to allow our code to exchange a username and password for an access token
- Demonstrate how to retrieve and use an OAuth access token to make authenticated requests

By the end of this tutorial you should know how to install and configure the Simple OAuth module and make authenticated API requests using an OAuth password grant flow.

Goal

Install and configure the Simple OAuth module and then validate that it works by making some example API requests.

Prerequisites

- Drupal 8.5.x (minimum) installed. If you have Drupal 8.7 or higher installed, JSON:API is already included.
- Install JSON:API Module
- Postman or other application for testing HTTP requests

1 Install and configure Simple OAuth module

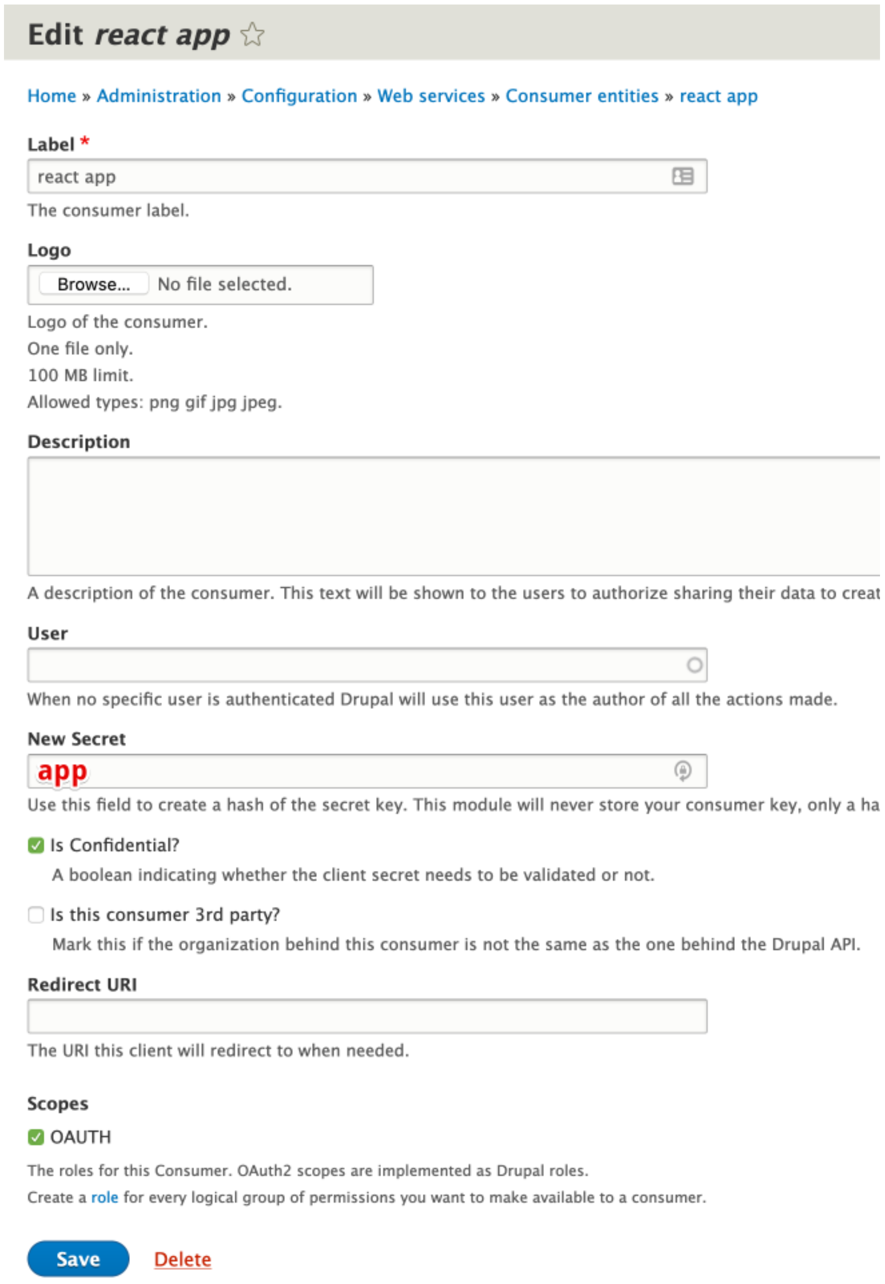
Start by downloading and installing the Simple OAuth module.

Use the following steps to configure Simple OAuth to allow our React application to use a password grant to retrieve access and refresh tokens from Drupal. (For detailed instructions see Install and Configure Simple OAuth.)

To follow along with building this app you'll need this configuration at a minimum:

- Create a Drupal role to use with OAuth; in this example we will use one named *OAUTH*.
- Give the *OAUTH* role permissions to add, edit, and delete content.
- Add a consumer at */admin/config/services/consumer/add* named "react app"
- Set an app secret.
    - New Secret: `app`. Make note of the app secret that you used if it's different.
- Check the box to enable *OAUTH* role for the scope of this client.

- Save the new configuration.
- Make note of the UUID for the client at */admin/config/services/consumer*, e.g. `0485fbc2-6ce3-4e34-9ffe-df7833c0476c`.



Edit *react app* ☆

Home » Administration » Configuration » Web services » Consumer entities » react app

Label *
react app
The consumer label.

Logo
Browse... No file selected.
Logo of the consumer.
One file only.
100 MB limit.
Allowed types: png gif jpg jpeg.

Description

A description of the consumer. This text will be shown to the users to authorize sharing their data to creat

User

When no specific user is authenticated Drupal will use this user as the author of all the actions made.

New Secret
app
Use this field to create a hash of the secret key. This module will never store your consumer key, only a ha

✓ Is Confidential?
A boolean indicating whether the client secret needs to be validated or not.

☐ Is this consumer 3rd party?
Mark this if the organization behind this consumer is not the same as the one behind the Drupal API.

Redirect URI

The URI this client will redirect to when needed.

Scopes
✓ OAUTH
The roles for this Consumer. OAuth2 scopes are implemented as Drupal roles.
Create a role for every logical group of permissions you want to make available to a consumer.

Save Delete

2 Give the *OAUTH* role the required permissions

We want users logged in with the *OAUTH* role to be able to add, edit, and delete *Article* content. Make sure the role has the following permissions:

- *Article:* Create new content
- *Article:* Delete any content
- *Article:* Edit any content

The following are not required, but nice to have:

- View content overview page
- View own unpublished nodes

> Otherwise Drupal will return a 403 forbidden error when our application attempts to add, update, or delete content.
>
> Not sure if you've got it configured right? Log into Drupal as the user you want to use via your app. If you can create an article as that user, you can create an article when authenticated as that user via OAuth.

3 Create a user account with the *OAUTH* role

OAuth allows us to exchange a username and password for an access token -- like how a browser can exchange a username and password for a session ID. The access token will grant you the permissions of the user for whom it was created. To do this, add a new Drupal user at *admin/people/create* and give them the *OAUTH* role.

Test requests in Postman

The following are examples of retrieving and using an OAuth access token from Drupal using a password grant, and then using it to make authenticated requests. The basic process is:

- Exchange a username and password for access and refresh tokens
- Use the access token in the Authorization header of a request
- Use the refresh token to get a new access token when the current one expires

These are examples of the different requests we'll need to make via our React application.

For more detailed information about how this works, read Get a Token for OAuth 2 Requests and Make an Authenticated Request Using OAuth 2.

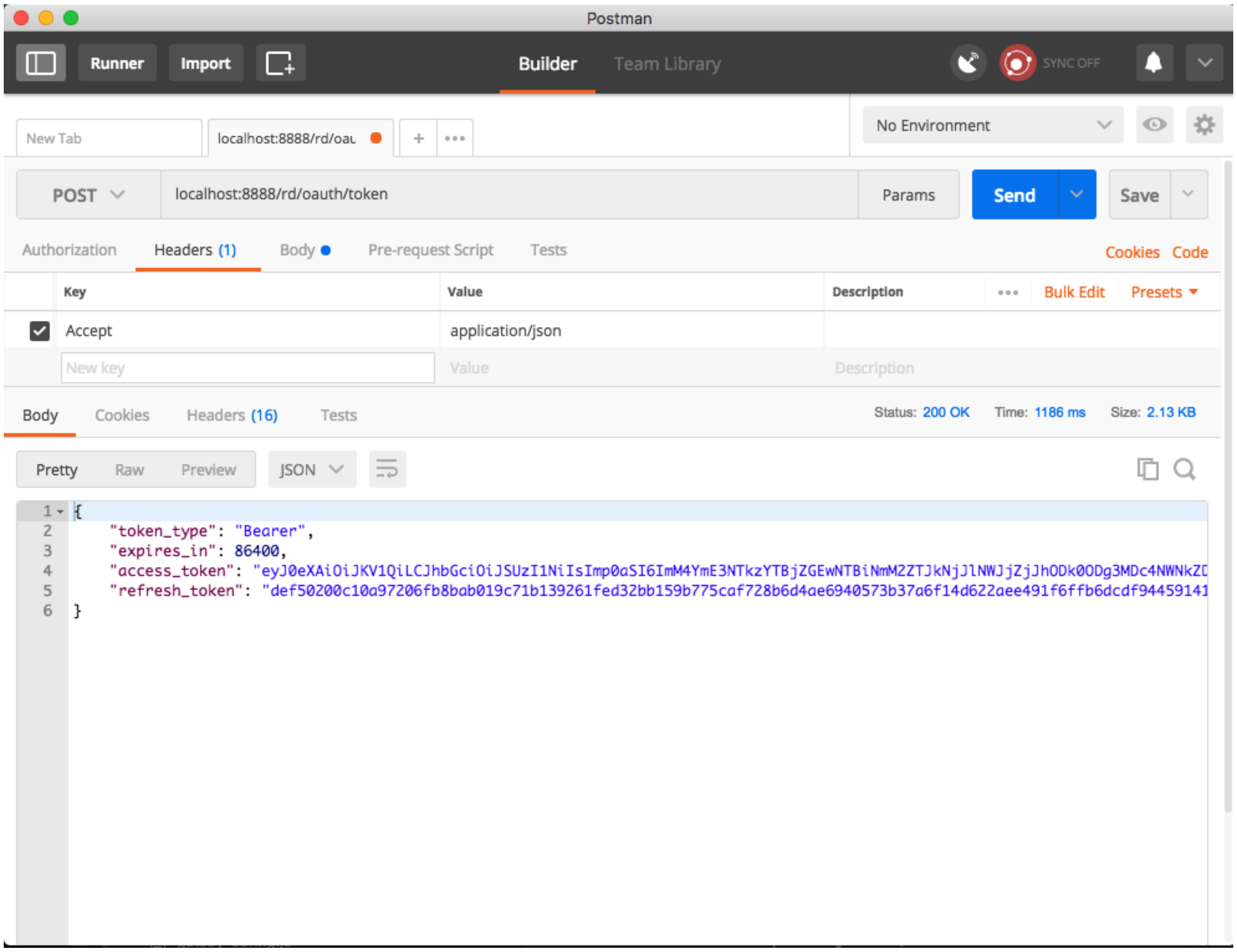
In this example we'll use Postman to make the test requests. You can use Postman, cURL, or any other application that allows you to generate HTTP requests.

Get an OAuth token

Make a POST request to `http://localhost:8888/oauth/token`

Headers:

```
Accept: application/vnd.api+json
Content-Type: application/x-www-form-urlencoded
```



POST localhost:8888/rd/oauth/token

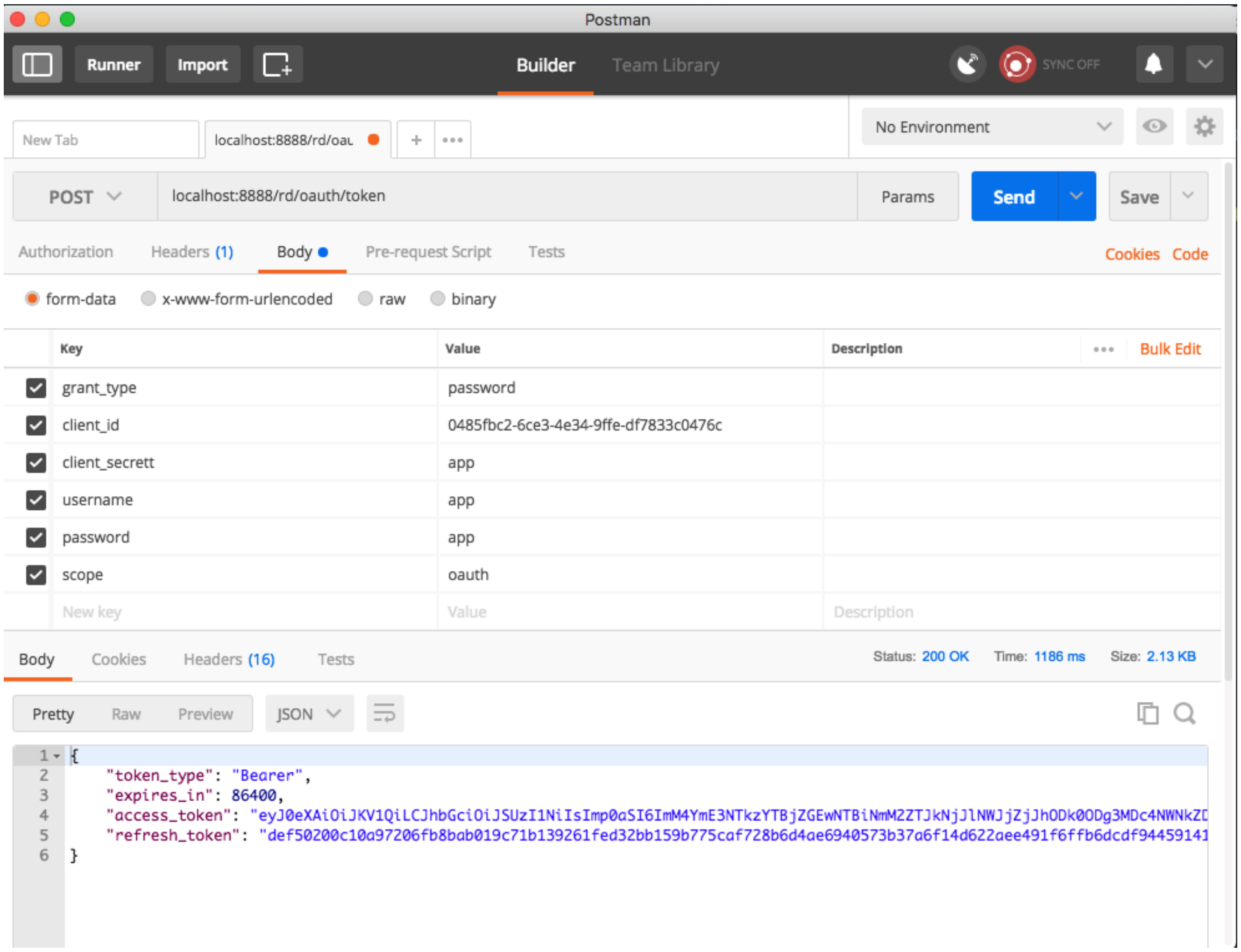
	Key	Value	Description
✓	Accept	application/json	

Status: 200 OK Time: 1186 ms Size: 2.13 KB

```
{
    "token_type": "Bearer",
    "expires_in": 86400,
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImM4YmE3NTkzYTBjZGEwNTBiNmM2ZTJkNjJlNWJjZjJhODk0ODg3MDc4NWNkZD
    "refresh_token": "def50200c10a97206fb8bab019c71b139261fed32bb159b775caf728b6d4ae6940573b37a6f14d622aee491f6ffb6dcdf94459141
}
```

Body:

```
grant_type = password
client_id = 0485fbc2-6ce3-4e34-9ffe-df7833c0476c
client_secret = app
scope = oauth
username = {username of Drupal user with OAUTH role}
password = {password}
```



POST localhost:8888/rd/oauth/token

Body: [✓ form-data] x-www-form-urlencoded raw binary

	Key	Value	Description
✓	grant_type	password	
✓	client_id	0485fbc2-6ce3-4e34-9ffe-df7833c0476c	
✓	client_secrett	app	
✓	username	app	
✓	password	app	
✓	scope	oauth	
	New key	Value	Description

Status: 200 OK Time: 1186 ms Size: 2.13 KB

```
{
    "token_type": "Bearer",
    "expires_in": 86400,
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImM4YmE3NTkzYTBjZGEwNTBiNmM2ZTJkNjJlNWJjZjJhODk0ODg3MDc4NWNkZ
    "refresh_token": "def50200c10a97206fb8bab019c71b139261fed32bb159b775caf728b6d4ae6940573b37a6f14d622aee491f6ffb6dcdf94459141
}
```

You should get a response that looks like:

```
{
    "token_type": "Bearer",
    "expires_in": 86400,
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImMwODNlYzVkMTJjNDk4M2JlMzE0MmZhMDc4NGU3NWEwNTJmMzlkNmM2YjkwNjY1YTFiZGI4O
    "refresh_token": "def5020036590d2c3c1bb057ec2683e4f0600e5e1fa467fd8f26e4a7d55fe33d929501cfc806dd7f4fbe8c39798088ce62fb2832ee29110640dd
}
```

Copy the value of the access token, and **note** that it is `token_type: Bearer`.

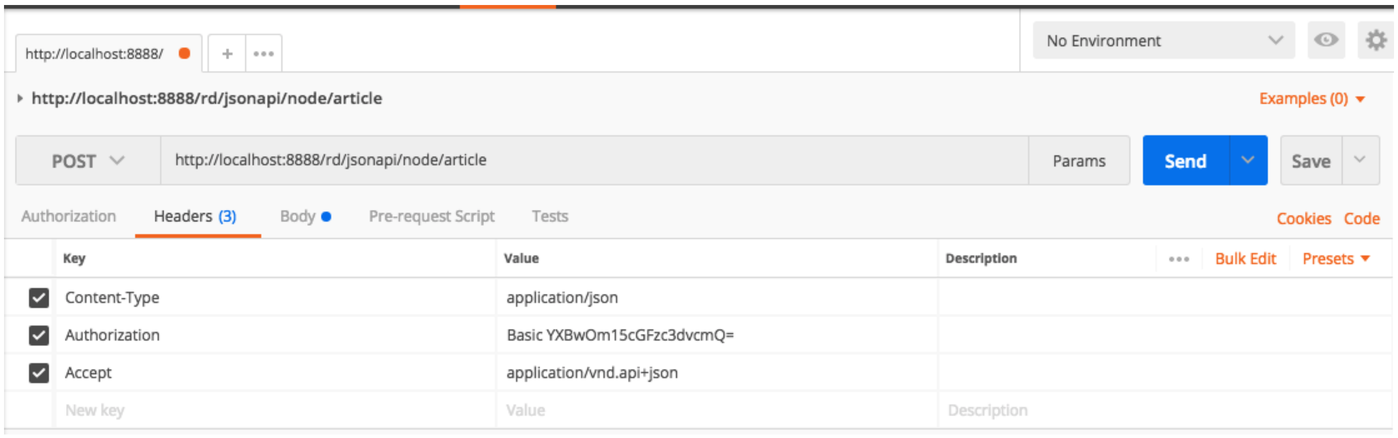
Use POST to create a new node

To add authentication to a request, you need to edit the header and add the token from above. Note, the token is truncated below to make it easier to read. Use the full token in your tests.

Make a POST request to `http://localhost:8888/jsonapi/node/article`

Headers:

```
Accept: application/vnd.api+json
Content-Type: application/vnd.api+json
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...
```



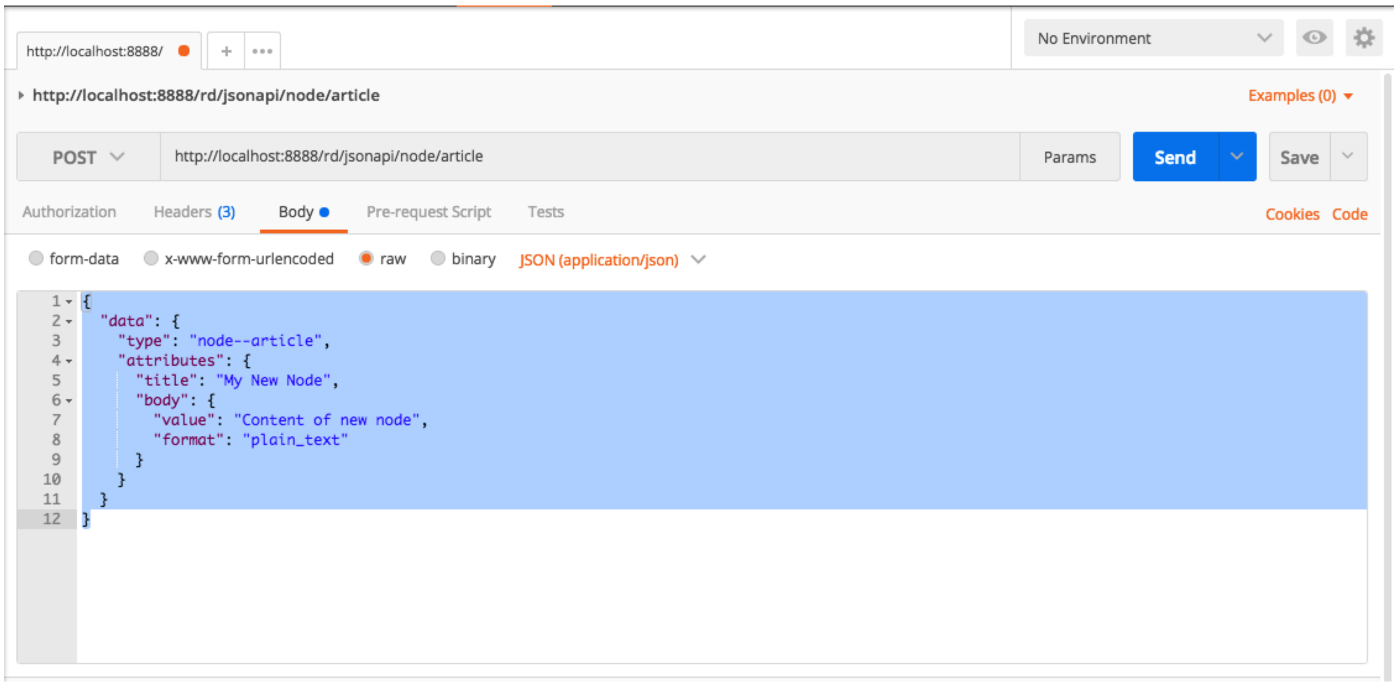
http://localhost:8888/rd/jsonapi/node/article

POST http://localhost:8888/rd/jsonapi/node/article

	Key	Value	Description
✓	Content-Type	application/json	
✓	Authorization	Basic YXBwOm15cGFzc3dvcmQ=	
✓	Accept	application/vnd.api+json	
	New key	Value	Description

Body:

```
{
  "data": {
    "type": "node--article",
    "attributes": {
      "title": "My New Node",
      "body": {
        "value": "Content of new node",
        "format": "plain_text"
      }
    }
  }
}
```

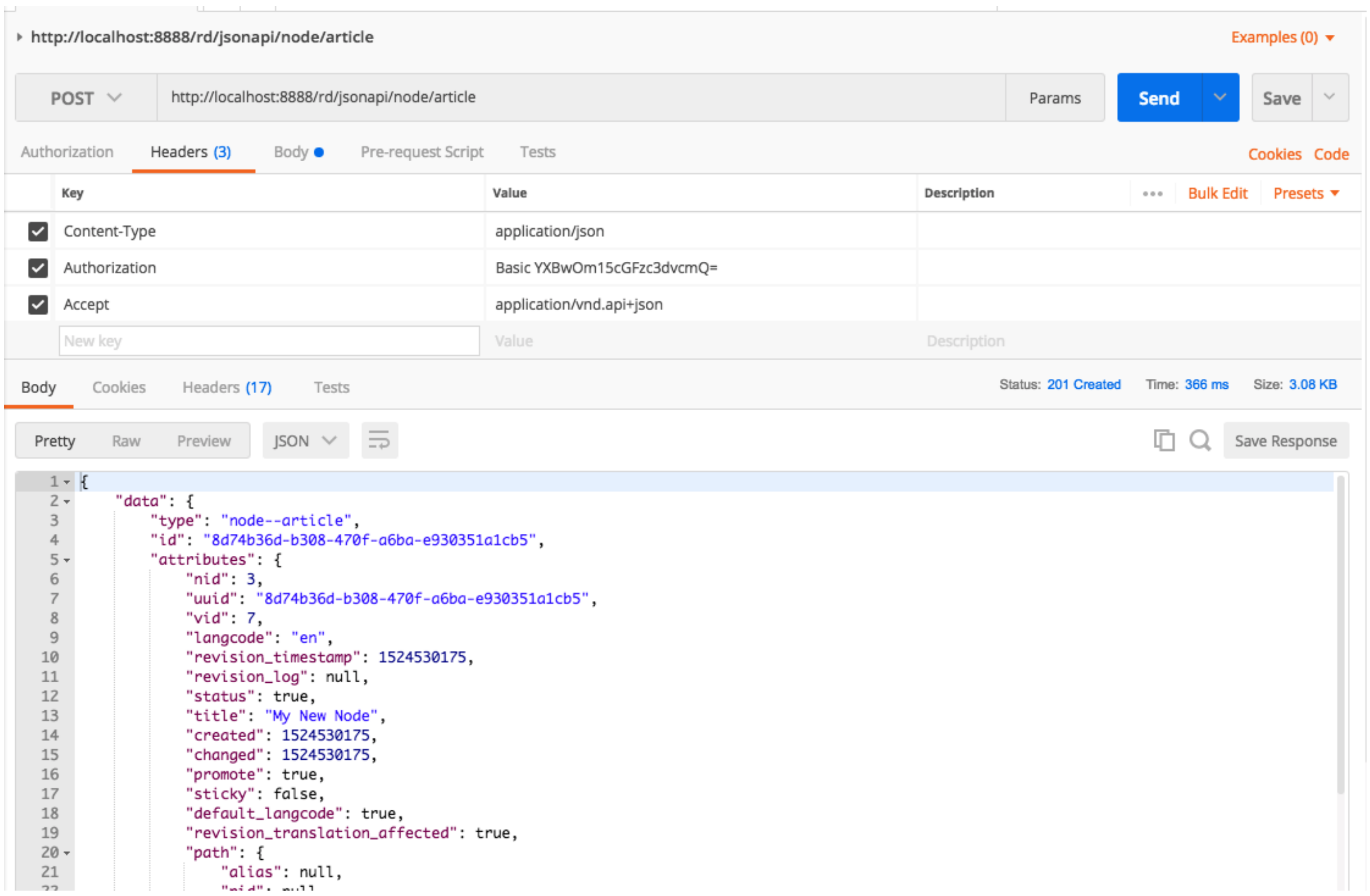


http://localhost:8888/rd/jsonapi/node/article

POST http://localhost:8888/rd/jsonapi/node/article

```
{
  "data": {
    "type": "node--article",
    "attributes": {
      "title": "My New Node",
      "body": {
        "value": "Content of new node",
        "format": "plain_text"
      }
    }
  }
}
```

If this is successful, you should get a `201 Created` response.



URL: http://localhost:8888/rd/jsonapi/node/article

Key	Value	Description
Content-Type	application/json	
Authorization	Basic YXBwOm15cGFzc3dvcmQ=	
Accept	application/vnd.api+json	

Status: 201 Created Time: 366 ms Size: 3.08 KB

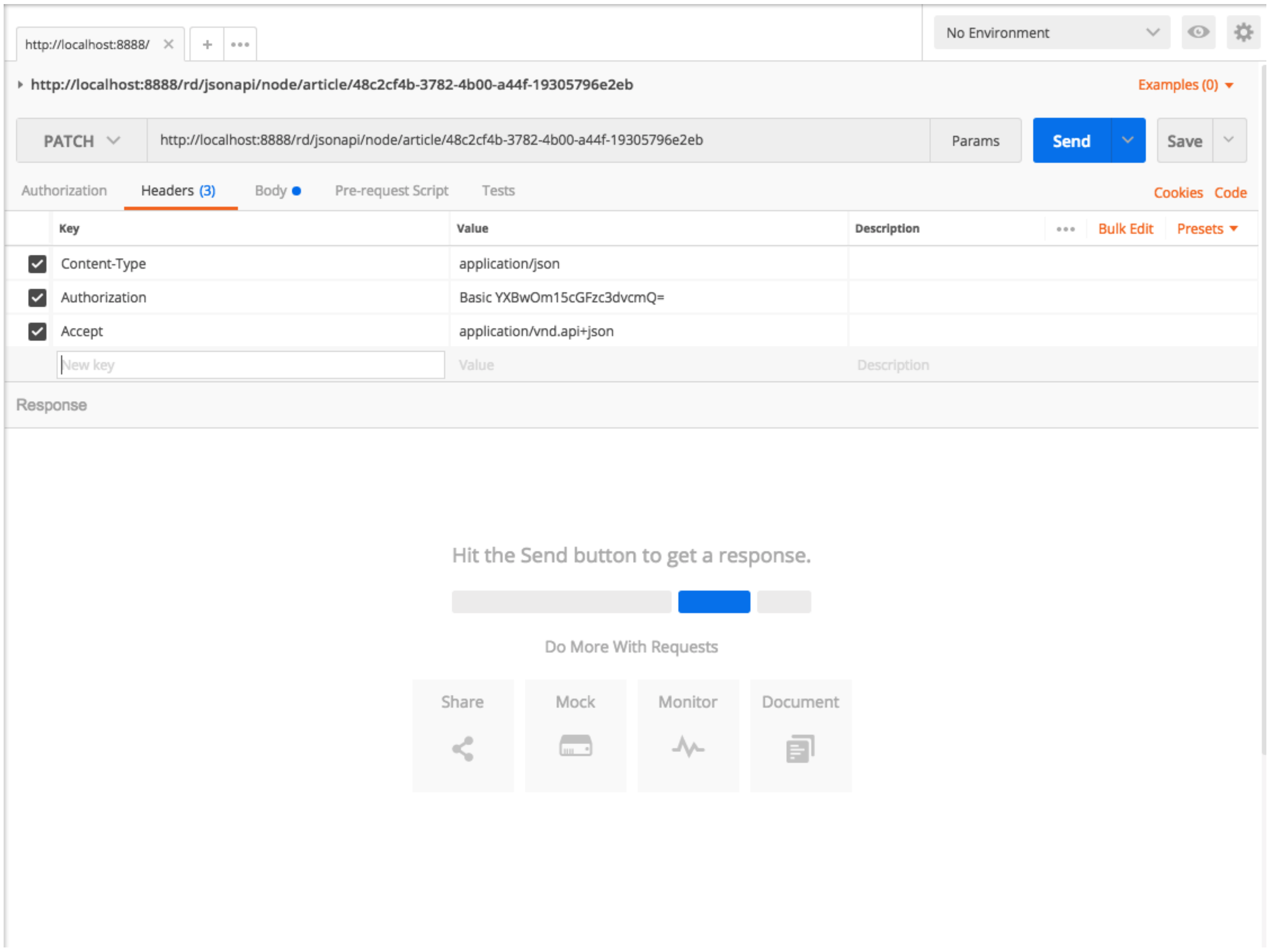
```
{
    "data": {
        "type": "node--article",
        "id": "8d74b36d-b308-470f-a6ba-e930351a1cb5",
        "attributes": {
            "nid": 3,
            "uuid": "8d74b36d-b308-470f-a6ba-e930351a1cb5",
            "vid": 7,
            "langcode": "en",
            "revision_timestamp": 1524530175,
            "revision_log": null,
            "status": true,
            "title": "My New Node",
            "created": 1524530175,
            "changed": 1524530175,
            "promote": true,
            "sticky": false,
            "default_langcode": true,
            "revision_translation_affected": true,
            "path": {
                "alias": null,
```

Use PATCH to edit a new node

Make a PATCH request to `http://localhost:8888/jsonapi/node/article/{NODE UUID}`

Headers:

```
Accept: application/vnd.api+json
Content-Type: application/vnd.api+json
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGci...
```



http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

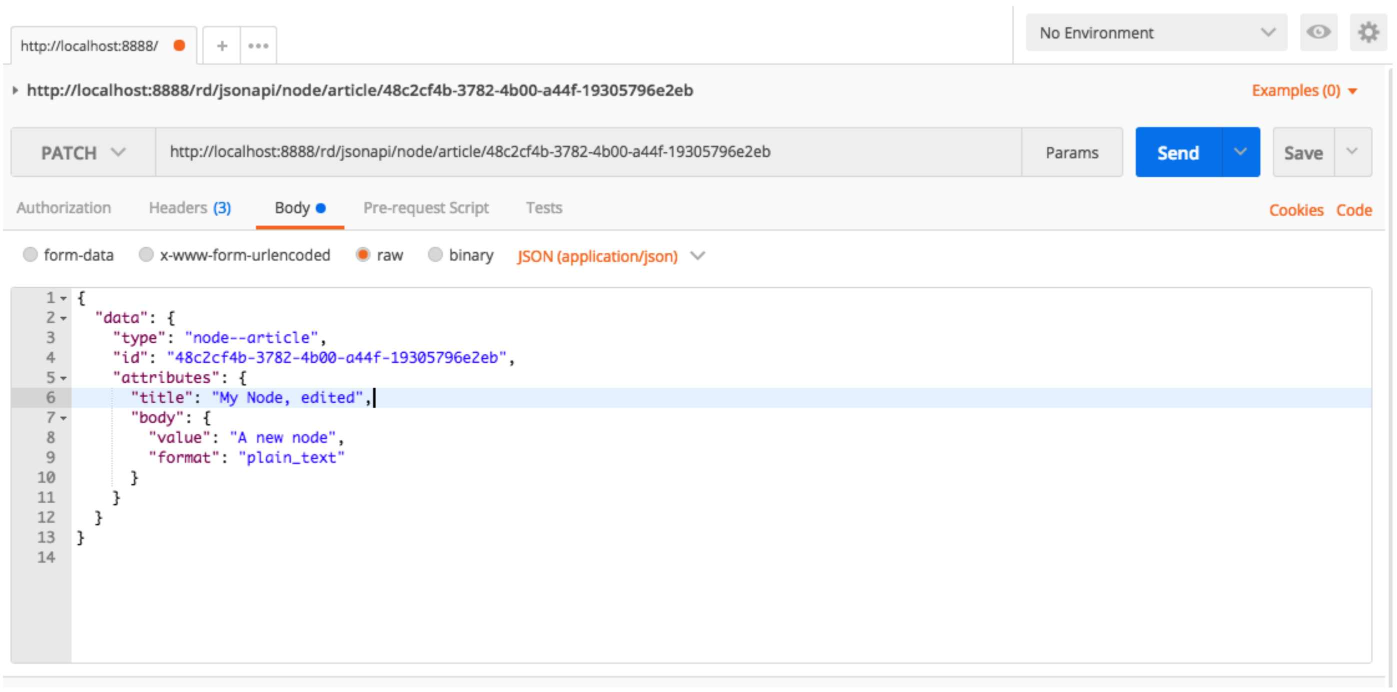
PATCH http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

Headers (3)

Key	Value	Description
✓ Content-Type	application/json	
✓ Authorization	Basic YXBwOm15cGFzc3dvcmQ=	
✓ Accept	application/vnd.api+json	

Body:

```
{
  "data": {
    "type": "node--article",
    "id": {NODE UUID},
    "attributes": {
      "title": "My Node, edited",
      "body": {
        "value": "A new node",
        "format": "plain_text"
      }
    }
  }
}
```

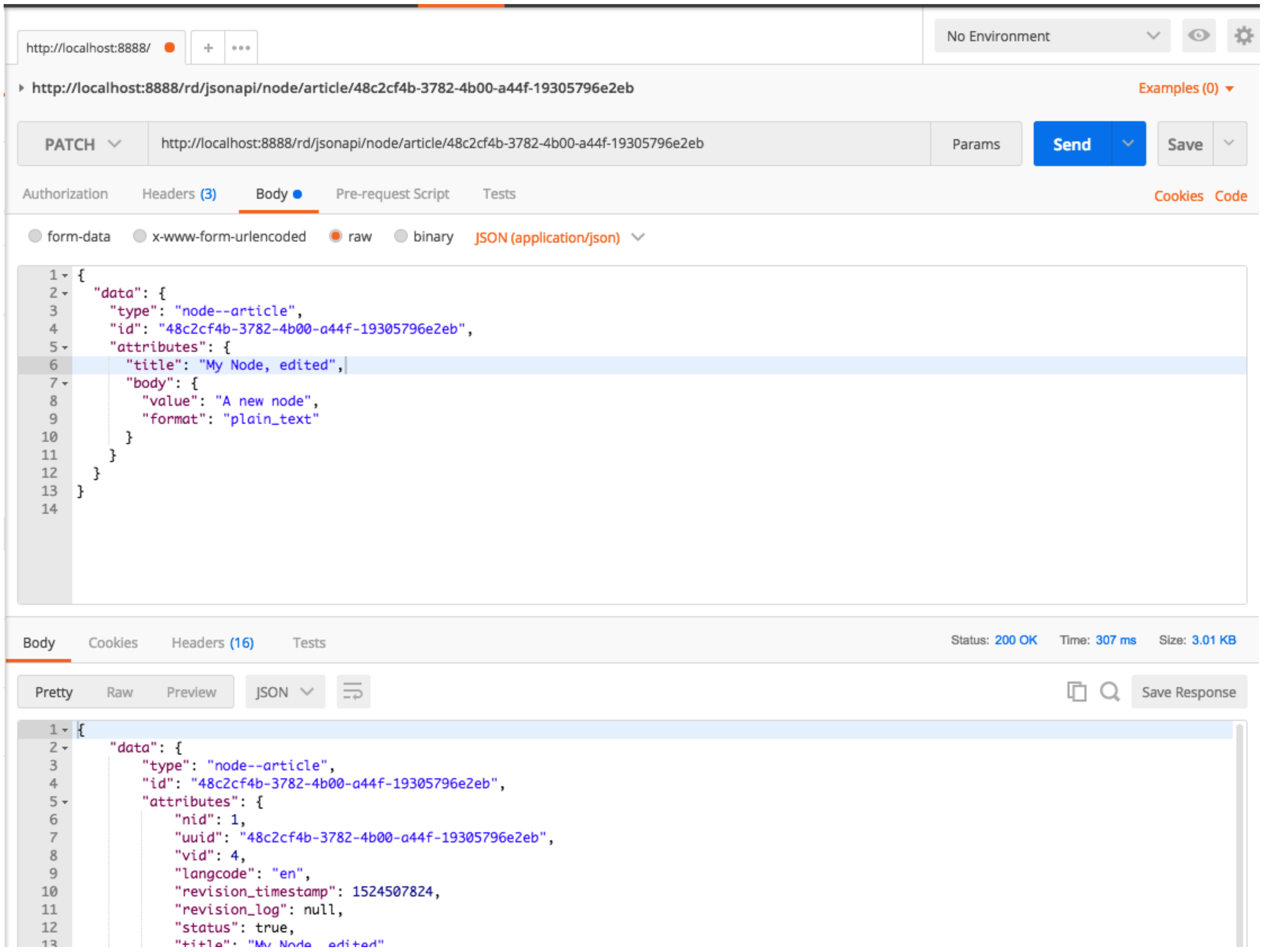


http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

PATCH http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

```
{
  "data": {
    "type": "node--article",
    "id": "48c2cf4b-3782-4b00-a44f-19305796e2eb",
    "attributes": {
      "title": "My Node, edited",
      "body": {
        "value": "A new node",
        "format": "plain_text"
      }
    }
  }
}
```

If this is successful, you should get a `200 OK` response.



http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

PATCH http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

```
{
  "data": {
    "type": "node--article",
    "id": "48c2cf4b-3782-4b00-a44f-19305796e2eb",
    "attributes": {
      "title": "My Node, edited",
      "body": {
        "value": "A new node",
        "format": "plain_text"
      }
    }
  }
}
```

Status: 200 OK Time: 307 ms Size: 3.01 KB

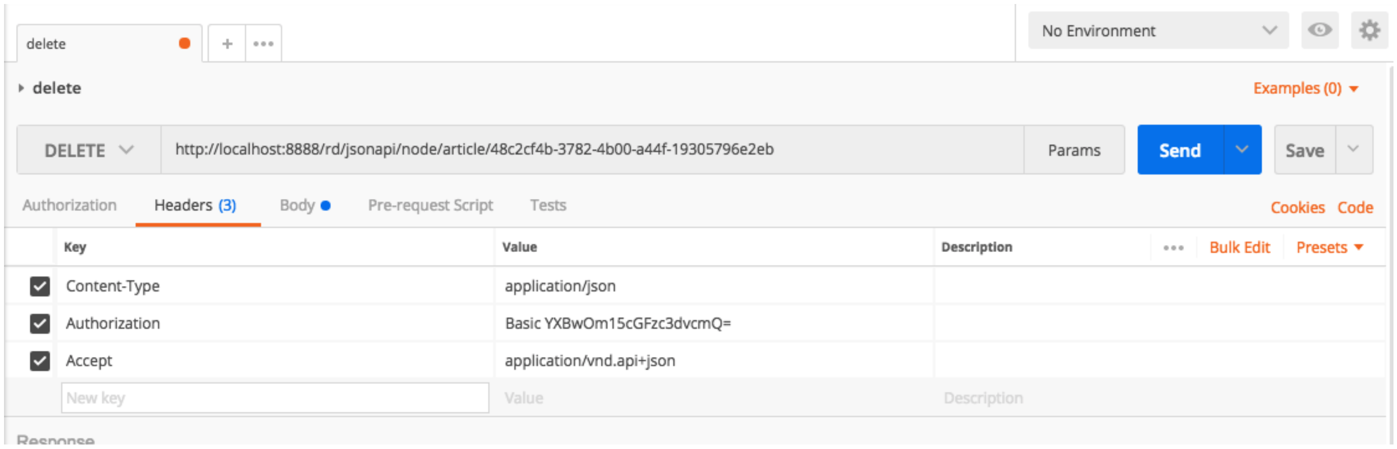
```
{
    "data": {
        "type": "node--article",
        "id": "48c2cf4b-3782-4b00-a44f-19305796e2eb",
        "attributes": {
            "nid": 1,
            "uuid": "48c2cf4b-3782-4b00-a44f-19305796e2eb",
            "vid": 4,
            "langcode": "en",
            "revision_timestamp": 1524507824,
            "revision_log": null,
            "status": true,
            "title": "My Node, edited"
```

Use DELETE to delete a node

Make a DELETE request to `http://localhost:8888/jsonapi/node/article/{NODE UUID}`

Headers:

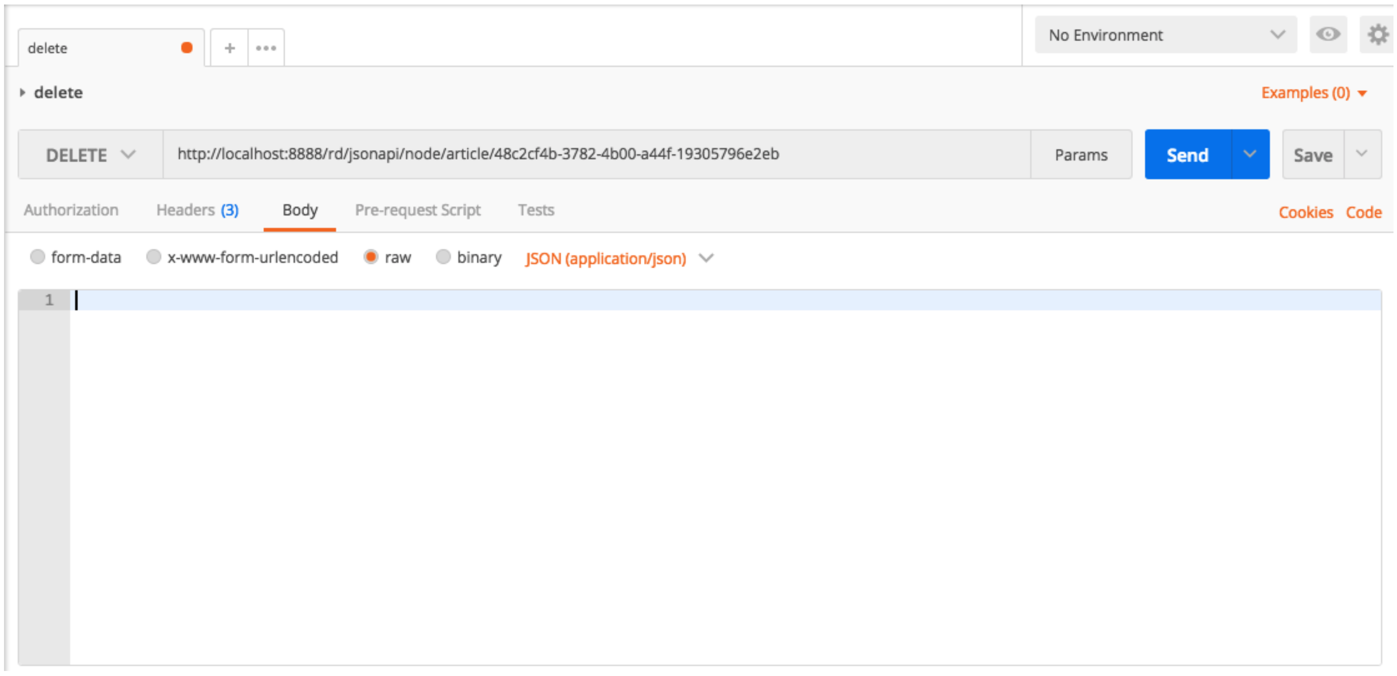
```
Accept: application/vnd.api+json
Content-Type: application/vnd.api+json
Authorization: Bearer eyJ0eXAiOiJKV1Qi...
```



http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

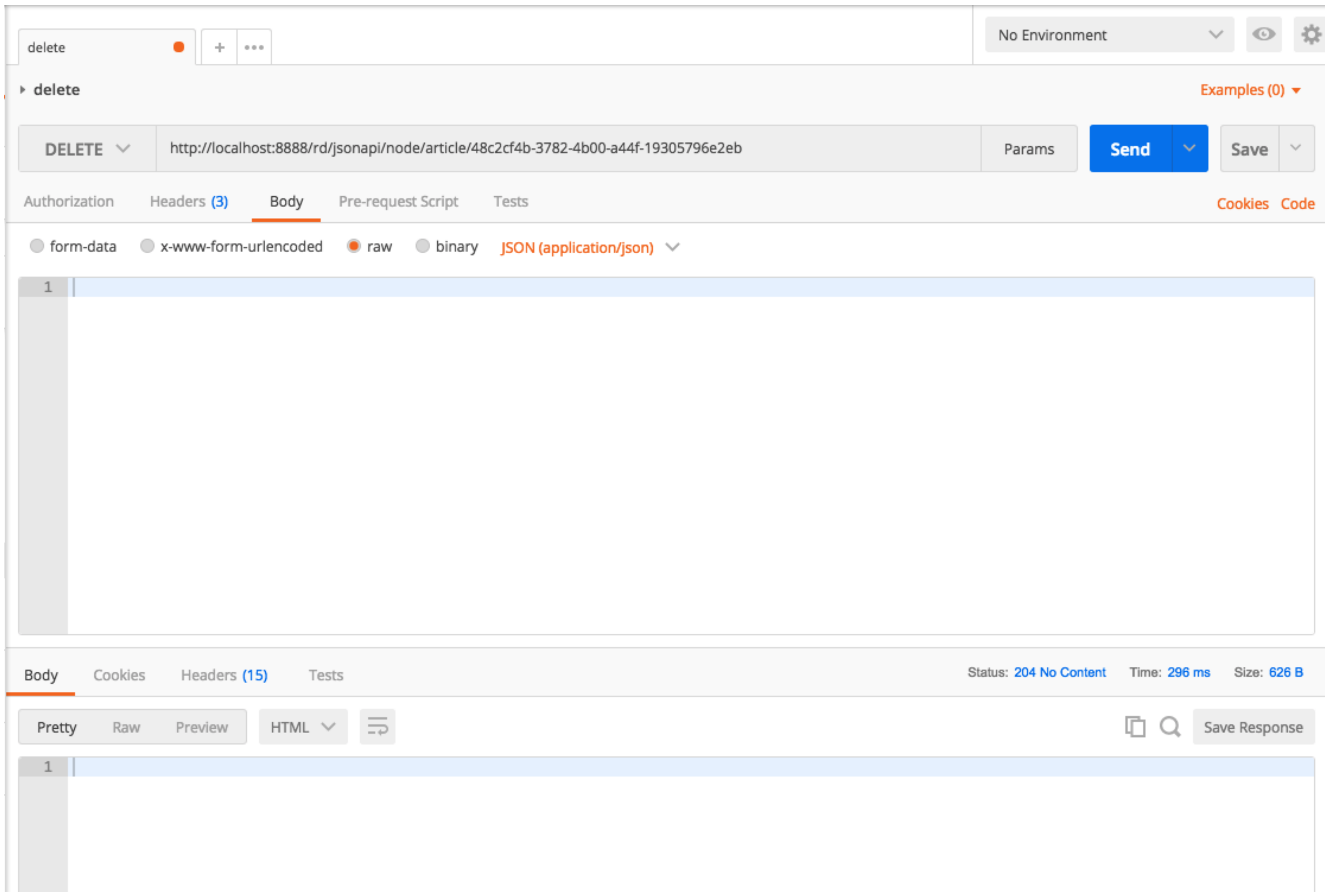
Key	Value	Description
✓ Content-Type	application/json	
✓ Authorization	Basic YXBwOm15cGFzc3dvcmQ=	
✓ Accept	application/vnd.api+json	
New key	Value	Description

No Body.



http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

If this is successful, you should get a `204 No Content` response with an empty body.



http://localhost:8888/rd/jsonapi/node/article/48c2cf4b-3782-4b00-a44f-19305796e2eb

Status: 204 No Content Time: 296 ms Size: 626 B

Troubleshooting

- Use double-quotes in your JSON, not single quotes, to avoid issues with special characters
- Make sure there are no trailing commas in your JSON objects

Recap

In this tutorial we installed and configured the Simple OAuth Drupal module to allow us to make authenticated requests to Drupal from a fully decoupled application, and then looked at some example requests.

Further your understanding

- Can you make OAuth requests using a different grant type other than "password"? How does that change the process? Which flow would be best for your use-case?
- Drupal can also be configured to authenticate API requests via HTTP Basic Authentication. When/why would you use OAuth versus Basic Authentication?

Additional resources

- What JSON:API DOESN'T do (drupal.org)
- API Authentication and Authorization (Drupalize.Me)
- Access an API from the Browser with Cross-Origin Resource Sharing (Drupalize.Me)
- Get a Token for OAuth 2 Requests (Drupalize.Me)

Was this helpful? Yes No

◀ Previous tutorial

Next tutorial ▶

Get Started Using React and Drupal Together

1 Introduction to React and Drupal Free

2 React Basics

3 Decoupled vs. Progressively Decoupled

4 Connect React to a Drupal Theme or Module Free

5 Create a React Component

6 Add Webpack Hot Module Replacement (HMR) to a Drupal Theme

7 Retrieve Data from an API with React

8 Use React to List Content from Drupal

9 Create, Update, and Delete Drupal Content with JavaScript

10 Build an Interface to Edit Nodes with React

11 Create a Fully Decoupled React Application Free

12 Use create-react-app to Start a Decoupled React Application

13 Make API Requests with OAuth

14 Use Fetch and OAuth to Make Authenticated Requests