Introduction to bl\_tools

run Bladelogic from the command line

# The bl\_tools allow one to do numerous bladelogic tasks from the commandline, including some things that are not possible from the GUI. You’ll never get kicked out of a session, changing directories is easy, and it’s quick and easy to change back and forth between roles.

# **Index**

[Part I. Preliminaries](#Preliminaries) – gets your CLI environment ready and gives some practical instruction on how to use the environment using some of the basic tools.

[Setting up your shell environment](#setting_up_shell_env)

[Setting up your profile with bl\_mkconf](#mk_conf)

[Using your profile with bl\_conf](#bl_conf)

[myGroup, the bl\_tools working directory – a brief examination of your profile](#myGroup_in_profile)

[Modify myGroup with bl\_chdir](#modify_myGroup_with_bl_chdir)

[Create myGroup with bl\_mkdir](#create_myGroup_bl_mkdir)

[Bladelogic Paths](#_Bladelogic_paths)

[Part II. Core utilities](#_Core_Utilities) – a closer look at how to use some core tools in bl\_tools

[bl\_jobrun](#bl_jobrun) – execute jobs from the command line

[bl\_printlog](#bl_printlog) – search for previous job runs and show their logs

[bl\_copyjob](#_bl_copyjob) – make copies of a job to customize the target list or parameters

[bl\_lsdir.u](#_bl_lsdir.u) – an unsupported utility for listing the contents of a job/depot/server group

# **Preliminaries**

## **Setting up your shell environment**

At the moment, the tools are available on lrdne0tg.

So, log in, and add this to your .profile:

. /unixworks/bl\_tools/etc/bl\_tools.sh

Only bash is supported right now. If your shell is ksh but you are not attached to it,

1. you can set your .profile like this:

/bin/bash –l

1. set your .bash\_profile like this:

. /unixworks/bl\_tools/etc/bl\_tools.sh

To test, get a new shell - you should see this:

(py27) -sh-4.1$

## **Setting up your profile with bl\_mkconf**

You’ll want to set up a profile for every role you use.

Please notice the bl\_mkconf command provides suggestions, but no defaults

You can set your default location to wherever the jobs are that you’ll execute most often, or to the base path where most of the folders you use are located. The default suggestion is to create a directory named after your uid and name, where you can place copies of jobs – for example, scheduled jobs.

(py27) -sh-4.1$ bl\_mkconf

Profile Name - how to refer to this role when executing bl\_conf - no whitespace: devprofile

Role Name: MYBLROLE

User Name - like USERID or USERID@DOMAIN: MYUSER@MYDOMAIN

Default Location - where your scripts and jobs will be kept - whitespace discouraged: /WorkAreas/MYBLROLE/MYUSER-mark-price

overwriting /home/MYUSER/.bl\_tools/conf/devprofile

If your preferred defaultMyGroup location does not exist, you can create it with

## **Using your profile with bl\_conf**

To configure your shell to use a certain profile, execute shell function bl\_conf

If you know which profile to use, just call with the profile name

Notice that your command prompt will now show the profile you are using and the current default search location for further commands

(py27) -sh-4.1$ bl\_conf devprofile

BL\_CONF SET is now set TO devprofile, myGroup set TO /WorkAreas/MYBLROLE/MYUSER-mark-price

BL\_PASS is not set. please run bl\_pass

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price)

If you don’t know which profile to pick, just call bl\_conf:

(py27) -sh-4.1$ bl\_conf

CONF=

Choose one of:

devprofile

prodprofile

engprofile

CONF> devprofile

BL\_CONF SET is now set TO devprofile, myGroup set TO /WorkAreas/MYBLROLE/MYUSER-mark-price

BL\_PASS is not set. please run bl\_pass

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price)

Now, set the password used by the userid in this profile

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_pass

Password:

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) echo $BL\_PASS

bWFkZSB5b3UgbG9vaw==

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price)

The password presented above is BASE64 encoded, and is stored in an environment variable.

A given shell instances environment variables are only visible to themselves and to subshells.

## **myGroup, the bl\_tools working directory – a brief examination of your profile**

Here are some notable entries from a profile.

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) grep Group ~/.bl\_tools/conf/eng

myGroup: /WorkAreas/DEPLOYERADV\_UNIX\_CSW\_NP\_BL/Operations/mark-price

defaultMyGroup: /WorkAreas/DEPLOYERADV\_UNIX\_CSW\_NP\_BL/Operations/mark-price

serverGroup: /Servers by Role/ENG\_UNIX\_CSW\_NP\_BL

myGroup and defaultMyGroup are also defined as environment variables.

myGroup is kind of like the unix current working directory – where you are right now.

defaultMyGroup is more like the $HOME variable

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) cd /tmp

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) pwd

/tmp

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) echo $HOME

/home/MYUSER

When using the bl\_tools, you’ll often specify a path to a job, script, or servergroup. If the path to that object does not start with a “/”, myGroup will be prepended.

For example:

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_jobrun --job myReport.nsh --servers host1

starting up /unixworks/bl\_tools/bin/bl\_jobrun --job myReport.nsh –servers host1 at 2019-01-12 18:38:28.125186-05:00 US/Eastern

attempting to run /WorkAreas/MYBLROLE/MYUSER-mark-price/myReport.nsh

## **modify myGroup with bl\_chdir**

Bladelogic job paths can be pretty long. So, setting shell variable $myGroup appropriately is very useful.. you can reference anything in a given folder using only its name.

That’s what bl\_chdir is for.

When you execute bl\_chdir, it will update your myGroup in the shell variable, as well as in the config file for your bl\_tools profile.

There are two ways to use bl\_chdir

If you call with an argument, the first argument will be used as the new myGroup. So, if there are any spaces, you’ll need to use quotes:

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_chdir "/new path/for future/commands"

BL\_CONF SET is now set TO devprofile, myGroup set TO /new path/for future/commands

(py27) -sh-4.1$ (devprofile:/new path/for future/commands)

If you call without arguments, the value of defaultMyGroup will be given as the default – just press enter again to revert.

(py27) -sh-4.1$ (cswdev:/new path/for future/commands) bl\_chdir

starting with profile devprofile and $myGroup=/new path/for future/commands

new value for myGroup[/WorkAreas/MYBLROLE/MYUSER-mark-price]:

BL\_CONF SET is now set TO devprofile, myGroup set TO /WorkAreas/MYBLROLE/MYUSER-mark-price

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price)

Calling without arguments is also a good way to go if you want to paste the value in without quotes:

new value for myGroup[/WorkAreas/MYBLROLE/MYUSER-mark-price]: /i/dont like/quotes

BL\_CONF SET is now set TO devprofile, myGroup set TO /i/dont like/quotes

(py27) -sh-4.1$ (devprofile:/i/dont like/quotes)

## **Create myGroup with bl\_mkdir**

If your preferred myGroup/defaultMyGroup location does not exist, you can quickly create it with bl\_mkdir

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/usr9999-bender-rodriguez) bl\_mkdir --jobgroup ''

logging to /tmp/bl\_tools\_log/MYUSER@MYDOMAIN.664597.1547685678.81

Logged in as: MYUSER@MYDOMAIN

Switching role to: MYBLROLE

groupType: JobGroup

creating /WorkAreas/MYBLROLE/usr9999-bender-rodriguez as JobGroup

bl\_mkdir: groupExists( /WorkAreas/MYBLROLE/usr9999-bender-rodriguez )

doing mkdir\_recursive of /WorkAreas/MYBLROLE/usr9999-bender-rodriguez on base / self.runBlcli( JobGroup createGroupWithParentName ['usr9999-bender-rodriguez', '/WorkAreas/MYBLROLE'] True

True

followup: groupExists( /WorkAreas/MYBLROLE/usr9999-bender-rodriguez ): True

## **Bladelogic paths**

For practical use, probably the most important first thing to know is that paths work just a little bit differently than what you might be used to.

Frequently, the path is prefixed with the repository type (i.e. /Jobs, /Depot, etc). For example, a doc to install [Media Server rev2](http://pe.bankofamerica.com/documents/rhel/view-html/5548d71cda7ba/5bff1b7270bfe/Netbackup%20Media%20Server%208.1%20RHEL%20rev2) might be at:

/Jobs/Library/Of/Jobs/Media Server/MediaServer\_1.1.noarch\_rev2

However, when using a bl\_tools command, the repository type (jobs vs depot etc) is already implicit based on the arguments passed to the tool so the /Jobs prefix is not used, so you would strip off “/Jobs” and reference:

/Library/Of/Jobs/ Media Server/MediaServer\_1.1.noarch\_rev2

# **Core Utilities**

## **bl\_jobrun**

Usage: bl\_jobrun [options] - executes once against each group of job targets in --defaults (--servers and --serverfile), and --servergroups

Options:

-h, --help show this help message and exit

--job=JOB $myGroup is prepended if not starting with a '/'

--defaultservers Run against the default targets defined on the job.

Enabled by default if no servers are specified by

other options.

--servers=SERVERS comma separated list. run against these servers

rather than the currently assigned ones.

--serverfile=SERVERFILE

newline delimited file. run against these servers

rather than the currently assigned ones.

--servergroups=SERVERGROUPS

comma delimited list. run against these server groups

rather than the currently assigned ones.

--servergroupfile=SERVERGROUPFILE

newline delimited list of server groups. run against

these server group rather than the currently assigned

ones.

--printlog print the bl logs for the jobs

--printstatus UNSUPPORTED - print the job exit status per server.

--reportwait=REPORTWAIT

How many seconds between loops waiting for jobs to

finish. Defaults: the interval increases logarithmically from 10 seconds

to 10 minutes.

--debug

A few notes about the command –

1. At this time, –defaultservers, --servers, --serverfile, --servergroups, --servergroupfile can all be used in combination with each other. However, mixing –defaullt options, –server\* options and –servergroup\* options will result in multiple job executions and multiple job run ids. The job run ids will be important when using [bl\_printlog](#_bl_printlog)
2. –printlog shows the same text you’d see in the console, but they are not broken down by server. If server-specific logs are important, it will depend on the job script whether that is displayed. If the job executes sequentially, it will be clear
3. –printstatus – if the job executes separately against each targeted server, --printstatus will show whether the job “succeeded” or “failed.”
4. Once a job has been submitted for execution, it will keep running even if you use ctrl-c. See [bl\_printlog](#_bl_printlog).

So, to run BAC\_Netbackup\_MediaServer\_8.1.noarch\_rev2 against server1.bankofamerica.com, and server2.bankofamerica.com, you could do this:

(py27) -sh-4.1$ (devprofile:/current/mygroup/location) bl\_jobrun --job /Library/Of/Jobs/ backupsolution/1.1/backupsolution\_rev2 --servers server1,server2 –printlog

Or, you could first use bl\_chdir. This makes the bl\_jobrun command easier to read, and if you use [bl\_lsdir.u](#bl_lsdir_u) as described below, you will also have easy access to any jobs in that directory

(py27) -sh-4.1$ (devprofile:/Library/Of/Jobs) bl\_chdir “/Library/Of/Jobs/backupsolution/1.1"

BL\_CONF SET is now set TO devprofile, myGroup set TO “/Library/Of/Jobs/backupsolution/1.1"

(py27) -sh-4.1$ (devprofile: /Library/Of/Jobs/backupsolution/1.1) bl\_jobrun –job backupsolution\_rev2 --servers server1,server2 –printlog

## bl\_printlog

Usage: bl\_printlog [options]

ex: bl\_printlog --job chkHome.nsh -- print logs from the last run initiated by current user and role

ex: bl\_printlog --job chkHome.nsh --jobrunkey DBKey:SJobRunModelKeyImpl:65836102-531990254 -- print logs from a specific job run"

Options:

-h, --help show this help message and exit

--job=JOB //WorkAreas/path/to/your/job - if not fully qualified

then relative to env varable $myGroup

--jobrunkey=JOBRUNKEY

key identifying the job run

--jobrunid=JOBRUNID runid identifying the job run

--lastrun filter down to the most recent match (default unless

--lookup is specified)

--anyone search for any user or role

--user=USER filter by user - like NBKID or

NBKID@MYDOMAIN - NBKID will match any

domain

--role=ROLE filter by bladelogic role name

--lookup print metadata: user/time/role of all jobruns returned

- do not print logs

--starttime=STARTTIME

filter by start time at or after YYYY/MM/DD HH:MM:SS

--endtime=ENDTIME filter by start time at or before YYYY/MM/DD HH:MM:SS

This utility lets you search for past executions of a given job, filtering based time windows and on whichever roles and users your current credentials have visibility on. That information can be used to make the bladelogic logs available for review on the commandline, so that you can copy the record and/or apply tools like awk to the output.

Example 1: get the logs for the last execution of the job by the current user and role:

The simplest of the tool is to just provide a jobname. It will find the most recent execution of the job from the current role and credentials.

(py27) -sh-4.1$ (deployDCustom:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_printlog --job myReport.nsh | grep Message | awk '{print $NF}'|grep , | sort | head

Logged in as: MYUSER@MYDOMAIN

Switching role to: MYBLROLE

host01,46168,27,31,=27/31,0.0%,99%,3

host02,45140,28,31,=28/31,0.0%,100%,3

host03,45328,29,31,=29/31,0.0%,95%,3

host04,59656,248,251,=248/251,6.7%,100%,23

host05,67712,248,251,=248/251,6.7%,100%,23

host06,59836,248,251,=248/251,66.7%,100%,23

host07,45364,248,251,=248/251,0.0%,100%,23

host08,45816,248,251,=248/251,0.0%,100%,23

host09,45436,248,251,=248/251,0.0%,100%,23

host10,44176,236,251,=236/251,0.0%,100%,55

Example 2: get the record for the last executions of the job

This example gets the run keys for every record my user and role has visibility on, and shows the two most recent. The runkey and the different userids are highlighted.

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_printlog --job myReport.nsh --lookup --starttime=2019/01/10 --anyone 2> /dev/null | grep ^DBKey:SJobR | head -2

DBKey:SJobRunModelKeyImpl:72614186-563116416 MYBLROLE SVCID@MYDOMAIN 2019/01/17 01:18:34 2019/01/17 01:18:49

DBKey:SJobRunModelKeyImpl:72600302-563032653 MYBLROLE NBKDZ5R@MYDOMAIN 2019/01/15 15:23:34 2019/01/15 15:24:30

Example 3: using a run key from example 2, get the run log from a particular execution. Show which hosts it ran on, how many hosts it ran on, and that it was successful on all hosts

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_printlog --job myReport.nsh DBKey:SJobRunModelKeyImpl:72614186-563116416 2> /dev/null | grep Exit\ Code | awk '{ print $(NF-2),$(NF-1),$NF }' | wc -l

106

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_printlog --job myReport.nsh DBKey:SJobRunModelKeyImpl:72614186-563116416 > /dev/nujll | grep Exit\ Code | awk '{ print $(NF-2),$(NF-1),$NF }' 2> /dev/null | sort | uniq

Exit Code 0

Example 4: get the most recent execution that happened between Saturday and Tuesday, and search for references to host1

(py27) -sh-4.1$ (cswprod:/WorkAreas/DEPLOYERADV\_UNIX\_CSW\_CUSTOM\_PROD\_BL/MYUSER-mark-price) bl\_printlog --job myReport.nsh --starttime 2019/01/13\ 00:00:00 --endtime 2019/01/13\ 23:59:59 | grep host1

Logged in as: MYUSER@MYDOMAIN

Switching role to: DEPLOYERADV\_UNIX\_CSW\_CUSTOM\_PROD\_BL

Type: Info Date: Sun Jan 13 16:43:34 EST 2019 Message: Executing work item NSH Script Job:myReport.nsh; Server:host1 on application server: blappserver

Type: Info Date: Sun Jan 13 16:43:34 EST 2019 Message: host1

Type: Info Date: Sun Jan 13 16:43:43 EST 2019 Message: host1,53188,227,314,=227/314,0.0%,100%,63

(py27) -sh-4.1$ (cswdev:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_printlog --job myReport.nsh --starttime 2019/01/12\ 00:00:00 --endtime 2019/01/15\ 23:59:59 2> /dev/null | grep host1

Type: Info Date: Tue Jan 15 15:24:23 EST 2019 Message: Executing work item NSH Script Job:myReport.nsh; Server:host1; on application server: blappserver

Type: Info Date: Tue Jan 15 15:24:24 EST 2019 Message: host1

Type: Info Date: Tue Jan 15 15:24:30 EST 2019 Message: host1,66472,5,7,=5/7,0.0%,90%,3

## bl\_copyjob

Usage: bl\_copyjob [options]

Options:

-h, --help show this help message and exit

--srcjob=SRCJOB source job

--destjob=DESTJOB new job name

This tool lets you copy jobs from one jobgroup to another. For example, if you want to copy it into your own folder, or if you want to make a new copy that can have its own default targets or parameters.

Example 1: Copy from the OS core tools archive into my own location for holding copied jobs

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE/MYUSER-mark-price) bl\_copyjob --srcjob "/Jo We/Want/To/Copy\_rev2" --destjob copies/Copy\_rev2

logging to /tmp/bl\_tools\_log/MYUSER@MYDOMAIN.664597.1547709917.21

Logged in as: MYUSER@MYDOMAIN

Switching role to: MYBLROLE

False

DBKey:SJobModelKeyImpl:14651212-5-557637467

{

'attachment': [],

'classInstance': False,

'comments': 'No comments',

'error': None,

'executionTime': 0L,

'list': False,

'memoryUsed': 0L,

'realInputArguments': [

'DBKey:SJobModelKeyImpl:14651212-5-557637467',

'/WorkAreas/MYBLROLE/MYUSER-mark-price/copies',

'Copy\_rev2'

],

'returnValue': 'DBKey:SJobModelKeyImpl:14905981-1-562774375',

'returnValues': None,

'success': True

}

## bl\_lsdir.u

Usage: bl\_lsdir.u [options] -- UNSUPPORTED - uses unpublished bladelogic API.

Options:

-h, --help show this help message and exit

--jobgroup=JOBGROUP

--depotgroup=DEPOTGROUP

--servergroup=SERVERGROUP

This utility’s name ends in .u for “unsupported.” This means it should not be used in production scripts. However, it’s quite useful for looking around job, depot and server trees.

Example 1: list the contents of the path show in the command prompt

This first example is like unix ls `pwd`. myGroup is set to the base jobgroup for the MYBLROLE role, and we pass bl\_lsdir.u the empty string.

In the output, the listings under “subgroups” are jobgroups, i.e. like folders. And, of course, under Jobs is a job.

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE) bl\_lsdir.u --jobgroup ''

Listing jobGroup /WorkAreas/MYBLROLE

Logged in as: MYUSER@MYDOMAIN

Switching role to: MYBLROLE

subgroups:

ID group description

16724749 "USER1-don-knots" ""

16187745 "MYUSER-mark-price" ""

17754345 "USER2-Frank-Sinatra" ""

17840448 "USER3-Fred-Flintstone" ""

Jobs:

"mypackage-9.9.99-136\_rhel.rev0"

""

Example 2: get a listing on an unqualified path:

Now, a listing on a unqualified path based on the previous example:

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE) bl\_lsdir.u --jobgroup MYUSER-mark-price/copies

Listing jobGroup /WorkAreas/MYBLROLE/MYUSER-mark-price/copies

logging to /tmp/bl\_tools\_log/MYUSER@MYDOMAIN.664597.1547698214.79

Logged in as: MYUSER@MYDOMAIN

Switching role to: MYBLROLE

subgroups:

ID group description

17225149 "scheduled" ""

Jobs:

"void"

(py27) -sh-4.1$ (devprofile:/WorkAreas/MYBLROLE) bl\_lsdir.u --jobgroup MYUSER-mark-price/copies/scheduled

Listing jobGroup /WorkAreas/MYBLROLE/MYUSER-mark-price/copies/scheduled

logging to /tmp/bl\_tools\_log/MYUSER@MYDOMAIN.664597.1547698223.67

Logged in as: MYUSER@MYDOMAIN

Switching role to: MYBLROLE

subgroups:

ID group description

Jobs:

"demo.nsh-start20180704\_115659"

"demo.nsh-start20180704\_114447"

""