jobgen - Job File Generator

For jobgen v0.5.

jobgen is under heavy development. This document may not be fully up to date.

Contents

Introduction

Jobgen accepts directives for a virtualized queueing system and generates directives for a target queue system.

Jobgen is useful for the vast majority of situations in which a basic set of job requirements suffice. jobgen can also be used to validate job requests and apply job file processors to transform the request before the final result is generated.

Jobgen provides generators for Gridengine, PBS, and PBSPro. These generators can be enhanced to support site-specific needs.

Usage

```
usage: jobgen [<options>] [-p <profile>] -j <jobfile>
       jobgen [-p <profile>] [-j <jobfile>] --show-meta[=<fmt>]
       jobgen [-p <profile>] [-j <jobfile>] --show-profile[=<fmt>]
       jobgen [-p <profile>] [-j <jobfile>] --show-profile-flat
       jobgen [<options>] [-p <profile>] --show-request[=<fmt>] -j <jobfile>
       jobgen -l|--list
       jobgen -h|--help

Generate job file targeted for specific queueing system.

Where:
-j <jobfile>        Job file.
-l|--list           List profiles.
-p <profile>        Profile to load.
--show-meta[=<fmt>] Show meta information. <fmt> is one of "text", "json".
                    Default is "text".
--show-profile[=<fmt>]
                    Show profile. <fmt> is one of "text", "json". Default is
                    "text".
--show-profile-flat
                    Show flattened (no sections) profile. json format only.
--show-request[=<fmt>]
                    Show request information. <fmt> is one of "text", "json".
                    Default is "text".

Options:
-c <name>=<value>   Request chunk setting.
-H <name>=<value>   Request hook setting.
-k <name>=<value>   Setting using full key name.
-r <name>=<value>   Request attribute setting.
-v <name>=<value>   Request environment variable.
--enforce y|n       Enforce property constraints. Default is y.

--debug             Enable debug.
--verbose           Enable verbosity.
```

Quick Start

Things to Know

- Configuration is done using profile files some of which are provided by the administrator, and others by the user (for overrides).
- A named profile is provided to select a target queueing system and cluster/cell.
- Metadata is used to configure settings: type, constraints (e.g., minimum, maximum), and other aspects.
- Command line options are available to show profile and metadata information for inspection.
- Directives may be provided in the job file or at the command line.
- The common settings can be seen below in Common Profile Settings.
- A queueing system is **not** required to run jobgen.

Example

The following is based on the sample settings provided in the package.

Job file (`hello.jgen`):

```
#! /bin/bash
#
#JGEN -r name=test
#JGEN -r joinouterr=y
#JGEN -r queue=dev
#JGEN -r wallclock=0:30
#JGEN -c nslots=4
#JGEN -c ncores=8
#JGEN -c memory=4G

echo "hello"
```

Notes:

- `#JGEN` is the jobgen directive prefix.
- `-r` is a shortcut to specify `request.<name>` settings.
- `-c` is a shortcut to specify `request.chunk.0.default.<name>` settings.

Generate new job file (using `gpsc1` profile which targets a site-specific Gridengine setup):

```
jobgen -p gpsc1 -j hello.jgen
```

Notes:

- Use `-p <profile>` to specify the profile.
- Use `-j <jobfile>` to specify the job file.

Generated job file:

```
#! /bin/bash
#

echo "hello"

# ---- jobgen -- start
# ---- jobgen -- timestamp (2019-10-18 12:23:18)
# ---- jobgen -- generator (<class 'jobgen.generators.gridengine.GPSCGridengineGenerator'>)
# ---- jobgen -- syshooks (check_memory_tmpfs,compact) (None) (None)
# ---- jobgen -- hooks (None) (None) (None)
#
#$ -j y
#$ -N test
#$ -S /bin/bash
#$ -v JOBGEN_JOINOUTERR=true
#$ -v JOBGEN_NAME=test
#$ -v JOBGEN_NSLOTS=4
#$ -v JOBGEN_QUEUE=dev
#$ -v JOBGEN_SHELL=/bin/bash
#$ -v JOBGEN_SLOT_MEMORY=4G
#$ -v JOBGEN_SLOT_NCORES=8
#$ -v JOBGEN_SLOT_TMPFS=500M
#$ -v JOBGEN_WALLCLOCK=30
#$ -pe dev 1
#$ -l h_rt=0:00:30
#$ -l res_mem=4096
#$ -l res_cpus=8
#$ -l res_tmpfs=500
#
# ---- jobgen -- end
```

Notes:

- Gridengine-specific directives are used (`-j`, `-N`, `-S`, `-v`, `-pe`, `-l`).
- `JOBGEN_*` environment variables are automatically provided with job information.

Generate new job file (using ppp1 profile which targets a site-specific PBSPro setup):

```
jobgen -p ppp1 -j hello.jgen
```

Generated job files:

```
#! /bin/bash

# ---- jobgen -- start
# ---- jobgen -- timestamp (2019-10-18 12:24:14)
# ---- jobgen -- generator (<class 'jobgen.generators.pbspro.PPPPBSProGenerator'>)
# ---- jobgen -- syshooks (check_memory_tmpfs,compact) (None) (None)
# ---- jobgen -- hooks (None) (None) (None)
#
#PBS -j oe
#PBS -N test
#PBS -S /bin/bash
#PBS -v JOBGEN_JOINOUTERR=true
#PBS -v JOBGEN_NAME=test
#PBS -v JOBGEN_NSLOTS=4
#PBS -v JOBGEN_QUEUE=dev
#PBS -v JOBGEN_SHELL=/bin/bash
#PBS -v JOBGEN_SLOT_MEMORY=4G
#PBS -v JOBGEN_SLOT_NCORES=8
#PBS -v JOBGEN_WALLCLOCK=30
#PBS -q development
#PBS -l walltime=0:00:30
#PBS -l select=1:ncpus=8:mem=4096M
#
# ---- jobgen -- end

#

echo "hello"
```

Notes:

- PBSPro-specific directives are used (`-j`, `-N`, `-S`, `-v`, `-q`, `-l`, `-l select`).
- `JOBGEN_*` environment variables are automatically provided with job information.

Configuration

Profile Files and Directives

Profiles are INI-style files containing various kinds of settings (metadata, and data). Profiles are most often set up by system administrators. Users can provide their own provides to modify, augment, and override the system settings.

Directives are provided in the job file itself, at the command line, or both. They are converted into profile settings to be loaded.

Profiles and directives are loaded in a specific order (`profname` may be provided when calling `jobgen`):

1. system base profile: `base.conf`
2. system site profile: `site.conf`
3. system named profile: `<profname>.conf`
4. user base profile: `~/.jobgen/profiles/base.conf`
5. user site profile: `~/.jobgen/profiles/site.conf`
6. user named profile: `~/.jobgen/profiles/<profname>.conf`
7. job file directives
8. command line directives

The settings of all sections (from all profiles) are combined / overlaid so that the last settings loaded take precedence.

Given the profiles and directives, a final, flattened (no sections) profile is then used for the job request to generate a job file suitable for the target queueing system.

Profile Settings

Sections and Values

Settings are organized by sections. Sections contain zero or more key=value settings.

```
[secname]
key = value
```

If not value is given, the value is None:

```
[secname]
key
```

Two kinds of settings are possible:

- meta - information about non-meta settings
- non-meta - values

Namespaces

Profile settings are organized by namespaces which are determined by the prefix of the key:

Namespace Prefix	Type
`info`	Informational.
`generator`	Job file generator.
`meta`	Metadata.
`qs`	Queueing system-specific.
`request`	Request.

The namespaces provide information used for different parts of the process of generating a final job file.

Examples

List profiles:

```
$ jobgen -l
sys:base
sys:site
sys:gpsc4
...
```

Show standard profiles:

```
jobgen --show-profile
```

Notes:

- loads system and user profiles

Show profile (profile given):

```
jobgen --show-profile -p gpsc4
```

Notes:

- loads `gpsc4` profile in addition to the system and user profiles

Show profile (profile, job file, and command-line directives are given):

```
jobgen --show-profile -p gpsc4 -j hello.jgen -r name=testing-123
```

Notes:

- after the profile files are loaded, the directives from `hello.jgen` and loaded, then the command line directives
- even though the job file is provided, nothing is submitted

Metadata Settings

Metadata settings are special. They are specified using `meta` which corresponds to the metadata namespace. Metadata settings provide information about the non-metadata settings and are used during the normalization step. Normalization is the process of applying metadata settings to a non-metadata setting to produce a normalized value.

Metadata settings take the form:

```
meta.<mtype>.<key> = <value>
```

where `<mtype>` is:

- `type`, which specifies the setting type
- type-specific constraint such as `min`
- other type-specific setting

Metadata Types

The following `meta.types` are supported:

Type	Normal Form	mtypes	Accepted Values	Description
boolean	True, False	• `default` • `forced`	• `y` • yes • true • `n` • no • false • `$default` • `$forced`	Boolean value.
firstlastrange	(Integer, Integer, Integer)	• `max` • `maxcount` • `min`		Range 3-tuple of (first, last, step).
float	Floating point value	• `clamp` • `default` • `forced` • `max` • `min`	• float value • `$default` • `$forced` • `$max` • `$min`	Floating point value.
integer	Integer value	• `clamp` • `default` • `forced` • `max` • `min`	• integer value • `$default` • `$forced` • `$max` • `$min`	Integer value.
memory	Bytes	• `clamp` • `default` • `forced` • `max` • `min`	• integer with optional suffix: B - bytes, K - kilobytes, M - megabytes, G - gigabytes, T - terabytes • `$default` • `$forced` • `$max` • `$min`	Memory value with optional suffix.
namelist	List of strings		• comma-separated list of unquoted strings	Strings.

string	String	• `default` • `forced`	• string value • string value within quotes (", ', """, "") • `$default` • `$forced`	String value.
stringmatch	String	• `values` • `default` • `forced`	• string value • string value within quotes (", ', """, "") • `$default` • `$forced`	String value which must match one in provided list of strings.
stringregexp	String	• `default` • `forced` • `regexp`	• string value • string value within quotes (", ', """, "") • `$default` • `$forced`	String value which must match regular pattern.
time	Integer in seconds	• `clamp` • `default` • `forced` • `max` • `min`	• integers in format: <s>, <m>:<s>, <h>:<m>:<s>, <d>:<h>:<m>:<s>. • `$default` • `$forced` • `$max` • `$min`	Time value which must match a format.

Notes:

- Normal form the internal form in which a value is kept
- Accepted values may using symbolic names which start with a $; these names are replaced with values

Common constraints are:

Value	Description
`clamp`	Clamp to min or max value.
`default`	Default value.
`forced`	Forced value.
`max`	Maximum value.
`min`	Minimum value.
`regexp`	Regular expression.
`values`	Comma-separated values.

Examples

If:

```
meta.type.request.name = string
request.name = "bigwork"
```

then:

- every instance of the profile setting `request.name` is treated as a string.
- strings normalize to strings.
- strings are usually quoted.

If:

```
meta.type.request.chunk.0.default.ncores = integer
meta.min.request.chunk.0.default.ncores = 1
meta.max.request.chunk.0.default.ncores = 44
request.chunk.0.default.ncores = 32
```

then:

- the setting `request.chunk.0.default.ncores` specifies a chunk setting, for chunkid 0, for nodegroup default, with name ncores.
- the profile setting `request.chunk.0.default.ncores` is treated as an integer.
- the min and max constraints are set using `meta.min.request.chunk.0.default.ncores` and `meta.max.request.chunk.0.default.ncores`.
- the min and max constraints are treated as and normalized to integer, are specified by `meta.type.request.chunk.0.default.ncores`.
- the min and max constraints for `request.chunk.0.default.ncores` are set to 1 and 44, respectively.

If:

```
meta.type.request.chunk.0.default.memory = memory
request.chunk.0.default.memory = 2G
```

then:

- the profile setting `request.chunk.0.default.memory` is treated as memory.
- `request.chunk.0.default.memory` is normalized to count of bytes.
- the memory type accepts common memory suffixes, as G in this case, for gigabytes.

If:

```
meta.type.request.chunk.0.default.memory = memory
meta.max.request.chunk.0.default.memory = 100G
request.chunk.0.default.memory = $max
```

then:

- the maximum value constraint is 100G
- when normalized, the symbolic name `$max` is resolved to the maximum value constraint, if available

Request Information

Once the profiles are loaded, request settings are drawn from the specific sections using a cascading method: select first instance from the sections in cascade order.

The cascade order (most to least significant) by section:

1. command line directives
2. job file directives
3. `user.<username>`
4. `group.<groupname>`
5. `queue.<queuename>`
6. `qs.<qsname>`
7. `default`

The request settings are then normalized based on the metadata settings.

Supported Settings

Standard

A common, standard set of profile settings that are supported across all queueing systems is provided by default.

Name	Value Type	Job Environment	Description
`generator.import`	string		Identifies the generator used to produce job file.
`request.chunk.0.default.nslots`	integer	`JOBGEN_NSLOTS`	Number of slots.
`request.chunk.0.default.memory`	memory	`JOBGEN_SLOT_MEMORY`	Memory per slot.
`request.chunk.0.default.ncores`	integer	`JOBGEN_SLOT_NCORES`	Number of cores per slot.
`request.chunk.0.default.raw.<name>`	string or other		Per slot setting where `<name>` is used for

request.env.*	string		Environment variables to pass.
request.errpath	string	JOBGEN_ERRPATH	Path to error output file (stderr).
request.jobarray	firstlastrange	JOBGEN_JOBARRAY	Job array range of <first>-<last>:<step>.
request.joinouterr	boolean	JOBGEN_JOINOUTERR	Join error and output to same output file.
request.mail	string	JOBGEN_MAIL	Email address to send notifications.
request.mailopts	CS strings	JOBGEN_MAILOPTS	Comma-separated list of strings. One or more: beginning, end, aborted, suspended, none. Support depends on the queueing system.
request.name	string	JOBGEN_NAME	Job name.
request.outpath	string	JOBGEN_OUTPATH	Path to output file (stdout).
request.project	string	JOBGEN_PROJECT	Project name.
request.qs	string	JOBGEN_QS	Queueing system name.
request.queue	string	JOBGEN_QUEUE	Queue name.
request.rerun	boolean	JOBGEN_RERUN	Rerun job.
request.shell	string	JOBGEN_SHELL	Shell to run job file.
request.wallclock	time	JOBGEN_WALLCLOCK	Time to run job.

Gridengine

QS Name	Name	Value Type	Job Environment	Description
gridengine	qs.gridengine.request.pe	string		Native PE. Fallback to request.queue.
gridengine	qs.gridengine.request.queue	string		Native queue. Fallback to request.subqueue.
gridengine-gpsc	request.chunk.0.default.gputype	stringmatch	JOBGEN_SLOT_GPUTYPE	Type of gpu.
gridengine-gpsc	request.chunk.0.default.image	stringmatch	JOBGEN_SLOT_IMAGE	Container image.
gridengine-gpsc	request.chunk.0.default.ngpus	integer	JOBGEN_SLOT_NGPUS	Number of gpus.
gridengine-gpsc	request.chunk.0.default.tmpfs	memory	JOBGEN_SLOT_TMPFS	Temporary filespace.

PBS

QS Name	Name	Value Type	Job Environment	Description
pbspro	qs.pbs.request.queue	string		Native queue. Fallback to request.queue.
pbspro-ppp	request.chunk.0.default.image	stringmatch	JOBGEN_SLOT_IMAGE	Container image.
pbspro-ppp	request.chunk.0.default.tmpfs	memory	JOBGEN_SLOT_TMPFS	Temporary filespace.

Slurm

QS Name	Name	Value Type	Job Environment	Description
slurm	qs.pbs.request.partitions	string		Native partition(s).
slurm-gpsc	request.chunk.0.default.image	stringmatch	JOBGEN_SLOT_IMAGE	Container image.
slurm-gpsc	request.chunk.0.default.tmpfs	memory	JOBGEN_SLOT_TMPFS	Temporary filespace.

Examples

The following assumes a basic setup of the system profile files base.conf, site.conf, and gpsc4.conf installed by the system administrator.

Given basic.jgen:

```
#! /bin/bash
#
# basic.jgen
#
#JGEN -r name=test
#JGEN -r joinouterr=y
#JGEN -r queue=dev
#JGEN -r wallclock=2:30
#JGEN -c nslots=6
#JGEN -c ncores=8
#JGEN -c memory=2G

echo "hello"
```

Notes:

- `#JGEN` prefix identifies `jobgen` directives.
- General request settings are specified with `-r`.
- Chunk-specific request settings are specified with `-c`.

Generate the job file using the gpsc4 profile:

```
jobgen -p gpsc4 -j basic.jgen
```

The generated job file:

```
#! /bin/bash
#
# basic.jgen
#

echo "hello"

# ---- jobgen -- start
# ---- jobgen -- timestamp (2019-05-31 12:14:07)
# ---- jobgen -- generator (jobgen.generators.gridengine.GridengineGPSCGenerator)
# ---- jobgen -- syshook (None)
# ---- jobgen -- userhook (None)
#
#$ -j y
#$ -N test
#$ -S /bin/bash
#$ -pe dev 6
#$ -q dev
#$ -l h_rt=2:30
#$ -l res_mem=2048
#$ -l res_cpus=8
#
# ---- jobgen -- end
```

Notes:

- Generated output contains information about the generation process.
- The generator `jobgen.generators.gridengine.GridengineGPSCGenerator` is used, which generates for Gridengine with GPSC-specific support.
- Final directives are generated based on `jobgen` settings.
- Final directives use formats tailored for the target queueing system; e.g., `res_mem` takes its value in MB, but has no suffix

Hooks

Under development.

Hooks provide a way to programatically modify the job request. Hooks are called **before** the targetted job file is created.

Files

Hooks are functions defined in files:

- system - `lib/jobgen/hooks/syshooks.py`
- user - `~/.jobgen/hooks/userhooks.py`

They are written in Python v2. To facilitate the future transition to Python v3, keep the hooks simple.

The standard form of a hook is:

```
def hookname(reqinfo):
    ...
```

Notes:

- A `RequestInfo` object is passed as an in+out parameter.
- All changes are made to the the `RequestInfo` object.

Settings

Name	Value Type	Description
`request.syshooks.allow`	namelist	List of system hooks allowed. Default is to allow all.
`request.syshooks.deny`	namelist	List of system hooks denied. Default is to deny none.
`request.syshooks.names`	namelist	Ordered list of system hooks to run.
`request.hooks.allow`	namelist	List of (all) hooks allowed (also applies to `request.syshooks.names`). Default is to allow all.
`request.hooks.deny`	namelist	List of (all) hooks denied (also applies to `request.syshooks.names`). Default is to deny none.
`request.hook.<hookname>.<subkey>`		Namespace for (all) hook-specific settings.
`request.hooks.names`	namelist	Ordered list of hooks to run (after those of `request.syshooks.names`).

The hooks that are run (`request.hooks.names`) are searched for in the user and system hooks files, in that order.

Examples

Identity

Profile (`base.conf`):

```
request.hooks.names = identity
```

Do nothing.

```
def identity(reqinfo):
    pass
```

Bump the Wallclock

Bump the wallclock time by a specific amount of time.

Profile (`base.conf`):

```
request.hooks.names = bump_wallclock
meta.type.request.hook.bump_wallclock.bump = time
```

In the job file:

```
#JGEN -k request.hook.bump_wallclock.bump = 30
```

Code:

```
def bump_wallclock(reqinfo):
    if "request.wallclock" in reqinfo:
        bump = reqinfo.get("request.hook.bump_wallclock.bump")
        if bump:
            wallclock = reqinfo.get("request.wallclock")
            reqinfo.set("request.wallclock", wallclock+bump)
```

Notes:

- `bump_wallclock` is specified as one of the hooks to call.
- `request.hooks.names` is specified in the user `base.conf` profile.
- `request.hook.bump_wallclock.bump` is of type time and is used to pass a value.
- A bump value can be specified as a directive using the fully specified key `request.hook.bump_wallclock.bump`.
- Operations on the value of `request.wallclock` are in normalized form (in seconds)
- All constraints are applied when setting the new value using `reqinfo.set()`.

Dump to File

Profile (`base.conf`):

```
request.hooks.names = dump_to_file
```

Dump the request information to a file.

```
import os
import os.path
import pprint
import time

def dump_to_file(reqinfo):
    path = os.path.expanduser("~/.jobgen/tmp")
    name = reqinfo.get("request.name", "noname")
    os.mkdir(path)
    f = open(os.path.join(path, "%s.%s" % (name, time.time())))
    f.write("meta:\n")
    f.write("%s\n-----" % pprint.pformat(reqinfo.meta.d, indent=4))
    f.write("%s\n" % pprint.pformat(reqinfo.d, indent=4))
```

Notes:

- Dump the output into the `~/.jobgen/tmp` directory.
- Generate a filename using the request name with a timestamp for uniqueness.
- Dump `reqinfo` meta and non-meta contents (access to this data will change in the future).