NOTE: All steps below assume you have Drupal 8.1.1+ installed with "standard" profile chosen during installation.

Demo version

Before we start, you may test how everything works on demo website

1. Demo website:

Url: http://demo1a3.project.experienceit.pl/

(optional) you might login into Drupal admin to create invoices manually
Login: Dev
Password: dev

2. Making api call

2.1 Get session token:

- Go to: http://demo1a3.project.experienceit.pl/rest/session/token
- I've got this token: JKN0jaysntR3ln_Wwxi7e_jA_9D9YtaY6pLH0TW90Lo

2.2 Make cURL call:

curl --include --request POST --user dev:dev --header "Content-type: application/json" --header "X-CSRF-Token: **JKN0jaysntR3ln_Wwxi7e_jA_9D9YtaY6pLH0TW90Lo**" --data-binary "{\"type\":[{\"target_id\":\"invoice\"}], \"title\":[{\"value\":\"Invoice\"}], \"field_no\":[{\"value\":\"No\"}], \"field_date\":[{\"value\":\"2016-06-15T02:59:52\"}], \"field_product\":[{\"value\":\"Product\"}], \"field_fees\":[{\"value\":\"123.45\"}], \"field_customer_name\":[{\"value\":\"John Doe\"}], \"field_customer_address\":[{\"value\":\"Street\nCity\nPost Code\"}]}" http://demo1a3.project.experienceit.pl/entity/node?_format=json

---

1. Define Invoice content type

1.1 Create new content type

- Go to: /admin/structure/types/add
- Fill required fields
- Important: machine name must be set to „invoice"
- Click "Save and manage fields"

1.2 Add fields

1.2.1 No

- Go to: /admin/structure/types/manage/invoice/fields/add-field
- Field type: Text (plain)
- Label: “No”
- Important: machine name must be set to “no” – it will show as “field_no”
- Click “Save and continue”, continue to new page
- Maximum length: “50”
- Click “Save field settings”, continue to new page
- Click “Save settings”

1.2.2 Date

- Go to: /admin/structure/types/manage/invoice/fields/add-field
- Field type: Date
- Label: “Date”
- Important: machine name must be set to “date” – it will show as “field_date”
- Click “Save and continue”, continue to new page
- Click “Save field settings”, continue to new page
- Click “Save settings”

1.2.3 Product

- Go to: /admin/structure/types/manage/invoice/fields/add-field
- Field type: Text (plain)
- Label: “Product”
- Important: machine name must be set to “product” – it will show as “field_product”
- Click “Save and continue”, continue to new page
- Maximum length: “50”
- Click “Save field settings”, continue to new page
- Click “Save settings”

1.2.4 Fees

- Go to: /admin/structure/types/manage/invoice/fields/add-field
- Field type: Number (decimal)
- Label: “Fees”
- Important: machine name must be set to “fees” – it will show as “field_fees”
- Click “Save and continue”, continue to new page
- Click “Save field settings”, continue to new page
- Click “Save settings”

1.2.5 Customer Name

- Go to: /admin/structure/types/manage/invoice/fields/add-field
- Field type: Text (plain)
- Label: “Customer Name”

- Important: machine name must be set to “customer_name” – it will show as “field_customer_name”
- Click “Save and continue”, continue to new page
- Click “Save field settings”, continue to new page
- Click “Save settings”

1.2.6 Customer Address

- Go to: /admin/structure/types/manage/invoice/fields/add-field
- Field type: Text (plain, long)
- Label: “Customer Address”
- Important: machine name must be set to “customer_address” – it will show as “field_customer_address”
- Click “Save and continue”, continue to new page
- Maximum length: “50”
- Click “Save field settings”, continue to new page
- Click “Save settings”

1.2.7 (optional) Remove Body

- Drupal 8 adds body field to every new content type. We don’t need it for Invoice, so it can be removed. Go to: /admin/structure/types/manage/invoice/fields
- Click “Delete” next to Body field, continue to new page
- Confirm Delete

1.3 Final result

When you are done, go to: /admin/structure/types/manage/invoice/fields - it should look like on image below.

LABEL	MACHINE NAME	FIELD TYPE	OPERATIONS
Customer Address	field_customer_address	Text (plain, long)	Edit
Customer Name	field_customer_name	Text (plain)	Edit
Date	field_date	Date	Edit
Fees	field_fees	Number (decimal)	Edit
No	field_no	Text (plain)	Edit
Product	field_product	Text (plain)	Edit

2. Enable modules

- Go to: /admin/modules
- Enable modules:
    - HTTP Basic Authentication

- RESTful Web Services
- Serialization

3. Set permissions

3.1 (optional) Create new user only for api calls

If you only have Administrator account, you might need to create new user that will be used only for API calls. This step is optional – you can just use your Admin account to make the calls.

3.1.1 Create new user role

- Go to: /admin/people/roles/add
- Set Role name, for example: “Api calls”
- Click “Save”

3.1.2 Create new account

- Go to: /admin/people/create
- Fill required fields, for example:
  - Username: Dev
  - Password: dev
- Important: Set “Roles” to role that have permissions to make api calls (see below about setting permissions) – in my example it is role called “Api calls” that we created a moment ago.
- Click “Create new account”

3.2 Set permissions

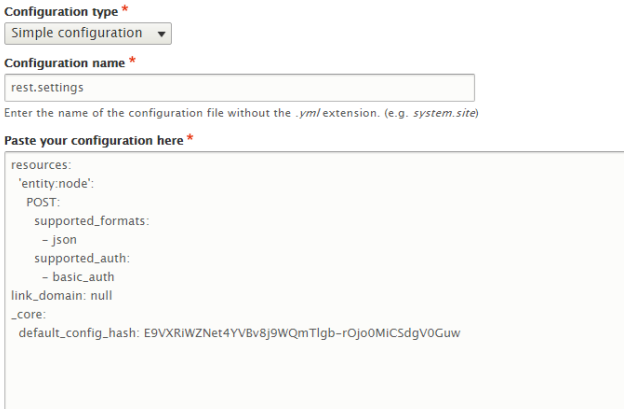
- Go to: /admin/people/permissions
- Set permissions for selected role (in my case for “Api calls”):
  - Invoice: Create new content
  - Access POST on Content resource
- Click “Save permissions”

4. Expose Invoice content type through RESTful Web Services API

Inside this project there is “settings” folder – inside you will find rest.settings.yml file. Open it in text editor – you will need content of that file in a moment.

- Go to: /admin/config/development/configuration/single/import
- Configuration type: Simple configuration
- Configuration name: “rest.settings”
- Paste your configuration here: paste content of rest.settings.yml file

- Double check that everything looks like the image below, and then click “Import”



- On next page click “Confirm”

5. Using cURL to add Invoice

Now we can make api calls. This requires two steps:

5.1 Get session token

- Go to /rest/session/token
- Copy token we will need it in a moment

5.2 Making the call

5.2.1 Technical details

Now we need to send actual data, important thing is it must be in JSON format. For testing purposes I’m using this string:

```
{
   "type":[{"target_id":"invoice"}],
   "title":[{"value":"Invoice"}],
   "field_no":[{"value":"No"}],
   "field_date":[{"value":"2016-06-15T02:59:52"}],
   "field_product":[{"value":"Product"}],
   "field_fees":[{"value":"123.45"}],
   "field_customer_name":[{"value":"John Doe"}],
   "field_customer_address":[{"value":"Street\nCity\nPost Code"}]
}
```

Explanation (values that you might change are in bold):

- "type":[{"target_id":"invoice"}] – node type, in our case “invoice” - this part must not change!

- "title":[{"value":"**Invoice**"}] – this is node title – we don't need it in our invoices, but Drupal requires it. You might give different title for each node, but naming every "Invoice" is just fine.
- "field_no":[{"value":"**No**"}] – invoice number
- "field_date":[{"value":"**2016-06-15T02:59:52**"}] – invoice date. It is required that date (2016-06-15T02:59:52) is in "Local date and time" format. More on that here: https://www.w3.org/TR/html-markup/datatypes.html#form.data.datetime-local
- "field_product":[{"value":"**Product**"}] – product name
- "field_fees":[{"value":"**123.45**"}] – fees. Number format is important. Only numbers and single dot is allowed.
- "field_customer_name":[{"value":"**John Doe**"}] – Customer Name
- "field_customer_address":[{"value":"**Street\nCity\nPost Code**"}] – Customer Address. The "\n" are there to divide string in lines, so when you display it it will be shown as:
  Street
  City
  Post Code

5.2.2 Using cUrl

This is how my curl command looks like:

curl --include --request POST --user **dev:dev** --header "Content-type: application/json" --header "X-CSRF-Token: **JKN0jaysntR3ln_Wwxi7e_jA_9D9YtaY6pLH0TW90Lo**" --data-binary "{\"type\":[{\"target_id\":\"invoice\"}], \"title\":[{\"value\":\"Invoice\"}], \"field_no\":[{\"value\":\"No\"}], \"field_date\":[{\"value\":\"2016-06-15T02:59:52\"}], \"field_product\":[{\"value\":\"Product\"}], \"field_fees\":[{\"value\":\"123.45\"}], \"field_customer_name\":[{\"value\":\"John Doe\"}], \"field_customer_address\":[{\"value\":\"Street\nCity\nPost Code\"}]}" **http://demo1a3.project.experienceit.pl**/entity/node?_format=json

Explanation:

- --user **dev:dev** – login credentials in format username:password
- --header "X-CSRF-Token: **WkCkMJrdGifL9iTeZXDDoMZGxnyVoa8B2AyjqjZlRTA**" – this is the token that we generated. Put YOUR token here.
- --data-binary: here you should put JSON data. Notice the backdslashes: **\"** – I'm making the call from Windows and it was the only way to make the call work. If you are on Unix or OSX you might use single quotes.
- http://demo1a3.project.experienceit.pl – website url

6. Configure Invoice permalink

[TODO]

7. Invoice template

[INCOMPLETE] Inside this project there is “template” folder - with invoice template inside. Copy whole “template” folder into front-end theme. I’m using “Bartik” theme so I would copy “template” folder to /core/themes/bartik. There is already “template” folder, so invoice template will be added to other bartik templates.