Jaryba® SmartSuspend™

Integration with Platform LSF

Version 2.1.1

August 2012

Jaryba, Inc.

2068 Tesuque CT Reno, Nevada 89511

Disclaimer

The information contained herein is subject to change without notice. Jaryba, Inc. is not liable for errors or damages, incidental or consequential, in connection with the furnishing, performance, or use of this material.

Jaryba SmartSuspend software users are bound by the terms and conditions of the Jaryba, Inc. license agreement.

Copyright

Trademarks

Jaryba, the Jaryba logo and Jaryba SmartSuspend are trademarks or registered trademarks of Jaryba, Inc. or its subsidiaries in the United States and/or other countries.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Contents

Preface

This guide provides:

- Instructions for integrating Jaryba SmartSuspend software with Platform LSF
- Instructions for submitting jobs to LSF using SmartSuspend

This guide is intended for SmartSuspend administrators and users.

Prerequisites

To use Jaryba SmartSuspend, you should have working knowledge of the operating system and shell for the machines on which you are performing the operations described in this guide.

In addition, you should have in-depth knowledge of any third-party applications you are using in conjunction with the SmartSuspend software.

Conventions Used in This Guide

Commands, URLs, and any text that you enter appear in **`this bold font`**.

Computer output, file names, and locations appear in `this font`.

Elements on the user interface appear in **bold like this**.

Important new terms appear in *italics like this*.

Variables, where you need to substitute site- or installation-specific details, appear in brackets and italics <*`like_this`*>.

Contacting Jaryba

For product operation and support inquiries send email to:
`support@jaryba.com`

The following information is needed when you contact Jaryba, Inc. for support:

- Company name, contact name, email address, and phone number
- Software version number: execute the software library name to obtain the version number, for example: `/lib/libssr.so` or `/lib64/libssr.so`
- Platform details: hardware, operating system type, and version

- Problem background information: describe the problem in as much detail as possible and any actions you have attempted to resolve the problem. In addition, be sure to include available log files and error message text.

You can also visit our support Web site at:
http://support.jaryba.com/

For general inquiries, send email to:
info@jaryba.com

For sales inquiries, send email to:
sales@jaryba.com

For general information about Jaryba and Jaryba SmartSuspend visit our Web site at:
http://www.jaryba.com/

1 Integrating SmartSuspend with LSF

Overview

Integrating Jaryba SmartSuspend with Platform LSF allows you to leverage the powerful suspend and resume capabilities of SmartSuspend, while using the familiar Platform LSF interface to submit and monitor jobs.

This document provides instructions for integrating SmartSuspend with LSF, including configuration details and working with an example LSF integration wrapper script supplied by Jaryba. This documentation assumes that you have prerequisite and working knowledge about configuring, administering, and using both the Jaryba SmartSuspend and the Platform LSF software.

Configuring the SmartSuspend Integration Script

The LSF integration script is included in the SmartSuspend distribution. After installing SmartSuspend according to the instructions in the *Using SmartSuspend* documentation, you will find the integration script, `ssrlsf.sh`, in `/usr/share/smartsuspend`.

The `ssrlsf.sh` integration script defines default values for the variables listed below:

```
SSRLSF_LOGDIR_ROOT_DEFAULT=<specify directory>
SSR_SAVE_PATH_DEFAULT=/tmp
SSR_SIG_RESUME_DEFAULT=58
SSR_SIG_SUSPEND_DEFAULT=56
```

- `SSRLSF_LOGDIR_ROOT_DEFAULT` specifies the default path for `SSRLSF_LOGDIR_ROOT`, which is the base directory where `ssrlsf.sh` creates the subdirectory `<SSRLSF_LOGDIR_ROOT>/<LSB_JOBID>`. This subdirectory contains the SmartSuspend status directory (required for storing job state communication data and process IDs (PIDs)), in addition to various log files. Ideally, `SSRLSF_LOGDIR_ROOT_DEFAULT` is a location on a shared file system, because the directories for jobs started on all machines are then kept in one location. A location on a local file system may also be used if desired.

Note: There is no valid default set for this variable; you must specify a valid path to which all users have read, write and execute permissions.

For instance, if:

```
SSRLSF_LOGDIR_ROOT_DEFAULT=/path/to/ssrlsf/logdir_root
```

use the following commands as root:

```
$ mkdir -p /path/to/ssrlsf/logdir_root
$ chmod 1777 /path/to/ssrlsf/logdir_root
```

- SSR_SAVE_PATH_DEFAULT sets the default location in which SmartSuspend stores the memory for suspended jobs. The default location is set to /tmp (which is also the default location for ssrcmd), but you can specify a different location by setting this variable. Override this default by passing the "-d <pathname>" argument to ssrlsf.sh, or by setting the SSR_SAVE_PATH environment variable.

- SSR_SIG_RESUME_DEFAULT and SSR_SIG_SUSPEND_DEFAULT variables define the signals that ssrlsf.sh sets to instruct SmartSuspend to use to suspend and resume jobs. Note that the values used for the defaults in ssrlsf.sh may differ from the default values that ssrcmd assumes (10 for suspend, and 12 for resume). Although you would not normally need to, you can change the signals that SmartSuspend uses, if one of your applications uses these signal values internally for another purpose.

Note: Be aware that $SSR_SIG_RESUME_DEFAULT and $SSR_SIG_SUSPEND_DEFAULT will not override any values already specified for $SSR_SIG_RESUME or $SSR_SIG_SUSPEND in your shell/application startup script or set from the command line through the -a (--sig_suspend) or -b (--sig_resume) options. If you encounter problems suspending a job, make sure you do not have variable specification conflicts.

Configuring LSF to Use SmartSuspend

The standard LSF product provides a preemptive scheduling feature that allows pending high-priority jobs to take resources away from running jobs of a lower priority. To keep the integration as simple and transparent as possible, we take advantage of LSF's queue-based model for preemptive scheduling, using SmartSuspend technology for handling the underlying job suspension and resumption operations.

To preempt based on license availability, see S.

You can specify an LSF queue as:

- Preemptive—Jobs in a *preemptive* queue can preempt jobs in any queue of lower priority, or jobs in specified queues only.

- Preemptable—Jobs in a *preemptable* queue can be preempted by jobs from any queue of a higher priority, or by jobs in specified queues only.

Add SmartSuspend Preemptive Scheduling Queues

Add a preemptive queue and a preemptable queue to `LSB_CONFDIR/<cluster_name>/configdir /lsb.queues`.

For example, add a preemptive queue called "hipri" and a preemptable queue called "lowpri" using the following queue definitions:

```
Begin Queue
QUEUE_NAME       = hipri
DESCRIPTION      = For HIGH Priority Jobs
JOB_STARTER      = /usr/share/smartsuspend/ssrlsf.sh -l license_server --
                   "%USRCMD"
JOB_CONTROLS     = SUSPEND[/usr/share/smartsuspend/ssrlsf.sh –s ] \
                   RESUME[/usr/share/smartsuspend/ssrlsf.sh –r ]

PREEMPTION       = PREEMPTIVE[lowpri]
RERUNNABLE       = YES
PRIORITY         = 100
NICE             = 20
End Queue


Begin Queue
QUEUE_NAME       = lowpri
DESCRIPTION      = For LOW Priority Jobs
JOB_STARTER      = /usr/share/smartsuspend/ssrlsf.sh -l license_server --

                   "%USRCMD"
JOB_CONTROLS     = SUSPEND[/usr/share/smartsuspend/ssrlsf.sh –s ] \
                   RESUME[/usr/share/smartsuspend/ssrlsf.sh –r ]

PREEMPTION       = PREEMPTABLE[hipri]
RERUNNABLE       = YES
PRIORITY         = 50
NICE             = 20
End Queue
```

The configuration `PREEMPTION = PREEMPTIVE[lowpri]` means that only jobs in the lowpri queue can be preempted by jobs in the hipri queue. If no queue name is specified for this configuration, e.g. `PREEMPTION = PREEMPTIVE`, then jobs in any queue of lower priority can be preempted. Likewise, specifying a queue name (or names) for a preemptable queue makes it preemptable only by jobs in the specified queues.

The addition of job controls for the hipri queue is optional, if you do not intend for jobs in this queue to be preempted. However, adding the job controls enables the administrator or user to manually suspend and resume these jobs using SmartSuspend.

Submitting Your Job to LSF Using SmartSuspend

Assuming you are using a `bsub` script to submit your job, prepend `/usr/share/smartsuspend/ssrlsf.sh` to the name of the application you are going to run to indicate that you are starting a job using SmartSuspend:

```
/usr/share/smartsuspend/ssrlsf.sh -- <application_name>
<application_arguments>
```

Note that if `ssrlsf.sh` is already in your execution path, you don't need to specify the full path.

`ssrlsf.sh` takes the same arguments, in the same format, that `ssrcmd` does. However, any `-n` argument passed to `ssrlsf.sh` will be ignored, since LSF is configured to place the `SSR_STATUS_PATH` directory in a specific location. Similarly, any value set for `SSR_STATUS_PATH` will be overwritten by `ssrlsf.sh`. Refer to the *Using SmartSuspend* documentation for more information about `ssrcmd`.

Submit your job to either the preemptive queue (hipri) or the preemptable queue (lowpri) that you created for SmartSuspend jobs, for example:

```
bsub –q hipri < script.bsub
```

If there are more LSF job slots needed than are available, LSF automatically uses SmartSuspend to suspend enough jobs submitted to the lowpri queue to make room for jobs in the hipri queue. When enough job slots become available, LSF uses SmartSuspend to resume any suspended jobs in the lowpri queue. LSF's overall scheduling policies and priorities determine which jobs are suspended or resumed.

Parallel Applications Using MPI

To submit parallel applications using MPI, add the 'mpirun' command and the appropriate MPI option to your `bsub` script:

SmartSuspend MPI options are:

Option	Description
`-H, --hpmpi`	Enables HP-MPI support.
`-M, --mpich`	Enables MPICH MPI support.
`-O, --openmpi`	Enables OpenMPI support.

In this example, an MPICH application <mpi_app> is submitted with mpirun:

```
/usr/share/smartsuspend/ssrlsf.sh --mpich -- mpirun <mpi_app> <mpi_app_args>
```

Note that you do not specify any arguments to the MPI wrapper; host names and the number of ranks to be used are determined by LSF (through the "-n" option to bsub, or the "#BSUB -n <# of ranks>" directive in your bsub script.

Controlling Your LSF Job Using SmartSuspend

Your job can be suspended, resumed or killed with the standard LSF commands, but instead of using the default LSF operations, the commands use the underlying SmartSuspend operations:

- bstop <*job_id*>— LSF uses SmartSuspend to suspend CPU usage, page out memory on demand, and free up licenses if required.
- bresume <*job_id*>— LSF uses SmartSuspend to restore CPU and memory usage and allow the application to obtain any needed licenses as required.

Use the LSF command bkill to remove/kill the job; no action by SmartSuspend is necessary:

bkill <*job_id*>—LSF sends its default signals to kill your job.

SmartSuspend Integration with LSF License Scheduler

SmartSuspend integrates with Platform LSF License Scheduler to provide a way to preempt jobs based on license availability, instead of job queue priority as discussed in Configuring LSF to Use . LSF License Scheduler enables pre-emptive scheduling that allows projects that own license resources to have guaranteed access to those licenses if another project which does not have ownership is currently using them.

As in the integration with LSF, SmartSuspend leverages the LSF License Scheduler's project-based pre-emptive scheduling model, but uses the SmartSuspend technology to handle the underlying job suspension and resumption operations.

The default LSF License Scheduler suspend/resume behavior is to send SIGTSTP and SIGCONT signals to running jobs. The SmartSuspend integration substitutes these signals with the ssrcmd which performs a suspend or resume with SSR_SIG_RESUME and SSR_SIG_SUSPEND defined signals.

Configuring LSF License Scheduler

1. Configure a distribution policy with license ownerships in the feature section of the Platform LSF License Scheduler configuration file (`lsf.licensescheduler`).

   In this example, license project "`p1`" owns 10 hsim licenses and "`p2`" does not own any hsim licenses.

   ```
   Begin Feature
   NAME = hsim
   DISTRIBUTION = SynopsysServer(p1 1/10 p2 1)
   End Feature
   ```

2. Edit the Platform LSF queues configuration file (`lsb.queues`); the `JOB_CONTROLS` parameter is required.

   ```
   Begin Queue
   QUEUE_NAME = normal
   DESCRIPTION = For License Scheduler Jobs
   PRIORITY = 100
   NICE = 20
   JOB_CONTROLS = SUSPEND[<path_to_lsf_integration>/ssrlsf.sh -s]\
       RESUME[<path_to_lsf_integration>/ssrlsf.sh -r]
   End Queue
   ```

   Where:

   <path_to_lsf_integration> specifies the path where the integration wrapper script (`ssrlsf.sh`) is installed. By default, it is installed in `/usr/share/smartsuspend`, but it can also be installed on a shared file system.

Note: `SSRLSF_LOGDIR_ROOT_DEFAULT` specifies the default path for `SSRLSF_LOGDIR_ROOT`, which is the base directory where `ssrlsf.sh` creates the subdirectory `<SSRLSF_LOGDIR_ROOT>/<LSB_JOBID>`. This subdirectory contains the SmartSuspend status directory (required for storing job state communication data and process IDs (PIDs)), in addition to various log files. Ideally, `SSRLSF_LOGDIR_ROOT_DEFAULT` is a location on a shared file system, because the directories for jobs started on all machines are then kept in one location. A location on a local file system may also be used if desired.

Unlike the Platform LSF queue pre-emption case, the PREEMPTION parameter is not required here.

---

3. Edit the main Platform LSF configuration file (`lsf.conf`) to direct Platform LSF License Scheduler to use JOB_CONTROLS in `lsb.queues` for suspension and resumption mechanisms and release the job slot on suspension:

   ```
   LSF_LIC_SCHED_PREEMPT_STOP=Y
   LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE=Y
   ```

   Jobs can then be submitted to their respective license projects (specified by `-Lp` on the `bsub`

command line).

Controlling LSF License Scheduler Jobs Using SmartSuspend

Assuming you are using a `bsub` script to submit your job, prepend `/usr/share/smartsuspend/ssrlsf.sh` to the job command to indicate that you are starting a job using SmartSuspend:

`/usr/share/smartsuspend/ssrlsf.sh -f <license_file> -- <application_name> <application_arguments>`

Either the `-f <license_file>` or `-l <license_host>` option must be specified, but not both. Note that if `ssrlsf.sh` is already in your execution path, you don't need to specify the full path.

- `<license_file>` specifies the FLEXnet $LM_LICENSE_FILE environment variable as the license file
- `<license_host>` specifies the FLEXnet license server host, or multiple hosts in a colon-separated list

For example:

```
# bsub –Lp p1 –R “rusage[hsim=1]” /usr/share/smartsuspend/ssrlsf.sh -f
"1718@sjlapl1" -- hsim_job_script
```

Controlling Your LSF Job Using SmartSuspend

Jobs will be suspended, resumed, killed, or will run to completion automatically based on their license ownerships. Much the same as the LSF integration case, administrators can manipulate SmartSuspend enabled jobs manually, using normal Platform LSF commands.

However, instead of using the default LSF operations, the commands use the underlying SmartSuspend operations:

- `bstop <job_id>`— LSF uses SmartSuspend to suspend CPU usage, page out memory on demand, and free up licenses if required.
- `bresume <job_id>`— LSF uses SmartSuspend to restore CPU and memory usage and allow the application to obtain any needed licenses as required.

Use the LSF command `bkill` to remove/kill the job; no action by SmartSuspend is necessary:

`bkill` <*job_id*>—LSF sends its default signals to kill your job.

Running Jobs in "Pass-Through" Mode

To submit a job to LSF without enabling SmartSuspend, set the environment variable `NO_SSR=yes` in your environment or `bsub` script:

```
export NO_SSR=yes
```

`ssrlsf.sh` also disables the use of SmartSuspend if you are not running your job on a supported operating system.

In either case, a job launched with `ssrlsf.sh` and SmartSuspend disables runs in “pass-through” mode; that is, your command is run with its arguments, but without SmartSuspend. When LSF suspends or preempts a job launched with `ssrlsf.sh` when SmartSuspend is disabled, the default action is to kill and requeue the job. This action is performed by the following line in `ssrlsf.sh`:

```
brequeue $LSB_JOBID
```

This action can be customized by changing this line to the appropriate action. Changing this pass-through behavior must be done with caution, so that it does not conflict with LSF actions. Contact Jaryba support if you would like to customize pass-through actions.