SALSIFY

v1.0

Token

SUGGEST EDITS

In order to call any Salsify APIs, you must provide your authorization token, or use OAuth2.

The auth token is specific to a user account (not an individual Salsify workspace). Treat this token as you would your own personal password -- don't send it to anyone, and make sure to store it in a safe place.

The Salsify API uses authentication tokens to allow access to the API. You can generate a personal token in the Salsify application on the My Profile page.

Token Authentication

SUGGEST EDITS

Once you have your auth_token, it can be passed in as an http header or via an access_token query parameter

> We recommend sending the authentication token using the Authorization header. Sending the authentication token as a query parameter is less secure because various systems may log the URL along the request path.

Header Authentication Examples

cURL Ruby Python

```
curl -X GET \
  https://app.salsify.com/api/v1/orgs/s-999-999-999-999/products/102918
  -H 'Authorization: Bearer YOUR-AUTH-TOKEN' \

  ## where org_id = s-999-999-999-999, salsify:id=102918, api_token = Y
```

Query Authentication Example

HTTP

```
https://app.salsify.com/api/v1/orgs/s-999-999-999-999/products/1234?acc
```

where org_id = s-999-999-999-999, product id=1234, api_token = YOUR-AUTH-TOKEN

OAuth2

SUGGEST EDITS

There are two ways to authenticate with the Salsify API: authentication via OAuth 2.0 and authentication by passing HTTP Authorization headers.

We also support passing the authentication token via an access_token query parameter:
`http://app.salsify.com/api/v1/products/1234?access_token=<your auth token>`

> Note: We recommend sending the authentication token using the Authorization header. Sending the authentication token as a query parameter is less secure because various systems may log the URL along the request path.

For more details on integrating with Salsify, click here.

OAuth Setup

1. Email success@salsify.com and provide your preferred URI to receive your *client_id* and *client_secret*.
2. Salsify will provide you with a *client_id* and *client_secret*, which you can use to kick off the flow.
3. Make the authorization request to get the *Authorization Code* a a response.
   Send a GET request to:
   `https://app.salsify.com/oauth/authorize&redirect_uri=<your_preferred_redirect_url>&respose_type=code&client_id=<we_provide_you_with_this>`
   You should get an *Authorization Code* in the response.
4. Exchange the *Authorization Code* for an *Access Token*.
   Send a POST request to: `https://app.salfisy.com/oauth/token`
   Include in the BODY:
   code=(the authorization code you get sent back from Step #1)
   grant_type=authorization_code
   redirect_uri=<your_preferred_redirect_uri>

Include in the HEADER:
client_id=<we_provide_you_with_this>
client_secret=<we_provide_you_with_this>
You should receive an *Access Token* in the response, which is what you'll include in the header of your requests to our API.

Ruby

```ruby
#!/usr/bin/env ruby
require 'sinatra'
require 'sinatra/reloader'

# Access tokens are stored in a session variable, encrypted in a cookie
# This removes the burden of maintaining sessions across multiple serve
enable :sessions

# This is the callback configured for the "Local Sinatra Application" r
# Use this route and the given UID/secret values below instead of regis
# for local use.
CALLBACK = "http://localhost:4567/auth/callback".freeze

# OAuth application UID. Can be obtained from production Rails console
# Doorkeeper::Application.find_by!(name: 'Local Sinatra Application').u
APP_ID = ENV['SALSIFY_APPLICATION_ID'].freeze
# OAuth application secret. Can be obtained from production Rails conso
# Doorkeeper::Application.find_by(name: 'Local Sinatra Application').se
# WARNING: Do NOT store the secret in version-controlled source code.
# Use environment variables or another mechanism to feed this informati
SECRET = ENV['SALSIFY_APPLICATION_SECRET'].freeze

# Change this if you're running against a local instance of Dandelion.
# DANDELION = 'http://localhost:5000'.freeze
DANDELION = 'https://app.salsify.com'.freeze

UNAUTHENTICATED_PATHS = ['/auth/callback'].freeze

# Configure the OAuth2 client for later use
require 'oauth2'
set :oauth_client, OAuth2::Client.new(APP_ID, SECRET, site: DANDELION)

# This application uses curl to retrieve data from Dandelion, but any H
# you to set custom headers is sufficient.
require 'curl' # comes from the "curb" gem

# The before block runs before each request is processed.
before do
 # The current token is serialized into a cookie, re-materialze it for
 if session.has_key?(:current_token)
 @current_token = OAuth2::AccessToken.from_hash(settings.oauth_client,
 end

 # If there is no current token in the user's session, redirect immedia
 # If only part of your application requires authorized access, you sho
 unless @current_token || UNAUTHENTICATED_PATHS.include?(request.path)
 redirect_to_sign_in_page!
 end
end

# The after block runs after each request is processed.
after do
 # Serialize the current token object back into the user's cookie
 if @current_token
 session[:current_token] = @current_token.to_hash
 end
end

# Get some data via the API and feed it to the information-hungry consu
get '/' do
 response = with_auth_token_refresh do
 curl_client = Curl::Easy.new
 curl_client.verbose = true # handy for debugging, e.g. making sure the
 curl_client.url = DANDELION + '/api/v1/products/100'
 # This is the important part. All of your requests to the API need thi
 # Authorization: Bearer 0123456789abcdef
 curl_client.headers['Authorization'] = "Bearer #{@current_token.token}
 curl_client.perform
 curl_client
 end

 # Catch all for handling a 401 response code
 # (e.g. if we weren't given a refresh token)
 if response.response_code == 401
 redirect_to_sign_in_page!
 end

 [response.response_code, {'Content-Type' => response.content_type}, re
end

# This is the callback that Dandelion, the provider, will send the clie
# succeeds. We only care about the access token and the user's original
get '/auth/callback' do
 @current_token = settings.oauth_client.auth_code.get_token(params[:coc
 redirect to(session.delete(:post_auth_redirect))
end

helpers do
 # Interprets the return code of a request. If the request
 # is unauthorized, attempt to use the refresh token and request again.
 def with_auth_token_refresh
 response = yield

 if response.response_code == 401 && @current_token.refresh_token
 begin
 @current_token = @current_token.refresh!
 rescue OAuth2::Error
 # If the token refresh fails, take the user
 # back to the sign in page
 redirect_to_sign_in_page!
 end
 response = yield
 end

 response
 end

 # Immediately redirect the user to the auth provider's sign in page.
 def redirect_to_sign_in_page!
 session[:post_auth_redirect] = request.path
 redirect to(settings.oauth_client.auth_code.authorize_url(redirect_uri
 end
end
```

Products Overview

SUGGEST EDITS

> **URL Path Update (April 2017)**
>
> Updates have been made to include organization ID (org ID) in the URL path as of April 2017. This is the preferred method, but for user IDs associated with a single Salsify organization, previously-working methods will continue to function as expected. If a user ID is associated with more than one Salsify organization, org ID is required.

Products are JSON objects whose members are the attributes of the product. They contain properties which hold information about the product. Properties can hold:

- system metadata about the product
- digital assets associated with products (if any)
- product relations (if any)
- parent/variant relationships (if any)

Product System Metadata

All product system metadata is prefixed with `salsify:`

attribute	description	user editable
	Unique product identifier which holds	

attribute	description	user editable
salsify:id *string*	the same value as the property that is set as the product ID in the organization. `salsify:id` cannot be modified directly. Updates can be made to the property in the organization's product ID role, and those updates will update the `salsify:id` value as well.	Y
salsify:created_at *date*	Date product record was created; in UTC. *Format: YYYY-MM-DDTHH:MM:SS.MSZ*	N
salsify:updated_at *date*	Date product record was most recently updated; in UTC. *Format: YYYY-MM-DDTHH:MM:SS.MSZ*	N
salsify:version *integer*	Counter which starts at 0 when record is created, and is incremented by 1 each time an update to the product is saved. In products with parent/variant relationships, variant product version is not incremented when inherited property values are updated in the parent product.	N
salsify:relations_updated_at *date*	Date product relations were recently updated; in UTC. *Format: YYYY-MM-DDTHH:MM:SS.MSZ*	N
salsify:profile_asset_id *string*	The digital asset ID for a thumbnail image manually selected to override system thumbnail defaults. In most cases, this will be `null`.	Y
salsify:parent_id *string*	Present in organizations with the parent/variant feature enabled. If parent ID contains a value, the product is a variant. Parent ID contains the ID for the parent product. By default, parent property values are inherited by the variant product. If parent_id is `null`, the product either has no parent/variant relationship, or the product is a parent to one or more variants. Learn more about parent/variant relationships.	Y
salsify:system_id	Unique, permanent identifier for product record. System ID cannot be	

string **attribute**	altered, and does not change with updates to the value in the property holding the product ID role. **description**	N **user editable**

Product ID

Product ID is a string format unique product identifier which is user defined when the first data import is completed. Updates can be made to the value in the property id role, and those updates will update the `salsify:id` value as well. The id cannot start with an `=`, `_`, or `salsify`.

Input Format

The serialization format for products will mirror the format used for JSON imports. Products are represented as key value pairs of `property_id`:`property_value`. With the exception of system properties, only property IDs that contain property values are stored with product records.

Input format example JSON body

```
{
    "Product ID": "102918",
    "Product Name": "Jetsetter Carry On Roller - Purple",
    "Main Image": "8d538dcbfd2d5547f7080f2408e7457e96c0451a",
    "UPC": "143287212918",
    "Certifications": "TSA-Approved",
    "Assembly Country of Origin": "US",
    "Component Country of Origin": "China",
    "Warranty": "<p>5 year limited warranty</p>",
    "Color": "Purple",
    "Tags": [
        "air travel",
        "TSA approved",
        "lightweight luggage",
        "rolling bag"
    ]
}
```

Ruby Example

Here's a Ruby example of API usage. It leverages the [rest-client gem[(https://github.com/rest-client/rest-client). Note that the referenced items (e.g. SKU, Name, Brand) need to be existing attributes exactly in that form in the existing schema, and they are case sensitive.

Ruby

```
require 'rest_client'

# Create the product
product = {'SKU' => '12345', 'Name' => 'Big TV', 'Brand' => 'Salsify'}
RestClient.post('https://app.salsify.com/api/v1/orgs/9999-9999-9999-999
                accept: 'application/json', content_type: 'application/
                Authorization: "Bearer <your auth token here>")

# Update the product
product = {'Inventory' => '10'}
RestClient.put('https://app.salsify.com/api/v1/orgs/9999-9999-9999-9999
               accept: 'application/json', content_type: 'application/j
               Authorization: "Bearer <your auth token here>")

# Get the product
result = RestClient.get('https://app.salsify.com/api/v1/orgs/9999-9999-
                        Authorization: "Bearer <your auth token here>")
product = JSON.parse(result)

# Delete the product
RestClient.delete('https://app.salsify.com/api/v1/orgs/9999-9999-9999-9
                  Authorization: "Bearer <your auth token here>")
```

Create Product

SUGGEST EDITS

PATH PARAMS

org_id* string

The Salsify organization's unique identifier. Visit your Salsify organization and pull from the URL path, immediately following /orgs/

BODY PARAMS

product_name string

Property ID for the property that is set as the product name role in Salsify. Product name will be left blank if not included.

additional_properties string

Additional information to store in properties associated with the product, sent in value pairs of `property_id : property_value`. Property IDs that don't exist in the Salsify org will be created as default string data type. Accepts multiple property values as arrays.

Usage

When you add products, you can add property values that contain product details and the following system properties:

- salsify:parent_id
- salsify:profile_asset_id
  Salsify will ignore updates to other system properties.

Property IDs are case sensitive. The id cannot start with any of the following: `=`, `_`, or `salsify`.

Products can be added individually, or as an array. See array example below.

The create and update endpoints will not yet support creating digital assets or relations. Those are on the roadmap.

Add values to new properties

You can also add new properties when you add a product. If you send values for property IDs that don't currently exist in the system, the properties will be created and the associated values will be stored with a default string data type.

Add multiple values to a single property

Properties accept and store multiple values, with the exception of properties assigned to the Product ID and Product Name roles, and system properties in Salsify. Send multiple values to a property in an array of `property_id`:`property_value` pairs.

JSON

```json
{
    "Product ID":"456",
    "Product Name":"Product 456",
    "Material":"plastic",
    "Tags":["headphones","noise cancelling"]
  }
```

In this example two property values will be created for the property ID `Tags`.

Returns

Successful result will return property values and system property values for the new product. If in-app computed properties are configured, the computed property values will also be returned.

Bulk Create Products

SUGGEST EDITS

Read Product Record

SUGGEST EDITS

Bulk Read Products - REPORT

SUGGEST EDITS

Update Product

SUGGEST EDITS

Bulk Update Products

SUGGEST EDITS

Delete Product

SUGGEST EDITS

Bulk Delete Products

SUGGEST EDITS

Properties Overview

SUGGEST EDITS

Create New Property

SUGGEST EDITS

Bulk Create New Properties

SUGGEST EDITS

Read Property

SUGGEST EDITS

Read Multiple Properties - REPORT

SUGGEST EDITS

Update Property

SUGGEST EDITS

Delete Property

SUGGEST EDITS

Bulk Delete Properties

SUGGEST EDITS

Import Format

SUGGEST EDITS

Create a mount point

SUGGEST EDITS

Upload to a mount point

SUGGEST EDITS

Creating a JSON Import from a mount point

SUGGEST EDITS

Creating a JSON Import from FTP

SUGGEST EDITS

Updating an Import to point at a new mount point

SUGGEST EDITS

Starting an Import Run

SUGGEST EDITS

Export Overview

SUGGEST EDITS

Start Export Run

SUGGEST EDITS

Get Export Status

SUGGEST EDITS

Digital Asset Overview

SUGGEST EDITS

Create a Digital Asset &

SUGGEST EDITS

Metadata

Create Multiple Digital Assets

SUGGEST EDITS

Read Digital Asset Metadata

SUGGEST EDITS

Where a user has permission and the digital asset exists, complete digital asset JSON object is returned with its most current information.

Read Multiple Digital Assets - REPORT

SUGGEST EDITS

Refresh Digital Assets

SUGGEST EDITS

Update Digital Asset Metadata

SUGGEST EDITS

Update Multiple Digital Assets

SUGGEST EDITS

Delete Digital Assets

SUGGEST EDITS

Delete Multiple Digital Assets

SUGGEST EDITS

List Membership Overview

SUGGEST EDITS

Add List Members

SUGGEST EDITS

Delete List Members

SUGGEST EDITS

Webhook Signature Headers

SUGGEST EDITS

Webhook Signature Verification Algorithm

SUGGEST EDITS

Certificate Caching

SUGGEST EDITS