Jaryba® SmartSuspend™

Using SmartSuspend

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 installing and configuring the Jaryba SmartSuspend software
- Detailed descriptions of and examples for using the SmartSuspend command line utility (`ssrcmd`) to perform basic job suspension and resume operations

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 Overview

About SmartSuspend

SmartSuspend allows the safe suspension of a currently running job, relinquishing all the system resources consumed by that job (CPU, memory, and even licenses). SmartSuspend also provides the proper resumption of a suspended job, reobtaining the necessary system resources and licenses for job completion. When used in conjunction with job queuing software like Platform LSF or Oracle Grid Engine, job suspension and resumption can be automated using the more advanced capabilities of SmartSuspend to perform these operations.

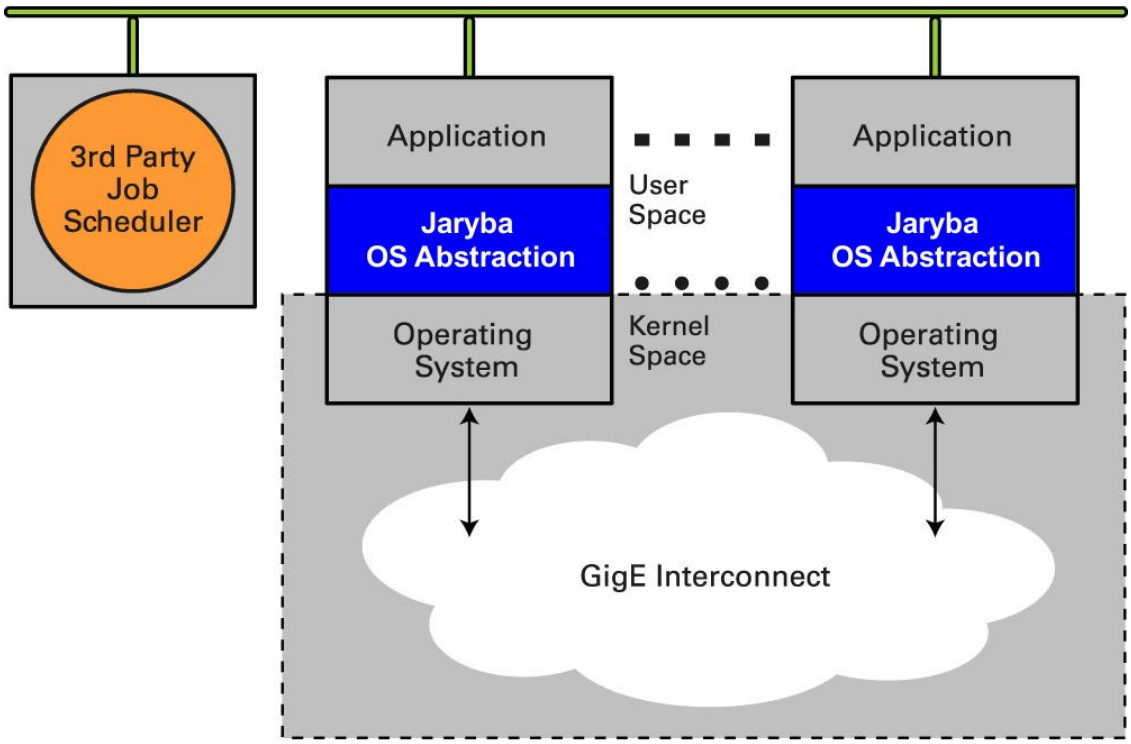
SmartSuspend is a lightweight solution with less than 1% performance overhead, and it is completely transparent to both applications and the operating system. SmartSuspend ensures no compute cycles are lost, increases job throughput based on administrator-defined priorities, and maximizes server utilization.

How SmartSuspend Works

Traditional job suspension methods send SIGSTOP or SIGTSTP to the jobs, but the downside to this approach is that suspended jobs are still holding on to the system memory, as well as any software licenses required for the job.

Based on Jaryba's Operating System Abstraction Layer technology, SmartSuspend preempts a running job by suspending the application's CPU usage, paging out memory, and relinquishing licenses (if applicable), so that they can be used by other jobs deemed higher in priority. SmartSuspend keeps a minimal memory footprint and the sockets alive, so that the suspended application is still active from an operating system perspective.

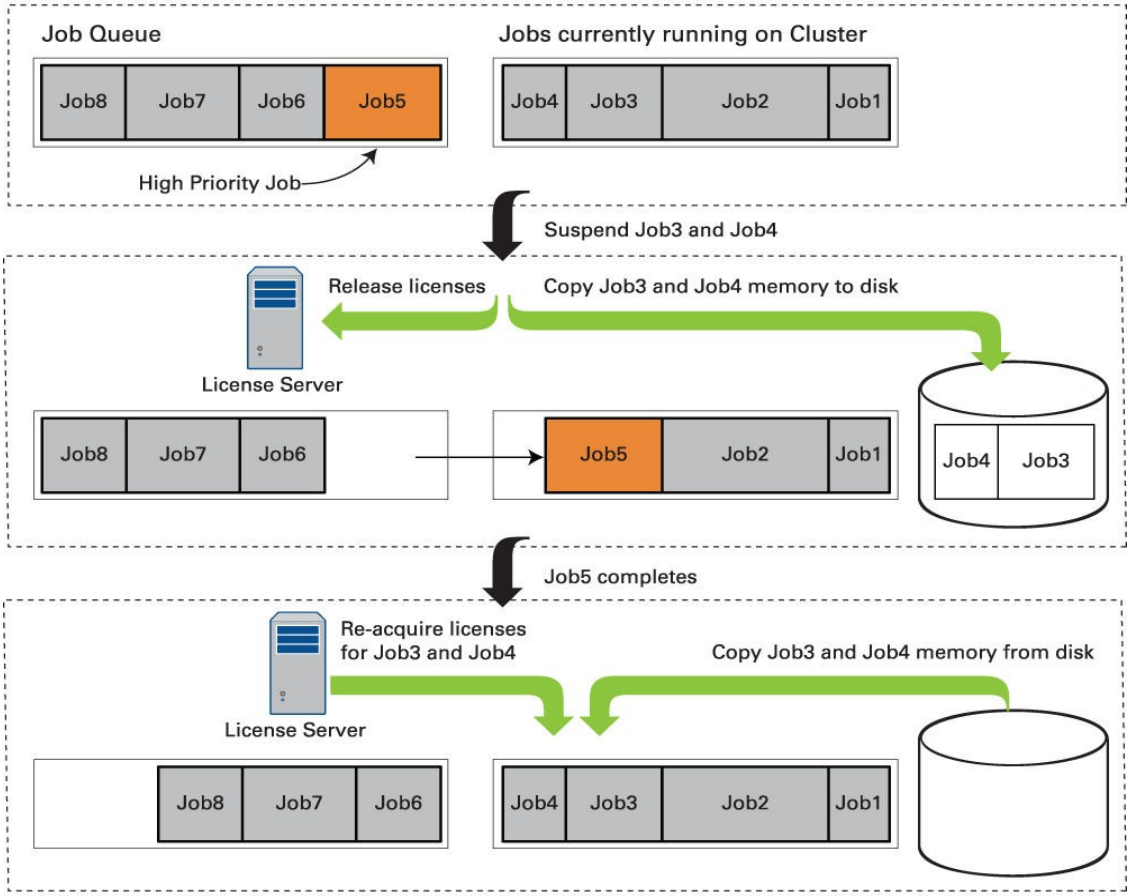
Jaryba’s OS Abstraction Layer



SmartSuspend can be used with popular queuing systems such as Platform LSF or Oracle Grid Engine. SmartSuspend provides a very simple command line interface enabling integration with third-party queuing systems with minimal effort. The SmartSuspend command line utility can be used interchangeably with both parallel and serial applications, and can be run from anywhere on the network as long as `ssh` access is available.

The following illustration shows two jobs being suspended when a higher priority job needs immediate access to a set amount of server resources. Upon the completion of the high priority job, the two suspended jobs are resumed from the point at which they were suspended.

Preemptive Scheduling with SmartSuspend



2 Installing the SmartSuspend Software

The SmartSuspend software installation is accomplished using an installer that you download from the Jaryba customer Web site (jaryba). The installer unpacks the required RPM package.

This chapter also provides instructions for uninstalling the SmartSuspend software.

Prerequisites

Review the following installation prerequisites before installing the SmartSuspend software:

- Ensure the machine targeted for SmartSuspend software installation meets the minimum system requirements documented in the *SmartSuspend Release Notes*.
- You must run the installation script as `root`.
- Verify the operating system and architecture of the machine you are installing on. To get OS version:

```
# cat /etc/*release
Red Hat Enterprise Linux Server release 5.1 (Tikanga)
```

Architecture:

```
# uname -a
Linux xeon26 2.6.18-53.el5 #1 SMP Wed Oct 10 16:34:19 EDT 2007 x86_64 x86_64 x86_64
GNU/Linux
```

Installing the SmartSuspend Software

The SmartSuspend software is contained in a self-extracting installer that extracts the RPM package with the required components into the current directory. You will then manually install the components using RPM commands (for a local installation) or create archived cpio files (for a centralized installation).

To install the SmartSuspend software:

1. Log in to the target installation machine as `root`. You must be `root` to install the SmartSuspend software.
2. Create a directory in which you want to download the installer and extract the RPM package. For example:

   ```
   # mkdir /tmp/ssr
   ```

3. Download the SmartSuspend software and documentation, as directed by your Jaryba sales

or support personnel.

There are different installers for each supported operating system. Make sure you download the installer that is appropriate for the operating system on the target installation machine.

4. Go to the directory in which you downloaded the software, and run the self-extracting installation script, for example:

   ```
   # cd /tmp/ssr
   # ./jaryba-ssr-<version-arch.os>.bin
   ```

   where <*version-arch.os*> is the version of the SmartSuspend software you are installing, the CPU architecture, and the operating system of the installation target. For example:

   ```
   # ./jaryba-ssr-1.1.9-x86_64.rhel5.bin
   ```

   would be used on a Red Hat Enterprise Linux Server, release 5.1, 64-bit machine.

   Type **1** and press **Enter** to extract the SmartSuspend software, or type **2** and press **Enter** if you want to exit out of the installation.

5. The installation script posts a message providing you with the name of the SmartSuspend RPM software package when it is finished extracting the files.

Local Installation Only

Continue with this step only if you are installing SmartSuspend on the local node. For a centralized installation, skip to Using a Centralized I.

6. Use the RPM install command to install the required SmartSuspend package:

   ```
   # rpm -ivh jaryba-ssr-<version.arch>.rpm
   ```

   where <*version.arch*> describes the version of the SmartSuspend software you are installing (for example, `1.1.9`) and the CPU architecture of the installation target. For example:

   ```
   # rpm -ivh jaryba-ssr-1.1.9-0.x86_64.rpm
   Preparing...                ########################################### [100%]
     1:jaryba-ssr              ########################################### [100%]
   ```

   Installation on the local node is now complete.

Using a Centralized Installation

SmartSuspend can be run as a centralized installation, in which the SmartSuspend libraries are stored in a shared location available to all machines, instead of on each individual node. Follow the steps in this section to extract the software to a centralized repository, and add symbolic links to each node to use the centralized repository.

6. Copy the RPM file to a central directory.

7. To extract files from the RPM into the current (central) directory, and maintain the correct

directory structure, issue this command:

```
rpm2cpio <rpm file> | cpio –idmv
```

8. Configure each node to use the Central Repository.

   Linux's security mechanism prevents libraries being loaded into setuid (Set User ID) programs if the library is not located in the standard `/lib` (32 bits) or `/lib64` (64 bits) directories. This prevents a rogue user from preloading a library that intercepts a simple call, such as file open, and running additional code as the effective user id (such as the super user). SmartSuspend's local installation installs 32- and 64-bit variants of the library in these standard directories.

   When SmartSuspend is installed and used from a central repository, symbolic links from the `/lib` and `/lib64` directories to the central repository must be created manually. Jaryba recommends making the local changes for central installations even if setuid programs are *not* expected to be used. This way each node is ready to support setuid programs in the future.

   Create the following symbolic links on each node:

   ```
   /lib/libssr.so -> /path/to/shared/32-bit/library/libssr.so
   /lib64/libssr.so -> /path/to/shared/64-bit/library/libssr.so
   ```

Verifying SmartSuspend Installation

To verify SmartSuspend installation:

Use the RPM query command to check the RPM database and ensure the SmartSuspend RPM package is installed on your machine. For example:

```
# rpm -qa | grep jaryba-ssr
jaryba-ssr-1.1.9
```

Alternatively, use the following RPM query command if you want to view the package details and want a list of all the SmartSuspend directories and files that are installed:

```
# rpm -qil jaryba-ssr
```

Understanding What Is Installed

The SmartSuspend key components are installed in the following locations:

Description	Directory
SmartSuspend command line utilities `ssrcmd`	`/bin`
SmartSuspend library—`libssr`	`/lib` `/lib64`
SmartSuspend-LSF integration script — `ssrlsf.sh` SmartSuspend-Oracle Grid Engine integration script — `ssroge.sh` Sample timeout script to use with ssrcmd `--timeout-script` option — `ssrcmd_timeout.sh`	`/usr/share/smartsuspend`
SmartSuspend man pages	`/usr/share/man`
Disable file (to disable SmartSuspend on specified programs)	`/etc/smartsuspend/disable.conf`
SmartSuspend log files	`/var/log/ssr`
SmartSuspend-related process ID (PID) file	`/var/run/ssr`
Jaryba SmartSuspend default license file location.	`/etc/smartsuspend/jaryba.lic`

Ensure your `$LD_LIBRARY_PATH`, `$PATH`, and `$MANPATH` environment variables accommodate the SmartSuspend software, and modify them as required.

Setting Up SSH Access

If you will run any MPI jobs with SmartSuspend, you need to set up SSH access before using SmartSuspend. SmartSuspend uses SSH to connect to the machine nodes and execute the processes. Public key logins must be set up and working for SmartSuspend to correctly connect to the nodes.

In a cluster environment, any compute nodes must mount the same `/home` directory as the head node and the head node must have password-less SSH access to each compute node in the cluster. The easiest way to do this is to generate a public key and set it up to accept that key as a login credential.

To generate a Public key :

1. Run the following command:

   ```
   $ ssh-keygen -t rsa
   ```

2. Accept the default values by pressing `Enter` at the prompts. (Note: Do not use a passphrase.)

   After completing this process, two files are created:

   ```
   $(HOME)/.ssh/id_rsa
   ```

   ```
   $(HOME)/.ssh/id_rsa.pub
   ```

3. Copy the public key (stored in the `$(HOME)/.ssh/id_rsa.pub` file) to the location where authorized/accepted login keys are stored:

   ```
   $ cp $HOME/.ssh/id_rsa.pub $HOME/.ssh/authorized_keys
   ```

   You will now be able to log in using the key you generated. See your SSH documentation for more information.

Upgrading SmartSuspend

To upgrade to a newer version, download the package as directed in Installing the S. Upgrade all nodes to avoid any confusion over what version of SmartSuspend is being preloaded.

Use the RPM upgrade command to install the newer version:

```
# rpm -U jaryba-ssr-<version.arch>.rpm
```

where <*version.arch*> describes the version of the SmartSuspend software you are installing (for example, `1.1.9`) and the CPU architecture of the installation target. For example:

```
# rpm -U jaryba-ssr-1.1.9-0.x86_64.rpm
```

Uninstalling the SmartSuspend Software

To uninstall the SmartSuspend software, remove the RPM package.

To uninstall the SmartSuspend software package:

1. Log in (as `root`) to the machine where the SmartSuspend software is installed.

   You must be `root` to uninstall the SmartSuspend software.

2. Use the RPM query command to check the RPM database and find the SmartSuspend package that is installed on your machine. For example:

   ```
   # rpm -qa | grep jaryba-ssr
   ```

```
jaryba-ssr-1.1.9
```

3. Use the RPM erase command to remove the SmartSuspend software:

```
# rpm -ev jaryba-ssr
```

To verify that the SmartSuspend software is uninstalled:

Use the RPM query command to verify the SmartSuspend software was removed:

```
# rpm -q jaryba-ssr
package jaryba-ssr is not installed
```

3 Using the SmartSuspend Command Line Interface

This chapter describes how to use SmartSuspend command line utility `ssrcmd` to perform basic job submission, suspension, resumption, and cancellation operations.

You do not need to be `root` to use the `ssrcmd` utility, however, a job can only be suspended by the same user who started the job or a user with sufficient read and write privileges to access the required SmartSuspend job-related files.

If a job contains processes that are owned by more than one user, commands submitted by the `ssrcmd` utility must be executed with superuser privileges.

Most options can also be specified through environment variables; see Chapter 4, "SmartSuspend Environment Variables" for more information.

About ssrcmd

`The ssrcmd` command provides the recommended interface for suspending, resuming, and killing jobs using SmartSuspend.

The basic syntax for the `ssrcmd` command to *start* jobs is:

```
ssrcmd -n <status_dir> [ssrcmd startup options] -- <application_exe>
[application_options]
```

The basic syntax for the `ssrcmd` command to *control* jobs is:

```
ssrcmd -n <status_dir> [ssrcmd command options]
```

In many cases, there is both a short version and long version of `ssrcmd` commands—the two versions are equivalent and perform the same operation. For example, to display help text for the `ssrcmd` command, you can use either:

```
$ ssrcmd --help
```

or

```
$ ssrcmd -h
```

Required ssrcmd Argument

You must specify the `-n <status_directory>` argument when issuing any `ssrcmd` command.

See SmartSuspend Environment Variables for information on variables.

`-n <status_directory>`	Specifies the path and directory location where the job state communication data is stored. If multiple hosts are involved in running the job, the path you specify must be on shared storage. This is a required `ssrcmd` argument. Alternatively, you can set the `$SSR_STATUS_PATH` environment variable prior to running the job.

Job Startup Options

You can specify the following options when starting a job with `ssrcmd`:

`-a, --sig_suspend <signal>`	Specifies the signal to use to suspend a job. By default, the signal used to suspend a job is `10`. Alternatively, you can set the `$SSR_SIG_SUSPEND` environment variable prior to running the job.
`-b, --sig_resume <signal>`	Specifies the signal to use to resume a job. By default, the signal used to suspend a job is `12`. Alternatively, you can set the `$SSR_SIG_RESUME` environment variable prior to running the job.
`-d <save_directory>`	Specifies the path and directory location where the suspended memory is stored. By default, suspended memory is stored in `/tmp`. SmartSuspend creates a hidden file in which to store the suspended memory, so you won't be able to actually see a file in `/tmp` or the directory you specify. The hidden file is also unlinked, which makes it accessible only to SmartSuspend itself, so if the program exits unexpectedly the file is automatically deleted by the operating system. Alternatively, you can set the `$SSR_SAVE_PATH` environment variable prior to running the job.
`-H, --hpmpi`	Enables HP-MPI support.
`-M, --mpich`	Enables MPICH MPI support.
`-O, --openmpi`	Enables Open MPI support.
`-J, --jaryba-license`	Path to Jaryba license file. You can use this option if you

	would like ssrcmd to look in an alternate location for the license file. The default location is `/etc/smartsuspend/jaryba.lic`

Operational Command Options

You can specify the following operational command options with `ssrcmd`:

`-s, --suspend`	Suspends a job using the default signal, or that specified in the `-a` option when you started the job.
`-r, --resume`	Resumes a job using the default signal, or that specified in the `-b` option when you started the job.
`-k, --kill`	Cancels (kills) a running or suspended job and cleans up the job state directory, specified with the `-n` option when you started the job, using the `SIGKILL` signal.
`-t, --terminate`	Terminates a running or suspended job and cleans up the job state directory, specified with the -n option when you started the job, using the `SIGTERM` signal.
`--timeout <`*`seconds`*`>`	Sets a timeout, in seconds, after which suspend/resume actions are abandoned, and the job is killed. If the `--timeout_script` option is also set, then the user-defined script is called first, then the job is killed.
`--timeout_script <`*`path_to_script`*`>`	Specifies the path to a user-generated timeout script. The script is called only if `--timeout <seconds>` is specified, and if the operation time exceeds the number of seconds specified there. When the timeout script is called, it is given the SmartSuspend status directory as an argument. A sample script is included in the SmartSuspend distribution at `/usr/share/smartsuspend/ssrcmd_timeout.sh`.
`-h, --help`	Displays help text for `ssrcmd` utility.
`-v, --version`	Displays `ssrcmd` version information.

Note: The `kill` (`-k`) option sends a `SIGKILL` signal to the job; whereas the `terminate` (`-t`) option sends a `SIGTERM` signal. Clean up of the job state directory takes place when you kill a job using either option with `ssrcmd`.

FLEXnet License Support Startup Options

License support allows any subsequent job suspend operation to properly relinquish the license and then re-obtain the license when the job is resumed. You can specify one of the following additional options to enable license support when using `ssrcmd` to start a job:

`-l, --licensehost <host>` `-l, --licensehost <host>:<host>`	Specifies the FLEXnet license server host or multiple hosts in a colon-separatedlist. **Chapter 4.** Specify either the `-l, --licensehost` option *or* the `-f, --licensefile` option, but not both.
`-f, --licensefile <$LM_LICENSE_FILE>`	Specifies the FLEXnet `$SSR_LICENSE_HOST` environment variable as the license file. **Chapter 4.** Specify either the `-f, --licensefile` option *or* the `-l, --licensehost` option, but not both

If neither one of these options is specified, SmartSuspend will not perform any FLEXnet license management.

Displaying ssrcmd Help

While written documentation and online man pages are provided, you can also display an online summary of `ssrcmd` actions and options from the command line.

To display help for ssrcmd:

Enter one of the following commands:

```
$ ssrcmd -h

$ ssrcmd --help
```

Displaying the ssrcmd Version

If you need to contact Jaryba about the `ssrcmd` interface of the SmartSuspend software, you will need to tell us what version you are running.

To display the ssrcmd version:

Enter one of the following commands:

```
$ ssrcmd -v

$ ssrcmd --version
```

Jaryba SmartSuspend displays version information similar to the following:

```
ssrcmd version 1.1.9
```

Version information is also in the `ssr_data` file generated when you run SmartSuspend-enabled jobs. See the example on $.

Running SmartSuspend-Enabled Jobs

Use the `ssrcmd` utility to start SmartSuspend-enabled jobs. There are environment variables that you can set (see Chapter 4), but using the `ssrcmd` utility to start SmartSuspend-enabled jobs is recommended because it performs error checking and cleanup operations that would otherwise need to be performed manually.

Note: The path you specify for the required `ssrcmd -n` option must be unique for each invocation of a job. Once a job has finished running or has been removed, you can reuse the directory, but you cannot submit another job specifying the same job state communication data directory when one is already running.

To run a SmartSuspend-enabled job:

Minimally, specify the `-n` option for `ssrcmd` and the name of application (as well as any application submission options) on the command line:

```
$ ssrcmd -n <job_dir> -- <application_name> <app_options>
```

If the job you are submitting will run on multiple hosts or be controlled from another location, specify a shared storage location for the `-n` option for `ssrcmd`.

```
$ ssrcmd -n /shared/location/job_dir -- <application_name> <app_options>
```

To run a SmartSuspend-enabled MPI job:

If you are submitting an MPI job, specify the appropriate command line option to enable MPI support:

```
--hpmpi(-H)

--mpich (M)

--openmpi (-O)
```

For example, for an HP-MPI job:

```
$ ssrcmd -n /shared/location/job_dir --hpmpi -- mpirun <application_name>
<options>
```

Note: The supported versions are: HP-MPI, version 2.2.5.1 or higher; Open MPI, version 1.2.2 or higher; MPICH version 1.2.7p1.

To run a SmartSuspend-enabled job, specifying suspend and resume signals:

Specify the `-n` option and the signals you will later use to suspend and resume the job using the `ssrcmd -a` (suspend signal) and `-b` (resume signal) options:

```
$ ssrcmd -n <job_dir> -a 10 -b 12 -- <application_name> <app_options>
```

Here, the `-a` option specifies signal `10` for suspend and the `-b` option specifies signal `12` for resume. Note, however, that `10` and `12` are the default signals used by SmartSuspend, so it would not be necessary to specify these signals if they are the ones you want to use.

An example of specifying non-default signal values would be:

```
$ ssrcmd -n <job_dir> -a 60 -b 62 -- <application_name> <app_options>
```

To run a SmartSuspend-enabled job, specifying a shared storage location to store suspended memory:

Specify the `-n` option and additionally provide the location to store suspended memory using the `ssrcmd -d` option:

```
$ ssrcmd -n <job_dir> -d /shared/location/mem_dir -- <application_name> <app_options>
```

To run a SmartSuspend-enabled job that enables license support:

You can specify one of the following additional options to enable license support when using `ssrcmd` to submit a job. If the application you are running has a FLEXnet license associated with it, license support allows a subsequent job suspend operation to relinquish the license so that it can be used by another application. When you resume the job, the license will be re-obtained.

You can provide either the FLEXnet license server host name (using `-l` option) or, if it is set, specify the `$LM_LICENSE_FILE` environment variable (using `-f` option) for the license file:

```
$ ssrcmd -n <job_dir> -l <hostname> -- <application_name> <app_options>
```

Or:

If the `$LM_LICENSE_FILE` environment variable is set in your environment, you can specify the variable with the `-f` option, for example:

```
$ ssrcmd -n  <job_dir> -f $LM_LICENSE_FILE -- <application_name> <app_options>
```

Examples

Example of submitting a serial job that runs on a single host:

Here is an example of running a serial application called LAMMPS on a single host with SmartSuspend enabled:

```
$ ssrcmd -n /tmp/ssr_jobinfo -- ./lmp_serial -in ./in.crack.serial
LAMMPS (17 July 2006)
Created box = (0 0 -0.278569) to (111.428 77.1994 0.278569)
  1 by 1 by 1 processor grid
Created 8141 atoms
302 atoms in group lower
302 atoms in group upper
604 atoms in group boundary
7537 atoms in group mobile
841 atoms in group leftupper
841 atoms in group leftlower
Setting atom values ...
  841 settings made
Setting atom values ...
  841 settings made
Setting atom values ...
  302 settings made
Setting atom values ...
  302 settings made
WARNING: Temperature for thermo pressure is not for group all
Setting up run ...
Memory usage per processor = 2.41665 Mbytes
Step Temp E_pair E_mol TotEng Press Volume
       0  0.010233422                      -3.2595015             0   -3.2500292
                -0.10147524             4792.6107
    1000 0.0051057052                     -3.2546357             0   -3.2499098
                -0.13099352             4792.6107
    2000 0.0051305159                     -3.2543693             0   -3.2496204
                -0.16341482             4807.4758
    3000  0.005186568                     -3.2539935             0   -3.2491927
                -0.17782954             4808.2338
    4000  0.005087842                     -3.2533474             0   -3.248638
             -0.18347851             4814.6291
    5000  0.004952798                     -3.2525326             0   -3.2479482
                -0.23680218             4819.1809
    6000  0.004991209                     -3.2516461             0   -3.2470261
                -0.30814679             4826.0224
    7000  0.004944807                     -3.2506197             0   -3.2460426
                -0.39136722             4830.7072
    8000 0.0049376929                     -3.2494443             0   -3.2448739
                -0.48674075             4839.3787
    9000 0.0050078092                     -3.2482053             0   -3.24357
              -0.53039175             4853.1439
   10000  0.004983781                     -3.2466688             0   -3.2420557
                -0.55385151             4857.416
```

The job started successfully and begins sending the output of the job to `stdout`.

Let's go have a look at what is created in the directory you specified with the -n option:

```
$ cd /tmp/ssr_jobinfo
$ ls -l
```

```
total 1
-rw------- 1 bsmith users 4096 Apr 23 23:01 host20_12686
-rw-r--r-- 1 bsmith users   37 Apr 23 23:01 ssr_data
```

You can see the process ID of the running LAMMPS job and the `ssr_data` file.

Let's look at the contents of the `ssr_data` file:

```
$ cat ssr_data
SmartSuspend 1.1.9
SSR_SIG_SUSPEND 10
SSR_SIG_RESUME 12
SSR_STATUS_PATH /tmp/ssr_jobinfo
SSR_SAVE_PATH /tmp
SSR_LOG_PATH /ssr_log
LD_LIBRARY_PATH /home/user1/lib:/home/user1/lib64
License tracking: Enabled
```

This file keeps track of the default or specified signals for the SmartSuspend suspend and resume operations. In this case, the suspend and resume signals were not specified on the command line, so the default signals of `10` (for suspend) and `12` (for resume) are shown in the `ssr_data` file.

The `ssr_data` file includes the SmartSuspend version number, and values for other environment variables that may be useful in troubleshooting.

If the job has been launched with license support enabled, the `ssr_data` file also displays that information.

Example of submitting an Open MPI job that runs on a multiple hosts:

Here is an example of running a parallel version of the application called LAMMPS on multiple hosts with SmartSuspend enabled. In the first example, `mpirun` is launched directly with `ssrcmd`:

```
$ ssrcmd -n /nfs/share/ssr_jobinfo -a 61 -b 62 --openmpi -- mpirun
-machinefile hostfile -np 4 lammps -in in.crack
```

Notice in this example, the `ssrcmd -a` (suspend signal) and `-b` (resume signal) options specify the signals to be used rather than the defaults of `10` and `12`, and the `--openmpi` option is specified so that SmartSuspend is enabled on the Open MPI processes.

For HP-MPI or MPICH jobs, replace `--openmpi` with `--hpmpi` or `--mpich`, respectively.

SmartSuspend also supports applications that call `mpirun` indirectly from the command line, such as in a wrapper script. In the example below, `mpi_wrapper_script` is launched with `ssrcmd`. If `mpi_wrapper_script` or any process spawned from it calls `mpirun`, SmartSuspend propagates its environment variables.

```
$ ssrcmd -n /nfs/share/ssr_jobinfo -a 61 -b 62 --openmpi --
mpi_wrapper_script -machinefile hostfile -np 4 lammps -in in.crack
```

Example of submitting a SmartSuspend-enabled job that enables license support:

Here is an example of running an application called ABAQUS with SmartSuspend and license management support enabled:

```
$ ssrcmd -n /nfs2/share/ssr_jobinfo -a 61 -b 62 -f $LM_LICENSE_FILE --
/nfs/share/apps/abaqus job=e2
```

In this example, the `ssrcmd -f` option specifies a variable for the FLEXnet license file because this environment variable was already set in the user's shell before submitting the job. With license management support enabled, a subsequent suspend operation will automatically relinquish the license and then reobtain the license when the job is resumed—the `ssrcmd` suspend and resume commands require no modification or additional options with license support already enabled for the job.

Suspending SmartSuspend-Enabled Jobs

When you suspend a job using `ssrcmd`, SmartSuspend suspends the application's CPU usage, pages out memory, and relinquishes licenses (if you enabled license support when you submitted the job), so that the resources can be used by other jobs deemed higher in priority. SmartSuspend keeps a minimal memory footprint and the sockets alive, so that the suspended application is still active from an operating system perspective.

To suspend a SmartSuspend-enabled job:

Minimally, specify the `-n` and `-s` options for `ssrcmd` on the command line:

```
$ ssrcmd -n <job_dir> -s
```

SmartSuspend suspends the job using the signal stored in the `ssr_data` file in the directory specified with the `-n` option. The signal stored in the `ssr_data` file is either the default (which is `10`) or the signal you specified with the `-a` (suspend signal) option when you started the job.

Examples

Here is an example of issuing the `ssrcmd` suspend command in another command window while the LAMMPS job we started in the previous example is running:

```
$ ssrcmd -n /tmp/ssr_jobinfo -s
Job successfully Suspended.
```

The output of the LAMMPS application to `stdout` stops.

To set a timeout on the suspend operation, use the `--timeout <seconds>` argument. This example suspends with a 600 second (10 minute) timeout:

```
$ ssrcmd -n /tmp/ssr_jobinfo --timeout 600 -s
```

In this case, if the suspend operation is not completed after 600 seconds, the job is killed.

To take other actions in the event of a timeout, set `--timeout_script` in addition to `--timeout <seconds>`, where the specified script contains user-defined actions. The script is called if the operation has not completed after the `--timeout` interval, then the job is killed.

```
$ ssrcmd -n /tmp/ssr_jobinfo --timeout 600 --timeout_script
/tmp/failure_script -s
```

A sample script is provided in `/usr/share/smartsuspend/ssrcmd_timeout.sh`. This script gathers data about the job from the status directory, and sends output to stderr. Edit the script to include a valid email address, to route output to the specified address.

Resuming Suspended Jobs

When you resume a job that was previously suspended, the system resources are automatically re-assigned to the job, and it continues running from where it left off.
If you enabled license support when you submitted the job, the required license is also reobtained.

To resume a previously suspended job:

```
$ ssrcmd -n <job_dir> -r
```

SmartSuspend resumes the job using the signal stored in the `ssr_data` file in the directory specified with the `-n` option. The signal stored in the `ssr_data` file is either the default (which is `12`) or the signal you specified with the `-b` (resume signal) option when you started the job.

A timeout interval, and optionally a timeout script, can be specified for the resume action. See the examples in Suspending S above.

Example

Here is an example of resuming the LAMMPS job we suspended in a previous example:

```
$ ssrcmd -n /tmp/ssr_jobinfo -r
Job successfully Resumed.
```

You will see the application start running again where it left off when it was suspended:

```
LAMMPS (17 July 2006)
Created box = (0 0 -0.278569) to (111.428 77.1994 0.278569)
  1 by 1 by 1 processor grid
Created 8141 atoms
302 atoms in group lower
302 atoms in group upper
604 atoms in group boundary
7537 atoms in group mobile
841 atoms in group leftupper
841 atoms in group leftlower
Setting atom values ...
  841 settings made
Setting atom values ...
  841 settings made
Setting atom values ...
  302 settings made
```

```
Setting atom values ...
  302 settings made
WARNING: Temperature for thermo pressure is not for group all
Setting up run ...
Memory usage per processor = 2.41665 Mbytes
Step Temp E_pair E_mol TotEng Press Volume
       0  0.010233422                      -3.2595015              0   -3.2500292
             -0.10147524            4792.6107
    1000 0.0051057052                      -3.2546357              0   -3.2499098
             -0.13099352            4792.6107
    2000 0.0051305159                      -3.2543693              0   -3.2496204
             -0.16341482            4807.4758
    3000  0.005186568                      -3.2539935              0   -3.2491927
             -0.17782954            4808.2338
    4000  0.005087842                      -3.2533474              0   -3.248638
              -0.18347851           4814.6291
    5000  0.004952798                      -3.2525326              0   -3.2479482
             -0.23680218            4819.1809
    6000  0.004991209                      -3.2516461              0   -3.2470261
             -0.30814679            4826.0224
    7000  0.004944807                      -3.2506197              0   -3.2460426
             -0.39136722            4830.7072
    8000 0.0049376929                      -3.2494443              0   -3.2448739
             -0.48674075            4839.3787
    9000 0.0050078092                      -3.2482053              0   -3.24357
              -0.53039175           4853.1439
   10000  0.004983781                      -3.2466688              0   -3.2420557
             -0.55385151            4857.416
<This is the point at which the job was suspended and the output stopped...>
   11000 0.0049723432                      -3.2451304              0   -3.2405279
             -0.56009417            4862.5452
   12000 0.0049013912                      -3.243408            0   -3.2388712
              -0.57381742           4865.356
   13000 0.0048043804                      -3.2415478              0   -3.2371007
             -0.62822628            4866.3058
   14000 0.0048263329                      -3.2395963              0   -3.2351289
             -0.69278151            4872.0789
   15000 0.0047872575                      -3.2375073              0   -3.2330761
             -0.77081332            4885.8904
   16000 0.0047945553                      -3.2353167              0   -3.2308787
             -0.82826028            4893.1664
   17000 0.0047216486                      -3.2330128              0   -3.2286423
             -0.86086905            4902.0617
   18000 0.0047886085                      -3.2306714              0   -3.226239
              -0.86709584           4912.8232
```

Canceling Suspended or Running Jobs

The `ssrcmd` utility lets you cancel (kill) or terminate a suspended or running job.

To cancel/kill a previously suspended or running job:

Specify the `-n` and `-k` options for `ssrcmd` on the command line to kill a job:

```
$ ssrcmd -n <job_dir> -k
```

Example

Here is an example of killing the previously suspended serial LAMMPS job, instead of resuming the job:

```
$ ssrcmd -n /tmp/ssr_jobinfo -k
Job successfully Killed.
```

After killing the job successfully, the `ssrcmd` utility removes the `ssr_data` file, as well as any host-specific process ID (PID) files. The directory name specified by the `-n` option is not deleted, but since the directory is now empty, you can reuse this directory for new runs of the application:

```
$ cd /tmp/ssr_jobinfo
$ ls
$
```

Example

Here is an example of terminating a previously suspended job, and cleaning up the job state directory specified when starting the job, using the `terminate` (`-t`) option of the `ssrcmd` utility:

```
$ ssrcmd -n /tmp/ssr_jobinfo -t
Job successfully Terminated.
```

Note: The `kill` (`-k`) option sends a `SIGKILL` signal to the job; whereas the `terminate` (`-t`) option sends a `SIGTERM` signal. Clean up of the job state directory takes place when you kill a job using either option with `ssrcmd`.

Disabling SmartSuspend for Specified Applications

A system administrator may define (at the node level) a list of specific applications that should not be SmartSuspend enabled.

Applications you wish to disable must be listed in the configuration file (a sample is installed in the SmartSuspend distribution):

```
/etc/smartsuspend/disable.conf
```

Permissions for this file are set to 644, so that it can be modified only by the system administrator, but read by any user on the system. In the disable file, include specific program names, one per line, that SmartSuspend will consult at runtime. If the current running program matches a name in the file, then it will disable all SmartSuspend capabilities (ability to suspend/resume, release resources). Comment lines can begin with "#".

Although all capabilities will be disabled for the listed processes, a process that is forked and executed from a disabled process *will* be enabled unless it also is specifically named in the `disabled.conf` file.

Example

Run the SmartSuspend-enabled job, “prog1”:

```
$ ssrcmd -n <job_dir> -- prog1
```

Contents of the `disable.conf`:

```
prog2
```

Process tree output:

```
prog1 (Enabled)-> prog2 (Disabled) -> prog3 (Enabled)
```

4 SmartSuspend Environment Variables

The Jaryba SmartSuspend library (`libssr.so`) is preloaded with an application using the standard Linux environment variable `$LD_PRELOAD`. The `$LD_PRELOAD` variable allows SmartSuspend application suspension and resumption capabilities without any modification to the application itself.

Hierarchy of Settings and Commands

Because values used by SmartSuspend can come from several sources, it is important to understand the order in which values are used. In the list below, each successive item takes precedence over the previous items, e.g. command line settings override environment variables.

- Default values
- Environment variables
- Command line settings
- Values stored in the `ssr_data` file. For example, if you specified suspend and resume signals when you started a job, those signal values are stored in `ssr_data` for that job. If you subsequently set new signal values, the job will ignore them and still use the values in `ssr_data`.

Setting the Environment Variables

A set of environment variables provide the configuration parameters for controlling SmartSuspend behavior. Alternatively, the SmartSuspend command line utility (`ssrcmd`) provides options that let you supply many of these configuration parameters directly on the command line.

While Jaryba recommends using the `ssrcmd` utility to specify SmartSuspend configuration parameters (see "Using the SmartSuspend Command Line Interface" on page 11 for details), there are environment variables you can use instead of or in conjunction with the `ssrcmd` utility options:

Environment Variable	Description
`LD_PRELOAD`	`$LD_PRELOAD` is a standard Linux environment variable that instructs the linker/loader to preload an application with a dynamic library, binding all symbols found in both the application and the library to the preloaded library. The `$LD_PRELOAD` variable allows SmartSuspend application suspension and resumption without any modification to the application itself. When you use the `ssrcmd` utility to start a job, the application is preloaded with the SmartSuspend library.

SSR_STATUS_PATH	$SSR_STATUS_PATH specifies a unique directory path for each job you start with SmartSuspend and specifies the location where the job state communication data is stored. $SSR_STATUS_PATH is a required variable unless you use the ssrcmd -n option when you start a job. This variable acts as a job key and is used to identify all processes within a particular job and their current state. If multiple hosts are involved in running a job (such as a parallel MPI job), or you want to submit commands from another host, then the path specified must be on shared storage. Suspended job data is *not* stored here, only the process information. Each SmartSuspend-capable process creates an entry in the $SSR_STATUS_PATH directory in the following format: <$*SSR_STATUS_PATH*>/<*hostname*>_<*PID*>.ssr While a job is running, you can determine the process ID (PID) and the name of the host where it is running for every process. The PID file contains status information about that particular process. The file is removed when a process exits normally. The suggested method for starting, suspending, resuming, and killing a job is to use the ssrcmd command line utility. This is especially true with multiprocess jobs, as the utility automates process signaling, synchronization, and determining state changes/errors though $SSR_STATUS_PATH.
SSR_SIG_SUSPEND	$SSR_SIG_SUSPEND specifies an integer value signal to be used for suspending an application. This signal properly handles the SmartSuspend job suspension when sent to the application by the controlling entity. Valid signal values are: • 10 (SIGUSR1) • 32 (SIGRTMIN) to 64 (SIGRTMAX) By default, $SSR_SIG_SUSPEND is set to 10 (SIGUSR1). When you use the ssrcmd utility to start a job, it automatically uses the suspend signal default of 10, or you can use the -a, --sig_suspend option to specify a different signal to use in the case that the application requires the use of the default signals.

SSR_SIG_RESUME	$SSR_SIG_RESUME specifies an integer value signal to be used for resuming an application. This signal properly handles the SmartSuspend job resumption when sent to the application by the controlling entity. Valid signal values are: • 12 (SIGUSR2) • 32 (SIGRTMIN) to 64 (SIGRTMAX) By default, $SSR_SIG_RESUME is set to 12 (SIGUSR2). When you use the ssrcmd utility to start a job, it automatically uses the resume signal default of 12, or you can use the -b, --sig_resume option to specify a different signal to use in the case that the application requires the use of the default signals.
SSR_SIGSTKSZ	$SSR_SIGSTKSZ specifies the amount of memory allocated for the separate stack on which suspend and resume handlers will be executed. By default, $SSR_SIGSTKSZ is set to 128K.
SSR_SAVE_PATH	$SSR_SAVE_PATH specifies the full path to the directory location in which to store suspended job memory. By default, $SSR_SAVE_PATH is set to /tmp. When you use the ssrcmd utility to start a job, it automatically uses the directory default of /tmp, or you can use the -d option to specify a different directory. SmartSuspend creates a hidden file in which to store the suspended memory, so you won't be able to actually see a file in /tmp or the directory you specify. The hidden file is also unlinked, which makes it accessible only to SmartSuspend itself, so if the program exits unexpectedly the file is automatically deleted by the operating system.
SSR_LICENSE_HOST	Setting $SSR_LICENSE_HOST enables license management support for SmartSuspend-enabled jobs and allows license relinquishment of FLEXnet licenses. When license support is enabled, a subsequent job suspension properly relinquishes the license, allowing it to be used by another job. When the suspended job is resumed, the required license is re-obtained for the job. This variable should be set to the hostname of the machine that will serve the licenses. In the case that more than one host handles this, it should be set as a colon-separated list of host names. For example: SSR_LICENSE_HOST = hosta:hostb

To set the environment variables:

You can set the environment variables on the command line or you can set variables in your shell startup script, for example .bashrc or .profile (for bash shells) or .cshrc or .tcshrc (for

csh/tcsh shells).

From the command line, use either the export command (bash) or the setenv command (csh/tcsh) to set the environment variables.

For example, for bash:

```
$ export SSR_SAVE_PATH=/nfs/share/ssr
```

For example, for csh/tcsh:

```
$ setenv SSR_SAVE_PATH "/nfs/share/ssr"
```

Index

T

U

V

.