Schlumberger

Altair | PBS Works™

Running Schlumberger Simulators in a PBS Professional Computing Environment:

Integration White Paper and How-To Guide

Owen Brazell and Steve Messenger - Schlumberger
Graham Russell and Dario Dorella - Altair Engineering

Contents

1 Abstract

The ECLIPSE* and INTERSECT† family of reservoir simulation software offers the industry's most complete and robust set of numerical solutions for fast and accurate prediction of dynamic behavior, for all types of reservoirs and degrees of complexity—structure, geology, fluids, and development schemes. In order to fully leverage these capabilities and ensure optimal performance and throughput of ECLIPSE and INTERSECT, it is important to provide a reliable workload management environment that can scale as needed to support large and globally distributed environments, with the features needed to simplify usage for reservoir engineers.

In response to user requests, Altair Engineering has collaborated with Schlumberger to deliver a tightly integrated version of ECLIPSE for the PBS Professional® workload management product, for optimized reservoir simulation. The integration requires no changes to the front end or user experience – the integrated solution abstracts the underlying workload management system away from the end user so there is no need for direct interaction with PBS Professional. Full license management is also included. The license requirements for each job are evaluated to ensure sufficient tokens are available, for both individual jobs and multiple realization jobs, before jobs are dispatched to run.

In this paper we document the integration of the PBS Professional solution with the simulation software and provide step-by-step instructions for configuration and use. The document includes detailed information on:

- Integration architecture
- What's included
- How-to guide for configuration and usage
- Steps for getting started

*Mark of Schlumberger
†Mark of Schlumberger, Chevron and TOTAL

2 About Schlumberger ECLIPSE Software

The ECLIPSE family of reservoir simulation software offers the industry's most complete and robust set of numerical solutions for fast and accurate prediction of dynamic behavior, for all types of reservoirs and degrees of complexity—structure, geology, fluids, and development schemes. ECLIPSE software covers the entire spectrum of reservoir simulation, specializing in blackoil, compositional and thermal finite-volume reservoir simulation, and streamline reservoir simulation. By choosing from a wide range of add-on options—such as coalbed methane, gas field operations, calorific value-based controls, reservoir coupling, and surface networks—simulator capabilities can be tailored to meet your needs, enhancing the scope of reservoir simulation studies. ECLIPSE reservoir simulators have been the benchmark for commercial reservoir simulation for more than 25 years because of their breadth of capabilities, robustness, speed, parallel scalability, and unmatched platform coverage.

3 About PBS Professional

Awarded the 'Best Use of HPC in Oil & Gas' award by HPCwire readers in 2011[1], Altair is a leader in the energy industry. Altair's PBS Professional is one of the most popular and widely used workload management products available. PBS Professional provides a flexible, on-demand computing environment that allows users of ECLIPSE simulators to easily share diverse computing resources across geographic boundaries to increase throughput for their ECLIPSE processing.

PBS Professional enables geoscientists to focus on their work by providing one easy-to use interface to all computing resources. IT managers can use PBS Professional to make the greatest possible use of available computing cycles and to dynamically distribute workloads across wide area networks. PBS Professional keeps large MPI jobs running, automatically detecting failed nodes and rescheduling around them, to deliver results as quickly as possible. The product has a proven history of reliability and scalability with leading high performance computing (HPC) facilities, including many Top500 resources. For more information about PBS Professional, visit http://www.pbsworks.com/pbs-professional.

4 Integration Overview

With the 2013 release of the Schlumberger Simulation software, PBS Professional is now able to handle all aspects of workload management and license management for ECLIPSE and INTERSECT jobs. This tight integration of the simulators with PBS Professional takes care of all aspects of job creation, job monitoring, license management and data management; the end user does not need to interact directly with PBS Professional. The integrated solution provides an abstraction layer between the back-end workload management system and the end user, so the user experience -- including creating and managing jobs and how they interact with the reservoir simulators -- remains unchanged.

The integration work is focused on modification of the simulation launch script '*eclrun'* to support the PBS Professional resource request format and to provide all license feature requests so that PBS Professional can carry out full license checking before job dispatch. The 2013.1 releases of ECLIPSE and INTERSECT ships with PBS Professional fully integrated into the *eclrun* script. The local site will need to install and configure PBS Professional and to integrate the supplied job submission plug-in into their simulation environment and with their local FLEXlm license manager.

Specific areas of integration, as well as configuration details and guidelines, are described in the next section.

[1] Press release: http://www.pbsworks.com/newsdetail.aspx?news_id=10616&news_country=en-US

5 What's Included with the Integration Package

5.1 eclrun Script Modification

The simulation launch script '*eclrun*' is responsible for generating job and resource requests for the back-end workload manager. In the integrated version, *eclrun* has been modified to support the resource request format of PBS Professional and to provide all license feature requests to allow PBS Professional to carry out full license checking before dispatching jobs to run. PBS Professional will only execute jobs if all requested resources (CPU, memory etc.) are available and all required license tokens are free. The license management logic for multiple realization jobs and non-multiple realization jobs is fairly complex and is handled by PBS Professional to ensure jobs do not fail because license tokens are not available. This is described in more detail in Section 6. Eclrun 2013.1 or later is required for all functionality to work.

PBS Professional supports integration of external code at various points during a job's execution using a call-out mechanism known as "plug-ins" (also referred to as "hooks" in code examples). Plug-ins are Python scripts that are able to change the behavior of the scheduler, query all aspects of a job, potentially modify resource requests etc. for a job, and make decisions about whether a job can proceed to the next step in the scheduling and dispatch process. The license checking logic for simulation jobs has been incorporated into the *runjob* plug-in that is called once all requested resources are available, just before a job is dispatched to run. The *runjob* plug-in checks the license features required for the job against the currently available licenses in the license management server. If all licenses are available, the *runjob* plug-in allows the job to start executing; otherwise the plug-in forces the job to remain queued. Because the *runjob* plug-in is called before all jobs are started, it checks the current job to determine whether it is an ECLIPSE/INTERSECT job or not.

Note that the complex license management discussed here is only applicable to ECLIPSE/INTERSECT jobs. All other jobs are allowed to execute immediately without any additional license checks.

Full details of how to install and configure the *runjob* plug-in for the simulators are given later in this paper.

5.2 Job Flow

PBS Professional jobs are created and submitted from the *eclrun* script. The flow through PBS Professional and the complex license token checks are illustrated in Figure 1.

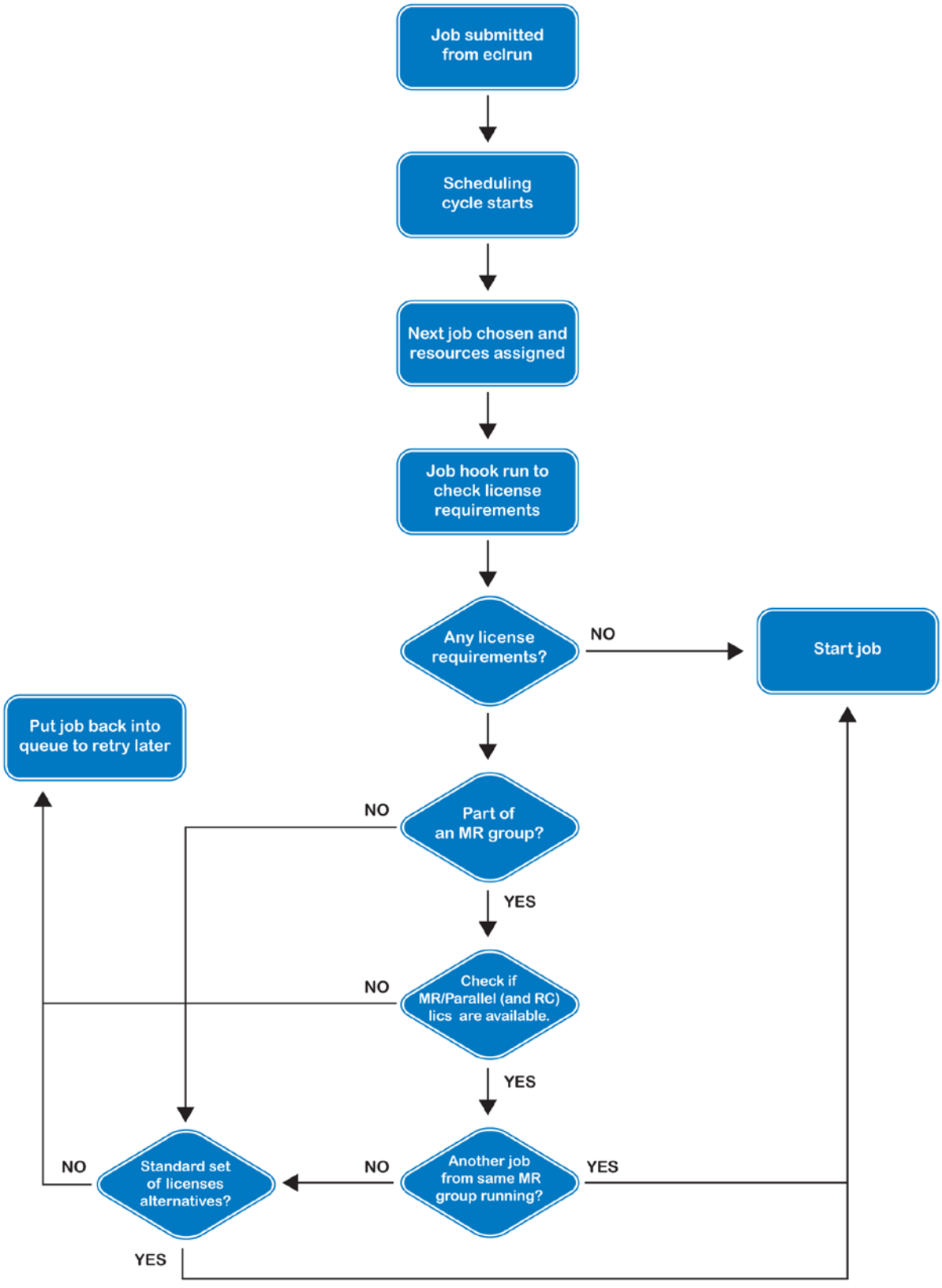

Figure 1: ECLIPSE job flow through PBS Professional with complex license token checks

5.3 License Checking and Management

Checking the licenses for simulation jobs is more complex than it is for most applications because multiple license alternatives can be specified and must be checked prior to running the job. Most applications request a certain number of license tokens and simply require all the tokens to be available before the job can run. Simulation jobs are able to request a number of license alternatives or scenarios, represented by rules (written as logical 'OR' statements) about how job requirements need to be met. As soon as one of the license alternatives is satisfied, the license requirements for the job are considered to be met and the job can start.

By default, PBS Professional does not support alternative feature or license requests that are separated by logical ORs. To implement this additional functionality for the simulation integration, license checking logic has been incorporated into the *runjob* plug-in. License alternatives are specified in a single resource request and are delimitated by '+' characters, which represents the OR in the Boolean logic expression.

Example:
licA=X:licB=Y+licA=K:licB=M
This license request specifies the job is able to run if:

X tokens of licA AND Y tokens of licB are available
OR
K tokens of licA AND M tokens of licB are available

The construction of these complex license alternatives happens automatically inside the *eclrun* script as part of the PBS job creation, so the user need not be concerned with it. Calculating these needs is part of the integration work that has been done, so nothing changes at the front end for the user.

In addition, FlexLM license management can be prone to race conditions caused by applications not necessarily checking out their licenses immediately after they start. Consider the situation in which there are only enough available tokens for one additional instance of an application. If the scheduler happens to query the number of available tokens just after a job has started but has not yet checked out its tokens, it is possible another job could be started. This creates a situation in which there are only enough available tokens for one instance of the application yet two jobs are running that expect to be able to start the application. The first application request will succeed and the second will fail, with the most likely result being the job will fail.

The integration of ECLIPSE software and the PBS Professional solution provides a workload management environment that minimizes these FlexLM race conditions. Enhancements have been incorporated into the runjob plug-in to throttle the starting of consecutive instances of the simulators. A tunable timer parameter *eclipseInterval* is available that defines the minimum duration between starts of the simulation jobs. Details on changing this parameter are given in Section 6.2.3.2.

6 Using ECLIPSE with PBS: How-To Guide

Integration of the ECLIPSE reservoir simulation suite and PBS Professional for job submission is configured in the *eclrun* script. The script calculates required compute and license resources, creates the PBS Professional job submission command, and submits the job.

Guidance on configuration is given in the following sections.

6.1 Requirements

To properly submit simulation jobs to PBS Professional, it is important to verify several items, including proper environment configuration with regard to filesystems and authentication.

This document does not cover product installations for ECLIPSE/INTERSECT and PBS Professional. For resource information, please refer to Section 8 and the installation manuals.

Before proceeding, verify the following:

1. You are familiar with workload management software concepts and procedures, such as those relevant to PBS Professional (knowledge of PBS Professional in particular is not required).
2. You have a valid and updated backup of your configuration.
3. You have ECLIPSE/INTERSECT version 2013.1 or later. (If you need support for previous versions of the simulators, contact Schlumberger.)
4. You have PBS Professional version 11.2 or later installed.
5. You have administrative rights on the computing environment (so you can perform the tasks described below).

6.2 Configuring PBS Professional

PBS Professional is very flexible and extensible, and it is able to accommodate very different application scheduling and launch strategies. By leveraging plug-ins and custom resources, the application integration demonstrates its adaptability.

6.2.1 Tune job session process limits

While not absolutely required, it is strongly advised (and under certain circumstances needed) to first set up the pbs_mom processes running environment properly so that user job scripts spawned from them will not encounter problems running MPI libraries over Infiniband networks: job sessions inherit their limits from generating pbs_mom process.

To do this:

1. On each execution node, edit the file $PBS_EXEC/lib/init.d/limits.pbs_mom (which is sourced by /etc/init.d/pbs at service startup) so that it looks like the following sample:

```
#This file will be sourced by the PBS startup script, pbs_init.d.
#It is here only for binary compatibility with previous releases.
#Feel free to replace its contents.
if [ -f /etc/sgi-release -o -f /etc/sgi-compute-node-release ] ; then
    MEMLOCKLIM=`ulimit -l`
    NOFILESLIM=`ulimit -n`
    STACKLIM=`ulimit -s`
    ulimit -l unlimited
    ulimit -n 16384
    ulimit -s unlimited
else
    ulimit -l unlimited
    ulimit -m unlimited
    ulimit -s unlimited
fi
```

2. Then restart pbs_mom using `service pbs restart`
    - Follow the guidelines in the PBS Professional Administration Guide to avoid harming any running jobs.

6.2.2 Create custom resources for simulation dynamic licensing

To allow PBS Professional to store job specific simulator license information, a few custom resources must be added to PBS Professional by appending the following lines in ${PBS_HOME}/server_priv/resourcedef:

```
debug_hooks               type=string
eclipse_alternatives      type=string
eclipse_mr_key            type=string
```

- **"debug_hooks"** is used to enable or disable logging debug information being written inside the pbs_server log file. Debug information produced by the *runjob* plug-in (or hook) used in this integration is extensive and grows rapidly with the number of jobs in the system and at every scheduling cycle; thus debug information should be turned off during normal operations.
- **"eclipse_alternatives"** will hold licensing alternatives for an simulation job. These are evaluated by the *runjob* plug-in against the tokens available from the simulation license server at the moment PBS Professional scheduler runs the job. License server location will be specified as the environment "SLBSLS_LICENSE_FILE" automatically by *eclrun* (i.e. "SLBSLS_LICENSE_FILE=27000@license.server").
- **"eclipse_mr_key"** holds the group identifier for jobs that are part of a multiple realization group. Such jobs are subjected to a further check on available licenses once they pass the license check for "eclipse_alternatives". If a job from the same group is already running, they are let go, but otherwise license alternatives from job submission time environment variable "ECL_LICS_REQD" are checked against the simulation license server using the same logic as that in the “eclipse alternatives” test.

After creating these custom resources, follow the PBS Professional Administration Guide instructions (http://www.pbsworks.com/SupportDocuments.aspx) to restart pbs_server in order for the newly created custom resource definitions to be taken in.

6.2.3 Create eclipse_licsched runjob plug-in in PBS Professional

The purpose of the simulation *runjob* plug-in is to hold a simulation job queued until a valid license alternative is available from the license server.

Creating this custom plug-in is a necessary step to handle simulation license scheduling, because this is where the simulation license scheduling logic gets plugged into the job scheduler. The simulation *runjob* plug-in will be evaluated every time an ECLIPSE job is able to start on the available computing resources, to see if a valid license alternative is available.

6.2.3.1 Brief description of ECLIPSE license logic

An ECLIPSE job asks for licenses differently, depending on whether the job is standalone or part of a multiple realization group.

If it is a standalone job, then it is enough to check the license server for the availability of one of the options reported in the job's 'eclipse_alternatives' property. This is a list of alternatives separated by "+", and if one is available the job can run.

If the job is part of a Multiple Realization group, then after verifying the availability of licenses against eclipse_alternatives, an additional condition must be checked: if a job from the same Multiple Realization group (identified by job's property eclipse_mr_key) is already running, then this new job can run as well, as it will level[2] with its companions. If no other job from the same group is already running, license availability must be checked against the additional alternatives listed in the job's ECL_LICS_REQD environment variables.

To obtain a full view of a submitted job's properties and environment, use the command "qstat -f" (see also PBS Professional User Guide).

6.2.3.2 Create the runjob plug-in

In this section we provide only the commands. For any additional information on PBS Professional plug-ins, please refer to the PBS Professional Administration Guide.

Before proceeding, please confirm that you have an up-to-date version of eclipse_licsched.py and that it is aligned with the simulation version you are running. If uncertain contact Schlumberger technical support.

After copying the file eclipse_licsched.py on your pbs_server host, edit the paths in lines 7 to 17 to make sure they are correct for your computing environment setup. (The given paths from the next example are the default -- you may have different ones.)

(Note: The *runjob* plug-in is executed by pbs_server process on the pbs_server host;

[2] (need to define leveling)

thus pathnames and permissions must be valid on the pbs_server host.)

```
'''
 These variables need to be customized to match pbs_server host configuration
'''
pbsExec='/opt/pbs/default'
lmutil='/usr/ecl/tools/linux_x86_64/flexlm118/lmutil'
eclipseInterval=30
eclipseLastRun='/tmp/ECLIPSELastRun'
'''
 Please do not change anything below this comment,
 unless you know what you're doing.
'''
```

- **"pbsExec"** is ${PBS_EXEC} from /etc/pbs.conf.
  "lmutil" is where the ECLIPSE Flexlm lmutil command is located.
  "eclipseInterval" is the minimum amount of time that will pass before dispatch of another ECLIPSE job. This is meant to avoid race conditions on license check out.
- **"eclipseLastRun"** is the path to a lock file through which the submission rate for ECLIPSE jobs is throttled to avoid FlexLM race conditions.

Once you have reviewed and modified what's necessary in the *runjob* plug-in code, you can import it inside PBS Professional complex by executing the following commands:

```
[root@e-server ~]# qmgr
Max open servers: 49
Qmgr: create hook eclipse_licsched
Qmgr: set hook eclipse_licsched event = runjob
Qmgr: import hook eclipse_licsched application/x-python default
eclipse_licsched.py
```

Note: Although eclipse_licsched.py was written in a way that it will not interfere with non-ECLIPSE jobs, if you have other plug-ins in place you should verify that those are not interfering with ECLIPSE jobs and eclipse_licsched.

6.2.4 Optimizing PBS Professional and MPI libraries used by ECLIPSE

Though not strictly necessary, it is strongly recommended that you perform these additional steps in order to let PBS Professional accurately account and clean up distributed jobs.
The MPI libraries we cover in this document are:

- Intel MPI
- Platform MPI

Modify $PBS_HOME/pbs_environment and add $PBS_EXEC/bin to PATH and set PBS_RSHCOMMAND as shown:

```
TZ=Europe/Rome
PATH=/bin:/usr/bin:/opt/pbs/default/bin
PBS_RSHCOMMAND=/usr/bin/ssh
MPI_REMSH=/opt/pbs/default/bin/pbs_tmrsh
```

6.2.4.1 Intel MPI specifics

To obtain the best performance with PBS Professional it is necessary to modify two parts of the Intel MPI distribution.

The first recommended modification is for the file mpdboot.py in the function launch_one_mpd(); this script launches mpd and it is best to tune the section holding default parameters for different shells. This is needed to avoid the positional parameter "-n" being passed through:

```
if rshCmd == 'ssh':
     rshArgs = '-x -n -q'
elif rshCmd == 'pbs_tmrsh':
     rshArgs = ''
elif rshCmd == 'pbs_remsh':
     rshArgs = ''
else:
     rshArgs = '-n'
```

The second recommended modification is made so that mpd.py will avoid loss of connection with pbs_mom (we comment out the second fork()):

```
if self.parmdb['MPD_DAEMON_FLAG']: # see if I should become a daemon with no
controlling tty
   rc = os.fork()
   if rc != 0:                                  # parent exits; child in
background
      sys.exit(0)
      #os.setsid()                              # become session leader; no
controlling tty
      #signal.signal(signal.SIGHUP,signal.SIG_IGN)  # make sure no sighup when
leader ends
      ## leader exits; svr4: make sure do not get another controlling tty
      #rc = os.fork()
      #if rc != 0:
      #    sys.exit(0)
```

6.2.4.2 Platform MPI specifics

Setting the MPI_REMSH variable in the job's execution environment is the only step needed to make PBS Professional fully aware of this class of MPI jobs.

The MPI_REMSH content points to the command used to generate remote processes on execution hosts and if this is a PBS Professional aware tool, all job processes will be correctly tracked.

This setting needs to be done on all execution hosts, and an easy way to achieve this is to insert the setting in pbs_environment as just described

6.3 Submitting and Monitoring Jobs

Prior to running *eclrun*, the license server must be running and the user environment must be properly set up (exact steps will vary depending on your environment configuration):

```
source /usr/ecl/profile.csh
cd [dir containing input files]
eclrun FILENAME.dat
```

ECLIPSE *eclrun* has a wide and complex set of options, for references see the eclrun reference manuals or:

```
[user01@e-clipse parallel]$ eclrun –help
```

To submit to PBS Professional you need to specify proper values for the parameters identifying and characterizing your batch system:

```
[user01@e-clipse data]$ eclrun --subserver=localhost --queue=workq --
queuesystem=PBSPRO    --comm=ilmpi --np=4 eclipse ONEM1
Cleaning up...
Preparing job for submission...
Analyzing the input file (it may take a while)...
Message Job ONEM1 submitted to queue workq with job_id=215
Message Simulation is queued
```

eclrun also embeds monitoring capabilities to check job status and kill it:

```
[user01@e-clipse data]$ eclrun check ONEM1
Message Simulation is queued

[user01@e-clipse data]$ eclrun check ONEM1
Error Simulation may have aborted. Check the .OUT file for more details
Cleaning up...

[user01@e-clipse data]$ eclrun check ONEM1
Message The state file ONEM1.ECLRUN was not found. Your run has probably already
finished
[user01@e-clipse data]$ eclrun kill ONEM1
Cleaning up...
```

eclrun can also be used to list the available queues in the PBS Professional complex:

```
[user01@e-clipse ~]$ eclrun --queuesystem=PBSPRO --report-queues
fast prio workq
```

7 Troubleshooting

To obtain debug information from the plug-in written inside pbs_server log files, use the aforementioned “debug_hook” custom resource:

```
[root@e-server ~]# qmgr
Max open servers: 49
Qmgr: set server resources_available.debug_hooks = eclipse_licsched
```

Changes made via qmgr take effect immediately without having to re-run any daemons. Any modifications to the value of “debug_hooks” will automatically be actioned the next time the plug-in is run. Be aware that the amount of debugging information returned by the plug-in is comprehensive and can overwhelm a poorly dimensioned pbs_server or an extremely busy one. We strongly advise that debug should not be kept on during normal operations. The suggested debug technique is employed to stop scheduling, enable plug-in debug information, and use qrun to inspect individual jobs. The following is an example of what the plug-in logs for job 1234.pbs_server:

```
qmgr -c “set server scheduling=False”
qmgr -c “set server resources_available.debug_hooks = eclipse_licsched”
qrun 1234.pbs_server
```

Then review pbs_server log files and reset debugging and scheduling:

```
qmgr -c “unset server resources_available.debug_hooks
qmgr -c “set server scheduling=True”
```

The PBS integration kit for the simulation software is packaged on the ECLIPSE & INTERSECT DVD in the 3rd party/PBSPro directory.

8 Resources

Refer to the following resources for:

- Schlumberger reservoir simulation information: http://www.slb.com/services/software/reseng.aspx
- Details on the ECLIPSE software family: http://www.slb.com/services/software/reseng/eclipse.aspx
- Details on the PBS Works software family: http://www.pbsworks.com/Default.aspx
- PBS Professional documentation: http://www.pbsworks.com/SupportDocuments.aspx
- PBS Professional white papers and case studies: http://www.pbsworks.com/resources
- ECLIPSE services and support: http://www.slb.com/services/software/reseng.aspx
- PBS Professional services and support: http://www.pbsworks.com/PBSSupportForm.aspx

t +1 248.614.2400 • f +1 248.614.2411 • 1820 E. Big Beaver Rd. • Troy, MI 48083-2031 USA • www.pbsworks.com