Manual for usage of *RUTHELDE* command line tools

V4.0 2024_07_12
R. Heller

Usage of RUTHELDE from terminal

First the RUTHELDE server must be started on any PC within the network.

It is started by:

java -jar Ruthelde_Server.jar

If you are running Ruthede server on a Unix machine make sure that port **9090** is open!

After starting the server it starts listening automatically and forever. The console should look like this:

```
it12:Server rene$ java -jar Ruthelde_Server.jar
Ruthelde - Server V4.0
```

Once the server is running the client can be started on any PC within the network by typing in:

java -jar Ruthelde_Client.jar requestType input.json output.json 192.168.12.12

- requestType can be SIMULATE or OPTIMIZE
- input.json is the path to the input file.
- output.json is the path to the output file.
- 192.168.12.12 is the IP address of the PC the server is running on.

If the server is running on the same machine like the client you can pass “localhost” as IP address.
Once the command is executed the console will spit out the following:

In case of a SIMULATION request:

```
it12:Client rene$ java -jar Ruthelde_Client.jar SIMULATE sim_input.json sim_outp
ut.json localhost
Ruthelde - Client V4.0
Ruthelde_Client: Loading input file.
Ruthelde_Client: Connecting to server.
Ruthelde_Client: Sending input to server.
Ruthelde_Client: Received simulation result.
Ruthelde_Client: Writing output file.
it12:Client rene$
```

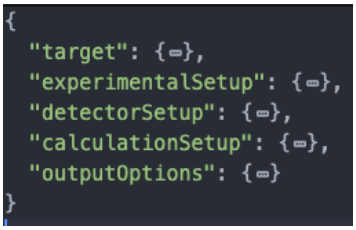
In case of an OPTIMIZATION or OPTMIZATION_MS request

```
it12:Client rene$ java -jar Ruthelde_Client.jar OPTIMIZE opt_input.json opt_outp
ut.json localhost
Ruthelde - Client V4.0
Ruthelde_Client: Loading input file.
Ruthelde_Client: Connecting to server.
Ruthelde_Client: Sending input to server.
Ruthelde_Client: Received intermediate DE result: 1
Ruthelde_Client: Received intermediate DE result: 2
Ruthelde_Client: Received intermediate DE result: 3
Ruthelde_Client: Received intermediate DE result: 4
Ruthelde_Client: Received intermediate DE result: 5
Ruthelde_Client: Received intermediate DE result: 6
Ruthelde_Client: Received intermediate DE result: 7
Ruthelde_Client: Received intermediate DE result: 8
Ruthelde_Client: Received intermediate DE result: 9
Ruthelde_Client: Received intermediate DE result: 10
Ruthelde_Client: Received optimisation result.
Ruthelde_Client: Writing output file.
it12:Client rene$
```

While the client will be closed after the simulation is done, the server remains open and waits for the next request forever.

Formats of RUTHELDE input & output files

The input file for a SIMULATION request must be in JSON format like this:

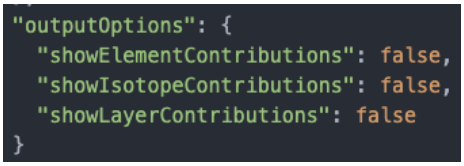


The structs “target”, “experimentalSetup” and “detectorSetup” are the same like in any conventional Ruthelde input file.

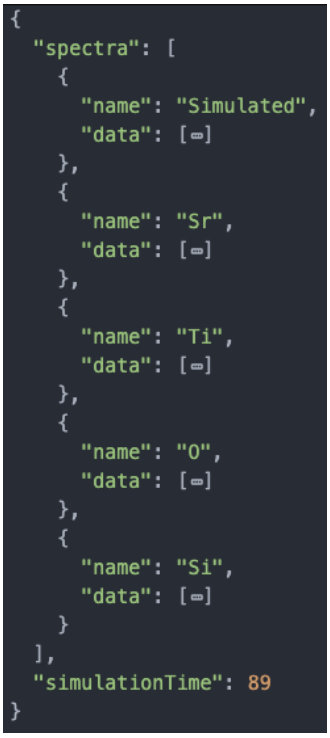
The struct “calculationSetup” is basically the same like in standard Ruthelde input file with the need for one additional parameter field called “numberOfChannels”. This one holds the length of the spectrum that will be simulated.

“stoppingData” can be an empty String. However in that case” stoppingPowerCalculationMode” must be set to “ZB”. The path to the stopping data is on the server! This means that the file must be on the server. Best practice would be to have it just in the root folder of the Ruthelde_Server.jar. The same counts for any cross section data file!

The struct “outputOptions” defines what degree of detail the output file will have. Usual if you are just interested in the (sum) simulated spectrum you can have set all detail off.

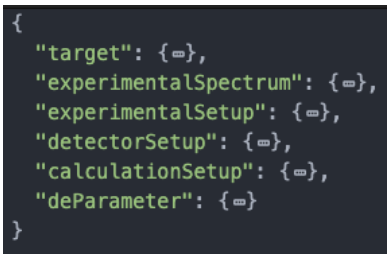


The output file of a SIMULATION request is a JSON file with the following structure:



It's basically an array of structs. The first entry is the total simulated spectrum and the following entries represent the individual elemental spectra (if selected in the "outputOptions" of the input file). Each "data" array has the length specified in the input file. "simulationTime" is given in milliseconds.

The input file for an OPTIMIZATION request must be in JSON format like this:



The structs "target", "experimentalSetup", "detectorSetup" and "deParamter" are the same like in any conventional Ruthelde input file.

The struct "calculationSetup" is basically the same like in standard Ruthelde input file with the need for one additional parameter field called "numberOfChannels". This one holds the length of the spectra for simulation so it needs to be the same like the length of the experimental spectrum!

"stoppingData" can be an empty String. However in that case" stoppingPowerCalculationMode" must be set to "ZB". The path to the stopping data is on the server! The server will find the file is it exists there, so you just need to pass the file name not a path!

The struct "experimentalSpectrum" consist of two fields. One String giving it a name and one array of data.

```
"experimentalSpectrum": {
  "name": "SrTiO3.json",
  "data": [
    33.94730705548802,
    48.46506967371671,
    ...
    0.0,
    0.0
  ]
},
```


The output file of an OPTIMIZATION request is a JSON file with the following structure:


```
{
  "target": {…},
  "charge": 24.6571265484325,
  "resolution": 15.576739946849802,
  "factor": 1.4978727229797588,
  "offset": 26.25813918536818,
  "fitness": 93.10687637190945,
  "optimizationTime": 11.512
}
```


It basically spits out the optimized parameters according to their name. “optimizationTime” is given in seconds.