

Installation

- [NORA inside a Docker](#)
- [Administration Backend](#)
- [Segmentation Assistant for ROItool \(nnInteractive\)](#)

NORA inside a Docker

Install Docker on your host system

First, install docker, git and jq on your host system

```
sudo apt-get install docker.io git jq
```

There might be an issue with the DNS address in docker. To check this, run

```
docker run busybox nslookup google.com
```

If the host cannot be reached, grep the address of your DNS server

```
nmcli dev show | grep 'IP4.DNS'
```

create the file `/etc/docker/daemon.json` and insert the address as

```
{  
  "dns": ["yourDNSip", "8.8.8.8"]  
}
```

then restart docker

```
sudo service docker restart
```

go back to the busybox above and check if it works now.

Now, you should add your user to the docker group, otherwise you have not to *sudo* all NORA commands

```
sudo adduser <username> docker
```

!!! you have to re-login to make these changes apply !!!!!

Clone the git repository

NORA was historically named DPX. Note that this name might still often be used in the following. Clone the repository into to a temp folder, or to your favorite program place.

Let's say we clone to your home directory into ~/nora

```
git clone https://<yourname>@bitbucket.org/reisert/dpx.git ~/nora
```

If want to use NORA in a multi-user environment you might want to create a specific group (we usually use *dpxuser*) and change the group of that directory (recursive) accordingly. In any case, you have to set the *s*-flag and some ACLs recursively to that dir (*you might have to run this as sudo. In this case make sure to apply it to the correct ~/nora directory !!*)

```
find ~/nora -type d -exec chmod g+s {} +
```

If want to manage rights by ar group user

```
chgrp -R NORA ~/nora # if you manage your right via usergroup NORA./dp
```

nonetheless

```
setfacl -R -d -m g::rwx ~/nora
```

Initial configuration

Go into the nora directory and run ./install

This will create some local config files in the "conf" directory (for details see separate section below).

NORA has 3 modules

- **Frontend:** *webserver with image viewer*
- **DICOM node** to receive images from a PACS
- **Backend:** processing module (depending on your preferences matlab or nodejs and slurm or SGE based)

By default, all three modules are enabled. To run NORA in this default configuration edit main.conf and set at least MATLBPATh: <your-path-to-matlab>

If you are behind a proxy, it might also be necessary to set the proxy (maybe even with user/password)

```
DOCKER_http_proxy:" ... "
```

Also set the user Nora should act as in the main.conf

```
DPXUSER: " ... "  
DPXGROUP: " ... "
```

Build the docker image

Your main function to control NORA is dpxcontrol. You can always run

```
./dpxcontrol
```

to get more help. To build the DOCKER image, first run

```
./dpxcontrol docker build
```

This will take a while. If everything went well, start all modules with

```
./dpxcontrol start
```

If something went wrong during installation / starts, there might be old 'zombie' containers. In this case you will be suggested docker commands to remove them. You can also start with --forceto autoremove old NORA containers.

To check the status, now use

```
./dpxcontrol status
```

If at least the first point, Docker, is running nicely, you should be able to log into the Webinterface. (see below)

Log into the Webinterface

Now, open a web-browser and go to localhost:81. Default login is

```
user:    root  
password: dpxuser
```

There are several options to create new users. You can either connect to an existing LDAP server, or just create users based on the internal user management of NORA (see Administration section for details).

Troubleshooting and testing

Most log files are written into

```
<path-to-nora>/var/syslogs/
```

These can also be seen from the *admin* dialog at the top of the webinterface. **If the daemon is not starting (or stopping again, red status) check the daemon.log for more info.** In case of a license error, maybe you have to forward your MAC address to Docker (see main.conf for more info)

For other system parts, there are also some test functions

```
./dpxcontrol test [slurm | email | more_to_be_programmed]
```

Upgrade the database

When your daemon is running nicely, it might be necessary to upgrade the initial database with

```
./dpxcontrol matlab updatedb
```

Autostart on system startup

To automatically start NORA when your computer starts, you can for example add `<path-to-nora>/dpxcontrol start --force` to your `/etc/rc.local`

Slurm setup

A proper configuration of slurm is a science on its own. Ask google for more information. If you are running slurm inside docker (default), the `conf/slurm.conf` is used and you do not have to do too much. Otherwise, if docker is installed outside (this should be the correct practice) you have to install via

```
sudo apt-get install slurm-wlm
```

on a deb-system and set

```
DOCKER_run_daemon_in_docker: 0,
```

in the `<path-to-nora>/conf/main.conf` file. For configuring slurm itself there is a configurator interface via html, which you usually find here

```
/usr/share/doc/slurmctld/slurm-wlm-configurator.easy.html
```

which generates a conf file `/etc/slurm-llnl/slurm.conf` where you have to replace the hostname by the machine you are running NORA on. Further, the partitions have to be specified. NORA has as default two partitions

(computing queues) `DPXproc` and `DPXimport`, which have to be specified in the `slurm.conf` like in this

example:

```
# COMPUTE NODES
NodeName=hostname CPUs=8 State=UNKNOWN
PartitionName=DPXproc Nodes=hostname Default=YES MaxTime=INFINITE State=UP
PartitionName=DPXimport Nodes=hostname Default=YES MaxTime=INFINITE State=UP
```

Alternatively, if you have an existing SLURM running with predefined partitions, you can change the partitions (queues) in NORA's configuration in `<path-to-nora>/conf/main.conf` (SLURM_QUEUES)

Also consider

```
./dpxcontrol test slurm
```

for testing whether your slurm configuration is working.

General Configuration

All configuration files are located in `<path-to-nora>/conf` directory. The configuration files are

- `main.conf` - includes all major configurations, responsible for
 - Location of external programs (MATLAB etc)
 - DOCKER port forwarding
 - Daemon behaviour
 - Import behavior (dicoms etc)
 - Mysql information
 - signin/signup behaviour, LDAP configuration
 - Backup behavior
- `pacs.conf` - includes all information how to connect to external dicom nodes
- `routes.conf` - ip-addresses of DB, and routes of mount dirs (depending on hostnames different mount locations are possible)
- `slurm.conf` - slurm configuration
- `smail.conf` - mail configuration

Administration Backend

[image.1600352934446.png](#)

Command line Interface (BASH)

You need a PATH to `<path-to-nora>/src/node`

Call

`nora`

any get help

```
NORA -  
backend
```

```
usage: nora [--parameter]
```

```
[value]
```

```
--project (-p)
```

```
[name]
```

```
    all preceding commands refer to this  
project
```

```
MANAGEMENT
```

```
--sql (-q) [sqlstatement] [--
```

```
csv]
```

executes SQL statment and return result as json (or as csv if option is given)

`--add (-a) [file]`

...

adds list of files to project

option: `--addtag tag1,tag2,...` (tags the files)

`--del (-d) [file]`

...

deletes list files

`--del_pat (-dp) [patient_id] [patients_id#studies_id]`

...

deletes list of patients/studies

`--out (-o) [filepattern]`

computes absolute outputfilepath from filepattern (see description of filepattern below)

`--pathout (-pa) [filepattern]`

computes absolute outfilepath from filepattern

`--select (-s) [--json (-j)] [--byfilename]`

`[filepattern]`

returns files matching filepattern, if `--json` is given, files are return as json

`filepattern`

if `--byfilename` (together with `--json`) is given, files are restructered by filename

`--addtag tag1,tag2,... -s [filepattern]`

add tag to file

`--rmtag tag1,tag2,... -s [filepattern]`

removes tags fromfile

`--launch (-l)`

[json]

launches batch from json

description

--createproject (-c) [name] [module]

creates a project with [name] from [module] definition

--admin [parameter1] [parameter2] ...

calls the admin tools (see nora --admin for more help)

PROCESSING

--launch json

launches a NORA job given by json

options: --throwerror

--handle_job_result (emits DONE signal, when ready)

--launchfile jsonfile

same as --launch but json given by fileref

--launch_autoexec psid

launches the autoexecution pipeline of the project for the patient specified by psid

--import srcpath

imports dicoms located in sourcepath to given project

options: --autoexec_queue (if given autoexecuter is initiated, use DEFAULT to select default autoexecution queue given in config)

--handle_job_result (emits DONE signal, when ready)

--extrafiles_filter: if there are non-dicom files in your dicomfolder and you want to add them to your patient folder, then, give here a comma-separated list of accepted extensions (e.g. nii,hdr,img)

--patients_id PIZ (overwrite patients_id of imported data with PIZ)

--studies_id SID (overwrite studies_id of imported data with SID)

--pattern REGEX (use REGEX applied on srcpath of dicoms to overwrite

patients_id/studies_id, example:

--pattern "(?<studies_id>[\w\-.]+)\V(?<patients_id>\w+)\$"

OTHER

--exportmeta [filepattern]

exports meta data as csv. Meta content from files matching [filepattern] are exported

options: --keys a comma separated list of key-sequences, which are exported [optional]

a key sequence is given by a dot-separated list, where the first key refers to the name of the json-file (as multiple jsons might be contained in the query). A wildcard * is possible select multiple keys at once

examples:

nora -p XYZ --exportmeta '*' 'nodeinfo.json' // gathers also patient info

```
nora -p XYZ --exportmeta '*' 'META/radio*.json' --keys '*.mask.volume,*.mask.diameter'
```

MATLAB Interface

To setup NORA for MATLAB you change to the folder

```
<path-to-nora>/src/matlab
```

at the MATLAB prompt and start `DPX_startup`. This sets up all necessary paths and the database connection. To test the DB connection, make a call to `DPX_Project`. This should list all created projects (initially only the default project). There are a variety of management commands to manage users/project/data etc. Here a list of important ones:

- General
 - `DPX_startup` - sets up NORA
 - `DPX_Project` - choose a project
 - `DPX_getCurrentProject` - get the current project information
 - `DPX_SQL_createUser` - create a user
 - `DPX_SQL_createProject` - create a project
- File selection
 - `DPX_getOutputLocation` - compute the path to certain output
 - `DPX_selectFiles` - select files based on pattern
 - `DPX_getFileInfo` - get additional file information (no database involved)
 - `DPX_SQL_getFileInfo` - get information from database for this file
- Processing
 - `DPX_demon` - start the processing daemon
 - `DPX_debug_job` - execute a debug job at the MATLAB command prompt

`DPX_SQL_clearGridCmds` - clear all jobs from the pipeline

The Database

There is one general database scheme containing information about users, projects, jobs, settings etc.

You can see below the table schemes.

- **Projects/User management**

- projects
- users
- rights - who can read which project in which role

- **Batch/Jobs**

- gridjobs - contains submitted and running jobs (mirror of qstat or queue)
- Commands - contains jobs submitted from web interface, but not yet submitted to resource management system.

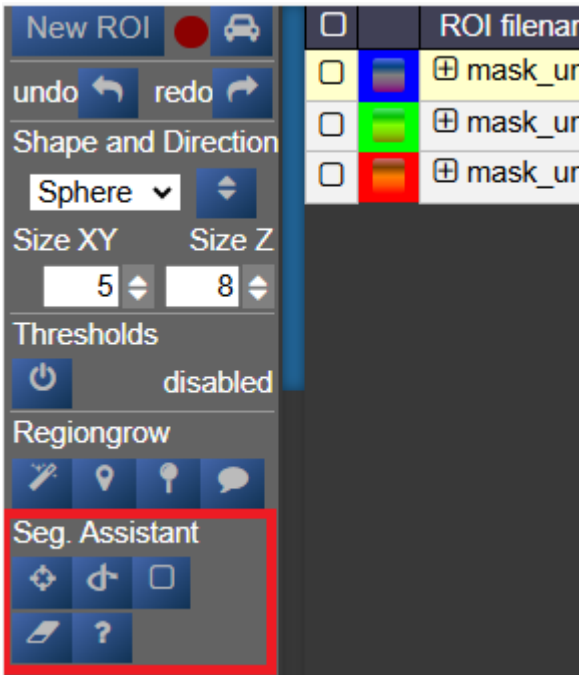
- **Miscellaneous**

- json - various types of jsons (saved batches, shares, settings, states etc.)
- notifications
- error_js

[image.1600353314610.png](#)

[image.1600353338398.png](#)

Segmentation Assistant for ROItool (nnInteractive)



This assistant allows users to create and refine 3D ROIs using interactions. Images, masks, and successive interactions are processed by a deep learning model hosted on a remote server.

The current version of the model is **nnInteractive** (Isensee, Rokuss, Krämer et. al., 2025).

[nnInteractive repository](#)

1. Usage

First, **select an ROI** from the ROI list to make it the active target for segmentation.

Point Interaction

- Click the **Point** icon (crosshairs) to activate Point mode
- **Left-click** on the image to add a **positive** point (to include an area)

- **Right-click** on the image to add a **negative** point (to exclude an area)
- The segmentation will be updated immediately. Click the icon again to deactivate
- To add multiple points before updating the segmentation, **hold down the Q key** while adding points. Release the Q key to send all accumulated points at once

Scribble Interaction

- Click the **Scribble** icon to activate Scribble mode and draw a scribble on the image
- Click the **+** button to run a **positive** interaction (add areas)
- Click the **-** button to run a **negative** interaction (remove areas)
- Scribbles are cleared after each interaction, allowing you to add more

BBox Interaction

- Click the **BBox** icon (square) to activate Bounding Box mode and draw a box on the image
- Click the **+** button for a **positive** interaction (segment inside the box)
- Click the **-** button for a **negative** interaction (exclude area in the box)
- The box is cleared after each interaction

Reset Interactions

- When editing the same ROI, all previous interactions are kept in memory server-side to guide the segmentation
- Click the **Reset** icon (eraser) to erase all previous interactions and start fresh

2. Server Setup

The segmentation server must be running, typically on a machine with a GPU. First, navigate to the `src/python/segmentation_assistant_server` directory and launch the `setup.sh` script (requires internet access):

bash

```
cd src/python/segmentation_assistant_server
./setup.sh
```

This will create a Python virtual environment and install the necessary dependencies.

Next, start the server from a computing node with:

bash

```
cd src/python/segmentation_assistant_server
./start.sh
```

3. Client Configuration

Once the server is running, it will log its port. You must update this client's configuration to point to that address.

Edit the file: `conf/segmentationserver.conf`

Set the `REMOTE_SEGMENTATION_BASE_URL` to the server's address, for example:

json

```
{"REMOTE_SEGMENTATION_BASE_URL": "http://<SERVER_IP>:PORT"}
```

Important: You must restart Nora for the changes to take effect.