IC Validator Distributed Processing Systems Quick Start Guide

Version T-2022.03-SP3, September 2022

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Feedback

Contents

About This Guide

This document provides an introduction to IC Validator distributed processing usage on SGE and LSF environments.

This preface includes the following sections:

New in This Release

Information about new features, enhancements, and changes, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the IC Validator Release Notes on the SolvNetPlus site.

Related Products, Publications, and Trademarks

For additional information about the IC Validator tool, see the documentation on the Synopsys SolvNetPlus support site at the following address:

https://solvnetplus.synopsys.com

You might also want to see the documentation for the following related Synopsys products:

- Synopsys® Custom Compiler™
- Synopsys IC Compiler™
- Synopsys IC Compiler™ II
- Synopsys Fusion Compiler™
- Synopsys StarRC™
- Synopsys IC Validator WorkBench

Feedback

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`Courier`	Indicates syntax, such as `write_file`.
`Courier italic`	Indicates a user-defined value in syntax, such as `write_file` *`design_list`*
`Courier bold`	Indicates user input—text you type verbatim—in examples, such as `prompt>` **`write_file top`**
`Purple`	• Within an example, indicates information of special interest. • Within a command-syntax section, indicates a default, such as `include_enclosing =` **`true`** `| false`
[ ]	Denotes optional arguments in syntax, such as `write_file [-format` *`fmt`*`]`
...	Indicates that arguments can be repeated as many times as needed, such as *`pin1 pin2 ... pinN`*.
|	Indicates a choice among alternatives, such as `low | medium | high`
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Bold	Indicates a graphical user interface (GUI) element that has an action associated with it.
Edit > Copy	Indicates a path to a menu command, such as opening the **Edit** menu and choosing **Copy**.
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.

Customer Support

Customer support is available through SolvNetPlus.

Accessing SolvNetPlus

The SolvNetPlus site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNetPlus site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNetPlus site, go to the following address:

https://solvnetplus.synopsys.com

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to sign up for an account.

If you need help using the SolvNetPlus site, click REGISTRATION HELP in the top-right menu bar.

Contacting Customer Support

To contact Customer Support, go to https://solvnetplus.synopsys.com.

Statement on Inclusivity and Diversity

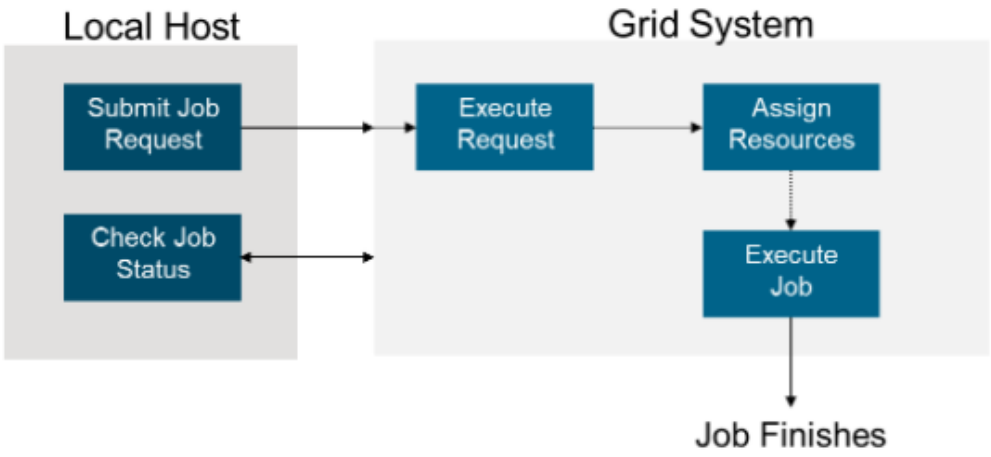
Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1

Getting Started With IC Validator on the SGE Grid System

IC Validator can run in a variety of different distributed processing situations. After obtaining the resources from the grid, IC Validator handles all of the distributed processing (DP) execution.

This section guides you in using IC Validator with distributed processing on SGE environments, and it illustrates how to launch IC Validator, monitor a submitted job, and add CPUs to a running job, as shown in the following figure.



Launching IC Validator Using SGE

To launch IC Validator, request to run the SGE binary, qsub from your local host.

```
qsub SGE_OPTIONS icv ICV_OPTIONS
```

The basic SGE command-line options are described in Command-Line Options for SGE.

Feedback

Examples of Using qsub to Launch IC Validator

The following examples demonstrate how to use qsub to launch IC Validator.

Examples

1. Request to launch IC Validator in the current directory using 16 CPUs and bnormal machines with at least 5 GB of free memory per host.

```
qsub -b y -cwd -V -pe mt 16 -P bnormal -l mem_free=5G  \
   icv -host_init SGE \
   ICV_OPTIONS
```

2. Request 40 CPUs using two dp20 bnormal machines with at least 25 GB of free memory per host. The qsub request returns two hosts each with 20 CPUs.

```
qsub -b y -cwd -V -pe dp20 40 -P bnormal -l mem_free=25G  \
   icv -host_init SGE \
   ICV_OPTIONS
```

3. Request 30 CPUs on the HostA.

```
qsub -b y -cwd -V -pe mt 30 -P bnormal -l hostname=HostA  \
   icv -host_init SGE \
   ICV_OPTIONS
```

   As an alternative to using the `-host_init` command-line option, set:

```
qsub -b y -cwd -V -pe mt 30 -P bnormal -l hostname=HostA  \
   icv -host_init HostA:30 \
   ICV_OPTIONS
```

The IC Validator command call and options are provided in a shell script file called icv_options.csh:

```
qsub SGE_OPTIONS icv_options.csh
```

icv.options.csh:

```
#!/bin/csh -f icv -host_init SGE  ICV_OPTIONS
```

In the *ICV_OPTIONS*, you must provide the list of command-line options for IC Validator that are specific to your IC Validator run. If, after running qsub, you do not get a successful submission error, check your qsub options syntax.

Monitoring the Job Status in SGE

To determine if your IC Validator job was launched successfully, type

```
qstat
```

The output of qstat provides the state of your current jobs. You can search for your job name and see if it is running (r) or waiting (qw). If your job is not in the qstat output list, check your qsub command output for any errors. If there are no errors, after qsub you should see a message indicating that “your job has been submitted.” For example, if you make a `quser` request,

```
qsub -b y -cwd -V -pe mt 16 -P bnormal -N -l mem_free=5G  \
   icv -host_init SGE \
   ICV_OPTIONS
```

After you submit your request in the output of qstat, you should see a job called `run1` using 16 slots. If there are resources to allocate the request, the status will be running (r). See qstat output next. Notice that the output format can change depending on your grid configuration.

```
job-ID  prior   name  user  state  submit/start at      queue       slots
-----------------------------------------------------------------------
0001    0.0361  run1  user  r      07/03/1991 23:00:48  q@HostName  16
```

If the job is scheduled but waiting, the output will be:

```
job-ID  prior   name  user  state  submit/start at      queue       slots
-----------------------------------------------------------------------
0001    0.0361  run1  user  w      07/03/1991 23:00:48              16
```

To end your job, use the qdel command along with the job id from the qstat output:

```
qdel job_id
```

If your job is in waiting status for a long time: Check your qsub options to ensure resources are available to allocate the job. Also, make sure it is possible to meet the resource constraints in your current grid. You can relax your constraints according to what is available in the grid.

To see the list of available resources, use the quser command. This command shows the resources you have access to and provides the information if the resources are free. You can also use qhost to see the status of each host. For more information, set

```
man quser
```

and

```
man qhost
```

To know if IC Validator is running correctly: Find the icv.log file in the run directory after the job starts. If the hosts were initialized correctly, you should see the following message for each host:

```
Host startup done: CPUS successes, 0 failures
```

Following this message, you see the host names assigned to your job. Finally, from your qstat, the host listed under queue is selected as your primary host:

```
Selecting HostName as Primary host
```

Next, you see the ICV % complete in the icv.log file.

If the job is not running correctly: Ensure you have the license server and ICV binaries set up correctly. The icv.log file should have more information. Check that you provided the IC Validator options correctly. If there are incorrect IC Validator options, you can find more information in the file icv.RESULTS file (icv.log is not created in this case).

You can also look at run_details/host_login to ensure that qsrh is automatically set to `qrsh -inherit`. Make sure that you have access to the right qsrh path for your grid configuration. You can manually specify this option with the `-host_login qrsh -inherit` IC Validator switch. Finally, look at run_details/host_login for any login error messages.

Adding a Host to a Running IC Validator Job in SGE

Use the following syntax to add a host to a running IC Validator job:

```
qsub SGE_OPTIONS icv -host_add SGE
```

For example:

```
cd DESIRED_RUN _DIRECTORY
   qsub -b y -cwd -V -pe mt 8 -P bnormal -N icv_add -l mem_free=20G \
   icv -host_add SGE
```

This command requests 8 CPUs from bnormal machines with at least 20 GB of free memory per host.

Use the following syntax to remove a host from a current IC Validator run:

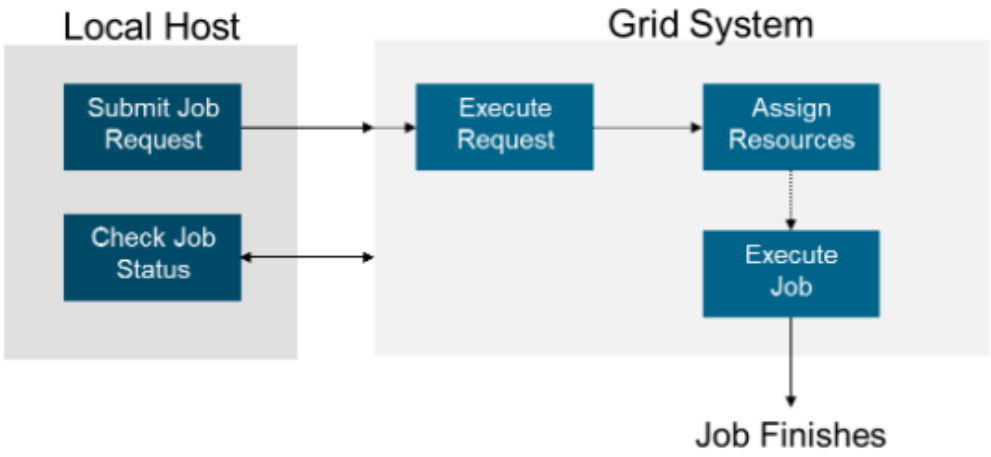
```
icv -host_remove
```

2

Getting Started With IC Validator on the LSF Grid System

IC Validator can run in a variety of different distributed processing situations. After obtaining the resources from the grid, IC Validator handles all of the distributed processing (DP) execution.

This section guides you in using IC Validator with distributed processing on LSF environments, and it illustrates how to launch IC Validator, monitor a submitted job, and add CPUs to a running job, as shown in the following figure.



Launching IC Validator

To launch IC Validator, request to run the The LSF binary, bsub from your local host.

```
bsub LSF_OPTIONS icv ICV_OPTIONS
```

The LSF_OPTIONS determine the resources that are assigned to the IC Validator job. For a list of the most common LSF command-line options, see Command-Line Options for LSF, or the bsub man page.

Feedback

Examples of Using bsub to Launch IC Validator

The following examples demonstrate how to use bsub to launch IC Validator.

Examples

1. Request to launch IC Validator in the current directory using 16 CPUs and normal machines with at least 5 GB of free memory per host.

```
bsub -cwd -n 16 -P normal -R "rusage[mem=5G]" \
   icv -host_init LSF \
   ICV_OPTIONS
```

2. Request 40 CPUs using two normal machines with at least 25 GB of free memory. The bsub request returns two hosts each with 20 CPUs.

```
bsub -cwd -n 40 -P normal -R "rusage[mem=25G]" -R "span[ptile=20]" \
   icv -host_init LSF \
   ICV_OPTIONS
```

3. Request 30 CPUs on the HostA.

```
bsub -cwd -n 30 -P normal -R "select[hname==HostA]" \
   icv -host_init LSF \
   ICV_OPTIONS
```

   As an alternative to using the `-host_init` command-line option, set:

```
bsub -cwd -n 30 -P normal -R "select[hname==HostA]" \
   icv -host_init HostA:30 \
   ICV_OPTIONS
```

The IC Validator command call and options are provided in a shell script file:

```
bsub LSF_OPTIONS icv_options.csh
```

icv.options.csh:

```
#!/bin/csh -f
   icv -host_init LSF ICV_OPTIONS
```

Monitoring the Job Status

To determine if your IC Validator job was launched successfully, run the LSF binary, bjobs from your local host.

```
bjobs
```

The output of bjobs provides the state of your current jobs. You can search for your job name and see if it is running (RUN) or waiting (PEND). If your job is not in the bjobs output

list, check your bsub command output for any errors. If there are no errors, after bsub you should see a message indicating that “your job has been submitted.” For example, if you make a bjobs request,

```
bsub -cwd -n 16 -P normal -J run1 -R rusage[mem=5G] \
   icv -host_init LSF \
   ICV_OPTIONS
```

You should see a job called `run1` after you submit your request in the bjobs output. If there are resources to allocate to the request, the status will be running (RUN). See bjobs output next. Notice that the output format can change depending on your grid configuration.

```
JOBID  USER  STAT  QUEUE   FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
-------------------------------------------------------------------------
0001   user  RUN   normal  localhost  HostA      run1      Jul 07 23:00
```

If the job is scheduled but pending, the output will be:

```
JOBID  USER  STAT  QUEUE   FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
-------------------------------------------------------------------------
0001   user  PEND  normal  localhost             run1      Jul 07 23:00
```

If you need to end your job, use the job id from the bjobs output and the command:

```
bkill JOBID
```

If your job is in pending status for a long time: Check your bsub options to ensure resources are available to allocate the job. Also, make sure it is possible to meet the resource constraints in your current grid. You can relax your constraints according to what is available in the grid.

To see the list of available resources, use the bhosts command. This command shows the resources you have access to and provides the information if the resources are free. For more information, set

```
man bhosts
```

To determine if IC Validator is running correctly: Find the icv.log file in the run directory after the job starts. If the hosts were initialized correctly, you should see the following message for each host:

```
Host startup done: CPUS successes, 0 failures.
```

Following this message, you see the host names assigned to your job. Finally, from bjobs, the host listed under EXEC_HOST is selected as your primary host:

```
Selecting HostName as Primary host.
```

Next, you see the ICV % complete in the icv.log file.

If the job is not running correctly: Ensure that you have the license server and ICV binaries set up correctly. The icv.log file should have more information. Ensure that you

provided the IC Validator options correctly. If there are incorrect IC Validator options, you can find more information in the file icv.RESULTS file (icv.log is not created in this case).

You can also look at run_details/host_login to ensure that qsrh is automatically set to `qrsh -inherit`. Make sure that you have access to the right qsrh path for your grid configuration. Finally, look at run_details/host_login for any login error messages that might exist.

Adding a Host to a Running IC Validator Job in LSF

Use the following command to add a host to a running IC Validator job:

```
bsub LSF_OPTIONS icv -host_add LSF
```

For example:

```
cd DESIRED_RUN_DIRECTORY
   bsub -cwd -n 8 -P normal -R "rusage[mem=20G]" \
   icv -host_add LSF
```

This command requests 8 CPUs from bnormal machines with at least 20 GB of free memory per host.

Use the following syntax to remove to remove a host from a current IC Validator run:

```
icv -host_remove
```

Feedback

A

Command-Line Options for SGE

The basic SGE command-line options and IC Validator command-line options for SGE are described in Table 1 and Table 2.

Table 1 SGE Command-Line Options

Option	Description
`-b y`	Enables the execution of a binary. This setting is required to start IC Validator binary.
`-cwd`	Runs IC Validator in the current working directory. If this switch is not provided, it runs IC Validator in your home directory.
`-V`	Exports current environment variables to the context of the submitted job. This setting is required to export license configuration and IC Validator binaries location.
`-N`	Names your job. This setting is optional but recommended to identify your jobs.
`-I` *`RESOURCE_CONSTRAINTS`*	Allows you to constraint the machine resources, such as OS, hostname, memory, and so on. For example, `-l` *`mem_free=5G`*.
`-pe` *`PE_INSTANCE NUMBER_CPUS`*	Specifies the parallel environment to instantiate. To obtain a list of available instances in the grid, set `qconf -spl`. You must choose the number of CPUs according to the instance type. For example, if the instance is a 20-CPU box (`dp20`), the number of CPUs must be a multiple of 20. The `-pe dp20 40` command provides up to dp20 hosts. You can find an instance that allows any number of CPUs (`mt`). The `-pe mt 14` command provides 14 CPUs in one or multiple hosts.

Table 1 SGE Command-Line Options (Continued)

Option	Description
`-P PROJECT`	Specifies the project to which the job submission is assigned. To see the list of project instances in the grid, set `qconf -sprjl`. A common default project name in SGE is bnormal. For more advanced options, see the `qsub` man page.

Table 2 IC Validator Command-Line Option With SGE

Option	Description
`-host_init SGE`	Required to start the ICV validator job on the host or multiple hosts returned from the qsub request. The `SGE` argument is not required, but it simplifies the run. SGE automatically determines host names and the number of CPUs from the grid engine environment by reading the PE_HOSTFILE environment variable. An alternative to using `-host_init SGE` is to set `-host_init hostname:number_of_cpus`. In this case you must know the host names and the grid engine assigned to your request. See Example 3. You must add the remaining IC Validator command-line options that are specific to your run `ICV_OPTIONS`, such as input database, runset, schematic, and so on.

B

Command-Line Options for LSF

The basic LSF command-line options and IC Validator command-line options for LSF are described in Table 3 and Table 4.

Table 3 LSF Command-Line Options

Option	Description
`-cwd`	Runs IC Validator in the current directory. If this switch is not provided, it runs IC Validator in your home directory.
`-R` *RESOURCE_CONSTRAINTS*	Allows you to constraint the machine resources, such as OS, host name, memory, and so on. For example, `-R select[hname!='hostA'] rusage[mem=5G]`. To obtain a detailed list of the resources you can configure in your grid, use the `lsinfo` command.
`-J`	Names your job. This setting is optional but recommended to identify your jobs.
`-P` *PROJECT*	Specifies the project to which the job submission is assigned. To see the list of project instances in the grid, set `bqueues`. In most cases, this option is optional, unless it is a requirement for your local grid.

Feedback

Table 4 IC Validator Command-Line Option With LSF

Option	Description
`-host_init` *`LSF`*	Required to start the ICV validator job on the host or multiple hosts returned from the bsub request. The `LSF` argument is not required, but it simplifies the run. LSF automatically determines host names and the number of CPUs from the grid engine environment by reading the LSB_HOSTS or LSB_MCPU_HOSTS environment variables. An alternative to using `-host_init` *`LSF`* is to set `-host_init hostname:number_of_cpus`. In this case you must know the host names and the grid engine assigned to your request. See Example 3. You must add the remaining IC Validator command-line options that are specific to your run *`ICV_OPTIONS`*, such as input database, runset, schematic, and so on.