Ops4J

This is a reference guide for Ops4J.

Operations

Operations provide single purpose utilities which can be combined in pipelines to accomplish more complex task. This section will document each of these utilities/operations.

backlog

> Service a backlog of task concurrently.

usage:

```
backlog [-C=<view>] [-iqt=<inputQueueType>] [-L=<logLevel>]
               [-max=<maxThreads>] [-min=<minThreads>] [-N=<name>]
               [-oqt=<outputQueueType>] <commands>

Run operations using a backlog feeding concurrent workers.

Additional CLI Options: [-hHP] [-D=<dataSource>] [-O=<outputType>]
                        [-S=<serializationType>] [-LL=<String=LogLevel>]...
```

help:

```
backlog -H
Usage: backlog [-C=<view>] [-iqt=<inputQueueType>] [-L=<logLevel>]
               [-max=<maxThreads>] [-min=<minThreads>] [-N=<name>]
               [-oqt=<outputQueueType>] <commands>

Run operations using a backlog feeding concurrent workers.

      <commands>          The commands to be executed.

      -iqt, --input-queue-type=<inputQueueType>
                          The input queue type.
      -max, --max-threads=<maxThreads>
                          The maximum number of threads.
      -min, --min-threads=<minThreads>
                          The minimum number of threads.
      -oqt, --output-queue-type=<outputQueueType>
                          The output queue type.

Class: org.ops4j.op.Backlog
```

examples:

```
# Insert example here...
```

bash-exec

> Execute a bash script.

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

bash-filter

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

bash-source

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

benchmark

> Benchmark something.

usage:

```
Usage: benchmark [<transactionThreshold>]

Benchmark something.
```

help:

```
benchmark [<transactionThreshold>]

Benchmark something.

      [<transactionThreshold>]
                          The number of transactions between reports.  Default =
                            0 = No progress reports

Class: org.ops4j.op.Benchmark
```

examples:

```
# Benchmark the creation of 100,000 people records.
map -D 100000 /=gen-person: | benchmark 10000 -O none
```

disruptor

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

filter

> Filter a stream based upon inclusion and exclusion conditions.

usage:

```
Usage: filter [-i=<includes>] [-x=<excludes>]

Filter records.
```

help:

```
filter [-i=<includes>] [-x=<excludes>]

Filter records.

  -i, --includes=<includes>
                          The includes.
  -x, --excludes=<excludes>
                          The excludes.

Class: org.ops4j.op.Filter
```

examples:

```
# Generate 100 people, keeping only females
map -D 100 /=gen-person: | \
  filter -i 'match(/sex -pattern="Female")'

# Generate 100 people, discarding females
map -D 100 /=gen-person: | \
  filter -x 'match(/sex -pattern="Female")'

# Generate 100 people, filter on Asian Females
map -D 100 /=gen-person: | \
  filter -i 'match(/sex -pattern="Female")' \
    -i 'match(/race -pattern="Asian")'
```

flatten

> Flatten a structured payload.

usage:

```
flatten

Flatten a nested JSON.
```

help:

```
flatten

Flatten a nested JSON.

Class: org.ops4j.op.Flatten
```

examples:

```
echo '{"student":{"name":"bob", "grades":[90,100]}}' | flatten
```

groovy-template

> Render a groovy template.

usage:

```
Usage: groovy-template [-C=<view>] [-L=<logLevel>] [-N=<name>]
                       [-t=<templatePath>]

Render a Groovy template.
```

help:

```
Usage: groovy-template [-C=<view>] [-L=<logLevel>] [-N=<name>]
                       [-t=<templatePath>]

Render a Groovy template.

  -t, --template=<templatePath>
                         The template.

Class: org.ops4j.groovy.op.GroovyTemplate
```

examples:

```
cat data.json | groovy-template tps-report.gt
```

http-client

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

http-get

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

http-server

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

http-view

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

jdbc-create

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

jdbc-drop

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

jdbc-insert

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

jdbc-stream

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

jhead

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

logphases

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

logtest

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

map

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

model-usl

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

mongo-insert

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

mongo-stream

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

noop

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

op-info

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

pause

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

pipeline

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

poe

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

print

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

remove-nulls

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

route

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

shell

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

shuffle

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

simulate

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

smile:cluster

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

sort

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

stream

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

stream:lines

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

tail

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

unwind

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

viz-flow

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

viz-sequence

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

viz-tree

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

vw

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

web-view

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

wss

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

xray

> Short description

usage:

```
# USAGE
```

help:

```
# HELP
```

examples:

```
# EXAMPLE
```

Node Operations

array

```
map -D 1 /=gen-person: | map '/data=array(/first /last /age)'
```

array-add

broken

avg

```
map -D 10 /n=int: | map /=/ /avg=avg:/n
map -D 10 /n=int: | map /=/ /avg='avg(/n)'
```

choose

```
map -D 10 /c='choose(a b c)'
```

cos

```
map -D 10 /n=seq: | map /=/ /cos='cos(/n)'
```

decrypt

```
echo '{"msg":"hello world!"}' | map /enc=encrypt:/
{"enc":"+ql7yKVgXJgZqgDCXh8aGI4h5uBfmnTwK5BhvpM2XhA="}

echo '{"enc":"+ql7yKVgXJgZqgDCXh8aGI4h5uBfmnTwK5BhvpM2XhA="}' | \
  map /=decrypt:/enc   {"msg":"hello world!"}
{"msg":"hello world!"}
```

dist

This node operation can be used to generate various distributions.

```
# Generate a uniform distribution from 0 to 100 with precision of 2
map -D 100 '/uniform=dist(-uniform -min=0 -max=100 -precision=2)'

# generate 100 records with a normal distribution with a
# mean of 10 and standard deviation of 5
map -D 100 '/normal=dist(-normal -mean=10 -variance=5)'

# Generate a tseries with 10 values and 9 degrees of freedom.
map -D 10 '/tseries=dist(-tseries -freedom=9)'

# Generate a logistic series with peak mu and scaling factor s
map -D 100 '/logistic=dist(-logistic -s=1 -mu=10)'
```

double

Generate random doubles.

```
map -D 1 /d=double:

map -D 10 /score='double(-min=1.0 -max=100.0 -precision 2)'
```

encrypt

```
echo '{"msg":"hello world!"}' | map /enc=encrypt:/
{"enc":"+ql7yKVgXJgZqgDCXh8aGI4h5uBfmnTwK5BhvpM2XhA="}

echo '{"enc":"+ql7yKVgXJgZqgDCXh8aGI4h5uBfmnTwK5BhvpM2XhA="}' | \
  map /=decrypt:/enc   {"msg":"hello world!"}
{"msg":"hello world!"}
```

eval

```
echo '{"x":1,"y":2}' | \
  map /=/ '/result=eval(-x="return x+y")'
```

gen-address

```
map -D 1 /address=gen-address:
```

gen-city

```
map -D 1 /city=gen-city:
```

gen-code

```
map -D 1 /code=gen-code:
```

gen-data

```
map -D 1 /data=gen-data:
```

gen-date

```
map -D 1 /date=gen-date:
```

gen-first

```
map -D 1 /first=gen-first:
```

gen-int-array

```
map -D 1 /array=gen-int-array:
# Generate even numbers from 2-100
map -D 1 '/array=gen-int-array(-s=2 -e=100 -i=2)'
```

gen-key

Generate a cryptographic key.

```
map -D 1 /key=gen-key:
```

gen-last

```
map -D 1 /last=gen-last:
```

gen-lat-long

```
map -D 1 /coordinates=gen-lat-long:
```

gen-name

```
map -D 1 /name=gen-name:
```

gen-person

```
map -D 1 /person=gen-person:
```

gen-phone

```
map -D 1 /phone=gen-phone:
```

gen-state

```
map -D 1 /state=gen-state:
```

gen-text

```
map -D 1 /text=gen-text:
map -D 1 /text='gen-text(-p=###-@@-####)'
```

int

```
map -D 1 /i=int:
map -D 1 /i='int()'
map -D 1 '/i='int()
```

jpath

```
map -D 1 /=gen-person: | map /name='jpath($.last)'
```

keywords

```
stream-lines book.txt | map /keywords=keywords:/lines
```

match

```
echo '{"name":"bob"}' | map /match='match(/name -pattern=bo)'
```

min

```
map -D 10 /n=int: | map /=/ /min=min:/n
```

missing

Generate a missing or null JSON node.

```
map -D 1 /missingValue=missing:
```

normalize

```
stream-lines book.txt | map /n=normalize:/lines
```

now

```
map -D 1 \
  /now=now: \
  /yesterday='now(-offset=-86400000)' \
  /tomorrow='now(-offset=86400000)'
```

null

```
map -D 1 /empty=null:
```

pct

```
map -D 100 /i=int: | map /=/ /pct='pct(/i -p 20 -w 10)'
```

plus

```
map -D 1 /=DELETEME:
map -D 1 /='DELETEME()'
```

random-text

```
map -D 1 /text=random-text:
map -D 5 /text='random-text(-min=10 -max=10)'
map -D 10 /matrix='random-text(-charset=01 -min=10 -max=10)'
```

run

```
map -D 1 /=DELETEME:
map -D 1 /='DELETEME()'
```

sentences

```
stream-lines book.txt | map /s=sentences:/lines
```

seq

```
map -D 100 /n=seq:
```

sin

```
map -D 10 /n=seq: | map /=/ /sin='sin(/n)'
```

slope

```
map -D 10 /i=int: | map /=/ /slope=slope:/i
```

split

```
echo '{"names":"jim,john,sue,bob"}' | map /n=split:/names
```

text

```
map -D 1 /message='text("hello world")'
```

to-double

```
echo '{"msg":"123.456"}' | map /d=to-double:/msg
```

to-float

```
echo '{"msg":"123.456"}' | map /f=to-float:/msg
```

to-int

```
echo '{"msg":"123"}' | map /i=to-int:/msg
```

to-lower

```
echo '{"msg":"HELLO WORLD"}' | map /msg=to-lower:/msg
```

to-month

```
map -D 10 /i=int: | map /month=to-month:/i
```

to-string

```
echo '{"pi":3.14}' | map /pi=to-string:/pi
```

to-upper

```
map -D 10 /=gen-person: | map /NAME=to-upper:/first
```

words

```
stream-lines book.txt | map /words=words:/
```