Viadat – installation (admin) guide

V1.0

Charles University, Prague, CZ

December 2019

Quick start guide

To get everything running should be fairly simple, at least for demonstration[1] purposes. The viadat package (http://hdl.handle.net/11234/1-3141 or https://github.com/ufal/viadat which contains all the modules) contains a docker-compose.yml. So the only prerequisites for the demonstration should be docker (docker.com) and docker-compose. To run it, simple `docker-compose up` should suffice as all the necessary configuration is provided. The repository is available at http://localhost:8080/xmlui the viadat frontend at http://localhost:3000/
The above steps will also create demo users. An admin account in the repository with login dspace@lindat.cz and password "dspace". A test account in viadat (login test, password test).
The system comprises of a postgresql database, a tomcat servlet container running the repository webapp, a react app for the "viadat" frontend, mongodb and a flask app that provides a rest api to the frontend. You need to assign enough resources for all this to run. Especially the forced alignment is memory heavy. Borderline amount is 5GB of memory, this will somehow run, if the files you are trying to align are not too big, the alignment might finish too, but you may occasionally see out of memory errors. The more the better.

Inspect the docker-compose and the Dockerfiles (also those at https://github.com/ufal/viadat-repo-docker which are used to build the repository images) to get a sense of what it takes to get the infrastructure up and running. Few highlights follow

Viadat

Inspect the mongo setup in backend/settings.py and the repository setup in repository.conf (there's the url, user and password. This is the user under which records are exported from viadat into the repository, in this case it's the administrator's account; so the setup is not polluted with configuring correct access rights).
Additional viadat users can be created with `python3 -m backend create-user test --password test`.

Repository

For the repository there is also a comprehensive guide at https://github.com/ufal/clarin-dspace/wiki and you should consult that as going through all the dspace.cfg and local.properties is out of the scope of this document.

What differs from a stock (clarin-)dspace repository installation is that in this version the repository structure is populated with one community and two collections, one collection is for items describing narrators the other is for items describing interviews. There are also two

[1] That means the result will be unoptimized, unsafe (it will expose parts of the system that should never be exposed to the public) and in certain sense unfinished (handle server is not configured/running, shibboleth authentication is not working etc.). So this should rather serve as a showcase of what components are where, not as a recommendation to configure your infrastructure this way.

template items created. Do not remove these. The collections and the items play a role in how the system functions, e.g. that you can see the interviews under a narrator or that you can use the facets on different items (narrators + interviews) and still get some results (there's a solr join query). If you add an item under a new collection (that you've created) it's very likely that something will break. Generally the users should use the add narrator/interview links to create new items in the repository (or interact with it only through the viadat app).

Eventually (depending on how you go about setting the production version), when you setup the handle server, you might need to update the collection handles in dspace/config/item-submission.xml to map the correct submission workflows.

Getting additional information

Many of the core parts of the repository system remain unchanged from the stock (clarin-)dspace. So to handle common tasks like changing what's displayed on the item-view page, configuring new browse index, adding users etc. the two following wikis should provide more than enough information.

https://github.com/ufal/clarin-dspace/wiki

https://wiki.lyrasis.org/display/DSDOC5x/DSpace+5.x+Documentation