Integration with the Spigit API

Set up an app within Spigit

Navigate to the administrator page from within your Spigit instance.

From there, find and click on the "API Management" icon.

Create a new app with all the required information. This new app will now appear under the "current apps" section in the API management page. Click on it to find the client ID, client secret, and redirect URL, which we will need to authenticate the API to work with our app.

Authenticate

Before you can interact with the Spigit API, you need to retrieve a OAuth2.0 token. Simply put, this is to ensure that only authorized users can interact with your instance's API.

To do this we need to make an API call using the HTTP POST method. We need to get an authorization code so that we can tell the Spigit API our calls are to be trusted.

```
https://[base_domain]/oauth/authorize?response_type=code&client_id={AppID}&redirect_uri={CallbackURL}
```

Enter this into your browser url bar, and replace [base_domain] with the instance subdomain. For example, the acme company's base domain would be "acme.spigit.com."
Replace {AppID} with the App ID of the app we created in the last step. Replace {CallbackURL} with a suitable redirect URI. If you don't have one, you can use the one provided by Spigit.

```
https://{base_domain}/main/html/api/testOauthRedirectPage.html
```

[Note: the callback URL (aka redirect URI) must be the same on both the API management UI and in the url you plug into a browser]

Now that we have a URL filled out with all of our information, we can enter it into a browser URL bar. That will take us to a page that asks if we want to allow our app to access data. When we accept, it will send us to a page with the authorization

code we wanted in the URL bar. Log that somewhere for when we need to get auth tokens.

Get an OAuth token

Now that we have manually gathered a code, we can get OAuth tokens that are used when making API calls. In order to get the OAuth token we use the following HTTP POST method:

```
POST https://acme.spigit.com/oauth/token

     Body:
grant_type: authorization_code
code: {code}
client_id: {app ID}
client_secret: {app Secret}
```

This will get a response in JSON format, with the field "access_token" set to the auth token we need. We should log this in our program and use the token for making Spigit API calls later.

The response will be a single JavaScript object bearing fruits:

```
{
    "access_token": "3712ced6164693ba08174e57fe05e359",
    "refresh_token": "o9rzvp3KBQLF",
    "token_type": "Bearer",
    "issued_at": "49609-07-28T07:19:52Z",
    "expires_in": 600
}
```

Make calls to the Spigit API

Now we have our authentication token and can make any calls to the Spigit API. For this example, we'll make a call to list all communities and their values, such as ID's, names and titles. We make a GET to the provided URL and specify the token we retrieved in the last section with the key "Authorization" and the value "Bearer {auth_token}" with auth-token being the token we received.

```
GET
https://acme.spigit.com/api/v1/communities?offset=0&limit=2&site_type=
```

```
regular&regular_status=active&challenge_status=scheduled&sort=id%2Ctit
le%2Ccommunity_name&include_hidden=true&include_disabled=false

Headers:
Authorization: Bearer {auth token}
```

This method, if the authentication token given should result in a response showing all communities and their properties. The response will be another Javascript object. To see the various fields and information, either try to do it yourself, or check the Spigit API reference.

Making HTTP calls and logging responses

If you're not sure how exactly to POST or GET, or even log responses, this section will be useful. I'm going to show the code I wrote that connects the Spigit integration with Slack. To achieve this, I used Node.JS, which allows us to use javascript to send and receive data with HTTP.

For the first part of the script, we want to acquire a module that allows us to easily request data with HTTPS (request library)

```
var request = require("request");
```

Now that we've imported the request library we can make HTTP calls

Let"s get a Spigit OAuth token using Node

```
var url = "https://acme.spigit.com/oauth/token";

var data = JSON.stringify({
      "grant_type": "authorization_code",
      "code":  "*****",
      "client_id": "*****",
      "client_secret": "*****"
      }
)

var header = {
      "Content-Type": "application/json"
}
```

```
var options = {
     url: url,
     body: data,
     headers: header
}

request.post(options, (error, response, body) => {
     var bodyObject  = JSON.parse(body);
}
```

This code creates an HTTP post to the specified **URL** and with the body declared in **data** and headers declared in **header**. By using the **request.post** method, we know we're using POST and not GET or some other method. To use another method, we can use choose another subroutine such as **request.get**. The variable **bodyObject** will contain the javascript response given by by the Spigit API. With this code block, the entirety of the Slack integration was created. For different functions, I reworded the code and added error catches and data loggers. With the Spigit API, we can accomplish nearly any task.

The create idea POST call

Here is a look at the post idea API call.

For this example we will say that we're working in an instance named "acme." In order to create an idea and post it to a community from the API, we need to specify which community we'll be posting to. The community ID is represented in the code here as ${communityID}. The ID of a community can be found by using the Listing Communities API GET call.

All of the content that makes up the idea will be in the body of the POST. Here title, tags, category id, description, and all other aspects of an idea, default or changed, will needed. Once the idea has been sent via the POST request, the API will give us a response body with either a success message or failure message as well as the idea ID.

```
var url =
"https://acme.spigit.com/api/v1/communities/${communityID}/ideas";

var idea = JSON.stringify({
```

```
      "title": "Post title",
      "tags": "Tag1, Tag2, Tag3",
      "category_id": "",
      "post_anonymously": false,
      "template_fields": {
            "Content": "This is a description"
            }
      }
)

var header = {
      "Authorization": "Bearer ${auth_token}",
      "Content-Type": "application/json"
}

var options = {
      url: url,
      body: idea,
      headers: header
}

request.post(options, (error, response, body) => {
      var bodyObject  = JSON.parse(body)
}
```