**Starting Okapi**

First open Okapi and run the Command

java -Dstorage=postgres -jar okapi/okapi-core/target/okapi-core-fat.jar dev

\*\*-Dstorage=postgres //to store data in database after running

In the end you should have Okapi running in a console window, and listening on port 9130. You can verify it is running by listing the tenants:

curl -w '\n' -D - http://localhost:9130/\_/proxy/tenants

HTTP/1.1 200 OK

Content-Type: application/json

X-Okapi-Trace: GET okapi-2.16.1-SNAPSHOT /\_/proxy/tenants : 200 11141us

Content-Length: 105

[ {

"id" : "supertenant",

"name" : "supertenant",

"description" : "Okapi built-in super tenant"

} ]

**Create Tenant:**

By default Okapi create tenant it’s called “Supertenant”

First we need to create Tenant to used it for examples.

First tenant:

cat > /tmp/okapi-tenant1.json <<END

{

"id": "tenant1",

"name": "First Tenant",

"description": "First Tenant"

}

END

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d @/tmp/okapi-tenant1.json \

http://localhost:9130/\_/proxy/tenants

Second Tenant

cat > /tmp/okapi-tenant2.json <<END

{

"id": "tenant2",

"name": "second tenant",

"description": "Our Own second tenant"

}

END

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d @/tmp/okapi-tenant2.json \

http://localhost:9130/\_/proxy/tenants

**Required modules**

## Order of operations

Basically we just need to declare, deploy, and enable the modules declared above, and load some data into them.

Declaring and deploying the modules can be done in any order, but we have to be careful with the order of enabling them. Specifically, we may not enable mod-authtoken until the very end, when we have all the other modules in place and loaded with data, as mod-authtoken will not let us finish the process, locking us out of our own system.

* *ModuleDescriptor*

See the built target/ModuleDescriptor.json for the interfaces that this module requires and provides, the permissions, and the additional module metadata.

* LaunchDescriptor

The launchDescriptor tells Okapi how this module is to be started and stopped. In this version we use a simple exec command line.

* **Mod-users**
* **Introduction**

[mod-users](https://github.com/folio-org/mod-users) – Provides user management

Module to provide central user management for FOLIO systems.

* **fetch and compile “Build” the modules**

git clone --recursive https://github.com/folio-org/mod-users

cd mod-users

git checkout master # v15.1.0 is okay

mvn clean install

cd ..

The install rule also runs tests. Tests should not fail. If they do then please report it, and in the meantime fall back to:

mvn install -DskipTests

after building run module by okapi without docker by default its use docker , in moduleDescriptor change the LaunchDescripptor from launch by docker to launch by exec command for okapi.

"exec" : "java -Dport=%p -jar ../mod-users/target/mod-users-fat.jar"

Then Post it to Okapi

### Declaring the modules

The module descriptors are generated from a template during the build process. Where ever they come from, we just need to POST them to Okapi.

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-users/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

Now we need to deploy the module.  Here we simply use the default deployment descriptors that are provided by each module for its development purposes, so there is some tweaking required.

cat mod-users/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-users.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-users.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable the modules for our tenants. We do not need to specify the version number here, Okapi will choose the latest (and only) version we have declared.

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-users"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-users"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-login**
* **Introduction**

[mod-login](https://github.com/folio-org/mod-login) – Handles username/password login.

This module is responsible for verifying the user's identity and issuing a valid JWT that can be used for system access. The implementation of this module may vary (username/password, SAML, OAuth, etc.), and it is possible for more than one Authentication module to exist in a running system. The default implementation uses a simple username and password for authentication.

* **fetch and compile “Build” the modules**

git clone --recursive https://github.com/folio-org/mod-login

cd mod-login

git checkout master # v4.0.1 is okay

mvn clean install

cd ..

### Declaring the modules

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-login/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-login/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-login.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-login.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-login"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-login"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-permissions**
* **Introduction**

[mod-permissions](https://github.com/folio-org/mod-permissions) – Handles permissions and permissions/user associations.

This module stores permissions and associations between permissions and users. It also maintains a hierarchy of permissions and sub-permissions, allowing for permissions to act as roles, rather than simple bits. It is used primarily by the Authtoken module, though it is possible that some Authentication implementations may have reason to make calls to the Permissions module as well.

* **fetch and compile “Build” the modules**

git clone --recursive https://github.com/folio-org/mod-permissions

cd mod-permissions

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the modules

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-permissions/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-permissions/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-permissions.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-permissions.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-permissions"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-permissions"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-authtkoen**
* **Introduction**

[mod-authtoken](https://github.com/folio-org/mod-authtoken) – Filtering requests based on JWT tokens.

This module is responsible for filtering all proxy traffic and checking for a valid token. In addition, it is responsible for retrieving the permissions for a given user and making decisions regarding access based on user permissions and defined requirements for a given path. It provides a token creation endpoint that privileged modules (such as Authentication) may make use of.

* **fetch and compile “Build” the modules**

git clone https://github.com/folio-org/mod-authtoken

cd mod-authtoken

git checkout master # v1.4.1 is okay

mvn clean install

cd ..

### Declaring the modules

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-authtoken/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-authtoken/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-authtoken.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-authtoken.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-authtoken"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-authtoken"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-codex-mux**
* **Introduction**

[mod-codex-mux](https://github.com/folio-org/mod-codex-mux) – Codex Multiplexer.

Codex Multiplexer backend module

* **fetch and compile the module**

git clone --recursive https://github.com/folio-org/mod-codex-mux

cd mod-codex-mux

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the module

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-codex-mux/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-codex-mux/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-codex-mux.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-codex-mux.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-codex-mux"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-codex-mux"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-codex-mux**
* **Introduction**

[mod-codex-inventory](https://github.com/folio-org/mod-codex-inventory) – Codex wrapper for local inventory.

Codex Wrapper for Inventory

* **fetch and compile the module**

git clone --recursive https://github.com/folio-org/mod-codex-inventory

cd mod-codex-inventory

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the module

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @ mod-codex-inventory/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-codex-inventory/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-codex-inventory.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-codex-inventory.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-codex-inventory"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":"mod-codex-inventory"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-codex-mock**
* **Introduction**

[mod-codex-mock](https://github.com/folio-org/mod-codex-mock) – Codex mock module - for testing and development..

Codex mock module - for testing and development

This module provides a very minimal implementation of the codex interface. It has a small number of hard coded records, that it can list, and provide by the id.

The module loads a hard-coded list of instances into its database at tenant-init time. These can be queried and sorted according to the usual RMB conventions.

* **fetch and compile the module**

git clone --recursive https://github.com/folio-org/mod-codex-mock

cd mod-codex-mock

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the module

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @ mod-codex-mock/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-codex-mock/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-codex-mock.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-codex-mock.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-codex-mock"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-codex-mock"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-codex-ekb**
* **Introduction**

[mod-codex-ekb](https://github.com/folio-org/mod-codex-ekb) – Codex wrapper for the EBSCO knowledge base

Codex wrapper for the EBSCO knowledge base.

* **fetch and compile the module**

git clone --recursive https://github.com/folio-org/mod-codex-ekb

cd mod-codex-ekb

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the module

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @ mod-codex-ekb/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-codex-ekb/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-codex-ekb.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-codex-ekb.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-codex-ekb"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-codex-ekb"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-erm-usage**
* **Introduction**

[mod-erm-usage](https://github.com/folio-org/mod-erm-usage) – Store ERM usage statistics and access data to these statistics.

* **fetch and compile the module**

git clone --recursive https://github.com/folio-org/mod-erm-usage

cd mod-erm-usage

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the module

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @ mod-erm-usage/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-erm-usage/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-erm-usage.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-erm-usage.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-erm-usage"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-erm-usage"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **Mod-licenses**
* **Introduction**

[mod-licenses](https://github.com/folio-org/mod-licenses) – Upload, manage and analyze licenses.

* **fetch and compile the module**

git clone --recursive https://github.com/folio-org/mod-licenses

cd mod-licenses

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the module

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @ mod-licenses/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-licenses/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-licenses.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-licenses.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-licenses"}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-licenses"}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **mod-inventory-storage**
* **Introduction**

[mod-inventory-storage](https://github.com/folio-org/mod-inventory-storage) – Persistent storage to complement the inventory module.

FOLIO compatible inventory storage module.

Missing module to mod-codex-inventory, Provides PostgreSQL based storage to complement the inventory module. Written in Java, using the raml-module-builder and uses Maven as its build system.

* **fetch and compile the module**

git clone --recursive https://github.com/folio-org/mod-inventory-storage

cd mod-inventory-storage

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the module

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @ mod-inventory-storage /target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-inventory-storage /target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-inventory-storage.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-inventory-storage.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-inventory-storage "}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-inventory-storage "}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

* **mod-configuration**
* **Introduction**

[mod-configuration](https://github.com/folio-org/mod-configuration) – Configuration module based on the raml-module-builder and a set of RAML and JSON Schemas backed by a PostgreSQL asynchronous implementation.

Missing for mod-coex-ekb

* **fetch and compile the module**

git clone --recursive https://github.com/folio-org/mod-configuration

cd mod-configuration

git checkout master # v5.2.5 is okay

mvn clean install

cd ..

### Declaring the module

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @ mod-configuration /target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

### Deploying the modules

cat mod-configuration /target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-configuration.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-configuration.json \

http://localhost:9130/\_/discovery/modules

### Enabling modules and loading data

Enable to first tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-configuration "}' \

http://localhost:9130/\_/proxy/tenants/tenant1/modules

Enable to second tenant

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d'{"id":" mod-configuration "}' \

http://localhost:9130/\_/proxy/tenants/tenant2/modules

# Conclusion

So I covered in this documentation the modules and the operations like

“Build, post ,deploy, enable” on the module which created.

**Modules Running and enabled to tenants**

* mod-login
* mod-users
* mod -erm-usage
* mod-codex-inventory
* mod-codex-mux
* mod-inventory-storage

And some modules have problem in enable operation like:

* mod-permissions
* mod-authtoken
* mod-codex-mock
* mod-codex-ekb