Parallel System Support Programs for AIX

# Command and Technical Reference, Volume 2

*Version 3 Release 1.1*

SA22-7351-01

Parallel System Support Programs for AIX

# Command and Technical Reference, Volume 2

*Version 3 Release 1.1*

SA22-7351-01

**Note!**

| Before using this information and the product it supports, read the information in “Notices” on page ix.

| **Second Edition (October 1999)**

| This edition applies to version 3 release 1 modification 1 of the IBM Parallel System Support Programs for AIX (PSSP) Licensed
| Program (product number 5765-D51) and to all subsequent releases and modifications until otherwise indicated in new editions. This
| edition replaces SA22-7351-00. Significant changes or additions to the text are indicated by a vertical line (|) to the left of the
| change.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+914+432-9405
FAX (Other Countries):
Your International Access Code +1+914+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMMAIL
Internet e-mail: mhvrcfs@us.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

# Contents

# Notices

## Trademarks

## Publicly Available Software

PSSP includes software that is publicly available:

| | |
|---|---|
| **expect** | Programmed dialogue with interactive programs |
| **Kerberos** | Provides authentication of the execution of remote commands |
| **NTP** | Network Time Protocol |
| **Perl** | Practical Extraction and Report Language |
| **SUP** | Software Update Protocol |
| **Tcl** | Tool Command Language |
| **TclX** | Tool Command Language Extended |
| **Tk** | Tcl-based Tool Kit for X-windows |

This book discusses the use of these products only as they apply specifically to the RS/6000 system. The distribution for these products includes the source code and associated documentation. (Kerberos does not ship source code.) **/usr/lpp/ssp/public** contains the compressed **tar** files of the publicly available software. (IBM has made minor modifications to the versions of Tcl and Tk used in the SP system to improve their security characteristics. Therefore, the IBM-supplied versions do not match exactly the versions you may build from the compressed **tar** files.) All copyright notices in the documentation must be respected. You can find version and distribution information for each of these products that are part of your selected install options in the **/usr/lpp/ssp/README/ssp.public.README** file.

# About This Book

This book provides detailed syntax and parameter information for all commands you can use to install, customize, and maintain the IBM RS/6000 SP system.

For a list of related books and information about accessing online information, see the bibliography in the back of the book.

This book applies to PSSP Version 3 Release 1 Modification 1. To find out what version of PSSP is running on your control workstation (node 0), enter the following:

```
splst_versions -t -n0
```

In response, the system displays something similar to:

```
0 PSSP-3.1.1
```

If the response indicates **PSSP-3.1.1**, this book applies to the version of PSSP that is running on your system.

To find out what version of PSSP is running on the nodes of your system, enter the following from your control workstation:

```
splst_versions -t -G
```

In response, the system displays something similar to:

```
1 PSSP-3.1.1
2 PSSP-3.1.1
7 PSSP-2.4
8 PSSP-2.2
```

If the response indicates **PSSP-3.1.1**, this book applies to the version of PSSP that is running on those nodes.

If you are running mixed levels of PSSP, be sure to maintain and refer to the appropriate documentation for whatever versions of PSSP you are running.

## Who Should Use This Book

This book is intended for anyone not familiar with the syntax and use of the RS/6000 SP commands.

## How This Book Is Organized

This book consists of two volumes. Volume 1 contains RS/6000 SP commands a - p. Volume 2 contains RS/6000 SP commands r - v, RS/6000 SP Files and Other Technical Information, and RS/6000 SP Subroutines. Both volumes share a common frontmatter, appendix, glossary, and bibliography. The indexes are customized for each volume.

## Command Format

The commands in this book are in the following format:

| | |
|---|---|
| **Purpose** | Provides the name of the command and a brief description of its purpose. |
| **Syntax** | Includes a diagram that summarizes the use of the command. |
| **Flags** | Lists and describes the options that control the behavior of the command. |
| **Operands** | Lists and describes the objects on which the command operates. |
| **Description** | Includes a complete description of the command. |
| **Extended Description** | Includes more detailed or additional description of the command. |
| **Environment Variables** | Lists any environment variables that affect the operation of the command. Lists any environment variables that are affected by the operation of the command. |
| **Files** | Lists any RS/6000 SP system files that are read, employed, referred to, or written to by the command, or that are otherwise relevant to its use. |
| **Standard Input** | Describes what this command reads from standard input. |
| **Standard Output** | Describes what this command writes to standard output. |
| **Standard Error** | Describes what and when this command writes to standard error. |
| **Exit Values** | Describes the values returned and the conditions that caused the values to be returned. |
| **Security** | Describes who can run this command and provides other security-related information. |
| **Restrictions** | Lists restrictions beyond the security restrictions described previously. |
| **Implementation Specifics** | Identifies the package of each individual command. |
| **Prerequisite Information** | Provides a pointer to other documents that would enhance the user's understanding of this command. |
| **Location** | Specifies the location of the command. |
| **Related Information** | Lists RS/6000 SP commands, functions, file formats, and special files that are employed by the command, that have a purpose which is related to that of the command, or that are otherwise of interest within the context of the command. Also listed are related RS/6000 SP documents, other related documents, and miscellaneous information related to the command. |
| **Examples** | Provides examples of how the command is typically used. |

## Typographic Conventions

This book uses the following typographic conventions:

| Typographic | Usage |
|---|---|
| **Bold** | • **Bold** words or characters represent system elements that you must use literally, such as commands, flags, and path names. • **Bold** words also indicate the first use of a term included in the glossary. |
| *Italic* | • *Italic* words or characters represent variable values that you must supply. • *Italics* are also used for book titles and for general emphasis in text. |
| `Constant width` | Examples and information that the system displays appear in `constant width` typeface. |
| [ ] | Brackets enclose optional items in format and syntax descriptions. |
| { } | Braces enclose a list from which you must choose an item in format and syntax descriptions. |
| \| | A vertical bar separates items in a list of choices. (In other words, it means "or.") |
| < > | Angle brackets (less-than and greater-than) enclose the name of a key on the keyboard. For example, <**Enter**> refers to the key on your terminal or workstation that is labeled with the word Enter. |
| ... | An ellipsis indicates that you can repeat the preceding item one or more times. |
| <**Ctrl-***x*> | The notation <**Ctrl-***x*> indicates a control character sequence. For example, <**Ctrl-c**> means that you hold down the control key while pressing <**c**>. |
| **\** | The continuation character is used in coding examples in this book for formatting purposes. |

# Part 1.  Command Reference Volume 2

# Chapter 1. Commands

This volume contains the RS/6000 SP Commands R - Z, RS/6000 SP Files and Other Technical Information (Chapter 2), and RS/6000 SP Subroutines (Chapter 3). See Volume 1 for RS/6000 SP Commands A - Q.

To access the RS/6000 SP online manual pages, set the MANPATH environment variable as follows:

for ksh

```
export MANPATH=$MANPATH:/usr/lpp/ssp/man
```

for csh

```
setenv MANPATH $MANPATH\:/usr/lpp/ssp/man
```

## System Partitioning and Commands

When you partition your system, you create one or more system partitions which, for most tasks, function as separate and distinct logical RS/6000 SP systems. Most commands function within the boundary of the system partition in which they are executed. A number of commands, however, continue to treat the RS/6000 SP as a single entity and do not respect system partition boundaries. That is, in their normal function they may affect a node or other entity outside of the current system partition. In addition, some commands which normally function only within the current system partition have been given a new parameter which, when used, allows the scope of that command to exceed the boundaries of the current system partition.

On the control workstation, the administrator is in an environment for *one system partition at a time*. The SP_NAME environment variable identifies the system partition to subsystems. (If this environment variable is not set, the system partition is defined by the `primary:` stanza in the **/etc/SDR_dest_info** file.) Most tasks performed on the control workstation that get information from the System Data Repository (SDR) will get the information for *that particular system partition*.

In managing multiple system partitions, it is helpful to open a window for each system partition. You can set and export the SP_NAME environment variable in each window and set up the window title bar or shell prompt with the system partition name. The following script is an example:

```
sysparenv:
# !/bin/ksh
  for i in 'splst_syspars'
  do
     syspar='host $i | cut -f 1 -d"."'
     echo "Opening the $syspar partition environment"
     sleep 2
     export SP_NAME=$syspar
     aixterm -T "Work Environment for CWS 'hostname -s' - View: $syspar" -ls -sb &
  done
  exit
```

```
.profile addition:
```

```
# Added for syspar environment setup
  if [ "'env | grep SP_NAME | cut -d= -f1'" = SP_NAME ]
     then
        PS1="['hostname -s'<p>"$SP_NAME] ['$PWD]> '
     else
        PS1="['hostname -s']["'$PWD]< '
  fi
  export ENV
```

As a user, you can check what system partition you're in with the command:

```
spget_syspar -n
```

The following table summarizes those commands which can exceed the boundary of the current system partition. Unless otherwise stated, commands not listed in this table have as their scope the current system partition.

| Command | Effect |
|---|---|
| **arp** | Can reference any node (by its host name) in any system partition. |
| Automounter commands | Host names need not be in the current system partition. |
| **chauthpar -p** | The **-p** flag allows specification of a system partition other than the current system partition. |
| **crunacct** | Merges accounting data from all nodes regardless of system partition boundaries. |
| **cshutdown -G** | The **-G** flag allows specification of target nodes outside of the current system partition. |
| **cstartup -G** | The **-G** flag allows specification of target nodes outside of the current system partition. |
| **dsh**<br>**dsh -w**{*hostname* \| -} | Hosts added to the working collective by host name need not be in the current system partition. |
| **dsh -aG** | The **-G** flag modifies the **-a** flag (all nodes in the current system partition) by expanding the scope to all nodes in the entire physical SP system. |
| **Eclock** | There is a single switch clock for the SP regardless of the number of system partitions. |
| **Efence -G** | The **-G** flag allows specification of nodes outside of the current system partition. |
| **emonctrl -c** | The system partition-sensitive control script for the **emon** subsystem supports the **-c** option, which crosses system partitions. |
| **Eunfence -G** | The **-G** flag allows specification of nodes outside of the current system partition. |

| Command | Effect |
|---|---|
| **haemctrl -c**<br>**haemctrl -u** | The system partition-sensitive control script for the **haem** subsystem supports the **-c** and **-u** options, which cross system partitions. |
| **haemd_SP** *syspar_IPaddr* | Specifies the IP address of the system partition in which the **haemd** daemon is to execute. |
| **haemqvar** | If the **SP_NAME** environment variable is not set, the default system partition is used. |
| **hagsctrl -c**<br>**hagsctrl -u** | The system partition-sensitive control script for the **hags** subsystem supports the **-c** and **-u** options, which cross system partitions. |
| **hatsctrl -c**<br>**hatsctrl -u** | The system partition-sensitive control script for the **hats** subsystem supports the **-c** and **-u** options, which cross system partitions. |
| **hmcmds -G** | The **-G** flag allows the **hmcmds** commands to be sent to any hardware on the SP system. |
| **hmmon -G** | The **-G** flag allows for the specification of hardware outside of the current system partition. |
| **hostlist**<br>**hostlist -f** *filename*<br>**hostlist -w** *hostname* | Host names need not be in the current system partition. |
| **hostlist -aG \| -nG \| -sG** | The **-G** flag modifies the **-a**, **-n**, or **-s** flag by expanding the scope to the entire physical SP system. |
| **hrctrl -c** | The system partition-sensitive control script for the **hr** subsystem supports the **-c** option, which crosses system partitions. |
| **hsdatalst -G** | The **-G** flag causes the display of HSD information to be for all system partitions. |
| **lppdiff -aG** | The **-G** flag modifies the **-a** flag (all nodes in the current system partition) by expanding the scope to all nodes in the entire physical SP system. |
| **lsauthpar -p** | The **-p** flag allows specification of a system partition other than the current system partition. |
| **nodecond -G** | The **-G** flag allows specification of a node outside of the current system partition. |
| **psyslrpt -w** *hostnames* | The host names supplied with the **-w** flag can be in any system partition (the **-a** flag will select all nodes in the current system partition). |
| **psyslclr -w** *hostnames* | The host names supplied with the **-w** flag can be in any system partition (the **-a** flag will select all nodes in the current system partition). |
| **penotify -w** *hostnames* | The host names supplied with the **-w** flag can be in any system partition (the **-a** flag will select all nodes in the current system partition). |
| **pmanctrl -c** | The system partition-sensitive control script for the **pman** subsystem supports the **-c** option, which crosses system partitions. |

| **Command** | **Effect** |
|---|---|
| Parallel commands: <br>• **p_cat** <br>• **pcp** <br>• **pdf** <br>• **pfck** <br>• **pexec** <br>• **pexscr** <br>• **pfind** <br>• **pfps** <br>• **pls** <br>• **pmv** <br>• **ppred** <br>• **pps** <br>• **prm** | Parallel commands can take the following options and will behave accordingly: <br>**-w** Host names specified with **-w** need not be in the current system partition. <br>*noderange* <br>Nodes specified by *noderange* must be in the current system partition. <br>*hostlist_args* <br>Host names specified with **hostlist** options **-w** or **-G** need not be in the current system partition (any other **hostlist** options operate within the current system partition). |
| **SDRArchive**, **SDRRestore** | Archives/restores the SDR representing the entire SP. |
| **SDRGetObjects -G** | The **-G** flag allows for retrieval of partitioned class objects from partitions other than the current system partition. Without the **-G**, objects which are in a partitioned class are retrieved from the current system partition only. |
| **SDRMoveObjects** | Moves objects from one system partition to another. |
| Other SDR commands | SDR commands that create, change or delete values work within the system partition. Note though that System classes (Frame, for example) are shared among all system partitions. Changes to system classes will affect other system partitions. |
| Security commands: <br>• **ext_srvtab** <br>• **kadmin** <br>• **kdb_destroy** <br>• **kdb_edit** <br>• **kdb_init** <br>• **kdb_util** <br>• **k4destroy** <br>• **k4init** <br>• **k4list** <br>• **kpasswd** <br>• **kprop** <br>• **ksrvtgt** <br>• **ksrvutil** <br>• **kstash** <br>• **rcmdtgt** <br>• **setup_authent** <br>• **spseccfg** | The function of these security commands is unchanged under system partitioning. That is, if they previously affected the entire SP, they continue to do so even if the system has been partitioned. If they previously had the ability to affect a remote node that function is unchanged in a system partitioned environment. |
| **sp_configdctrl -c** | The system partition-sensitive control script for the **sp_configd** subsystem supports the **-c** option, which crosses system partitions. |
| **spapply_config** | Applies a system partition configuration to the entire SP. |
| **spbootins** | If a boot server outside of the current system partition is specified, that node is prepared appropriately. |
| **spbootlist** | The command targets nodes in any system partition. |
| **spchvgobj** | The command targets nodes in any system partition. |
| **spframe** | Configures data for one or more frames across the entire SP. |
| **sphardware** | Global system partition can be selected from within the Perspective. |

| Command | Effect |
|---|---|
| **splm** | The target nodes defined in the input table can include nodes from any system partition. |
| **splst_versions -G** | The **-G** flag allows for retrieval of PSSP version information from nodes outside the current system partition. |
| **splstdata -G** | The **-G** flag allows display of information on nodes and adapters outside of the current system partition. |
| **splstadapters -G** | The **-G** flag lists information about target adapters outside of the current system partition. |
| **splstnodes -G** | The **-G** flag lists information about target nodes outside of the current system partition. |
| **spmirrorvg** | The command targets nodes in any system partition. |
| **spmkvgobj** | The command targets nodes in any system partition. |
| **spmon -G** | The **-G** flag allows specification of nodes outside of the current system partition. The **-G** flag is required when performing operations on any frame or switch. |
| **sprestore_config** | Restores the entire SP SDR from a previously made archive. |
| **sprmvgobj** | The command targets nodes in any system partition. |
| **spsitenv** | Site environment variables are specified for the SP system as a whole. The specification of **acct_master=** can be any node in the SP regardless of system partition. The specification of **install_image=** may cause boot server nodes outside of the current system partition to refresh the default installation image they will serve to their nodes. |
| **spsyspar** | Command is always in global mode. |
| \| **sptg** | Can launch TaskGuides that affect nodes in any system partition. |
| **spunmirrorvg** | The command targets nodes in any system partition. |
| **spverify_config** | Verifies the configuration of all system partitions in the SPsystem. |
| **st_clean_table** | Can specify a node name which is outside the current partition. |
| **st_status** | Can specify a node which is outside the current partition. |
| **supper** | File collections are implemented and managed without respect to system partition boundaries. |
| **sysctl** | The Sysctl client can send requests to any node in the SP. |
| **syspar_ctrl -c -G** | The **-c** and **-G** flags allow for the crossing of system partitions in providing a single interface to the control scripts for the system partition-sensitive subsystems. |
| **s1term -G** | The **-G** flag allows specification of a node outside of the current system partition. |
| **vsdatalst -G** | The **-G** flag causes the display of IBM Virtual Shared Disk information to be for all system partitions. |
| **vsdsklst -G** | The **-G** flag specifies the display of information for disks outside the current system partition. |

## rcmdtgt

### Purpose

**rcmdtgt** – Obtains and caches a Kerberos Version 4 ticket-granting-ticket for the local realm, with a maximum allowed lifetime, using the service key for the instance of **rcmd** on the local host.

### Syntax

**rcmdtgt**

### Flags

None.

### Operands

None.

### Description

Use this command to obtain a Kerberos Version 4 ticket-granting-ticket with a maximum allowed lifetime, using the service key for **rcmd.***localhost* found in the service key file at **/etc/krb-srvtab**. When using SP authentication services, these tickets have an unlimited lifetime. When using AFS authentication services, a maximum of 30 days is enforced.

This command must be run as root and is intended for use in shell scripts and other batch-type facilities. The **rcmdtgt** command retrieves your initial ticket and puts it in the ticket file specified by your KRBTKFILE environment variable. If the KRBTKFILE variable is undefined, your ticket is stored in the **/tmp/tkt***uid* file, where *uid* specifies your user identification number.

**Note:** These tickets are shared by all processes running under the user's IDs. The KRBTKFILE environment variable can be set to change the location of the ticket cache file.

Because the ticket obtained using this command does not expire, the user should be careful to delete the temporary ticket file.

When using **/usr/lpp/ssp/rcmd/bin/rcmdtgt**, remember to check that the authentication method is in fact Kerberos V4 before using **k4destroy** or **/usr/lpp/ssp/kerberos/bin/k4destroy** to destroy credentials. While Kerberos V4 may be configured, the authentication method may be superseded by DCE and you could be destroying credentials obtained by the system administrator through a Kerberos V4 login.

### Files

| | |
|---|---|
| **/etc/krb.conf** | Contains the name of the local realm. |
| **/etc/krb-srvtab** | Specifies the service key file. |
| **/tmp/tkt***uid* | Specifies the ticket cache file. |

## Location

**/usr/lpp/ssp/rcmd/bin/rcmdtgt**

## Related Information

Commands: **k4destroy**, **k4init**

File: **krb.conf**

Refer to the "RS/6000 SP Files and Other Technical Information" section of *PSSP: Command and Technical Reference* for additional **Kerberos** information.

## Examples

The following example, excerpted from the sample **script.cust** file, shows how **rcmdtgt** can be used in a shell script to perform the authentication required to use the SP **rcp** command:

```
# set the host name from which you will copy the file.
SERVER='cat /etc/ssp/server_host_name | cut -d" " -f1'

# Define a temporary ticket cache file, then get a ticket
export KRBTKFILE=/tmp/tkt.$$
/usr/lpp/ssp/rcmd/bin/rcmdtgt
#
# Perform kerberos-authenticated rcp
/usr/lpp/ssp/rcmd/bin/rcp $SERVER:/etc/resolv.conf /etc/resolv.conf

# Remove the ticket cache file
/bin/rm $KRBTKFILE
```

## rcp

### Purpose

**rcp** – Transfers files between a local and a remote host.

**Note:** The remote-to-remote copy is restricted for the Kerberos V4 execution path.

The **/usr/lpp/ssp/rcmd/bin/rcp** path points to the AIX Secure **/usr/bin/rcp**. This version supports DCE authentication, Kerberos V4 authentication and Standard Unix authentication. The authentication method in use on the host can be checked using the **lsauthent** command. The SP Kerberos 4 support is provided to AIX Secure remote commands through library routines. For more information on the AIX Secure remote commands shipped with AIX 4.3.1, refer to the AIX **rcp** online man pages for that release.

### Syntax

**rcp** [–**p**] [–**k** *Kerberos_realm*] {*User*@*Host*:*File* **|** *Host*:*File* **|** *File*}
{*User*@*Host*:*File* **|** *Host*:*File* **|** *File* **|** *User*@*Host*:*Directory* **|**
*Host*:*Directory* **|** *Directory*} **|** [–**F**] **|** [–**r**] {*User*@*Host*:*Directory* **|**
*Host*:*Directory* **|** *Directory*} {*User*@*Host*:*Directory* **|**
*Host*:*Directory* **|** *Directory*}

### Flags

Refer to the AIX Secure **rcp** online man page.

### Operands

Refer to the AIX Secure **rcp** online man page.

### Description

The **/usr/lpp/ssp/rcmd/bin/rcp** path points to the AIX Secure **/usr/bin/rcp**. The AIX Secure **/usr/bin/rcp** supports the SP Kerberos V4 **rcp** by executing an SP KV4 **rcp** routine when the authentication mechanism is set to Kerberos Version 4. The SP KV4 **rcp** routine communicates with the AIX Secure remote command daemon, **krshd**. The **krshd daemon** supports Kerberos V4 using the V4 compatibility library included with the Kerberos V5 that is used by DCE (Distributed Computing Environment).

The execution path of **rcp** will depend on the authentication methods that have been set using **chauthent**.

In the case of DCE and Kerberos V4, the Kerberos V4 execution path will be taken when DCE authentication is unsuccessful.

In the case of Kerberos V4 and AIX Standard, the Kerberos V4 execution path will be tried. If the user is unsuccessful with Kerberos V4 authentication, then the standard AIX execution path is tried. The standard AIX execution path requires a **.rhosts** file.

If the **/usr/kerberos/bin/rcp** path does not exist, a link is created from **/usr/bin/rcp** to **/usr/kerberos/bin/rcp**. The Kerberos V4 execution path of **rcp** uses this path when initially starting an **rcp** process on a remote host for compatibility purposes.

The authentication method used by the AIX Secure **rsh** can be listed using the AIX command **lsauthent** and can be changed by root using the AIX command **chauthent**. In order for the Kerberos V4 execution path to be taken, the Kerberos V4 authentication method must be set.

## Security

The remote host allows access only if at least one of the following conditions is satisfied:

- The local user ID is listed as a principal in the authentication database and had performed a **k4init** to obtain an authentication ticket.
- If a **$HOME/.klogin** file exists, it must be located in the local user's **/home** directory on the target system. The local user must be listed as well as any users or services allowed to **rsh** into this account. This file performs a similar function to a local **.rhosts** file. Each line in this file should contain a principal in the form of "principal.instance@realm." If the originating user is authenticated as one of the principals named in **.klogin**, access is granted to the account. The owner of the account is granted access if there is no **.klogin** file.

For security reasons, any **$HOME/.klogin** file must be owned by either the remote user or root, and only the owner should have read and write access.

## Restrictions

The SP Kerberos V4 **rcp** execution path does not support remote-to-remote copy.

## Files

**$HOME/.klogin**
Specifies remote users that can use a local user account.

**/usr/lpp/ssp/rcmd/bin/rcp**
Link to AIX Secure **/usr/bin/rcp** which calls the SP Kerberos 4 **rcp** routine if applicable.

**/usr/kerberos/bin/rcp**
Link to AIX Secure **/usr/bin/rcp** which calls the SP Kerberos 4 **rcp** routine if applicable.

## Prerequisite Information

Refer to *AIX Version 4 System Management Guide: Communications and Networks* for a network overview.

Refer to the chapter on security in *PSSP: Administration Guide* for an overview.

Refer to the "RS/6000 SP Files and Other Technical Information" section of *PSSP: Command and Technical Reference* for additional **Kerberos** information.

## Location

The **/usr/lpp/ssp/rcmd/bin/rcp** path points to the AIX Secure **/usr/bin/rcp**.

## Related Information

SP Commands: **k4init**, **rsh**

AIX Commands: **cp**, **ftp**, **rlogin**, **rsh**, **rshd**, **tftp**, **umask**

## Examples

In the following examples, the local user is listed in the authentication database and a ticket was obtained by issuing a **k4init** and the password. In the following examples, host1 is the local host and host2 is the remote host:

1. To copy a local file to a remote host, enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rcp localfile host2:/home/eng/jane
   ```

   The file **localfile** from the local host is copied to the remote host host2.

2. To copy a remote file from one remote host to the local host, enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rcp host2:/home/eng/jane/newplan/home/eng/mary
   ```

   The file **/home/eng/jane/newplan** is copied from remote host host1 to local host host2.

3. To send the directory subtree from the local host to a remote host and preserve the modification times and modes, enter:

   ```
   /usr/lpp.ssp/rcmd/bin/rcp -p -r report jane@host2:report
   ```

   The directory subtree report is copied from the local host to the home directory of user jane at remote host host2 and all modes and modification times are preserved. The remote file **/home/jane/.rhosts** includes an entry specifying the local host and user name.

4. This example shows how the root user can issue an **rcp** on a remote host. The root user must be in the authentication database and must have already issued **k4init** on the local host. The command is issued at the local host to copy the file, stuff, from node r05n07 to node r05n05.

   ```
   /usr/lpp/ssp/rcmd/bin/rsh r05n07 'export KRBTKTFILE=/tmp/rcmdtkt$$; \
   /usr/lpp/ssp/rcmd/bin/rcmdtgt; \
   /usr/lpp/ssp/rcmd/bin/rcp /tmp/stuff r05n05:/tmp/stuff;'
   ```

   The root user sets the KRBTKTFILE environment variable to the name of a temporary ticket-cache file and then obtains a service ticket by issuing the **rcmdtgt** command. The **rcp** command uses the service ticket to authenticate from host r05n07 to host r05n05.

## removehsd

### Purpose

**removehsd** – Removes one or more hashed shared disks, the virtual shared disks associated with them, and the System Data Repository (SDR) information for virtual shared disks on the associated nodes.

### Syntax

**removehsd** {**–v** *hsd_names* **|** –**a**} [–**f**]

### Flags

| | |
|---|---|
| –**v** | Specifies the hashed shared disk name or names that are to be removed by this command. |
| –**a** | Specifies that the command should remove all hashed shared disks in the system or system partition. |
| –**f** | Forces the system to unconfigure the hashed shared disks and its underlying virtual shared disks and remove them. If –**f** is not specified and any of the virtual shared disks that constitute the hashed shared disks to be removed are configured or the hashed shared disk itself is configured, the command is unsuccessful. |

### Operands

None.

### Description

Use this command to remove the logical volumes associated with virtual shared disks in the set of hashed shared disks. The order in which the virtual shared disks that make up the hashed shared disks and the hashed shared disks themselves are removed is the reverse of the order in which they were created.

If the virtual shared disk or hashed shared disk is configured on any of the nodes on the system partition, this command is unsuccessful, unless the –**f** flag is specified.

### Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/removehsd**

## Related Information

Commands: **createhsd**, **removevsd**

## Examples

To unconfigure and remove the virtual shared disks associated with the hashed shared disks DATA and remove the hashed shared disk as well, type:

```
removehsd -d DATA -f
```

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Remove a Hashed Shared Disk option.

## removevsd

### Purpose

**removevsd** – Removes a set of virtual shared disks that are not part of any hashed shared disk.

### Syntax

**removevsd** {–**v** *vsd_names* **|** –**a**} [–**f**]

### Flags

| | |
|---|---|
| –**v** | Specifies the virtual shared disk name or names that are to be removed by this command. |
| –**a** | Specifies that the command should remove all virtual shared disks in the system or system partition. |
| –**f** | Forces the system to unconfigure the virtual shared disks and remove them. If –**f** is not specified and any of the virtual shared disks that are to be removed are configured, the command is unsuccessful. |

### Operands

None.

### Description

Use this command to remove the logical volumes associated with the virtual shared disks and update the backup nodes' Object Data Managers (ODMs), if any exist. The virtual shared disk information will be deleted from the System Data Repository (SDR). The removal of the virtual shared disks is done in the reverse of the order in which they were created. Volume groups are not removed with this command.

If the virtual shared disk is configured on any of the nodes on the system partition, this command is unsuccessful, unless the –**f** flag is specified.

**Note:** This command is unsuccessful if one of the virtual shared disks named in *vsd_names* belongs to a hashed shared disk. To remove virtual shared disks that belong to a hashed shared disk, use **removehsd**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Remove a Virtual Shared Disk option.

### Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Related Information

Commands: **createvsd**, **removehsd**

## Location

**/usr/lpp/csd/bin/removevsd**

## Examples

To unconfigure and remove all defined virtual shared disks in a system or system partition, enter:

```
removevsd -a -f
```

## resource_center

### Purpose

**resource_center** – Invokes the RS/6000 SP Resource Center.

### Syntax

**resource_center** [–**c**] [**Netscape flags ...**]

### Flags

–**c** Forces the SP Resource Center to prompt the user for the Netscape location.

**Netscape flags**
Refer to Netscape documentation or run **netscape –h** to list available flags.

### Operands

None.

### Description

The RS/6000 SP Resource Center provides one single interface for all softcopy SP documentation and information resources. It consists of HTML files, Java and JavaScript, and runs in Netscape Navigator. The SP Resource Center provides access to the following RS/6000 SP information:

- Publications, READMEs, Redbooks, White Papers
- Up-to-date Service Information
- SP Product Information (Software & Hardware)
- Many other online resources useful to the RS/6000 SP user and administrator.

Upon invoking the **resource_center** command for the first time, a dialog box will ask you for the location of the Netscape executable that is installed on your system. Enter the full pathname to the Netscape program (for example, **/usr/local/bin/netscape**). This path information is stored in your **$HOME/.resctr** file, and you will only be prompted for it once.

The **resource_center** command will bring up Netscape Navigator with the top level RS/6000 SP Resource Center page loaded. There are three frames on the SP Resource Center interface. The frame on the top is the Title frame. The frame on the left is the Navigation frame. The large frame on the right is the Display frame.

The Title frame lets you access the IBM Home Page on the Internet (click on the IBM logo), go back to the top of the SP Resource Center (the "Home" link), search the contents of the SP Resource Center (the "Search" link), display an index of the SP Resource Center's contents (the "Index" link), and obtain help about the use of the SP Resource Center (the "Help" link).

The Navigation frame lets you select content to view. This frame contains categories that may be expanded and collapsed to display their sub-categories. Click on a category with a right-pointing arrow to expand the category, and click on a category with a down-pointing arrow to collapse the category. When one category is expanded, all other categories are collapsed. When a category is expanded, any of the sub-categories may be selected, and the contents are displayed in the Display frame.

Some sub-categories on the Navigation frame include a small "world" icon that indicates that the link will take you to the Internet. If you do not have an internet connection, these links will not work. When you select a link to the internet, a new Netscape window appears. This ensures that the SP Resource Center Title and Navigation frames do not get in the way of the internet web page.

The Display frame is used to display all local information. When a non-internet link is selected from the Navigation frame, the resulting information is displayed in the Display frame.

To quit the SP Resource Center, exit Netscape Navigator.

## Environment Variables

The environment variable **NETSCAPE** is used (if set) to specify the pathname to the Netscape Navigator web browser that will be used to display the RS/6000 SP Resource Center.

## Files

| | |
|---|---|
| **$HOME/.resctr** | Stores the pathname to Netscape Navigator for each user. |

## Restrictions

If your machine does not have a connection to the internet, some of the SP Resource Center's hyperlinks will not function.

Web Pages on the internet that the SP Resource Center points to may not be available due to the dynamic nature of the web.

## Prerequisite Information

Netscape Navigator version 4 or later is required to run this command. The first time each user invokes the **resource_center** command, the pathname to the Netscape program is stored in **$HOME/.resctr**.

If online publications are installed on the system, the SP Resource Center will use the local copies, otherwise it will look on the web for the online publications.

## Location

**/usr/lpp/ssp/bin/resource_center**

## Examples

To invoke the RS/6000 SP Resource Center, enter:

```
resource_center
```

This example assumes the directory **/usr/lpp/ssp/bin** is in your path.

## resumevsd

### Purpose

**resumevsd** – Activates an available virtual shared disk.

### Syntax

**resumevsd [–p | –b]** **–a |** {*vsd_name* ...}

### Flags

| | |
|---|---|
| **–p** | Specifies that the primary server node defined for the global volume group is to be the active server. |
| **–b** | Specifies that the secondary server node defined for the global volume group is to be the active server. **Note:** This flag is used only by the Recoverable Virtual Shared Disk subsystem. |
| **–a** | Specifies that all the virtual shared disks that have been defined are to be resumed. |

### Operands

| | |
|---|---|
| *vsd_name* | Specifies a virtual shared disk. |

### Description

The **resumevsd** command brings the specified virtual shared disks from the suspended state to the active state. The virtual shared disks remains available. Read and write requests which had been held while the virtual shared disk was in the suspended state are resumed.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Resume a Virtual Shared Disk option.

### Security

You must have root privilege to run this command.

### Restrictions

1. If you have the Recoverable Virtual Shared Disk software installed and operational, do not use this command. The results may be unpredictable.

   See *PSSP: Managing Shared Disks*
2. The **–b** flag is used only by the Recoverable Virtual Shared Disk subsystem.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/resumevsd**

## Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **startvsd**, **stopvsd**, **suspendvsd**, **ucfgvsd**

## Examples

To bring the virtual shared disk **vsd1vg1n1** from the suspended state to the active state, enter:

```
resumevsd vsd1vg1n1
```

## rmkp

### Purpose

**rmkp** – Removes Kerberos principals.

### Syntax

**rmkp** –**h**

**rmkp** [–**n**] [–**v**] {*name*[.*instance*]**|***name.***|**.*instance*} ...

### Flags

–**h** Displays usage information.

–**n** Suppresses prompting for confirmation.

–**v** Specifies verbose mode (displays informational messages).

### Operands

**{***name***[**.*instance***]|***name.***|**.*instance***} ...**
Identifies specific principals to remove. When the command is invoked interactively (without the –**n** flag and not through Sysctl), you can use special notation to select all principals with a particular name or instance that you want to remove. Specify *name.* to remove all principals with a specific name or *.instance* to remove all principals with a specific instance.

**Note:** The *name* must be followed by a period and the *instance* must be preceded by a period.

### Description

Use this command to remove principals from the local Kerberos database. You will be prompted to confirm each deletion prior to its execution. This command will not remove any of the four principals that were predefined by Kerberos when the database was created. Deleted entries are saved in the **/var/kerberos/database/rmkp.save.**<**PID>** file, in the readable ASCII format produced by the **kdb_util dump** command. The **rmkp** command should normally be used only on the primary server. If there are secondary authentication servers, the **push-kprop** command is invoked to propagate the change to the other servers. The command can be used to update a secondary server's database, but the changes may be negated by a subsequent update from the primary.

### Files

**/var/kerberos/database/admin_acl.add**
Access control list for **kadmin**, **mkkp**, and **rmkp**.

**/var/kerberos/database/principal.***
Kerberos database files.

**/var/kerberos/database/rmkp.save.**<**pid>**
File containing removed Kerberos database entries.

## Standard Output

When the –**v** option is omitted, only the prompt for confirmation is written to standard output. When the –**v** flag is specified, the disposition of each selected principal is indicated by a message, and the name of the file containing the removed entries is printed. The –**v** flag has no effect on error messages written to standard error.

## Exit Values

**0** Indicates the successful completion of the command. At least one principal was found that matched the specified names. Whether or not any were removed depends on the responses you entered when prompted. If you entered a principal that does not exist, or if you entered an operand of the form *name.* or *.instance* in noninteractive mode, a message is written to standard error and processing continues with any remaining principals.

**1** Indicates that an error occurred and no principal was removed. One of the following conditions was detected:

- The command was incorrectly specified with no operand or a flag that is not valid.
- No principal was found matching the names specified.
- The host on which the command was issued is not an authentication server.
- The database was changed by another process while this command was executing.
- The **kdb_util** command was unsuccessful.

## Security

The **rmkp** command can be run by the root user logged in on a Kerberos server host. It can be invoked indirectly as a Sysctl procedure by a Kerberos database administrator who has a valid ticket and is listed in the **admin_acl.add** file.

## Restrictions

When you execute the **rmkp** command through the Sysctl procedure of the same name, the –**n** flag is added to your command invocation. This is required because Sysctl does not provide an interactive environment that supports prompting for confirmation. Suppressing confirmation increases the risk of unintentionally removing the wrong principal. In this mode, each principal to be removed must be named explicitly; selection of multiple principals by name or instance alone is not allowed. Since nonroot Kerberos administrators can execute this command only through Sysctl, you must be root on the server to use the special notation for selecting multiple principals.

## Location

**/usr/kerberos/etc/rmkp**

## Related Information

Commands: **chkp**, **kadmin**, **kdb_util**, **lskp**, **mkkp**, **sysctl**

## Examples

1. To remove Kerberos principal tempuser, enter:

   ```
   rmkp tempuser
   ```

   You should receive a prompt similar to the following:

   ```
   Confirm removal of principal tempuser? (y or n): y
   ```

2. To remove (be given the option to remove) all instances of joe, frank, and the rcmd service principal with instance node 25tr, enter:

   ```
   rmkp -v joe. frank rcmd.node25tr
   ```

   You should receive prompts similar to the following:

   ```
   Confirm removal of principal joe? (y or n): n

   joe was not removed

   Confirm removal of principal joe.admin? (y or n): y

   joe.admin was removed

   Confirm removal of principal frank? (y or n): y

   frank was removed

   Confirm removal of principal rcmd.node25tr? (y or n): y

   rcmd.node25tr was removed

   Removed entries were saved in /var/kerberos/database/rmkp.save.7942
   ```

## rsh

### Purpose

**/usr/lpp/ssp/rcmd/bin/rsh** – Executes the specified command at the remote host through the AIX Secure **/usr/bin/rsh** command.

This path points to the AIX Secure **/usr/bin/rsh**. This version supports DCE authentication, Kerberos V4 authentication and Standard Unix authentication. The authentication method in use on the host can be checked using the lsauthent command. The SP Kerberos 4 support is provided to AIX Secure remote commands through library routines. For more information on the AIX Secure remote commands shipped with AIX 4.3.1, refer to the AIX **rsh** online man pages for that release.

### Syntax

{**rsh** | **remsh**} *RemoteHost* [–**n**] [–**l** *user*] [–**k** *Kerberos_realm*] [–**F**] [–**f**]*command*

### Flags

Refer to the AIX Secure **rsh** online man page.

### Operands

Refer to the AIX Secure **rsh** online man page.

### Description

The **/usr/lpp/ssp/rcmd/bin/rsh** and **remsh** path point to the AIX Secure **/usr/bin/rsh**. The AIX Secure **/usr/bin/rsh** supports the SP Kerberos V4 **rsh** by executing an SP KV4 **rsh** routine when the authentication mechanism is set to Kerberos Version 4. Ths SP KV4 **rsh** routine communicates with the AIX Secure remote command daemon, **krshd**. The **krshd daemon** supports Kerberos V4 using the V4 compatibility library included with the Kerberos V5 that is used by DCE (Distributed Computing Environment).

The execution path of **rsh** will depend on the authentication methods that have been set using **chauthent**. In the case of DCE and Kerberos V4, the Kerberos V4 execution path will be taken when DCE authentication is unsuccessful. In the case of Kerberos V4 and AIX Standard, the Kerberos V4 execution path will be tried. If the user is unsuccessful with Kerberos V4 authentication, then the standard AIX execution path is tried. The standard AIX execution path requires a **.rhosts** file.

The authentication method used by the AIX Secure **rsh** can be listed using the AIX command **lsauthent** and can be changed by root using the AIX command **chauthen**t. In order for the Kerberos V4 execution path to be taken, the Kerberos V4 authentication method must be set.

## Files

**$HOME/.klogin**
Specifies remote users that can use a local user account.

**/usr/lpp/ssp/rcmd/bin/rsh**
Link to AIX Secure **/usr/bin/rsh** which calls the SP Kerberos 4 **rsh** routine if applicable.

**/usr/lpp/ssp/rcmd/bin/remsh**
Link to AIX Secure **/usr/bin/rsh** which calls the SP Kerberos 4 **rsh** routine if applicable.

## Security

The remote host allows access only if at least one of the following conditions is satisfied:

- The local user ID is listed as a principal in the authentication database and had performed a **k4init** to obtain an authentication ticket.
- If a **$HOME/.klogin** file exists, it must be located in the local user's **/home** directory on the target system. The local user must be listed as well as any users or services allowed to **rsh** into this account. This file performs a similar function to a local **.rhosts** file. Each line in this file should contain a principal in the form of "principal.instance@realm." If the originating user is authenticated as one of the principals named in **.klogin**, access is granted to the account. The owner of the account is granted access if there is no **.klogin** file.

For security reasons, any **$HOME/.klogin** file must be owned by either the remote user or root, and only the owner should have read and write access.

## Restrictions

The SP Kerberos V4 **rsh** execution path does not support a default to **rlogin** when there is no command listed on the command line. A message is issued indicating this restriction.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP)

## Prerequisite Information

Refer to *AIX Version 4 System Management Guide: Communications and Networks* for a network overview.

Refer to the chapter on security in *PSSP: Administration Guide* for an overview.

Refer to the "RS/6000 SP Files and Other Technical Information" section of *PSSP: Command and Technical Reference* for additional **Kerberos** information.

## Location

**/usr/lpp/ssp/rcmd/bin/rsh**

## Related Information

SP Commands: **k4init**, **rcp**

AIX Commands: **lsauthent**, **chauthent**, **/usr/bin/rsh**, **/usr/bin/rcp**, **rlogin**, **krshd**, **telnet**

## Examples

In the following examples, the local user is listed in the authentication database and has obtained a ticket by issuing a **k4init** and the password.

1. To check the amount of free disk space on a remote host, enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh host2 df
   ```

   The amount of free disk space on host2 is displayed on the local system.

2. To append a remote file to another file on the remote host, place the >> metacharacters in quotation marks, and enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh host2 cat test1 ">>" test2
   ```

   The file test1 is appended to test2 on remote host host2.

3. To append a remote file at the remote host to a local file, omit the quotation marks, and enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh host2 cat test2 >> test3
   ```

   The remote file test2 on host2 is appended to the local file test3.

4. To append a remote file to a local file and use a remote user's permissions at the remote host, enter:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh host2 -l jane cat test4 >> test5
   ```

   The remote file test4 is appended to the local file test5 using user jane's permissions at the remote host.

   This example shows how the root user can issue an **rcp** on a remote host. The root user must be in the authentication database and must have already issued **k4init** on the local host. The command is issued at the local host to copy the file, stuff, from node r05n07 to node r05n05.

   ```
   /usr/lpp/ssp/rcmd/bin/rsh r05n07 'export KRBTKTFILE=/tmp/rcmdtkt$$; \
   /usr/lpp/ssp/rcmd/bin/rcmdtgt; \
   /usr/lpp/ssp/rcmd/bin/rcp /tmp/stuff r05n05:/tmp/stuff;'
   ```

   The root user sets the KRBTKTFILE environment variable to the name of a temporary ticket-cache file and then obtains a service ticket by issuing the **rcmdtgt** command. The **rcp** uses the service ticket to authenticate from host r05n07 to host r05n05.

## rvsdrestrict

### Purpose

**rvsdrestrict** – Displays and sets which level of the IBM Recoverable Virtual Shared Disk software is to run when you have a system partition with mixed levels of the PSSP or IBM Recoverable Virtual Shared Disk software.

### Syntax

**rvsdrestrict** {**-l** | **-s** {RVSD1.2 | RVSD2.1 | RVSD3.1 | Reset}}

### Flags

**–l** Lists the current rvsd subsystem run level as recorded in the SDR.

**–s** Sets the rvsd subsystem run level to RVSD 1.2, RVSD 2.1, or RVSD 3.1 in the SDR, or resets the rvsd subsystem run level.

### Operands

None.

### Description

The **rvsdrestrict** command is used to restrict the level that the IBM Recoverable Virtual Shared Disk software will run at. This command must be used when in a system partition with mixed levels of PSSP and or mixed levels of the IBM Recoverable Virtual Shared Disk software. If a node has a lower level of the IBM Recoverable Virtual Shared Disk software installed than what is set with this command, then the rvsd subsystem will not start on that node.

This command does not dynamically change rvsd subsystem run levels across the SP. An rvsd subsystem instance will only react to this information after being restarted. Thus, if your cluster runs at a given level, and you want to override this level you must stop rvsd subsystem on all nodes, override the level, and restart.

### Standard Output

Current rvsd subsystem run level as recorded in the SDR.

### Location

**/usr/lpp/csd/bin/rvsdrestrict**

### Examples

1. If you were to have nodes with RVSD1.2, RVSD2.1, and RVSD3.1 all in the same system partition, and you wanted them to all run and coexist, you would issue:

   ```
   rvsdrestrict -s RVSD1.2
   ```

   This will force all the rvsd subsystems to run at the functionality level of RVSD1.2.

2. If you were to have nodes with RVSD1.2, RVSD2.1, and RVSD3.1 all in the same system partition, but you only wanted the rvsd subsystem to start on nodes that were capable of running RVSD3.1, then you would issue:

```
rvsdrestrict -s RVSD3.1
```

## SDR_config

### Purpose

**SDR_config** – Queries the existing hardware on the SP and updates the System Data Repository (SDR) as necessary.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

Use this command only under the direction of the IBM Support Center. You should only run this command if SDR configuration problems have been diagnosed.

### Syntax

**SDR_config** [**-l**] [–**u**] [**-e** *frame:slot:type...*] [**-v**] [**-d**]

### Flags

–**l** Logs output. All output will be logged to the **SDR_config** log file **/var/adm/SPlogs/sdr/SDR_config.log**.

–**u** Updates switch information.

–**e** *frame:slot:type*
Adds one or more external or dependent nodes. Each node is specified by the triple *frame:slot:type* where frame and slot indicate the physical location of the node, and type is the hardware type for the node (for example, 99 for dependent nodes). Additional nodes can be specified by separating the triples with a colon (:).

–**v** Verbose mode. Status messages, informational messages, and SDR commands will be echoed to **stdout**.

–**d** Debug mode. No SDR changes will be made. This option is most useful if run with the verbose (**-v**) option to see what the command would do during normal execution.

### Operands

None.

### Description

**SDR_config** creates and configures objects in the SDR. It queries **hardmon** for a current list of all frames, nodes, and switches. It then queries the existing data in the SDR. New objects are created as necessary and existing objects are updated to match the **hardmon** data for SDR frames, nodes, dependent nodes, switches, host responds, switch responds and the system partitioning map. Data in the SDR for which no hardware information is available is **NOT** deleted, since the hardware may be temporarily disabled or cannot be sensed by **hardmon**.

**SDR_config** is normally invoked during system initialization when the SP logging daemon is started, when **hardmon** records hardware changes through the logging daemon, and by the **hmreinit**, **spframe**, and **endefnode** commands.

If **SDR_config** encounters severe problems, no changes will be made and the SDR will be restored to the same state as it was before the command was invoked.

When **SDR_config** is invoked, it creates a lock file in **/var/adm/SPlogs/sdr/SDR_config.lock** and removes it when the command has completed execution. This lock file prevents two instances of **SDR_config** from executing at the same time.

## Files

**/var/adm/SPlogs/sdr/SDR_config.log**
Records output if the **-v** option was specified.

**/var/adm/SPlogs/sdr/SDR_config.lock**
Locks the command to prevent more than one instance of the **SDR_config** command from running at a time.

## Standard Output

All informational messages generated as a result of the **-v** option will be written to standard output.

If the **-l** option is specified, standard output will be redirected to the log file **/var/adm/SPlogs/sdr/SDR_config.log**.

## Standard Error

All error messages will be written to standard output.

If the **-l** option is specified, standard error will be redirected to the log file **/var/adm/SPlogs/sdr/SDR_config.log**.

## Exit Values

**0** Indicates the successful completion of the command.

**1** Indicates that a recoverable error occurred.

**2** Indicates that an irrecoverable error occurred.

**3** Indicates that **SDR_config** lock exists - command is already in use.

## Security

You must have root privilege to run this command or be a member of the system group.

You must run this command from a user who has monitor authority in **/spdata/sys1/spmon/hmacls**. The user must have a non-expired authentication ticket.

## Restrictions

Use this command only under the direction of the IBM Support Center. You should only run this command if SDR configuration problems have been diagnosed.

This command can only be issued on the control workstation.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP) **ssp.basic** file set.

## Location

**/usr/lpp/ssp/install/bin/SDR_config**

## Related Information

Commands: **endefnode**, **hmreinit**, **SDR_init**, **spframe**, **splogd**

## Examples

1. To check if SDR configuration changes may be required, but not actually perform those changes, login to the control workstation and enter:

   ```
   /usr/lpp/ssp/install/bin/SDR_config -v -d
   ```

2. To update the SDR to reflect the existing hardware, login to the control workstation and enter:

   ```
   /usr/lpp/ssp/install/bin/SDR_config
   ```

## SDR_init

### Purpose

**SDR_init** – Initializes the System Data Repository (SDR) during PSSP installation.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

Use this command only under the direction of the IBM Support Center. You should only run this command if SDR configuration problems have been diagnosed.

### Syntax

**SDR_init** [**-l**] [**-v**] [**-d**]

### Flags

| | |
|---|---|
| **–l** | Logs output. All output is logged to the **SDR_config** log file **/var/adm/SPlogs/sdr/SDR_config.log**. |
| **–v** | Verbose mode. Status messages, informational messages, and SDR commands is echoed to **stdout**. |
| **–d** | Debug mode. No SDR changes will be made. This option is most useful if run with the verbose option (**-v**) to see what the command would do during normal execution. |

### Operands

None.

### Description

**SDR_init** initializes the SDR during PSSP installation. It creates and initializes new SDR classes and attributes as required by the new level of PSSP being installed.

### Files

**/var/adm/SPlogs/sdr/SDR_config.log**
Records output if the **-l** option was specified.

### Standard Output

All informational messages generated as a result of the **-v** option will be written to standard output.

If the **-l** option is specified, standard output will be redirected to the log file **/var/adm/SPlogs/sdr/SDR_config.log**.

## Standard Error

All error messages are written to standard error.

If the **-l** option is specified, standard error will be redirected to the log file **/var/adm/SPlogs/sdr/SDR_config.log**.

## Security

You must have root privilege to run this command or be a member of the system group.

You must run this command from a user who has monitor authority in **/spdata/sys1/spmon/hmacls**. The user must have a non-expired authentication ticket.

## Restrictions

Use this command only under the direction of the IBM Support Center. You should only run this command if SDR initialization problems have been diagnosed.

This command can only be issued on the control workstation.

## Location

**/usr/lpp/ssp/install/bin/SDR_init**

## Related Information

Commands: **SDR_config**

## Examples

1. To check if SDR initialization changes may be required, but not actually perform those changes, login to the control workstation and enter:

   ```
   /usr/lpp/ssp/install/bin/SDR_init -v -d
   ```

2. To initialize the SDR for the current level of PSSP, login to the control workstation and enter:

   ```
   /usr/lpp/ssp/install/bin/SDR_init
   ```

## SDR_test

### Purpose

**SDR_test** – Verifies that the installation and configuration of the System Data Repository (SDR) component of the SP system completed successfully.

### Syntax

**SDR_test** [–**q**] [–**l** *log_file*]

### Flags

| | |
|---|---|
| –**q** | Specifies quiet mode; suppresses output to standard error. |
| –**l** *log_file* | Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.) |

### Operands

None.

### Description

This command verifies that the SDR is functioning properly. After clearing out a class name, it creates the class, performs various SDR tasks, and removes the class when done.

A return code of 0 indicates that the test completed as expected; otherwise it returns the number of errors. If you do not specify the –**q** flag, a message is displayed on standard output that indicates if the tests were successful or not. In either case, the command returns 0 if successful, 1 if unsuccessful. If errors are detected, more detailed information is recorded in the log file. If you do not specify the –**l** flag, error messages are recorded in **/var/adm/SPlogs/SDR_test.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the System Data Repository option.

### Files

**/var/adm/SPlogs/SDR_test.log**
Default log file.

### Related Information

Commands: **CSS_test**, **st_verify**, **SYSMAN_test**, **spmon_ctest**, **spmon_itest**

### Location

**/usr/lpp/ssp/bin/SDR_test**

## Examples

To verify the System Data Repository following installation, saving error messages in **sdr_err** in the current working directory, enter:

```
SDR_test -l sdr_err
```

## SDRAddSyspar

### Purpose

**SDRAddSyspar** – Creates a new daemon using the System Resource Controller (SRC). The new daemon creates a subdirectory under the **/spdata/sys1/sdr/partitions** directory.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRAddSyspar** *IP_address*

### Flags

None.

### Operands

| | |
|---|---|
| *IP_address* | Specifies a TCP dotted decimal address (real or alias). |

### Description

This command creates a new instance of the SDR daemon and passes it the IP address of the system partition. It does not perform all of the system management tasks involved in creating a system partition.

### Location

**/usr/lpp/ssp/bin/SDRAddSyspar**

## SDRArchive

### Purpose

**SDRArchive** – Archives the entire contents of the System Data Repository (SDR), except for the archives directory, for later retrieval.

**Attention**

**Migration Note:** Each new PSSP release may introduce new SDR classes and attributes. Use caution when using **SDRArchive** and **SDRRestore** to avoid overwriting new SDR classes and attributes. IBM suggests that after migration you do not execute **SDRRestore** from a back level system since it will overwrite any new SDR classes and attributes.

### Syntax

**SDRArchive** [*append_string*]

### Flags

None.

### Operands

| | |
|---|---|
| *append_string* | If specified, the *append_string* is appended to the name of the backup file. |

### Description

Use this command to tar the contents of the SDR and put the file in the **/spdata/sys1/sdr/archives** subdirectory. You might want to mount this directory from another machine or physical disk drive to protect against an error in the drive holding the SDR. The file name is **backup.JULIANdate.***HHMM.append_string*, where **JULIANdate.***HHMM* is a number or string uniquely identifying the date and time of the archive and *append_string* is the argument entered in the command invocation, if specified.

### Location

**/usr/lpp/ssp/bin/SDRAddSyspar**

## SDRChangeAttrValues

### Purpose

**SDRChangeAttrValues** – Changes attribute values of one or more objects.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRChangeAttrValues** *class_name* [*attr***==***value* ... ] *attr***=***value* ...

### Flags

None.

### Operands

| | |
|---|---|
| *class_name* | Identifies the target object and checks the class to see if it is a system class or a partitioned class. |
| *attr***==***value* | Specifies attribute values to match for the change to be made (note double equal signs signifying comparison.) |
| *attr***=***value* | Specifies target attribute to change and value to be assigned. |

### Description

This command changes one or more attribute values in a specified object with certain other attribute values.

### Location

**/usr/lpp/ssp/bin/SDRChangeAttrValues**

## SDRClearLock

### Purpose

**SDRClearLock** – Unlocks a System Data Repository (SDR) class.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRClearLock** *class_name*

### Flags

None.

### Operands

*class_name* Identifies the target class and removes the lock on that class if a lock exists.

### Description

Use this command when a process that obtained a lock ends abnormally and does not unlock the class.

### Location

**/usr/lpp/ssp/bin/SDRClearLock**

## SDRCreateAttrs

### Purpose

**SDRCreateAttrs** – Creates new attributes for a System Data Repository (SDR) class.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRCreateAttrs** *class_name attr***=***datatype* ...

### Flags

None.

### Operands

*class_name* Identifies the target object.

*attr***=***datatype*
Names the new attribute and defines the data type as an integer (**int**), a floating-point value (**float**), or a string (**string**).

### Description

This command creates one or more new attributes for a target class.

### Location

**/usr/lpp/ssp/bin/SDRCreateAttrs**

## SDRCreateClass

### Purpose

**SDRCreateClass** – Creates a partitioned class.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRCreateClass** *class_name attr***=***datatype* ...

### Flags

None.

### Operands

*class_name* Identifies the new object class.

*attr***=***datatype*
Names the new attribute and defines the data type as an integer (**int**), a floating-point value (**float**), or a string (**string**).

### Description

This command creates a partitioned class and defines its attributes.

### Location

**/usr/lpp/ssp/bin/SDRCreateClass**

## SDRCreateFile

### Purpose

**SDRCreateFile** – Reads the specified AIX file and puts it in the System Data Repository (SDR) under the specified SDR file name.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRCreateFile** *AIX_filename SDR_filename*

### Flags

None.

### Operands

*AIX_filename*
Identifies the AIX file name to be written to the SDR.

*SDR_filename*
Specifies the name of the new SDR file.

### Description

This command creates a partitioned SDR file from an AIX file. Use **SDRCreateSystemFile** to create a system file. Use **SDRRetrieveFile** to retrieve the file.

### Location

**/usr/lpp/ssp/bin/SDRCreateFile**

## SDRCreateObjects

### Purpose

**SDRCreateObjects** – Creates new objects in a system class or a partitioned class.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRCreateObjects** *class_name attr***=***value* ...

### Flags

None.

### Operands

| | |
|---|---|
| *class_name* | Identifies the class of the new objects. |
| *attr***=***value* | Specifies the target attribute and value to be assigned. |

### Description

This command creates one or more new objects. Not all attributes for an object need to be specified in this call; however, a subset of the attributes that uniquely identify this object must be entered at this time.

### Location

**/usr/lpp/ssp/bin/SDRCreateObjects**

## SDRCreateSystemClass

### Purpose

**SDRCreateSystemClass** – Creates a system class.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRCreateSystemClass** *class_name attr***=***datatype* ...

### Flags

None.

### Operands

*class_name* Identifies the new object class.

*attr***=***datatype*
Names the new attribute and defines the data type as an integer (**int**), a floating-point value (**float**), or a string (**string**).

### Description

This command creates a system class and defines its attributes.

### Location

**/usr/lpp/ssp/bin/SDRCreateSystemClass**

## SDRCreateSystemFile

### Purpose

**SDRCreateSystemFile** – Creates a file that can be retrieved from any system partition.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRCreateSystemFile** *AIX_filename SDR_filename*

### Flags

None.

### Operands

*AIX_filename*
: Specifies the AIX file name.

*SDR_filename*
: Specifies the System Data Repository (SDR) file name.

### Description

This command reads the AIX file and puts it in the repository under the SDR file name. Note that only ASCII files can be saved. Results are unpredictable if binary files are used with this command. Clients connected to any system partition can read this file.

Use **SDRRetrieveFile** to retrieve this file. If a system file and a partitioned file exist with the same name, the partitioned file will be returned from **SDRRetrieveFile**.

### Location

**/usr/lpp/ssp/bin/SDRCreateSystemFile**

## SDRDeleteFile

### Purpose

**SDRDeleteFile** – Deletes the specified System Data Repository (SDR) file.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRDeleteFile** *SDR_filename*

### Flags

None.

### Operands

*SDR_filename*
Specifies the name of the SDR file to be deleted.

### Description

This command deletes the partitioned file *SDR_filename*, if it exists. If the *SDR_filename* partitioned file does not exist, it will delete the *SDR_filename* system file. This command will not delete both the partitioned file and the system file.

### Location

**/usr/lpp/ssp/bin/SDRDeleteFile**

## SDRDeleteObjects

### Purpose

**SDRDeleteObjects** – Deletes objects from the System Data Repository (SDR).

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRDeleteObjects** *class_name* [*attr*==*value* ... ]

### Flags

None.

### Operands

| | |
|---|---|
| *class_name* | Identifies the class of the object to be deleted. |
| *attr*==*value* | Specifies specific attribute values to match to qualify the object for deletion. |

### Description

This command deletes one or more objects. All objects in the specified class with attribute values matching those specified are deleted. If no *attr*==*value* pairs are specified, this command will match all objects in the class and all objects will be deleted.

### Location

**/usr/lpp/ssp/bin/SDRDeleteObjects**

## SDRGetObjects

### Purpose

**SDRGetObjects** – Sends contents of attributes in specified object to standard output.

### Syntax

**SDRGetObjects** [–**G**] [–**x**] [–**q**] [–**d** *delimiter*] *class_name* [*attr*==*value* ...] [*attr* ...]

### Flags

| | |
|---|---|
| –**G** | For a partitioned class, returns the objects that match the *attr*==*value* arguments in the class specified from **all** system partitions. For system classes, the –**G** option has no effect. |
| –**x** | Inhibits the output of the header line. |
| –**q** | Specifies quiet mode; suppresses message output to standard error. |
| –**d** *delimiter* | Allows the user to specify a delimiter in the output. |

### Operands

| | |
|---|---|
| *class_name* | Identifies the target class. |
| *attr*==*value* | Specifies attribute values to match to qualify the objects for the operation (note the double equal signs, which signify comparison.) |
| *attr* | Specifies which attribute values should be returned by the command. The order of these arguments is the order they are written in the output. If no *attr* arguments are entered, all attributes will be selected. |

### Description

This command retrieves and sends to standard output attribute values in the specified objects.

### Location

**/usr/lpp/ssp/bin/SDRGetObjects**

### Examples

1. To query the SDR Adapter class for the node number and network address of all switch adapters, enter:

   ```
   SDRGetObjects -G Adapter adapter_type==css0 node_number netaddr
   ```

   You should receive output similar to the following:

```
node_number netaddr
          1 129.40.102.129
          3 129.40.102.131
          5 129.40.102.133
          6 129.40.102.134
          7 129.40.102.135
          8 129.40.102.136
          9 129.40.102.137
         10 129.40.102.138
         11 129.40.102.139
         12 129.40.102.140
         13 129.40.102.141
         14 129.40.102.142
         15 129.40.102.143
         16 129.40.102.144
```

2. To determine the reliable host name, switch node number, and switch chip port of every Node object, enter:

   ```
   SDRGetObjects -G Node reliable_hostname switch_node_number \
   switch_chip_port
   ```

   You should receive output similar to the following:

   ```
   reliable_hostname switch_node_number switch_chip_port
   k3n01.hpssl.kgn.ibm.com           0           3
   k3n03.hpssl.kgn.ibm.com           2           0
   k3n05.hpssl.kgn.ibm.com           4           1
   k3n06.hpssl.kgn.ibm.com           5           0
   k3n07.hpssl.kgn.ibm.com           6           2
   k3n08.hpssl.kgn.ibm.com           7           3
   k3n09.hpssl.kgn.ibm.com           8           3
   k3n10.hpssl.kgn.ibm.com           9           2
   k3n11.hpssl.kgn.ibm.com          10           0
   k3n12.hpssl.kgn.ibm.com          11           1
   k3n13.hpssl.kgn.ibm.com          12           1
   k3n14.hpssl.kgn.ibm.com          13           0
   k3n15.hpssl.kgn.ibm.com          14           2
   k3n16.hpssl.kgn.ibm.com          15           3
   ```

3. To save each node's node number and hardware Ethernet addresses (which is needed for netboot) in a file called **bootptab.info** without the SDR header class information, enter:

   ```
   SDRGetObjects -G -x Node node_number hdw_enet_addr > /etc/bootptab.info
   ```

   You should receive output similar to the following:

   ```
             1 02608C2D58D2
             3 10005AFA2375
             4 10005AFA22CE
             5 10005AFA22B2
             6 10005AFA2410
             7 10005AFA223F
             8 10005AFA2417
             9 02608C2DA0C7
            11 02608C2D9F62
            13 02608C2D9E75
            15 10005AFA1B03
            16 10005AFA2B9B
   ```

## SDRListClasses

### Purpose

**SDRListClasses** – Lists the class names in the System Data Repository (SDR).

### Syntax

**SDRListClasses**

### Flags

None.

### Operands

None.

### Description

This command outputs all of the class names (system and partitioned) currently defined in the SDR to standard output.

### Location

**/usr/lpp/ssp/bin/SDRListClasses**

## SDRListFiles

### Purpose

**SDRListFiles** – Lists all of the files in the system file area first, then lists all of the files in the system partition area.

### Syntax

**SDRListFiles**

### Flags

None.

### Operands

None.

### Description

This command outputs all the system partition files first, then the system files.

### Location

**/usr/lpp/ssp/bin/SDRListFiles**

## SDRMoveObjects

### Purpose

**SDRMoveObjects** – Moves objects from one system partition to another.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRMoveObjects** *source_syspar target_syspar class_name* [*attr*==*value* ...]

### Flags

None.

### Operands

| | |
|---|---|
| *source_syspar* | Specifies the system partition from which objects are to be moved. |
| *target_syspar* | Specifies the system partition to which objects should be moved. |
| *class_name* | Identifies the target object. |
| *attr*==*value* | Specifies attribute values to be moved. |

### Description

This command moves any objects in *class_name* that match all of the *attr*==*value* pairs from the *source_syspar* to the *target_syspar*.

### Location

**/usr/lpp/ssp/bin/SDRMoveObjects**

## SDRRemoveSyspar

### Purpose

**SDRRemoveSyspar** – Removes all of the partitioned classes in the System Data Repository (SDR) associated with the system partition whose address is *IP_address*. It removes the daemon that serves this system partition using the System Resource Controller (SRC).

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRRemoveSyspar** *IP_address*

### Flags

None.

### Operands

| | |
|---|---|
| *IP_address* | Specifies the dotted decimal address (real or alias) of a system partition. |

### Description

This command deletes a system partition in the SDR. It does not perform all of the system management tasks involved in deleting a system partition.

### Location

**/usr/lpp/ssp/bin/SDRRemoveSyspar**

## SDRReplaceFile

### Purpose

**SDRReplaceFile** – Replaces the specified System Data Repository (SDR) file with the specified AIX file.

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

### Syntax

**SDRReplaceFile** *AIX_filename SDR_filename*

### Flags

None.

### Operands

*AIX_filename*
Identifies the AIX file name to be written to the SDR.

*SDR_filename*
Specifies the name of the SDR file to be overwritten.

### Description

This command searches first for a partitioned file, then for a system file, and replaces the first one found.

### Location

**/usr/lpp/ssp/bin/SDRReplaceFile**

## SDRRestore

### Purpose

**SDRRestore** – Extracts the contents of the archived System Data Repository (SDR).

**Attention**

The System Data Repository (SDR) commands are to be used by the IBM Parallel System Support Programs for AIX (PSSP) system management software. Use of these commands by a user can cause damage to system configuration data. Exceptions are: **SDRArchive**, **SDRGetObjects**, **SDRListClasses**, **SDRListFiles**, **SDRRetrieveFile**, **SDR_test**, and **SDRWhoHasLock**.

**Attention**

**Migration Note:** Each new PSSP release may introduce new SDR classes and attributes. Use caution when using **SDRArchive** and **SDRRestore** to avoid overwriting new SDR classes and attributes. IBM suggests that after migration you do not execute **SDRRestore** from a back level system since it will overwrite any new SDR classes and attributes.

### Syntax

**SDRRestore** *archive_file*

### Flags

None.

### Operands

*archive_file* Indicates the name of the archive file. The file name is **backup.JULIANdate.***HHMM*, where **JULIANdate.***HHMM* is a number or string uniquely identifying the time of the archive.

### Description

Use this command to remove the contents of the SDR and retrieve the archived contents of the *archive_file*. The *archive_file* must be in the **/spdata/sys1/sdr/archives** directory. Any new SDR daemons that represent partitions in the restored SDR are then started and any daemons that are not in the new SDR are stopped.

### Location

**/usr/lpp/ssp/bin/SDRRestore**

## Related Information

Command: **SDRArchive**

## SDRRetrieveFile

### Purpose

**SDRRetrieveFile** – Retrieves the specified System Data Repository (SDR) file into an AIX file.

### Syntax

**SDRRetrieveFile** *SDR_filename AIX_filename*

### Flags

None.

### Operands

*SDR_filename*
Specifies the name of the SDR file to be retrieved.

*AIX_filename*
Identifies the name of the AIX file to be written.

### Description

This command searches first for a partitioned file, then for a system file if a partitioned file was not found.

### Location

**/usr/lpp/ssp/bin/SDRRetrieveFile**

## SDRWhoHasLock

### Purpose

**SDRWhoHasLock** – Returns transaction ID of lock on specified class.

### Syntax

**SDRWhoHasLock** *class_name*

### Flags

None.

### Operands

| | |
|---|---|
| *class_name* | Identifies the target object class. |

### Description

The lock transaction ID returned from this command takes the form *host_name:pid:session*, where *host_name* is the long name of the machine running the process with the lock, *pid* is the process ID of the process that has the lock, and *session* is the number of the client's session with the System Data Repository (SDR).

### Location

**/usr/lpp/ssp/bin/SDRWhoHasLock**

## seqfile

### Purpose

**seqfile** – Creates node sequence files for system startup and shutdown using information in the System Data Repository (SDR).

### Syntax

**seqfile** [–**b**]

### Flags

–**b** Includes lines for boot/install servers as well as **/usr** servers. Specify this option to create lines for **/etc/cstartSeq**.

### Operands

None.

### Description

**seqfile** uses information in the SDR to determine dependencies of SP nodes on **/usr** and, optionally, and boot/install servers, and to write the dependencies to standard output in the format of the node sequence files **/etc/cstartSeq** and **/etc/cshutSeq**.

**/usr** servers must shut down after and start before their clients. Boot-install servers must start before their clients. The node sequence files, **/etc/cshutSeq** and **/etc/cstartSeq**, have lines that describe these dependencies. The **seqfile** command eliminates the need for you to create these files from scratch. If the nodes in your system have sequencing dependencies in addition to those related to boot/install and **/usr** servers and clients, you can edit the output of **seqfile** to define those relationships.

**seqfile** defines only the nodes that have dependencies; if there are no **/usr** or boot/install dependencies, **seqfile** generates no output.

If you do not have a **/etc/cstartSeq** or **/etc/cshutSeq** file, the **cstartup** and **cshutdown** commands use **seqfile** to determine the default startup or shutdown sequence.

### Files

The following files reside on the control workstation:

**/usr/lpp/ssp/bin/seqfile**
The **seqfile** command.

**/etc/cshutSeq** Describes the sequence in which the nodes should be shut down. Nodes not listed in the file are shut down concurrently with listed nodes. If the file is empty, all nodes are shut down concurrently. If the file does not exist, **cshutdown** uses the output of **seqfile** as a temporary sequencing default.

**/etc/cstartSeq** Describes the sequence in which the nodes should be started. Nodes not listed in the file are started up concurrently with listed nodes. If the file is empty, all nodes are started up concurrently. If the file does not exist, **cstartup** uses the output of **seqfile** as a temporary sequencing default.

## Location

**/usr/lpp/ssp/bin/seqfile**

## Related Information

Commands: **cshutdown**, **cstartup**

## Examples

1. To create the node sequence file for system startup from information in the SDR, enter:

   ```
   seqfile -b > /etc/cstartSeq
   ```

2. To create the node sequence file for system shutdown from information in the SDR, enter:

   ```
   seqfile > /etc/cshutSeq
   ```

3. To view the sequence used during system shutdown in the absence of a **/etc/cshutSeq** file, enter:

   ```
   seqfile | more
   ```

## services_config

### Purpose

**services_config** – Configures designated services on nodes or the control workstation.

### Syntax

**services_config**

### Flags

None.

### Operands

None.

### Description

Use this command to configure SP services on the node or control workstation.

### Standard Error

This command writes error messages (as necessary) to standard error.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **nonzero** | Indicates that an error occurred. |

### Security

You must have root privilege to run this command.

### Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

### Location

**/usr/lpp/ssp/install/bin/services_config**

### Related Information

Commands: **setup_server**, **spbootins**

### Examples

To configure the SP services on a node or the control workstation, enter:

```
services_config
```

## sethacws

### Purpose

**sethacws** – Sets the state of the control workstation.

### Syntax

**sethacws** *state*

### Flags

None.

### Operands

| | |
|---|---|
| *state* | Specifies a number of the set: 0, 1, 2, 16, 32. |

### Description

Use this command to set the current state of the control workstation. It is valid only when issued on the control workstation. When the command is executed and the calling process is not on a control workstation, an error occurs.

**Note:** The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set the control workstation state. The state is changed during fail over or reintegration in the HACWS supplied pre- and post-event scripts for HACMP. The administrator should not normally have to set the control workstation state.

### Exit Values

| | |
|---|---|
| **0** | Indicates successful completion of the command. |
| **1** | Indicates that the command could not access the control workstation state location. |
| **2** | Indicates that the command was executed with a control workstation state that was not valid. |
| **3** | Indicates that the command was not executed on a control workstation. |

The following are the valid state values and their defined control workstation state:

| | |
|---|---|
| **0** | Indicates that the configuration is not an HACWS configuration, but is a control workstation. |
| **1** | Indicates that this is the primary control workstation, but not the active control workstation. |
| **2** | Indicates that this is the primary and active control workstation. |
| **16** | Indicates that this is the backup control workstation and not the active control workstation. |
| **32** | Indicates that this is the backup and active control workstation. |

## Security

You must have root privilege to run this command.

## Prerequisite Information

Refer to *PSSP: Administration Guide* for information on the HACWS option.

## Location

**/usr/bin/sethacws**

## Related Information

Command: **lshacws**

Subroutines: **hacws_set**, **hacws_stat**

## Examples

1. To set the control workstation state as a backup and active control workstation, enter:
```
sethacws 32
```
2. To set the control workstation state as a backup and inactive control workstation, enter:
```
sethacws 16
```
3. To set the control workstation state as a primary and active control workstation, enter:
```
sethacws 2
```
4. To set the control workstation state as a primary and inactive control workstation, enter:
```
sethacws 1
```
5. To set the control workstation state as a control workstation but not an HACWS configuration, enter:
```
sethacws 0
```

## setup_authent

### Purpose

**setup_authent** – Sets up a workstation to use SP authentication services.

### Syntax

**setup_authent**

### Flags

None.

### Operands

None.

### Description

The **setup_authent** command configures SP authentication services during SP installation on the control workstation and on other IBM RS/6000 workstations connected to an SP system. It is not executed on SP nodes, where authenticated client services are automatically installed. Executing this command invokes an interactive dialog, in which instructions for the various steps are displayed and various utility programs are invoked to accomplish the configuration.

There are several ways that **setup_authent** can configure these services. The method chosen is based on runtime choice, the combination of SP options installed on the workstation, and the contents of any predefined authentication configuration file that you have supplied.

**Primary Server:** When the local system is to be configured as the primary server, both **ssp.clients** and **ssp.authent** SP options must have been installed. You may supply the configuration files, **/etc/krb.conf** and **/etc/krb.realms**, or let the system create one that lists the local system as the sole authentication server in the local realm. This command creates the files used by the authentication and ticket granting services. These include the configuration files, the authentication database files, and the master key cache file. The server daemon, **kerberos**, is added to the inittab and started.

The administration of the authentication database is handled by the **kadmind** daemon, which is also added to inittab and started. The **setup_authent** command requires you to define the initial principal who administers the database. Access control list files are created containing this name, to be used by **kadmind** for authorization.

This command invokes **k4init** to log you in as this administrator, to define additional principals used by the SP authenticated services for monitoring and system administration. A server key file is created for use by the monitor commands, SP remote commands, and Sysctl remote command execution facility.

**Backup Server:** When the local workstation is to be configured as a secondary server, **ssp.clients** and **ssp.authent** must be installed. You must supply the configuration files, listing the local host as a slave server and some other workstation as the primary authentication server. The primary server must be configured and running and be available by standard TCP/IP connection to the local host.

You are required to authenticate your identity as the administrative user that you defined when you configured the primary server. The service principals for the local host are added to the primary database, and the server key file is created for them. Then the **kpropd** daemon is used in conjunction with the **kprop** command (executed remotely on the primary server) to copy the master database onto the local system. The server daemon, **kerberos**, is then added to the inittab and started.

**Authentication Server:** When the local host is to be configured only to provide authentication client services, just **ssp.clients** needs to be installed. As in the case of the slave server, you must supply the configuration files. In this case, however, the local host is not listed as a server. **setup_authent** simply requires the information to know how to get to the primary authentication server (already configured and accessible).

You are required to authenticate your identity as the administrative user that you defined when you configured the primary server. The service principals for the local host are added to the primary database, and the server key file is created for them.

**Using AFS Authentication Services:** When AFS authentication is to be configured, the local host must have already been established as either an AFS server or an AFS client. The **CellServDB** and **ThisCell** files are expected to exist in the **/usr/vice/etc** directory (or linked to that path). **ssp.clients** is the only required SP authentication option. When **setup_authent** finds these AFS configuration files on the local system, it allows you the choice of whether to use AFS authentication. If you choose not to use AFS, processing follows one of the other three variations described previously. When using AFS, you must supply an AFS user name and password that is a valid administrative ID in the local AFS cell. **setup_authent** creates the local service principals in the AFS database and creates a server key file for the SP authenticated services to use on the local host.

If you choose AFS authentication, you must do so for all workstations you configure with **setup_authent**, including the control workstation for your SP system.

You can reexecute **setup_authent** to change the configuration of your authentication services, but you add varying degrees of risk to system operations depending on how far you have progressed in the installation of the control workstation and nodes. Running it again on the control workstation prior to executing **install_cw** is not a problem. Reconfiguring a client workstation has little risk of disruption. A slave can be reconfigured provided the primary server is available. If the primary server must be reconfigured, all slave and client systems have to be reconfigured after the new primary server is up. If the control workstation is an authentication server, you have to recustomize any SP nodes previously booted, after running **setup_authent**.

## Files

| | |
|---|---|
| **/.k** | Master key cache file. |
| **/etc/krb.conf** | Authentication configuration file. |
| **/etc/krb.realms** | Authentication configuration file. |
| **/etc/krb-srvtab** | Server key file. |
| **/usr/kerberos/admin_acl.{add,get,mod}** | Access Control List files. |
| **/var/kerberos/database/principal.pag, /var/kerberos/database/principal.dir** | Authentication database files. |
| **/usr/vice/etc/CellServDB, /usr/vice/etc/ThisCell** | AFS configuration files. |

## Location

**/usr/lpp/ssp/bin/setup_authent**

## Related Information

Commands: **add_principal**, **ext_srvtab**, **kadmind**, **kdb_edit**, **kdb_init**, **kdb_util**, **kerberos**, **k4init**, **k4list**, **krb_conf**, **krb_realms**, **ksrvutil**, **kstash**

Refer to the "RS/6000 SP Files and Other Technical Information" section of *PSSP: Command and Technical Reference* for additional **Kerberos** information.

## Examples

See *PSSP: Installation and Migration Guide*.

## setup_CWS

### Purpose

**setup_CWS** – Updates control workstation files and directories for installation tasks.

### Syntax

**setup_CWS** [−**h**]

### Flags

| | |
|---|---|
| −**h** | Displays usage information. If the command is issued with the −**h** flag, the syntax description is displayed to standard output and no other action is taken. |

### Operands

None.

### Description

Use this command to update control workstation files and directories for installation tasks. This includes control workstation-specific Kerberos and other files. This command can only be run on the control workstation.

| You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

| If you do not have a ticket-granting-ticket, you must run **k4init**.

### Standard Error

This command writes error messages (as necessary) to standard error.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **nonzero** | Indicates that an error occurred. |

### Security

You must have root privilege to run this command.

### Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/setup_CWS**

## Related Information

Commands: **setup_server**

## Examples

To update the control workstation environment for installation, enter:

```
setup_CWS
```

## setup_logd

### Purpose

**setup_logd** – Sets up the logging daemon (**splogd**). This is called by installation scripts when the IBM RS/6000 control workstation is installed. It can also be run by root on a different workstation to have **splogd** spawned by the System Resource Controller (SRC).

### Syntax

**setup_logd**

### Flags

None.

### Operands

None.

### Description

To run the **splogd** logging daemon on a workstation other than the control workstation, install the **ssp.clients** option on that workstation and run **setup_logd**. You may want to do this so that:

1. Offload error logging from the control workstation
2. Have your own script called when a state change on a particular variable or variables occurs

By default the **/spdata/sys1/spmon/hwevents** file is set up to do error logging and state change logging for all frames. If you are installing **splogd** on a workstation besides the control workstation to call your own script, you should edit the **/spdata/sys1/spmon/hwevents** file, removing the entries for SP_STATE_LOG and SP_ERROR_LOG and add a call for your own script. Refer to the **splogd** command for instructions.

The **setup_logd** command performs the following steps:

1. Creates directories in **/var/adm** that the logging daemon uses, if they do not already exist.
2. Adds an entry to **syslog.conf** for **daemon.notice** and sends a HUP signal to **syslogd** to reread its configuration file.
3. Adds errlog templates for SP messages.
4. Adds the **splogd** daemon to SRC as the **splogd** subsystem.
5. Adds an entry for **splogd** to **/etc/inittab**.

If you do not want to perform any of the preceding steps on your workstation, do not run **setup_logd**. If you are only using **splogd** to call your own script, you might only want to do step 4 and step 5 (add **splogd** to SRC and **/etc/inittab**).

To run the logging daemon on a separate workstation, you must add the following to the **/etc/environment** file:

```
SP_NAME={control_workstation}
```

To move a subset of error logging off of the control workstation, edit **/spdata/sys1/spmon/hwevents** on the control workstation to define the subset that you want to monitor. Then **stopsrc** and **startsrc** the logging daemon on the control workstation to reread the **hwevents** file.

**Starting and Stopping the splogd Daemon**

The **splogd** daemon is under System Resource Controller (SRC) control. It uses the signal method of communication in SRC. The **splogd** daemon is a single subsystem and not associated with any SRC group. The subsystem name is **splogd**. To start the **splogd** daemon, use the **startsrc −s splogd** command. This starts the daemon with the default arguments and SRC options. The **splogd** daemon is setup to be respawnable and be the only instance of the **splogd** daemon running on a particular node or control workstation. Do **not** start the **splogd** daemon from the command line without using the **startsrc** command to start it.

To stop the **splogd** daemon, use the **stopsrc −s splogd** command. This stops the daemon and does not allow it to respawn.

To display the status of the **splogd** daemon, use the **lssrc −s splogd** command.

If the default startup arguments need to be changed, use the **chssys** command to change the startup arguments or the SRC options. Refer to *AIX Version 4 Commands Reference* and *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for more information about daemons under SRC control and how to modify daemon arguments when under SRC.

To view the current SRC options and daemon arguments, use the **odmget −q "subsysname=splogd" SRCsubsys** command.

## Files

| | |
|---|---|
| **/etc/inittab** | AIX file that contains a list of parameters to be brought up during initialization. |
| **/spdata/sys1/spmon/hwevents** | File that describes what logging is performed and what user exits are called. |
| **/etc/syslog.conf** | Describes where syslog messages are logged. |

## Location

**/usr/lpp/ssp/bin/setup_logd**

## Related Information

Daemon: **splogd**

Refer to *PSSP: Installation and Migration Guide* for more information on setting up Hardware Monitor clients on separate workstations and the System Resource Controller.

## Examples

1. To start the **splogd** daemon, enter:

   ```
   startsrc -s splogd
   ```

2. To stop the **splogd** daemon, enter:

   ```
   stopsrc -s splogd
   ```

3. To display the status of the **splogd** daemon, enter:

   ```
   lssrc -s splogd
   ```

4. To display the status of all the daemons under SRC control, enter:

   ```
   lssrc -a
   ```

5. To display the current SRC options and daemon arguments for the **splogd** daemon, enter:

   ```
   odmget -q "subsysname=splogd" SRCsubsys
   ```

## setup_server

### Purpose

**setup_server** – Configures a node or control workstation as a boot/install server.

### Syntax

**setup_server** [–**h**]

### Flags

| | |
|---|---|
| –**h** | Displays usage information. If the command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken. |

### Operands

None.

### Description

Use this command to set up the node on which it is run as a boot/install server for client nodes as defined in the System Data Repository (SDR).

On a boot/install server, this command:

- Defines the boot/install server as a Network Installation Management (NIM) master
- Defines the resources needed for the NIM clients
- Defines each node that this server installs as a NIM client
- Allocates the NIM resources necessary for each NIM client
- Creates the **node.install_info** file containing netinstall information
- Creates the **node.config_info** file containing node-specific configuration information be used during network boot
- Creates server key files containing the service keys for the nodes
- Copies the install images from the control workstation for nodes which are boot/install servers

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **k4init**.

Creation of the Network Installation Management (NIM) lppsource resource on a boot/install server will result in **setup_server** creating a lock in the lppsource directory on the control workstation. The **setup_server** command calls **mknimres** which creates the lock.

## Standard Error

This command writes error messages (as necessary) to standard error.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **–1** | Indicates that an error occurred. |

## Security

You must have root privilege to run this command.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/setup_server**

## Related Information

Commands: **allnimres**, **create_krb_files**, **delnimclient**, **delnimmast**, **export_clients**, **mkconfig**, **mkinstall**, **mknimclient**, **mknimint**, **mknimmast**, **mknimres**, **setup_CWS**, **unallnimres**

## Examples

To prepare a boot/install server node, enter the following on that node:

```
setup_server
```

## sp_configd

### Purpose

**sp_configd** – Starts the Simple Network Management Protocol (SNMP) SP Proxy daemon as a background process.

### Syntax

**sp_configd** [–**T**] [–**t** *secs*] [–**s** *secs*] [–**e** *secs*]

### Flags

| | |
|---|---|
| –**T** | Specifies whether to perform internal tracing. Trace entries are placed in the **/var/tmp/sp_config.log** file. The default is off. |
| –**t** *secs* | Specifies the amount of time, data instance values that are nonconstant should be kept in cache before being considered stale. This is used to improve the performance associated with a dump of the ibmSPEMVarValuesTable (which is a series of SNMP getnext-requests). When a specific instance value from this table is requested by an SNMP get-request, the latest value is obtained from the SP Event Manager (EM) regardless of the amount of elapsed time since the last request for this data. If the –**t** flag is not specified, a default value of 720 seconds is used. |
| –**s** *secs* | Specifies the amount of elapsed time between sending requests for the SP EM to determine the set of EM variables for which a resource monitor is currently active. Current EM resource instance values can only be obtained for those EM resource which are currently being monitored. If the –**s** flag is not specified, a default value of 1200 seconds is used. |
| –**e** *secs* | Specifies the amount of elapsed time between retrying unsuccessful EM connection attempts. EM connection initialization causes requests to be sent to the System Data Repository (SDR) which may hinder performance if attempted too frequently. If the –**e** flag is not specified, a default value of 60 seconds is used. |

### Operands

None.

### Description

The **sp_configd** daemon is internally configured as an SNMP Multiplexing Protocol (SMUX) peer, or proxy agent, of the **snmpd** daemon on the control workstation and on each node of the SP. For more information, refer to the “Managing SP System Events in a Network Environment” chapter in *PSSP: Administration Guide*.

The **sp_configd** daemon provides the following functions:

- It receives requests from network monitors for data from the ibmSP MIB (these requests are routed from the **snmpd** daemon to the **sp_configd** daemon over the SMUX interface). The results are returned by the **sp_configd** daemon to the **snmpd** daemon by the SMUX interface and are then sent to the originating monitor by the **snmpd** agent.

- It sends trap notifications about events occurring on the SP to all hosts listed in the **snmpd** daemon configuration file.

The **snmpd** daemon should be active before the **sp_configd** daemon is started. The following command activates the **snmpd** daemon:

```
startsrc -s snmpd
```

The **snmpd** daemon is controlled by the System Resource Controller (SRC) and activated whenever the system is initialized.

The **sp_configd** daemon has several sessions with the EM. These sessions are used to maintain SP EM variable instance data and information from the last trap issued associated with an SP EM event. See the **haem** command for information on starting the SP Event Manager.

The **sp_configd** daemon should be controlled using the SRC. IBM suggests that you do not enter **sp_configd** at the command line.

**Manipulating the sp_config Daemon with the System Resource Controller**

The **sp_configd** daemon is a subsystem controlled by the SRC. Use the following SRC commands to manipulate the **sp_configd** daemon:

**lssrc** Gets the status of a subsystem, group of subsystems, or a subserver. The long status form of the **lssrc** command is not supported.

**startsrc** Starts a subsystem, group of subsystems, or a subserver. Issuing the **startsrc** command causes the **sp_configd** daemon to generate a coldStart trap. Use the **–a** switch to override default switch values.

**stopsrc** Stops a subsystem, group of subsystems, or a subserver.

## Files

**/etc/services** Contains port assignments for required services. The following entry must be present in the **/etc/services** file if the entries are not already present:

```
smux 199/tcp
```

**Notes:**

1. The SMUX port must be 199.
2. The **/etc/services** file is shipped with this entry already in place.
3. If the **/etc/services** file is being served from a server, this entry must be present in the server's **/etc/services** file.

**/etc/snmpd.conf**
Specifies the SMUX association configuration for the **sp_configd** Proxy Agent. The following entry must be present:

```
smux    1.3.6.1.4.1.2.6.117    sp_configd_pw    # sp_configd
```

These entries are created when the SP is installed.

**/etc/snmpd.peers**
Specifies the configuration for the **sp_configd** SMUX peer. The following entry must be present:

```
sp_configd    1.3.6.1.4.1.2.6.117    sp_configd_pw
```

These entries are created when the SP is installed.

## Security

You must have root privilege to run this command or be a member of the system group.

## Location

**/usr/lpp/ssp/bin/sp_configd**

## Related Information

See the “Understanding the SNMP Daemon” and “Problem Determination for the SNMP Daemon” chapters in *AIX Version 3.2 System Management Guide: Communications and Networks*.

See the “Understanding the Simple Network Management Protocol (SNMP)”, “Understanding the Management Information Base (MIB)”, and “xgmon Overview for Programmers” chapters in *AIX Version 3.2 Communication Programming Concepts*.

See the “Managing SP System Events in a Network Environment” chapter in *PSSP: Administration Guide*.

## Examples

1. To start the **sp_configd** daemon, enter a command similar to the following:

   ```
   startsrc -s sp_configd -a '-T'
   ```

   This command starts the **sp_configd** daemon and logs information to the **/var/tmp/sp_configd.log** file.

2. To stop the **sp_configd** daemon normally, enter:

   ```
   stopsrc -s sp_configd
   ```

   This command stops the daemon. The **–s** flag specifies the subsystem that follows to be stopped.

3. To get short status from the **sp_configd daemon**, enter:

   ```
   lssrc -s sp_configd
   ```

   This command returns the daemon name, process ID, and state (active or inactive).

## sp_configdctrl Script

### Purpose

**sp_configdctrl** – A control script that is used to manage the installation of the SP Simple Network Management Protocol (SNMP) Proxy Agent subsystem.

### Syntax

**sp_configdctrl** { **–a** | **–s** | **–k** | **–d** | **–c** | **–t** | **–o** | **–r** | **–h** }

### Flags

| | |
|---|---|
| –**a** | Adds the subsystem. |
| –**s** | Starts the subsystem. |
| –**k** | Stops the subsystem. |
| –**d** | Deletes the subsystem. |
| –**c** | Cleans the subsystem, that is, deletes it. |
| –**t** | Turns tracing on for the subsystem. |
| –**o** | Turns tracing off for the subsystem. |
| –**r** | Refreshes the subsystem. |
| –**h** | Displays usage information. |

### Operands

None.

### Description

Use this command to install or remove the SP SNMP Proxy Agent daemon. This command can be issued only by a user with root privileges or by a member of the system group.

The **sp_configdctrl** control script controls the operation of the SP SNMP Proxy Agent subsystem. The subsystem is under the control of the System Resource Controller (SRC). The subsystem is called **sp_configd**.

An instance of the SP SNMP Proxy Agent subsystem executes on the control workstation and on every node of a system partition. Because the information about SP nodes and Event Manager (EM) variables exists in system partitions, it is said to be system partition-sensitive. This control script operates in a manner similar to the control scripts of other system partition-sensitive subsystems. It can be issued from either the control workstation or any of the system partition's nodes.

From an operational point of view, the SP SNMP Proxy Agent subsystem group is organized as follows:

| | |
|---|---|
| **Subsystem** | SP SNMP Proxy Agent |
| **Subsystem Group** | None |

**SRC Subsystem Name**

**sp_configd**

The **sp_configd** subsystem is associated with the **sp_configd** daemon.

The subsystem name on the nodes and the control workstation is **sp_configd**. There is one daemon per node and control workstation.

**Daemons** **sp_configd**

The **sp_configd** daemon provides the SP SNMP Proxy Agent function.

The **sp_configdctrl** script is not normally executed from the command line. It is normally called by the **syspar_ctrl** command during installation of the system, and partitioning or repartitioning of the system.

The **sp_configdctrl** script provides a variety of controls for operating the SP SNMP Proxy Agent subsystem:

- Adding, starting, stopping, and deleting the subsystem
- Cleaning up the subsystem, that is, deleting it from all system partitions
- Turning tracing on and off

**Adding the Subsystem**

When the **–a** flag is specified, the control script uses the **mkssys** command to add the SP SNMP Proxy Agent subsystem to the SRC. The control script operates as follows:

1. It makes sure that the **sp_configd** daemon is stopped.
2. It removes the **sp_configd** subsystem from the SRC (just in case it is still there).
3. It adds the **sp_configd** subsystem to the SRC.
4. It adds an entry for the **sp_configd** subsystem to the **/etc/inittab** file. The entry ensures that the subsystem is started during boot.
5. It adds a smux entry to the **/etc/snmpd.conf** file and a password entry to the **/etc/snmpd.peers** file for the **sp_configd** Proxy Agent if they do not currently exist.
6. It appends the ibmSP MIB definitions to the **/etc/mib.defs** file if they do not currently exist.
7. It issues a **refresh -s snmpd** command so that **snmpd** processes the new entries placed in the **/etc/snmpd.conf** and **/etc/snmpd.peers** files.
8. It adds an errnotify stanza for the **snmp_trap_gen** function to the Object Data Manager (ODM). This function notifies the SP SNMP Proxy Agent when an entry is written to the AIX errlog which has a template specifying **Alert = true**.

**Starting the Subsystem**

When the **–s** flag is specified, the control script uses the **startsrc** command to start the SP SNMP Proxy Agent subsystem, **sp_configd**.

### Stopping the Subsystem

When the **−k** flag is specified, the control script uses the **stopsrc** command to stop the SP SNMP Proxy Agent subsystem, **sp_configd**.

### Deleting the Subsystem

When the **−d** flag is specified, the control script uses the **rmssys** command to remove the SP SNMP Proxy Agent subsystem from the SRC. The control script operates as follows:

1. It makes sure that the **sp_configd** daemon is stopped.
2. It removes the **sp_configd** subsystem from the SRC using the **rmssys** command.
3. It removes the entry for the **sp_configd** subsystem from the **/etc/inittab** file.
4. It removes entries from **/etcsnmpd.conf** and **/etc/snmpd.peers** and removes ibmSP MIB definitions from **/etc/mib.defs**.

### Cleaning Up the Subsystem

When the **−c** flag is specified, the control script stops and removes the SP SNMP Proxy Agent subsystem from the SRC. The control script operates as follows:

1. It stops the subsystem using the **stopsrc -s sp_configd** command.
2. It removes the subsystem from the SRC using the **rmssys** command.
3. It removes entries from **/etcsnmpd.conf** and **/etc/snmpd.peers** and removes ibmSP MIB definitions from **/etc/mib.defs**.

### Turning Tracing On

When the **−t** flag is specified, the control script turns tracing on for the **sp_configd** daemon, by stopping the daemon and restarting it with the **−T** option.

### Turning Tracing Off

When the **−o** flag is specified, the control script turns tracing off for the **sp_configd** daemon, by stopping the daemon and restarting it without the **−T** option.

### Refreshing the Subsystem

The **−r** flag has no effect for this subsystem.

## Files

| | |
|---|---|
| **/etc/snmpd.peers** | Contains password entries. |
| **/etc/snmpd.conf** | Contains smux entries. |
| **/etc/mib.defs** | Contains the ibmSP MIB definitions. |

## Standard Error

This command writes error messages (as necessary) to standard error.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that an error occurred. |

## Security

You must be running with an effective user ID of root.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Prerequisite Information

*AIX Version 4 Commands Reference*

Information about the System Resource Controller (SRC) in *AIX Version 4 General Programming Concepts: Writing and Debugging Programs*

## Location

**/usr/lpp/ssp/bin/sp_configdctrl**

## Related Information

Commands: **sp_configd**

## Examples

1. To add the SP SNMP Proxy Agent subsystem to the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -a
   ```

2. To start the SP SNMP Proxy Agent subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -s
   ```

3. To stop the SP SNMP Proxy Agent subsystem in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -k
   ```

4. To delete the SP SNMP Proxy Agent subsystem from the SRC in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -d
   ```

5. To clean up the SP SNMP Proxy Agent subsystem on all system partitions, enter:

   ```
   sp_configdctrl -c
   ```

6. To turn tracing on for the **sp_configd** daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -t
   ```

7. To turn tracing off for the **sp_configd** daemon in the current system partition, set the **SP_NAME** environment variable to the appropriate system partition name, enter:

   ```
   sp_configdctrl -o
   ```

8. To display the status of the SP SNMP Proxy Agent subsystem on a node or the control workstation, enter:

   ```
   lssrc -s sp_configd
   ```

## spacctnd

### Purpose

**spacctnd** – Enters accounting data into the System Data Repository for a node or group of nodes.

### Syntax

**spacctnd** {[–**c** *acct_class_id*] | [–**e** {**true** | **false** | **default**}]
[–**j** *acct_job_charge*] [–**x** {**true** | **false**}]}
{*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*}

### Flags

–**c** *acct_class_id*
: Indicates that the accounting class identifier attribute of each specified node should be changed to the value of *acct_class_id*. The accounting class identifier is an arbitrary string. All nodes with the same string value constitute a class for purposes of grouping and merging accounting data.

–**e**
: Indicates that the accounting enabled attribute of each specified node should be changed. The accounting enabled attribute is an indicator of whether accounting is enabled for the node. The possible values are:

| | |
|---|---|
| **true** | Accounting is enabled |
| **false** | Accounting is disabled |
| **default** | Accounting is enabled based on the value of the SP accounting enabled attribute |

–**j** *acct_job_charge*
: Indicates that the accounting job charge value of each specified node should be changed to the value of *acct_job_charge*. The job charge value is used to determine the number of *charge fee units* to charge a user for exclusive use of the node. Its value is in units of seconds per charge fee unit. This value must be expressed as a float value with one or more digits followed by a decimal point which is followed by one or more digits.

–**x**
: Indicates whether accounting start and end job records and thus chargefee records are generated for jobs having exclusive use of the node. A value of **true** specifies that exclusive use accounting is enabled and start and end job records are generated. A value of **false** specifies that exclusive use accounting is not enabled and start and end job records are not generated.

–**N** *node_group*
: Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

–**l** *node_list*
: Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white

space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

## Operands

*start_frame*
: Indicates which frame is the starting frame for the range of nodes in this operation. If you use the *start_frame*, *start_slot*, and *node_count* fields, do not use the *node_list* field. Select a value from 1 through 64.

*start_slot* Indicates which slot is the starting slot for the range of nodes in this operation. The slot is the position in the rack that a node occupies. For example, for a thin node which is the second node in a rack that has a wide node in the first slot, the slot number is 3. If you use *start_frame*, *start_slot*, and *node_count*, do not use the *node_list* field. Specify the start slot as a number from 1 through 16.

**Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition.

*node_count*
: Indicates which nodes are to be used for the range of nodes in this operation. If the combination of *start_slot* and *node_count* goes past the nodes in a frame, the next sequential frame is used for the operation. If you use *start_frame*, *start_slot*, and *node_count*, do not use the *node_list* field. Specify a value from 1 through 1024.

**Note:** The *node_count* is considered to be within the current system partition.

## Description

Run this command during installation of the SP or later to set the accounting class identifier, the accounting enabled attribute, job charge value or the exclusive use accounting enabled attribute of a node or set of nodes.

You can use the System Management Interface Tool (SMIT) to run the **spacctnd** command. To use SMIT, enter:

```
smit node_data
```

and select the Accounting Information option.

**Note:** This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

## Location

**/usr/lpp/ssp/bin/spacctnd**

## Examples

The following example adds accounting SDR information for a system with 2 frames and 32 nodes. Accounting and exclusive use accounting is to be enabled for each node and 60 seconds of exclusive use by a user is to constitute one *charge fee unit*.

```
spacctnd -e true -j 60.0 -x true 1 1 32
```

## spacs_cntrl

### Purpose

**spacs_cntrl** – Controls interactive access to SP nodes.

### Syntax

**spacs_cntrl** [–**v** –**s**] [–**d**] [–**h**] [–**l**] [–**f** *file_name*] [–**n** *netgroup_name*]
{**allow** | **block** | **deny** | **unblock**} *user_name* ...

### Flags

**–d** Specifies debugging mode. Displays additional messages, when available, to trace program flow and system errors.

**–f** *file_name*
Specifies the name of a file in which the user names to be allowed or denied access are listed in a column.

**–h** Displays the usage message when present on the command line.

**–l** Specifies log messages. This flag logs messages to the **/var/adm/SPlogs/spacs/spacs.log** file. Included are all messages regarding user states that are be displayed if the –**v** flag is specified, as well as any debug messages if –**d** is specified. (This is lowercase **l**, as in **l**ist.)

**–s** Specifies suppress mode. No error messages displayed. If –**l** is specified, error messages are sent to log file.

**–v** Specifies verbose mode. A message is displayed for each user, containing the date, time and state of the user. User states resulting from this command include:

Access was removed
Access was allowed
Access removed, user allowed
Access removed, user denied
Access allowed, user allowed
Access allowed, user denied
User name is not valid or is root.

**–n** *netgroup_name*
Accepts one NIS netgroup name of a netgroup that contains user names in the user field. Netgroups embedded in a given netgroup name is resolved.

**allow** Used by job submission systems such as the Resource Manager. Requests that interactive access be granted to run a parallel job. Result depends on user state.

**block** Used by root user to set user state to a known denied state and remove user state information used by job submission systems.

**deny** Used by job submission systems such as the Resource Manager. Requests that interactive access be denied after running parallel job. Result depends on user state.

| | |
|---|---|
| **unblock** | Used by root user to set user state to a known allowed state and remove user state information used by job submission systems. |

## Operands

*user_name*
Specifies the user name for which access is to be allowed or denied. Delineate with a blank space if specifying more than one user name.

## Description

Use caution when issuing this command while the Resource Manager is running. If the Resource Manager is configured to use Login Control, you may cause loss of user state information. For more Login Control information, refer to *PSSP: Administration Guide*.

The following types of access can be disallowed when **spacs_cntrl block** or **deny** is used:

**login**
**rlogin**
AIX **rsh**
AIX **rcp**
AIX **rexec**
SP **rsh**
SP **rcp**

The **spacs_cntrl** command does not allow individual types of access to be disallowed.

Duplicate user names are removed from the user list whether entered from the command line or in a file.

If you add a new user to a node for which all users are denied, you must reissue **spacs_cntrl** to deny the new user as well.

**Flags and Logging**

Flags specified in combination have the following results:

| | |
|---|---|
| **None** | Error messages go to standard output. |
| –**l** | Error messages got to standard output and are logged. |
| –**l** –**s** | Error messages go to log only. |
| –**s** | Error messages suppressed. |
| –**l** –**d** –**v** –**s** | All messages go to log only. |
| –**d** –**v** –**s** | Messages suppressed. |
| –**d** –**l** | Debug and error messages go to standard output and log. |

Use of the verbose flag (–**v**) causes the command to run longer due to extra processing required to find state information.

## Location

**/usr/lpp/ssp/bin/spacs_cntrl**

## Examples

1. To block a single user (Betty) on a single parallel node, on that node enter:

   ```
   spacs_cntrl block betty
   ```

2. To block users on multiple nodes, enter:

   a. Create the **block_usr_sample** file after adjusting threshold uid.

   b. Send file to all nodes in the current system partition. Note this example would require **rsh** privileges on the nodes.

   ```
   dsh -a rcp  root@mynode:/tmp/usr.input /tmp/usr.input
   ```

   c. Issue the **spacs_cntrl** command to block users to all the nodes in the current system partition.

   ```
   dsh -a spacs_cntrl -f /tmp/usr.input block
   ```

## spadaptrs

### Purpose

**spadaptrs** – Enters configuration data for an additional adapter for a node or series of nodes in the System Data Repository (SDR).

### Syntax

| **spadaptrs** [–**s** {**yes** | **no**}] [–**t** {**bnc** | **dix** | **fiber**| **NA** | **tp**}]
| [–**r** {**4** | **16** | **autosense**}] [–**d** {**full** | **half** | **auto**}]
| [–**f** {**10** | **100** | **1000** | **auto**}] [–**a** {**no** | **yes**}]
| [–**n** {**yes** | **no**}] [–**o** *IP_address*]
| {–**l** *node_list* **|** –**N** *node_group* **|** *start_frame start_slot node_count*}
| *adapter_name starting_IP_address netmask*

### Flags

| –**s yes | no** Indicates whether IP addresses should be skipped, as needed, when assigning IP addresses. If –**s no** is specified, no skipping occurs; each IP address assigned is equal to the previous address assigned plus one. If –**s yes** is specified, each IP address assigned is equal to the previous address assigned plus the difference in their respective node numbers.

| –**t bnc | dix | fiber | NA | tp**
| Designates the Ethernet type. Use **bnc** to designate a thin Ethernet. Use **dix** to designate a thick Ethernet (also called a twisted pair). Use **fiber** to designate a 1000 Base SX network. Use **NA** for an integrated Ethernet. Use **tp** to designate a twisted pair. The default is **bnc**.

–**r 4 | 16 | autosense**
Specifies the token-ring network speed. This is required for **tr0** and **tr1** token-ring adapters. Specify **4** for 4MB per second, **16** for 16MB per second, and **autosense** for adapters that automatically choose the network speed.

| –**d full | half | auto**
| Specifies the communication transfer as one way (**half**) or two way (**full**). The default is **auto**. If –**t fiber** is set then –**d** is set to **full**.

| –**f 10 | 100 | 1000 | auto**
| Specifies ethernet speed in megabits (Mb/s). Specify **10** for 10Mb per second, **100** for 100Mb per second or **1000** for 1000Mb per second. The default is **auto**.

–**a no | yes** Indicates whether you want Address Resolution Protocol (ARP) to be used for the switch. If you want to assign IP addresses freely, as for other adapters, you must specify **yes**. If you specify –**a no**, you must not specify –**n no**. Do not use this flag unless you are specifying IP addresses for the **css0** adapter. If you do not specify –**a**, the default is **yes**.

–**n yes | no** Indicates whether you want to use switch node numbers for assigning IP addresses for the switch. If you want to assign IP addresses freely, as for other adapters, you must specify –**n no** and

–**a yes**. If you specify –**n yes**, you must not specify –**s yes**. If you specify –**n yes**, you must not specify your nodes with a node list. Do not use this flag unless you are specifying IP addresses for the **css0** adapter. If you do not use –**n**, the default is **yes**.

–**o** *ip_list* Specifies a list of additional IP addresses associated with this adapter. The *ip_list* is a comma delimited list of dotted decimal IP addresses. This flag is used in an HACMP environment only.

–**l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

–**N** *node_group*
Specifies a node group to be used for this operation. This node group must be bound to the current system partition.

## Operands

*start_frame* Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive.

*start_slot* Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive.

**Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition.

*node_count* Specifies the number of nodes to be used for this operation. The information is added sequentially to nodes in slots within a frame and, if the slots in a frame are exhausted, to slots in the next sequential frame. Specify a value between 1 and 1024 inclusive.

**Note:** The *node_count* is considered to be within the current system partition.

*adapter_name* Specifies the name of the adapter. Valid adapter types are: ethernet (**en**), fddi (**fi**), token ring (**tr**), and switch (**css**).

*starting_IP_address*
Specifies the IP address of the first node on the network. IP addresses of subsequent nodes are created via incrementing the IP address for each node.

Each IP address used in the operation must be resolved by the **host** command on the control workstation.

*netmask* Specifies the netmask for the network on which the adapter resides. Specify a valid IP address.

## Description

Execute this command during installation of the SP to identify the IP addresses, netmask, and default route associated with node adapters other than en0. If all your IP addresses are in the same block, run this command once. If you have “holes” in your IP addressing scheme, run this command once for each block of addresses you want to assign.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **k4init**.

You can use the System Management Interface Tool (SMIT) to run the **spadaptrs** command. To use SMIT, enter:

```
smit node_data
```

and select the Additional Adapter Information option.

**Notes:**

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. After running this command, you must issue the **syspar_ctrl –r** command to refresh system partition-sensitive subsystems in each system partition where node customization was performed. Subsystems like **hats**, **hb**, and **hr** need to be refreshed whenever nodes or adapters are added or deleted.
3. Any changes made will not take effect on the nodes until they are customized.

## Location

**/usr/lpp/ssp/bin/spadaptrs**

## Related Information

Commands: **syspar_ctrl**

## Examples

1. If you specify the –**s** flag to skip IP addresses when setting the **css0** switch addresses, you must also specify –**n no** to not use switch numbers for an IP address assignment. You must specify –**a yes** to use ARP.

   ```
   spadaptrs -s yes -n no -a yes 1 1 30 css0 129.33.34.1 255.255.255.0
   ```

2. The following example configures the Gigabit Ethernet adapter as (**en1**) on node (**7**) following the guidelines of: Adapter Type = **fiber**; Duplex = **full**; Adapter Speed = **1000**.

   ```
   spadaptrs -t fiber -f 1000 -d full -l 7 en1 129.33.34.1 255.255.255.0
   ```

## spapply_config

### Purpose

**spapply_config** – Applies a system partition configuration to the SP system.

### Syntax

**spapply_config** [–**h**] [–**v**] [–**A**] [–**F**] [–**q**] *config_dir*/*layout_dir*

### Flags

| | |
|---|---|
| –**h** | Displays usage information. If this command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid options or operands are entered with the –**h** flag). |
| –**v** | Verifies, but does not apply, the specified configuration layout. With this option, the command: • Checks the contents of each **custom** file in the specified layout to ensure that the required stanza entries exist • Describes which system partitions would be changed and which would be unchanged by applying the specified configuration layout • Lists all nodes in the changed system partitions which are not shutdown |
| –**A** | Archives the System Data Repository (SDR). With this flag, a copy of the SDR prior to applying the configuration is saved using the **SDRArchive** command. |
| –**F** | Corrects recoverable errors encountered in the IBM Virtual Shared Disk subsystem in the application of the specified configuration layout. Irrecoverable errors encountered cause the command to terminate prior to applying the specified layout. |
| –**q** | Specifies quiet mode. This option suppresses all status messages as well as the output from most internally called commands. The list of changed and unchanged system partitions, the list of nodes in changed system partitions which are not shutdown, and any warning or error messages are still displayed with this option. |

### Operands

| | |
|---|---|
| *config_dir* | Specifies the directory name for a configuration directory. |
| *layout_dir* | Specifies the directory name for a layout directory within the configuration directory. |

### Description

This command functions in two phases: verification and application. Before applying a new system partition configuration, the administrator should back up the SP system SDR. This can be accomplished by using either the **SDRArchive** command or by using the –**A** flag on **spapply_config**. If your system has an SP switch, the **Eunpartition** command must be run before applying a new system partition configuration. Otherwise there will be unpredictable results in the new system partitions. Refer to the “Managing System Partitions” chapter in *PSSP: Administration Guide* for additional information.

The layout directory contains one system partition directory for each system partition in the configuration. Each partition directory contains the switch **topology** file and **nodelist** file. It also contains the **custom** file (created and updated by the **spcustomize_syspar** command). The **spapply_config** command verifies that these files exist. It also verifies the contents of the **custom** file. If an error is encountered in this verification phase, the command issues an appropriate message and terminates without attempting to apply a configuration layout that is not valid. As part of its verification phase, this command also calls the **verparvsd** command to determine the impact on the IBM Virtual Shared Disk subsystem of applying the specified configuration layout. If any errors or warnings are returned from **verparvsd**, the **spapply_config** command reports those messages and stops. The **–F** flag can be used to alter this behavior by correcting recoverable IBM Virtual Shared Disk errors encountered in the analysis of the IBM Virtual Shared Disk subsystem.

As part of its processing, **spapply_config** displays to standard output the list of changed system partitions and the list of unchanged system partitions. A changed system partition is a currently-defined partition which will be changed in some way by the application of the specified configuration layout. Nodes in changed system partitions should be shutdown prior to applying that configuration. Conversely, an unchanged system partition is a currently-defined partition which will be unchanged by the application of the specified configuration layout. Nodes in unchanged system partitions can remain in operation during the application of this configuration layout. The **spapply_config** command issues the **Eannotator**, **Eprimary**, and **Etopology** commands as necessary.

The **spapply_config** command issues status messages which track the progress of operation to standard output. These messages along with the lists of changed and unchanged system partitions can be suppressed by the using the **–q** flag.

In the event that **spapply_config** encounters an error during the application phase, a descriptive error message is displayed and the command stops. In this case, it will be necessary to restore the SP SDR and the system partition-sensitive subsystems (for example, **hats**, **hb**, and **hr**) to their previous state by using the **sprestore_config** command.

**Note:** Due to system partitioning changes, your SP_NAME environment variable may no longer be set to a valid system partition name. To get a list of valid system partition names, enter the **splst_syspars -n** command. Then verify that your SP_NAME environment variable is either unset or set to one of the system partition names in the list.

## Files

| | |
|---|---|
| **nodelist** | Contains a list of switch node numbers contained in a system partition (used internally, not by end users). |
| **topology** | Contains the wiring configuration information for switch-to-switch and node-to-switch cabling in a switch network. This information is used during switch initialization. |

## Related Information

Commands: **Eunpartition**, **SDRArchive**, **spcustomize_syspar**, **spdisplay_config**, **sprestore_config**, **spverify_config**, **syspar_ctrl**, **verparvsd**

Files: **nodelist**, **topology**

## Location

**/usr/lpp/ssp/bin/spapply_config**

## Examples

1. To apply the system partition configuration represented by the **config.4_12/layout.2** layout directory, enter:

   ```
   spapply_config config.4_12/layout.2
   ```

2. To check (but not apply) the system partition configuration represented by the **config.8_8/layout.1** layout directory, enter:

   ```
   spapply_config -v config.8_8/layout.1
   ```

## spauthconfig

### Purpose

**spauthconfig** – Called from **rc.sp**, run each time a node boots. Installs and configures node based on selected authentication methods.

### Syntax

**spauthconfig** [–**h**] *node number*

### Flags

| | |
|---|---|
| –**h** | Presents syntax of command. |

### Operands

*node number*

### Description

The **spauthconfig** command invokes the **updauthfiles** command and sets the authentication methods.

This command may be run manually to update the node to pick up any changes to the authentication configuration that occurred on the control workstation. This command is similar to the process run on the control workstation except this is automated and relies on information in the SDR to control program flow. The ODM will be updated based on authentication methods set in the SDR.

### Files

The **spauthconfig** command reads from the SDR database, ODM database, and **.rhost** file.

The command writes to the log file created in **/var/adm/SPlogs/auth_install/log**, SDR database, ODM database, and **.rhost** file.

### Standard Input

SDR database

ODM database

**.rhost**

### Standard Output

Log file created in **/var/adm/SPlogs/auth_inst/log**

SDR database

ODM database

**.rhost**

## Exit Values

**0** Indicates successful completion of the command.

**1** Indicates errors occurred during the execution of this program. Review any reported errors either on the console or in the Log file.

## Security

Root authority is required to run this command.

## Location

**/usr/lpp/ssp/bin/spauthconfig**

## Related Information

Commands: **lsauthent**, **chauthent**

Files: **/etc/inittab**

## Examples

This command may be run locally by the user, but generally will be run at boot time out of the **/etc/rc.sp** file. The following example will run the command on the local node 2:

```
/usr/lpp/ssp/bin/spauthconfig 2
```

## spbootins

### Purpose

**spbootins** – Enters boot/install configuration data for a node or series of nodes in the System Data Repository (SDR).

### Syntax

| **spbootins** {–**c** *volume_group_name* **|**
| –**r** {**install | customize | disk | maintenance | diag | migrate**}}
| [–**s** {**yes | no**}] {*start_frame start_slot node_count* **|** –**l** *node_list*}

### Flags

–**c** *volume_group_name*
: Specifies the name of the volume group to select for the target nodes. This volume group will become the current volume group for subsequent installations, and customizations.

–**r**
: Specifies the boot/install server's response to the bootp request from the nodes.

**install** Indicates that you should specify **install** if you want the server to perform a network install (overwrite install) and customize each node.

**customize** Indicates that you should specify **customize** if you want the server to place node-specific configuration information from the SDR into each node's local Object Data Management (ODM).

**disk** Indicates that you should specify **disk** if you want the server to ignore the bootp request and have each node boot from its local disk.

**maintenance** Indicates that you should specify **maintenance** to have each node boot in a prompted mode.

A node that boots in a prompted mode, comes up with the “Install/Maintenance” panel. From this panel, you can choose option 3 to start a limited function maintenance shell. You may access files in the root volume group (**rootvg**) by choosing the panels to mount the root volume group and enter a shell.

**diag** Sets the bootp_response to **diag**. The next time the node is network booted, a diagnostic menu will be displayed on the tty. From the diagnostic menu, you can execute simple or advanced diagnostics on the node or execute service aids. Service aids allow you to perform such tasks as formatting and certifying the hard drive on the node, or downloading microcode to a device attached to the node. When diagnostics are complete, set the bootp_response back to disk and reboot the node.

The **diag** parameter can be used only on the IBM Parallel System Support Programs for AIX (PSSP) Version 2 Release 1 and later nodes. PSSP Version 1 Release 2 nodes cannot run remote diagnostics.

**migrate** Indicates that you want the server to perform a migration installation on the specified nodes. See the *PSSP: Installation and Migration Guide* for more details on the migration installation method.

**–s no | yes** Indicates whether **setup_server** should be run on the boot servers (including the control workstation) of the indicated nodes. If you specify **–s no**, **setup_server** is not run on the node's boot server, and it must be run later to make any necessary changes to installation-related files. Specify **–s yes** if you have finished entering boot/install/usr server data during your initial installation or if you are changing data after the initial installation. Otherwise, specify **–s no**. If **–s** is not specified, the default is **–s yes**.

**–l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

## Operands

*start_frame* Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive.

*start_slot* Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive.

**Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition.

*node_count* Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive.

**Note:** The *node_count* is considered to be within the current system partition.

## Description

Use this command to select a volume group for the target nodes to use as their root volume group and to select what action to perform using that volume group the next time this node is booted or network booted. Each time this command is run, the **setup_server** command is run on each of the affected boot/install servers.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **k4init**.

You can use the System Management Interface Tool (SMIT) to run the **spbootins** command. To use SMIT, enter:

```
smit node_data
```

and select the Boot/Install Information option.

You cannot use SMIT if you are using AFS authentication services.

**Notes:**

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. Any changes made will not take effect on the nodes until they are customized.

## Location

**/usr/lpp/ssp/bin/spbootins**

## Examples

1. To change the root volume group for node 1 and install that volume group, enter:

   ```
   spbootins -c rootvg2 -r install -s yes -l 1
   ```

2. To customize nodes 3 and 7 using their current volume group, enter:

   ```
   spbootins -r customize -s yes -l 3,7
   ```

## spbootlist

### Purpose

**spbootlist** – Sets the bootlist on a node or set of nodes based on the values in the Node and Volume Group objects.

### Syntax

| **spbootlist** {*start_frame start_slot node_count* **-l** *node_list*}

### Flags

| | |
|---|---|
| –**l** *node_list* | Specifies a list of nodes for this operation. This list can be a single numeric node number, or a list of numeric node numbers separated by commas. |

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of the first node to be used for this operation. |
| *start_slot* | Specifies the slot number of the first node to be used for this operation.<br><br>**Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition. |
| *node_count* | Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame.<br><br>**Note:** The *node_count* is considered to be within the current system partition. |

### Description

The **spbootlist** command is used to set the bootlist on a node or set of nodes based on the values in the Node and Volume Group objects. The selected_vg attribute of the Node object will point to a unique Volume_Group object for a node. **spbootlist** will look at the vg_name of the Volume_Group object and determine which physical volumes are in the volume group, and set the bootlist to "ent0" followed by all the physical volumes which contain boot logical volumes. In a mirrored environment, more than one physical volume will contain a boot logical volume.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that a recoverable error occurred, some changes may have succeeded. |
| **2** | Indicates that an irrecoverable error occurred and no changes were made. |

## Security

A user must run as "root" and have a valid Kerberos ticket.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/spbootlist**

## Related Information

Commands: **spchvgobj**

## Examples

1. To set the bootlist on node one, enter:

   ```
   spbootlist -l 1
   ```

2. To set the bootlist on a list of nodes, enter:

   ```
   spbootlist -l 1,2,3
   ```

## spchuser

## Purpose

**spchuser** – Changes the attributes of an SP user account.

## Syntax

**spchuser** *attribute***=***value* ... *name*

## Flags

None.

## Operands

| | |
|---|---|
| *attribute***=***value* | Pairs of the supported attributes and values as follows. |
| *name* | Name of the user account whose information you want to change. |

**Supported Attributes and Values**

| | |
|---|---|
| **id** | ID of the user specified by the *name* parameter. |
| **pgrp** | Principle group of the user specified by the *name* parameter. |
| **gecos** | General information about the user. |
| **groups** | The secondary groups to which the user specified by the *name* parameter belongs. |
| **home** | Host name of the file server where the home directory resides and the full path name of the directory. You can specify a host and directory in the format *host:path*, just specify the directory and have the host default to a value set in SMIT site environment panel or the **spsitenv** command, or just specify a directory and have the host default to the local machine. |
| **login** | Indicates whether the user specified by the *name* parameter can log in to the system with the **login** command. This option does not change the **/etc/security/user** file. Instead, it alters the user password field in **/etc/security/passwd**. |
| **shell** | Program run for the user specified by the *name* parameter at the session initiation. |

## Description

No flags are supported. Except for **home**, the rules for the supported attributes and values correspond to those enforced by the AIX **chuser** command.

You can only change the values of the supported attributes.

You can use the System Management Interface Tool (SMIT) to run the **spchuser** command. To use SMIT, enter:

```
smit spusers
```

and select the Change/Show Characteristics of a User option.

**Note:** This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

## Location

**/usr/lpp/ssp/bin/spchuser**

## Examples

To change the default shell to **/bin/csh**, and change the secondary group membership to **dev** and **dev2** for the user account **charlie**:

```
spchuser groups=dev,dev2 shell=/bin/csh charlie
```

## spchvgobj

### Purpose

**spchvgobj** – Changes the contents of a **Volume_Group** object.

### Syntax

| **spchvgobj** –**r** *volume_group_name* [–**h** *pv_list*] [–**i** *install_image*]
| [–**p** *code_version*] [–**v** *lppsource_name*] [–**n** *boot_server*]
| [–**c** {**1** | **2** | **3**}] [–**q** {**true** | **false**}]
| {*start_frame start_slot node_count* | –**l** *node_list*}]

### Flags

–**r** *volume_group*
Specifies the root volume group name to apply the changes towards.

–**h** *pv_list* Indicates the physical volumes to be used for installation for the volume group specified. The root volume group is defined on the disks indicated, and all data on the disks is destroyed. The physical volumes may be specified as logical names (such as **hdisk0**), hardware location (such as **00-00-00-0,0**), or connwhere (such as **ssar//012345678912345**). If multiple physical volumes are specified, separate them by commas for logical names and by colons for hardware location and connwhere. At installation, the value for each node's *pv_list* is **hdisk0**.

**Note:** IBM strongly suggests that you use the hardware location or connwhere format. It ensures that you install on the intended disk by targeting a specific disk at a specific location. The logical naming of physical volumes may change depending on hardware installed or possible hardware problems. This is especially true when there are external drives present, as the manner in which the device names are defined may not be obvious.

–**i** *install_image*
Specifies the name of the install image to be used for the volume group when they are next network-installed. Specify a file in the **/spdata/sys1/install/images** directory on the control workstation. At installation, the value for each volume group's install image name is default, which means that the default install image name for the system partition or the system is used for each node. The default install image name is found in the Syspar or the SP object in that order.

–**p** *code_version*
Sets the volume group's code version. Use this to indicate the PSSP level to install on the node. The *code_version* value you choose must match the directory name that the PSSP installation files are placed under in the **/spdata/sys1/install/pssplpp** directory during installation. See the *PSSP: Installation and Migration Guide* for more details.

–**v** *lppsource_name*
Sets the volume group's lppsource name. Use this to indicate the AIX level to install on the node. The *lppsource_name* value you choose must match the directory name you choose to place the lppsource files under

in the **/spdata/sys1/install** directory during installation. See the *PSSP: Installation and Migration Guide* for more details.

–**n** *boot_server*
Identifies the boot/install server for the volume groups you have specified. The boot/install server is identifies by a node number. Node number 0 represents the control workstation. The value of the boot/install server at installation depends on how many frames are in your system. In a single frame system, the control workstation (node 0) is the default server for each node. In a multiple frame system, the default server for the first node in each frame is the control workstation, and the default server for the rest of the nodes in a frame is the first node in that frame.

–**c copies**
Specifies the number of mirrors to create for the volume group. To enable mirroring, set this to 2 or 3. Setting this to 1 disables mirroring. When enabling mirroring, be sure that there are enough physical volumes to contain all the copies of the volume group. Each copy must have at least 1 physical volume.

–**q true | false**
Specifies whether quorum should be enabled. If quorum is enabled, a voting scheme will be used to determine if the number of physical volumes that are up is enough to maintain quorum. If quorum is lost, the entire volume group will be taken off line to preserve data integrity. If quorum is disabled, the volume group will remain on line as long as there is at least 1 running physical volume.

–**l** *node_list*
Specifies a list of nodes to be used for this operation. Specify a comma-delimited list of node numbers. If you use the **-l** flag, do not use the *start_frame*, *start_slot*, or *node_count* operands.

## Operands

*start_frame*
Specifies the frame number of the first node to be used for this operation.

*start_slot*
Specifies the slot number of the first node to be used for this operation.

**Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition.

*node_count*
Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame.

**Note:** The *node_count* is considered to be within the current system partition.

## Description

This command is used to change the configuration information for an existing volume group on a node or group of nodes in the System Data Repository (SDR). When this command is run and the SDR is changed, **setup_server** must be run on the affected boot/install servers and affected nodes may need to be customized or installed to apply the changes. Certain volume group information such as mirroring and the *pv_list* may be updated using the **spmirrorvg** or **spunmirrorvg** commands.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that a recoverable error occurred, some changes may have succeeded. |
| **2** | Indicates that an irrecoverable error occurred and no changes were made. |

## Security

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets. If you do not have a ticket-granting-ticket, you must run **k4init**.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/spchvgobj**

## Related Information

Commands: **spbootins**, **spmirrorvg**, **spmkvgobj**, **sprmvgobj**, **spunmirrorvg**

## Examples

1. To specify node 1 as the boot/install server for the volume group "rootvg" on nodes 2-16, enter:

```
spchvgobj -r rootvg -n 1 1 2 15
```

2. To enable mirroring with 2 copies, no quorum and 2 SSA physical volumes for the volume group "rootvg" on nodes 2 and 3, enter:

```
spchvgobj -r rootvg -c 2 -q false -h \
ssar//567464736372821:ssar//67464736372821 -l 2,3
```

## spcustomize_syspar

### Purpose

**spcustomize_syspar** – Enters or verifies customization information to be used in creating a system partition.

### Syntax

**spcustomize_syspar** [–**h**] [–**n** *syspar_name* **|** *IP_address*]
[–**l** *PSSP_code_level*]
[–**d** *default_install_image* **|** **default**]
[–**e** *primary_node* **|** **default**]
[–**b** *backup_primary_node* **|** **default**]
*config_dir/layout_dir/syspar_dir* **|**
*fully_qualified_path_name*
[–**r** **k4** **|** **std**] [–**m** *k5* **|** **k4** **|** **std**]

### Flags

–**h** Displays usage information.

–**n** *syspar_name* **|** *IP_address*
Specifies the system partition name (the control workstation host name or host name alias) or IP address (which corresponds to the system partition name) associated with this system partition.

–**l** *PSSP_code_level*
Specifies the IBM Parallel System Support Programs for AIX (PSSP) code level for the system partition. For mixed system partitions, partitions that have multiple supported levels of PSSP coexisting in the same partition, should be set to the minimum (earliest) level of PSSP in this system partition.

–**d** *default_install_image* **| default**
Specifies the default install image for the system partition or **default** to direct the system to use the system-wide default install image. Refer to *PSSP: Installation and Migration Guide* for additional information on the default install image.

–**e** *primary_node* **| default**
Specifies the primary node number for switch operations or **default** to direct the system to automatically set the default which is the first node in the node list.

–**b** *backup_primary_node* **| default**
Specifies the primary backup node number for switch operations or **default** to direct the system to automatically set the default which is the last node in the node list. This flag is valid only on SP Switch systems.

–**r** Authorization Method for root access to remote commands.

–**m** Authentication Method enabled for **usstem** use.

## Operands

*config_dir* Specifies the directory name for a configuration directory.

*layout_dir* Specifies the directory name for a layout directory within the configuration directory.

*syspar_dir* Specifies the directory name for a system partition directory within the layout directory.

*fully_qualified_path_name*
Specifies the fully qualified path name to a system partition directory.

## Description

Use this command to customize a system partition customization file (**custom**) or to display the previously-entered customization information.

For a specified system partition, the customization data can be entered with the optional parameters. If the **custom** file does not exist, you can create one by specifying the –**n** and –**l** flags. The –**d** and –**e** flags are optional when creating a **custom** file. If –**d** and –**e** are not specified, the system automatically specifies **default** to set the default install image and primary node in the newly-created **custom** file. Once the **custom** file is created, any combination of the optional parameters can be used to update the contents of the file.

If none of the optional parameters are specified with the **spcustomize_syspar** command, the contents of the customization file for the specified system partition are displayed to standard output as in the **spdisplay_config** command with the –**c** flag.

## Location

**/usr/lpp/ssp/bin/spcustomize_syspar**

## Related Information

Commands: **spapply_config**, **spdisplay_config**, **spverify_config**

Files: **nodelist**, **topology**

## Examples

1. To display the customization information for the specified system partition, enter:

```
spcustomize_syspar config.4_12/layout.1/syspar.1

syspar-name:             my-partition-1
IP-address:              9.102.55.301
PSSP-code-level:         PSSP-2.2
default-install-image:   bos.4.1.5
primary-node:            9
backup-primary-node:     16
```

2. To modify the system partition name, PSSP code level, and primary node information for the specified system partition, enter:

```
spcustomize_syspar -n my-new-partition-name -l PSSP-2.2 \
-e 7 config.4_12/layout.1/syspar.1
```

3. To use the default primary node information for the specified system partition, enter:

   ```
   spcustomize_syspar -e default config.4_12/layout.1/syspar.1
   ```

## spcw_addevents

### Purpose

**spcw_addevents** – Identifies the High Availability Cluster Multiprocessing (HACMP) event scripts supplied by the High Availability Control Workstation (HACWS) to the AIX High Availability Cluster Multi-Processing (HACMP) software.

### Syntax

**spcw_addevents**

### Flags

None.

### Operands

None.

### Description

HACWS customizes the recovery of control workstation services by providing HACMP event scripts, which get executed by the HACMP software. The **spcw_addevents** command is a shell script which identifies the HACMP event scripts to HACMP, without requiring the system administrator to go through all the equivalent HACMP SMIT panels.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **nonzero** | Indicates that an error occurred. |

### Security

You must have root privilege to run this command.

### Prerequisite Information

Refer to *PSSP: Administration Guide* for additional information on the HACWS option.

### Location

**/usr/sbin/hacws/spcw_addevents**

## spcw_apps

### Purpose

**spcw_apps** – Starts or stops control workstation applications in a High Availability Control Workstation (HACWS) configuration.

### Syntax

**spcw_apps** {–**u** | –**d**} [–**i** | –**a**]

### Flags

| | |
|---|---|
| –**u** | Starts control workstation applications on the local host. |
| –**d** | Stops control workstation applications on the local host. |
| –**i** | Sets the local host to be the inactive control workstation before starting or after stopping control workstation applications. |
| –**a** | Sets the local host to be the active control workstation before starting or after stopping control workstation applications. |

### Operands

None.

### Description

The control workstation services are started at boot time on a regular control workstation via entries in **/etc/inittab**. An HACWS configuration requires the capability to stop control workstation services on one control workstation and restart them on the other. The **install_hacws** command removes most of the control workstation entries from **/etc/inittab**, and the **spcw_apps** command is provided as a means to stop and start control workstation services in the HACWS configuration. In addition, the **spcw_apps** command can be used to make the inactive control workstation act as a client of the active control workstation to keep the two control workstations synchronized.

**Note:** The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP) will start or stop the control workstation applications during a fail over or reintegration. The administrator should not normally have to start or stop the control workstation applications.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **nonzero** | Indicates that an error occurred. |

## Prerequisite Information

Refer to *PSSP: Administration Guide* for additional information on the HACWS option.

## Location

**/usr/sbin/hacws/spcw_apps**

## Related Information

Command: **install_hacws**

## Examples

In the following example, assume that the primary control workstation is currently the active control workstation. This means that the primary control workstation is providing control workstation services to the SP system. When a control workstation failover occurs, the AIX High Availability Cluster Multi-Processing (HACMP) software moves the control workstation network and file system resources from the primary to the backup control workstation. In addition, control workstation applications must be stopped on the primary and restarted on the backup. HACWS provides the **spcw_apps** command to HACMP as the method to accomplish this. The HACMP software issues the following command on the primary:

```
spcw_apps -di
```

This command stops control workstation services on the active primary and then sets the primary to be the inactive control workstation. Next, the HACMP software issues the following command on the backup:

```
spcw_apps -ua
```

This command sets the backup to be the active control workstation and then starts the control workstation services on the backup. Finally, the HACMP software issues the following command on the primary:

```
spcw_apps -u
```

This command configures the primary to be a client of the backup (which is active) control workstation.

## spdeladap

### Purpose

**spdeladap** – Removes configuration data for adapters for a node or series of nodes from the System Data Repository (SDR).

### Syntax

**spdeladap** {*start_frame start_slot node_count* **|** –**N** *node_group* **|** –**l** *node_list*} *adapter_name*

### Flags

| | |
|---|---|
| –**N** *node_group* | Specifies a node group to be used for this operation. This node group must be bound to the current system partition. |
| –**l** *node_list* | Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.) |

### Operands

| | |
|---|---|
| *start_frame* | Frame number of first node to be used for this operation. Specify a value between 1 and 64 inclusive. |
| *start_slot* | Slot number of first node to be used for this operation. Specify a value between 1 and 16 inclusive. **Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition. |
| *node_count* | Number of nodes to be used for this operation. The adapter information is deleted for successive nodes within a frame and, when the count of nodes causes the nodes in a frame to be exhausted, for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive. **Note:** The *node_count* is considered to be within the current system partition. |
| *adapter_name* | Specifies the name of the adapter. Valid adapter types are: ethernet (**en**), fddi (**fi**), token ring (**tr**), and switch (**css**). |

### Description

Use this command to remove configuration data for adapters for a node or series of nodes from the SDR. You cannot use this command to delete data for the **en0** adapter. If you want to remove configuration data for the **en0** adapter, you should use the **spdelnode** command.

You can use the System Management Interface Tool (SMIT) to run the **spdeladap** command. To use SMIT, enter:

```
smit delete_data
```

and select the Delete Adapter Information option.

**Notes:**

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. After running this command, you must issue the **syspar_ctrl –r** command to refresh system partition-sensitive subsystems in each system partition where node customization was performed. Subsystems like **hats**, **hb**, and **hr** need to be refreshed whenever nodes or adapters are added or deleted.
3. Any changes made will not take effect on the nodes until they are customized.

## Location

**/usr/lpp/ssp/bin/spdeladap**

## Related Information

Commands: **syspar_ctrl**

## Examples

This example deletes **tr0** adapter information for the the first two nodes in frame 2:

```
spdeladap 2 1 2 tr0
```

## spdelexp

### Purpose

**spdelexp** – Removes configuration data for node expansion units from the System Data Repository (SDR).

### Syntax

**spdelexp** {*start_frame start_slot expansion_count* **|** –**x** *expansion_list* **|** –**l** *associated_node_list* **|** –**N** *node_group*}

### Flags

–**x** *expansion_list*
: Specifies a list of node expansion units to be used for this operation. Either specify a comma-delimited list of expansion numbers, or a file containing one line of data which is a comma-delimited list of expansion numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *expansion_list* field, do not use the *start_frame*, *start_slot*, or *expansion_count* fields, the **-l** flag or the **-N** flag.

–**l** *associated_node_list*
: Specifies a list of nodes whose attached node expansion units are to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *associated_node_list* field, do not use the *start_frame*, *start_slot*, or *expansion_count* fields, the **-x** flag or the **-N** flag.

–**N** *node_group*
: Specifies a node group containing nodes whose attached node expansion units are to be used for this operation. If you use the *node_group* field, do not use the *start_frame*, *start_slot*, or *expansion_count* fields, the **-x** flag or the **-l** flag.

### Operands

*start_frame*
: Specifies the frame number of the first expansion unit to be used for this operation. Specify a value between 1 and 64 inclusive.

*start_slot*
: Specifies the slot number of the first expansion unit to be used for this operation. Specify a value between 1 and 16 inclusive.

: **Note:** The *start_frame* and *start_slot* must resolve to an expansion unit in the current system partition.

*expansion_count*
: Specifies the number of expansion units to be used for this operation. The expansion information is removed for successive node expansion units within a frame and, when the count of expansion units exceeds those in a frame, processing continues with the next sequential frame. Specify a value between 1 and 1024 inclusive.

**Note:** The *expansion_count* is considered to be within the current system partition.

## Description

Execute this command to remove configuration data for a node expansion unit or series of units from the SDR. All the nodes to be deleted must be shut down. The *expansion_number* is also removed from the *expansion_list* of the node that it is connected to.

This command can be used to delete node expansion units in the current system partition. To delete expansion units in another system partition, you must make it your current system partition. Node expansion units reside in the system partition that their associated node resides in, or in the default system partition if the node expansion unit is not connected to a node.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **k4init**.

You can use the System Management Interface Tool (SMIT) to run the **spdelexp** command. To use SMIT, enter:

```
smit delete_data
```

and select the Delete Node Expansion Information option.

**Note:** This command should be run only on the control workstation.

## Location

**/usr/lpp/ssp/bin/spdelexp**

## Related Information

Commands: **syspar_ctrl**

## Examples

This example removes configuration data for node expansion unit 2 from the SDR:

```
spdelexp -x 2
```

## spdelfram

### Purpose

**spdelfram** – Removes configuration data for a frame or series of frames from the System Data Repository (SDR).

### Syntax

**spdelfram** *start_frame frame_count*

### Flags

None.

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of first node to be used for this operation. Specify a value between 1 and 64 inclusive. |
| *frame_count* | Specifies the number of frames to be used for this operation. Specify a value between 1 and 64 inclusive. |

### Description

Execute this command to remove configuration data for a frame or series of frames from the SDR. Any node information for nodes on the frames is also removed, as well as adapter information for any of the nodes on the frames. All the nodes on all the frames to be deleted must be shut down. A frame containing a node acting as a server for another node on a frame not being removed with this operation cannot be removed with this command. To remove a frame containing such a node, you must configure a different **boot/install** or **/usr** server for the client nodes on the other frames. If a definition for a node being removed exists in **/etc/switch.info**, this definition will be removed from that file. If a non-SP frame is using a switch port number within an SP frame, that SP frame may not be deleted.

| All node expansion information is removed from the SDR for expansion units that
| reside on frames being deleted, or that reside on other frames and are connected
| to nodes residing on frames being deleted. These node expansion units must be
| shut down before invoking this command.

The **spdelfram** command removes all extension node and extension node adapter information for extension nodes whose node numbers are within the range of node numbers represented by a frame being deleted.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **k4init**.

You can use the System Management Interface Tool (SMIT) to run the **spdelfram** command. To use SMIT, enter:

```
smit delete_data
```

and select the Delete Frame Information option.

**Notes:**

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. You should stop the Resource Manager before the command and start it again afterward.
3. There must be only one system partition defined when deleting a frame. This is to aid in the configuration of the system once the frame is deleted.

## Security

You must be logged into the control workstation as root to run this command.

## Location

**/usr/lpp/ssp/bin/spdelfram**

## Examples

1. This example deletes the first two node expansion units in frame 2:

   ```
   spdelexp 2 1 2
   ```

2. To delete the node expansion units connected to node 5, enter

   ```
   spdelexp -l 5
   ```

## spdelnode

### Purpose

**spdelnode** – Removes configuration data for a node or nodes from the System Data Repository (SDR).

### Syntax

**spdelnode** {*start_frame start_slot node_count* **|** –**N** *node_group* **|** –**l** *node_list*}

### Flags

| | |
|---|---|
| –**N** *node_group* | Specifies a node group to be used for this operation. This node group must be bound to the current system partition. |
| –**l** *node_list* | Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.) |

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive. |
| *start_slot* | Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive. **Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition. |
| *node_count* | Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame and, when the count of nodes causes the nodes in a frame to be exhausted, for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive. **Note:** The *node_count* is considered to be within the current system partition. |

### Description

Execute this command to remove configuration data for a node or series of nodes from the SDR. Any adapter information associated with the node is also removed. All the nodes to be deleted must be shut down. A node acting as a server for another node cannot be removed with this command. To remove a node which is a server, you must configure a different boot/install server for the client nodes.

| All node expansion information is removed from the SDR for expansion units that
| are connected to nodes being deleted. These node expansion units must be shut
| down before invoking this command.

This command can be used to delete nodes in the current system partition. To delete nodes in another system partition, you must make it your current system partition.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **k4init**.

You can use the System Management Interface Tool (SMIT) to run the **spdelnode** command. To use SMIT, enter: smit delete_data and select the Delete Node Information option.

**Notes:**

1. This command should be run only on the control workstation.
2. After running this command, you must issue the **syspar_ctrl -r** command to refresh system partition-sensitive subsystems in each system partition where node customization was performed. Subsystems like **hats**, **hb**, and **hr** need to be refreshed whenever nodes or adapters are added or deleted.

## Security

You must be logged into the control workstation as root to run this command.

## Related Information

Commands: **syspar_ctrl**

## Location

**/usr/lpp/ssp/bin/spdelnode**

## Examples

1. This example deletes the first two nodes in frame 2:

   ```
   spdelnode 2 1 2
   ```

2. To delete the nodes in the node group temp_nodes, enter:

   ```
   spdelnode -N temp_nodes
   ```

## spdisplay_config

### Purpose

**spdisplay_config** – Displays system partition configuration information which can be used to partition an SP system.

### Syntax

**spdisplay_config** [–**h**] [–**R**] [–**d**] [–**c**] [–**n**] [*config_dir* [**/***layout_dir* [**/***syspar_dir*]] **|** *fully_qualified_path_name*]

### Flags

| | |
|---|---|
| –**h** | Displays usage information. If this command is issued with the –**h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid options are entered with the –**h** flag). |
| –**R** | Recursively displays information for all levels below the specified directory level. |
| –**d** | Displays the description file (**layout.desc**) for the specified layout. This flag is valid only if the specified directory (or any subdirectories below it if –**R** is specified) is a layout directory. |
| –**c** | Displays the customization file (**custom**) for the specified system partition. This flag is valid only if the specified directory (or any subdirectories below it if –**R** is specified) is a system partition directory. |
| –**n** | Displays the node list file (**nodelist**) for the specified system partition. This flag is valid only if the specified directory (or any subdirectories below it if –**R** is specified) is a system partition directory. |

### Operands

| | |
|---|---|
| *config_dir* | Specifies the directory name for a configuration directory. |
| *layout_dir* | Specifies the directory name for a layout directory. |
| *syspar_dir* | Specifies the directory name for a system partition directory. |
| *fully_qualified_path_name* | Specifies the fully qualified directory path for a configuration directory, layout directory, or system partition directory. |

### Description

This command displays system partition information stored in the system partition information directory structure. Depending on the option and operand specified, the information displayed is at the configuration, layout, or system partition level. The output of this command is normally restricted to the SP on which it is executed. To display information for configurations applicable to SP systems other than the one on which it is executed, a fully qualified path name must be provided. This command does not display current system partition information (that function is provided with the **splstdata** command). If the command is issued without specifying a directory path name, the list of all valid configuration directories for the SP on which the command is running is displayed. If none of the file option flags (–**c**, –**d**, –**n**) are entered, the names of the files and subdirectories located on (and all levels below if **–R** is specified) the specified directory are displayed. If any of the file option flags are entered, the contents of the requested files are displayed instead.

## Files

| | |
|---|---|
| **custom** | Contains customization information for a partition (such as, partition name, IP address, code level, and so on). |
| **layout.desc** | Describes the node slot breakdown for a partitioning configuration (which nodes in which partition). |
| **nodelist** | Contains a list of switch node numbers contained in a system partition (used internally, not by end users). |
| **topology** | Contains the wiring configuration information for switch-to-switch and node-to-switch cabling in a switch network. This information is used during switch initialization. |

## Related Information

Commands: **spapply_config**, **spcustomize_syspar**, **splstdata**, **spverify_config**

Files: **nodelist**, **topology**

## Location

**/usr/lpp/ssp/bin/spdisplay_config**

## Examples

1. To display all valid configurations for this SP, enter:

   ```
   spdisplay_config
   config.2_14
   config.4_12
   config.8_8
   ```

2. To display the list of layout directory names for a specific configuration directory to standard output, enter:

   ```
   spdisplay_config config.4_12
   layout.1
   layout.2
   layout.3
   ```

3. To display the name of the file describing the layout and the list of system partition directory names for a specific layout directory to standard output, enter:

   ```
   spdisplay_config config.4_12/layout.2
   layout.desc
   syspar.1
   syspar.2
   ```

4. To display the list of files located in a specific system partition directory to standard output, enter:

   ```
   spdisplay_config config.4_12/layout.2/syspar.1
   custom
   nodelist
   topology
   ```

If the **–c** flag is also supplied, only the customization information is displayed for that system partition. For example:

```
spdisplay_config -c config.4_12/layout.2/syspar.1
custom:
syspar-name:              my-partition-1
IP-address:               9.102.55.301
primary-node:             9
default-install-image:    bos.4.1.5
PSSP-code-level:          PSSP-2.2
```

If the **–n** flag is supplied, only the list of nodes is displayed for that system partition. For example:

```
spdisplay_config -n config.4_12/layout.2/syspar.1
nodelist:
switch node numbers:      4 5 6 7 8 9 10 11 12 13 14 15
node numbers:             5 6 7 8 9 10 11 12 13 14 15 16
```

All of the commands can be issued with the **–R** flag to recursively display the information on the specified directory and all the levels below it.

To display the entire system partition information directory structure for the **config.4_12** configuration on this SP, enter:

```
spdisplay_config -R config.4_12
layout.1:
layout.1/layout.desc
layout.1/syspar.1:
layout.1/syspar.1/custom
layout.1/syspar.1/nodelist
layout.1/syspar.1/topology
layout.1/syspar.2:
layout.1/syspar.2/custom
layout.1/syspar.2/nodelist
layout.1/syspar.2/topology
layout.2:
layout.2/layout.desc
layout.2/syspar.1:
layout.2/syspar.1/custom
layout.2/syspar.1/nodelist
layout.2/syspar.1/topology
layout.2/syspar.2:
layout.2/syspar.2/custom
layout.2/syspar.2/nodelist
layout.2/syspar.2/topology
```

Another example to recursively display all of the customization information for the **config.4_12** on this SP follows:

```
spdisplay_config -R -c config.4_12
layout.1/syspar.1/custom:
syspar-name:              my-partition-1
IP-address:               9.102.55.301
primary-node:             9
default-install-image:    bos.4.1.5
PSSP-code-level:          PSSP-2.2

layout.1/syspar.2/custom:
syspar-name:              my-partition-2
IP-address:               9.102.55.302
primary-node:             9
default-install-image:    bos.4.1.5
PSSP-code-level:          PSSP-2.2

layout.2/syspar.1/custom:
syspar-name:              my-partition-1
IP-address:               9.102.55.501
primary-node:             9
default-install-image:    bos.4.1.5
PSSP-code-level:          PSSP-2.2

layout.2/syspar.2/custom:
syspar-name:              my-partition-2
IP-address:               9.102.55.502
primary-node:             9
default-install-image:    bos.4.1.5
PSSP-code-level:          PSSP-2.2
```

5. To display all valid configurations for an SP, specify the fully qualified path name to its system partition information directory. For example:

```
spdisplay_config /spdata/sys1/syspar_configs/1nsb0isb
config.16
config.4_12
config.4_4_4_4
config.4_4_8
config.8_8
```

## spethernt

### Purpose

**spethernt** – Enters configuration data for a node or series of nodes in the System Data Repository (SDR).

### Syntax

**spethernt** [–**s** {**yes** | **no**}] [–**t** {**bnc** | **dix** | **tp**}]
[–**d** {**full** | **half** | **auto**}] [–**f** {**10** | **100** | **auto**}]
{*start_frame start_slot node_count* | –**N** *node_group* | –**l** *node_list*}
*starting_IP_address netmask default_route*

### Flags

| | |
|---|---|
| –**s yes \| no** | Indicates whether IP addresses should be skipped, as needed, when assigning IP addresses. If –**s no** is specified, no skipping occurs; each IP address assigned is equal to the previous address assigned plus one. If –**s yes** is specified, each IP address assigned is equal to the previous address assigned plus the difference in their respective node numbers. |
| –**t bnc \| dix \| tp** | Designates the Ethernet type. Use **dix** to designate a thick Ethernet (also called a twisted pair). Use **bnc** to designate a thin Ethernet. Use **tp** to designate a twisted pair. The default is **bnc**. |
| –**d full \| half \| auto** | Specifies the communication transfer as one way (**half**) or two way (**full**). The default is **auto**. |
| –**f 10 \| 100 \| auto** | Specifies ethernet speed in megabits (Mb/s). Specify **10** for 10Mb per second or **100** for 100Mb per second. The default is **auto**. |
| –**N** *node_group* | Specifies a node group to be used for this operation. This node group must be bound to the current system partition. |
| –**l** *node_list* | Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.) |

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive. |
| *start_slot* | Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive. |

**Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition.

*node_count* Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame and, when the count of nodes causes the nodes in a frame to be exhausted, for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive.

**Note:** The *node_count* is considered to be within the current system partition.

*starting_IP_address*
Specifies the IP address or host name of the first node in this operation. IP addresses of subsequent nodes are created by incrementing the IP address for each node, depending on how the **–s** flag is set. Specify a valid IP address or host name.

Ensure that the combination of the starting IP address, the node count operand, and the **–s** flag do not result in the incrementing of the IP address to an IP address that is not valid.

Each IP address used in the operation must be resolved by the **host** command on the control workstation.

*netmask* Specifies the netmask for the en0 network. Specify a valid IP netmask.

| *default_route* The *default route* that you enter is not the same as the default route on the node. The route that you enter goes in the SDR Node Class. It is the route over which the node communicates with its boot/install server (for example, install, customize, and so on). The default route must be a valid Ethernet en0 path to the node's boot/install server and the control workstation.

| The default route on the node is the route it will use for its network communications if there is no specific route to the destination. During the boot process, this is set to the default route in the SDR. It can be changed later on in the boot process or after the node is running, but should not be changed permanently in the SDR. For FDDI, token ring, or other Ethernet adapters, create the route in **script.cust**. For the switch, set the route up in **/etc/inittab** after the line that runs **rc.switch**.

## Description

Execute this command during installation of the SP to identify the IP addresses, netmask, and default route associated with the en0 adapters of the nodes. If all your IP addresses are in the same block, run this command once. If you have “holes” in your IP addressing scheme, run this command once for each block of addresses you want to assign.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **k4init**.

You can use the System Management Interface Tool (SMIT) to run the **spethernt** command. To use SMIT, enter:

```
smit node_data
```

and select the SP Ethernet Information option.

**Notes:**

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. After running this command, you must issue the **syspar_ctrl –r** command to refresh system partition-sensitive subsystems in each system partition where node customization was performed. Subsystems like **hats**, **hb**, and **hr** need to be refreshed whenever nodes or adapters are added or deleted.
3. Any changes made will not take effect on the nodes until they are customized.

## Location

**/usr/lpp/ssp/bin/spethernt**

## Related Information

Commands: **syspar_ctrl**

## Examples

The following example adds SDR information for an en0 network of 15 nodes (frame 1 slot 1 to frame 1 slot 16 with the first node being a wide node and the rest of the nodes thin nodes all in a single system partition) with IP addresses from 129.33.32.1 to 129.33.32.15, a netmask of 255.255.255.0, and a default route of 129.33.32.200. The addresses are to be assigned to correspond with the nodes in the frame; for example, they do not increment the IP address of a wide node by 2 before assigning the IP address of the next node.

```
spethernt -s no 1 1 15 129.33.32.1 255.255.255.0 129.33.32.200
```

## spevent

### Purpose

**spevent** – Directly invokes the Event Perspective graphical user interface (GUI).

### Syntax

**spevent** [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**]
[–**backgroundColor** *colorName*]
[–**foregroundColor** *colorName*] [–**fontFamily** *name*]
[–**fontSize** *Size*] [–**fontBold**] [–**fontItalic**] [–**nosplash**] [–**h**]

### Flags

–**userProfile** *name*
: Upon initialization, loads the specified user profile. If a user profile named "Profile" exists in the user's home directory, it will be loaded by default if the –**userProfile** flag is not specified.

–**systemProfile** *name*
: Upon initialization, loads the specified system profile instead of the default system profile. The default system profile is named "Profile."

–**noProfile**
: Upon initialization, does not read either profile.

–**backgroundColor** *colorName*
: Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**foregroundColor** *colorName*
: Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
: Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
: Overrides any font point size with the specified size. Valid values are 6–30 points.

–**fontBold**
: Sets the font to bold.

–**fontItalic**
: Sets the font to italics.

–**nosplash**
: Does not display the splash screen before the Perspectives main window is displayed.

–**h**
: Displays usage information on the options available for the command.

**Note:** Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

## Operands

None.

## Description

Use this command to launch the Event Perspective. The Event Perspective is a graphic vehicle for managing events and event definitions in the system.

This Perspective allows the user to interact with the Event Manager and the Problem Manager. The user can create event definitions, register or unregister event definitions, and delete event definitions. A properties dialog box is provided for viewing and editing the condition, the resource identifier, and other attributes of an event definition. The properties dialog box also provides users with the capability for creating new conditions. Event definitions are viewed and manipulated within system partition boundaries.

By default, the Event Definition pane displays all the event definitions with the Kerberos principal of the user within the current system partition. The user can use the filter function provided on the pane to expand or further confine the event definitions being displayed.

When the command is invoked, preferences which define the look and layout of the spevent window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

## Files

The Users Preferences are read from and saved to **$HOME/.spevent(User Profile Name)**.

The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.spevent(System Profile name)**.

## Restrictions

Any user can run the **spevent** command. Many actions require root privilege to run.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Prerequisite Information

For information on using the Event Perspective, see the online help and the “Using SP Perspectives” chapter in the *PSSP: Administration Guide*.

For information on the Event Management services, refer to “The Event Management Subsystem” and “Using the Problem Management Subsystem” chapters in *PSSP: Administration Guide*.

## Location

**/usr/lpp/ssp/bin/spevent**

## Related Information

You can also access the Event Perspective by using the SP Perspectives Launch Pad. The **perspectives** command invokes the Launch Pad. Other Perspectives may be launched by invoking the following commands: **sphardware**, **spsyspar**, **spvsd**, and **spperfmon**.

## Examples

1. To invoke the spevent window, enter:

   ```
   spevent
   ```

2. To bring up the Event Perspective using a specific user profile, enter:

   ```
   spevent -userProfile myProfile
   ```

## spframe

### Purpose

**spframe** – Enters configuration data for a frame or series of frames and, optionally, set up the initial System Data Repository (SDR).

### Syntax

**spframe** [–**p SP**] [–**r yes | no**] *start_frame frame_count starting_tty_port*

**spframe** –**p SAMI** –**n** {*starting_switch_port*} [–**s** {*s1tty*}] [–**r yes | no**] *start_frame frame_count starting_tty_port*

| **spframe**
| For use with SP-controlled Netfinity servers;
| **-p SLIM** [**-r yes | no**] *start_frame frame_count starting_tty_port*

### Flags

–**r no | yes** Indicates whether you want to initialize the System Data Repository. If this is the last or only time you are invoking this command during installation, specify –**r yes**. If –**r yes** is specified, the **/spdata/sys1/spmon/hmacls file** has the default entries created.

The default is –**r no**.

–**n** *starting_switch_port*
Indicates the switch port number to which the node in the non-SP frame is connected. If a frame count greater than one has been specified, this field indicates the starting switch port number in a range of contiguous switch port numbers. (Note: "switch port number" is also known as "switch node number.")

–**s** *s1tty* Indicates the s1 tty port for the single non-SP frame. If an s1 tty
| is specified, the frame count must be one. If a hardware
| protocol (–**p**) of SAMI is specified, the –**s** flag must be
| specified. The *s1tty* must be specified as a fully qualified special device file name (such as */dev/tty2*).

–**p** *hardware_protocol*
Indicates the hardware protocol of the node(s) within the frame. For example, the hardware protocol of the SP-attached server
| is SAMI. The hardware protocol for the SP-controlled Netfinity
| server is SLIM. The default is –**p SP**.

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive. |
| *frame_count* | Specifies the number of frames to be used for this operation. Specify a value between 1 and 64 inclusive. |
| *starting_tty_port* | Specifies the device special file name of the tty port to be assigned to the first frame on this operation. tty ports for subsequent frames are assigned by incrementing the tty |

number for each frame. Specify the full path of a valid tty device special file.

## Description

Execute this command during installation of the SP to identify the tty ports to which your frames are connected. If the tty special files for your frames are consecutively numbered, run this command once during installation, specifying –**r yes**. If you have “holes” in your tty special file numbering scheme, run this command once for each block of ttys you want to assign, specifying –**r yes** only on the last invocation of the command.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

You can use the System Management Interface Tool (SMIT) to run the **spframe** command. To use SMIT, enter:

```
smit enter_data
```

and select the SP Frame Information or non-SP Frame Information options.

**Note:** This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

## Location

**/usr/lpp/ssp/bin/spframe**

## Examples

1. The following example enters information for four frames (frame 1 to frame 4) and indicates that frame 1 is connected to **/dev/tty1**, frame 2 to **/dev/tty2**, and so on. The System Data Repository is to be initialized with this invocation of **spframe**.

   ```
   spframe -r yes 1 4 /dev/tty1
   ```

2.

| For SP-controlled Netfinity servers the following example enters information for
| 1 frame connected to **/dev/tty5** and reinitializes the SDR.

```
|   spframe -r yes -p SLIM 5 1 /dev/tty5
```

## spget_syspar

### Purpose

**spget_syspar** – Returns the IP address or name of the current system partition.

### Syntax

**spget_syspar** [–**n**] [–**d**]

### Flags

| | |
|---|---|
| –**n** | Returns a host name instead of an address. |
| –**d** | Returns the name or IP address of the default system partition rather than the current system partition. |

### Operands

None.

### Description

Use this command to display to standard output the IP address (in dotted decimal format) of the current or default system partition. The current system partition indicates the system partition to which System Data Repository (SDR) client requests are directed. The result is displayed in dotted decimal format unless –**n** is specified.

### Restrictions

The –**d** flag will not work if the command is not issued on a control workstation or SP node.

### Location

**/usr/lpp/ssp/bin/spget_syspar**

### Examples

1. To display the IP address associated with the current system partition, enter:

   ```
   spget_syspar
   ```

   You should receive output similar to the following:

   ```
   129.40.127.122
   ```

2. To display the name (host name alias of the control workstation) of the current system partition, enter:

   ```
   spget_syspar -n
   ```

   You should receive output similar to the following:

   ```
   k47sp1
   ```

## spgetdesc

### Purpose

**spgetdesc** – Obtains the description information from the nodes specified and, optionally, enters it into the SDR.

### Syntax

**spgetdesc** [–**h**] [–**u**] [–**c**] [–**f**] {–**a** **|** –**l** *node_list*}

### Flags

| | |
|---|---|
| –**h** | Displays help information for this command (syntax message). If the command is issued with the **-h** flag then the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the **-h** flag). |
| –**u** | Updates the description attribute in the SDR with the description information found. |
| –**c** | Outputs the description information in a colon delimited format. The output will be of the form: `"node_number:hostname:description"` for all nodes that successfully obtained the description information. |
| –**f** | Forces the command to obtain description information from the specified nodes regardless of the host responds value. |

One of the following flags must be specified:

| | |
|---|---|
| –**a** | Obtains description information from all nodes found in the SDR. |
| –**l** *node_list* | Indicates by *node_list* the SP nodes to obtain the description information from. The *node_list* is a comma-separated list of node numbers. |

### Operands

None.

### Description

This command will obtain the description information from the nodes specified and, optionally, update the description attribute in the SDR Node class. Unless the -f flag is specified, the node's host_responds will be checked before it will attempt to dsh to the nodes and obtain their description information. This command requires that the user be authenticated to Kerberos using the **k4init** command. This command is primarily intended as a migration tool for obtaining description information from existing nodes. The description information will be obtained from new nodes when they are installed or customized.

## Standard Output

The model information that is obtained will be printed to standard output as well as placed in the SDR.

## Standard Error

This command writes error messages (as necessary) to standard error. Errors will be printed if any attempt to get the description information on a node is unsuccessful.

If the command does not run successfully it terminates with an error message and a nonzero return code. Messages will inform the user of the cause of the error. For a terminal error, no description information will be obtained. For a nonterminal error, no description information will be obtained for the node that had the error.

## Exit Values

**0** Successful completion

**1** A nonterminal error occurred for 1 or more nodes. Processing continued for any remaining nodes.

**2** A terminal error occurred and all processing was stopped.

## Security

Since this command uses **dsh**, it requires that the user issuing the command obtain a valid Kerberos ticket.

## Implementation Specifics

This command is part of the IBM Parallel System Support Program (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/spgetdesc**

## Related Information

SP Commands: **dsh, k4init, SDRChangeAttrValues, SDRGetObjects**

AIX Commands: **uname**

## Examples

To obtain description information for all existing nodes enter:

```
spgetdesc -a
```

To obtain description information for nodes 3 & 4 and update the SDR, enter:

```
spgetdesc -ul 3,4
```

To obtain description information from all nodes in a colon delimited format, enter:

```
spgetdesc -ac
```

## sphardware

### Purpose

**sphardware** – Directly launches the Hardware Perspective graphical user interface (GUI).

### Syntax

**sphardware** [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**]
[–**backgroundColor** *colorName*]
[–**foregroundColor** *colorName*] [–**fontFamily** *name*]
[–**fontSize** *size*] [–**fontBold**] [–**fontItalic**] [–**nosplash**] [–**h**]

### Flags

–**userProfile** *name*
Upon initialization, loads the specified user profile. If a user profile named "Profile" exists in the user's home directory, it will be loaded by default if the –**userProfile** flag is not specified.

–**systemProfile** *name*
Upon initialization, loads the specified system profile instead of the default system profile. The default system profile is named "Profile."

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**foregroundColor** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6–30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

–**nosplash** Does not display the splash screen before the Perspectives main window is displayed.

–**h** Displays usage information on the options available for the command.

**Note:** Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

## Operands

None.

## Description

Use this command to launch the Hardware Perspective. From the Hardware Perspective, the user can monitor and manipulate objects within the SP system. The SP objects included in this Perspective are the control workstation, SP system and system partitions, nodes, node groups, and frames and switches.

By default, the Hardware Perspective will display the CWS, System and Syspars pane and the Nodes pane. The CWS, System and Syspars pane contains all system partitions and indicates the current system partition. The Node pane contains all the nodes in the current system partition.

The Node Groups and Frame and Switches panes can be added to the Hardware Perspective by using the Add Pane tool bar icon. Panes can be deleted from the window by using the Delete Pane tool bar icon.

When the command is invoked, preferences which define the look and layout of the sphardware window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

## Files

The Users Preferences are read from and saved to **$HOME/.sphardware(User Profile Name)**.

The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.sphardware(System Profile name)**.

## Restrictions

Any user can run the **sphardware** command. Many actions require root privilege.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Prerequisite Information

For information on using the Hardware Perspective, see the online help and the “Using SP Perspectives” chapter in the *PSSP: Administration Guide*.

## Location

**/usr/lpp/ssp/bin/sphardware**

## Related Information

The Hardware Perspective can also be accessed by using the SP Perspectives Launch Pad. The **perspectives** command invokes the Launch Pad. Other Perspectives windows can be launched by invoking the following commands: **spevent**, **sphardware**, **spperfmon**, **spsyspar**, and **spvsd**.

## Examples

1. To invoke the **sphardware** window, enter:

   ```
   sphardware
   ```

2. To force **sphardware** to display text in chartreuse, regardless of what is set in the preference files, enter:

   ```
   sphardware -foregroundColor chartreuse
   ```

3. To start the **sphardware** window in the background, enter:

   ```
   sphardware &
   ```

## sphostnam

### Purpose

**sphostnam** – Enters host name configuration data for a node or series of nodes in the System Data Repository.

### Syntax

**sphostnam** [–**a** *adapter_name*] [–**f** {**long** | **short**}]
{*start_frame start_slot node_count* **|** –**N** *node_group* **|** –**l** *node_list*}

### Flags

| | |
|---|---|
| –**a** | Indicates the adapter to be used to derive the host name for the nodes specified. If –**a** is not specified, the default is **en0**. |
| –**f** | Specifies which form of the host name is to be used. Specify **long** if you want the host name to be the fully qualified host name and **short** if you want to use the short form of the host name. If –**f** is not specified, the default is **long**. |
| –**N** *node_group* | Specifies a node group to be used for this operation. This node group must be bound to the current system partition. |
| –**l** *node_list* | Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.) |

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive. |
| *start_slot* | Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive. **Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition. |
| *node_count* | Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted the operation continues in the next sequential frame. Specify a value between 1 and 1024 inclusive. **Note:** The *node_count* is considered to be within the current system partition. |

## Description

Execute this command during installation of the SP to specify the adapter type to be used to determine the host name by which your nodes are known. You can also use this command to indicate whether you want the long (fully qualified) or short form of a host name to be used.

You can use the System Management Interface Tool (SMIT) to run the **sphostnam** command. To use SMIT, enter:

```
smit node_data
```

and select the Hostname Information option.

**Notes:**

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. Any changes made will not take effect on the nodes until they are customized.

## Location

**/usr/lpp/ssp/bin/sphostnam**

## Examples

The following example selects the **css0** adapter for the host name for a system with two frames and 32 nodes. The long form of the host name is to be used.

```
sphostnam -a css0 1 1 32
```

## sphrdwrad

### Purpose

**sphrdwrad** – Obtains hardware Ethernet addresses for SP nodes so they can be written to the System Data Repository.

### Syntax

**sphrdwrad** {*start_frame start_slot* {*node_count* | **rest**} |
–**N** *node_group* | –**l** *node_list*}

### Flags

| | |
|---|---|
| **rest** | Indicates that, beginning with the node determined by *start_frame* and *start_slot*, all the rest of the nodes should be used for this operation. |
| –**N** *node_group* | Specifies a node group to be used for this operation. This node group must be bound to the current system partition. |
| –**l** *node_list* | Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.) |

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of the first node to be used for this operation. Specify a value between 1 and 64 inclusive. |
| *start_slot* | Specifies the slot number of the first node to be used for this operation. Specify a value between 1 and 16 inclusive. **Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition. |
| *node_count* | Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted the operation continues in the next sequential frame. Specify a value between 1 and 1024 inclusive. **Note:** The *node_count* is considered to be within the current system partition. |

### Description

Execute this command only at installation or when adding new frames or nodes. The **spframe** command must be run **before** this command so that frame information is already in the System Data Repository.

If you know your hardware Ethernet addresses, you can speed this process by putting the addresses in **/etc/bootptab.info**, as follows:

Create a file named **/etc/bootptab.info** (if it does not already exist), listing your SP nodes by node number (or frame, slot) followed by a blank and the hardware Ethernet address. The first token represents the node and the second token represents the hardware address. The file should look similar to this:

```
17 02608C2E48D9
19 02608C2D6712
21 02608C2E49A4
23 02608C2E48E2
```

If you do not know your hardware Ethernet addresses, use the **sphrdwrad** command to find them.

**Notes:**

1. The nodes should be physically powered on (but logically powered off) when you run this command.
2. The LEDs change values while this command is running.
3. You should not have a tty open to any of the nodes to be used for this command.
4. If the addresses are not found in **/etc/bootptab.info**, the **sphrdwrad** command takes a few minutes to run, and the addresses are obtained from the nodes in parallel.
5. Any nodes specified will be powered off to acquire the Ethernet addresses. The nodes remain in the powered off state, even after the addresses are received.
6. If you are adding a node, only the new node needs to be specified because any selected nodes will be powered off.
7. To avoid possible file system damage, you should always shut down a node cleanly before powering it off. You can do this by using the **cshutdown** command or by using the SHUTDOWN/POWER OFF option in the System Monitor Graphical User Interface.

You can use the System Management Interface Tool (SMIT) to run the **sphrdwrad** command. To use SMIT, enter:

```
smit enter_data
```

and select the Get Hardware Ethernet Addresses option.

**Note:** This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

| **Note:** If you are using a **bootptab.info** file, you must only place nodes in the
| current partition in the file. If you have multiple partitions, you must update
| the **bootptab.info** file and run **sphrdwrad** for each partition.

## Location

**/usr/lpp/ssp/bin/sphrdwrad**

## Examples

To obtain Ethernet addresses for a new frame containing 8 nodes (4 wide nodes and 4 thin nodes), enter:

```
sphrdwrad 2 1 8
```

You should receive output similar to the following:

```
Acquiring hardware Ethernet address for node 17
Acquiring hardware Ethernet address for node 19
Acquiring hardware Ethernet address for node 21
Acquiring hardware Ethernet address for node 23
Acquiring hardware Ethernet address for node 25
Acquiring hardware Ethernet address for node 26
Acquiring hardware Ethernet address for node 27
Acquiring hardware Ethernet address for node 28
Hardware ethernet address for node 17 is 02608C2D481C
Hardware ethernet address for node 19 is 02608C2D78DF
Hardware ethernet address for node 21 is 02608C2D93B3
Hardware ethernet address for node 23 is 02608C2D8C3C
Hardware ethernet address for node 25 is 10005AFA22B9
Hardware ethernet address for node 26 is 10005AFA230A
Hardware ethernet address for node 27 is 10005AFA2229
Hardware ethernet address for node 28 is 10005AFA2210
```

## spled

### Purpose

**spled** – Displays SP node LCD or LED information in a graphical user interface.

### Syntax

**spled** [–**G**] [–**n** *title*] [–**p**] [–**f** *fontname*]
[**-r** *pollrate*] [**-b** *background color*] [**-l**] [**-h**]

### Flags

| | |
|---|---|
| –**G** | Displays LED or LCD information for SP nodes in all system partitions. |
| –**n** *title* | Sets the window title. |
| –**p** | Output the process id to stdout. |
| –**f** *fontname* | Sets the font. |
| –**r** *pollrate* | Sets the time increment to poll for LCD or LED updates. The default is to poll every 5 seconds. IBM suggests that you not change this value. |
| –**b** *background color* | Sets the background color. |
| –**l** | Changes the "Window" menu bar item labeled "Exit" to "Close." |
| –**h** | Displays usage information on the options available for the command. |

### Operands

None.

### Description

**spled** is an X Windows based application that displays the three digit seven segment light emitting diode (LED) or the two line by 16 character liquid crystal display (LCD) information found on the front of an SP node.

The **spled** application contains a menubar at the top of the window which contains one button, "Window." This button has one menu item which by default is labeled "Exit." Selecting this button will close the **spled** application.

Below the menubar is an area which by default contains a graphical representation of the frames and the SP nodes in those frames for the current system partition. If the **-G** flag is specified then all of the SP nodes in the system will be shown in their respective frames.

The size of each node is represented by the number of slots it uses in the frame. A thin node occupies one slot; a wide node occupies two slots; and a high node occupies four slots.

Pressing mouse button one inside a frame will display the slot number of each node in that frame. Pressing mouse button two inside a frame will display the node number of each node in that frame.

For nodes that have an LCD:

- For high nodes, all 2 lines by 16 characters will be shown.
- For wide nodes, the first line of 16 characters will be shown.
- For thin nodes, up to the first 8 characters of the first line will be shown.

Thin and wide nodes that contain an LCD may display "..." after some characters in the LCD. This indicates there is more LCD data available than can be shown. Pressing mouse button three in a frame containing nodes with an LCD displaying "..." will open a window displaying all of the LCD information available for those nodes.

**Note:** A node in the process of having the microcode on its supervisor card updated will not be displayed in the window.

## Security

The user must have **hardmon** vfop or monitor permission to use this command.

## Location

**/usr/lpp/ssp/bin/spled**

## Examples

1. To start **spled** and set the window title to "My System" enter:

   ```
   spled -n "My System"
   ```

2. To start **spled** and display all of the nodes in a multiple partition system, enter:

   ```
   spled -G
   ```

## splm

### Purpose

**splm** – Views, gathers, archives, or collects log and system data.

### Syntax

| | |
|---|---|
| **splm** | [–**a archive**] [–**cny**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**t** *table*] |
| **splm** | [–**a archive**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**n**] [–**r**] [–**t** *table*] [–**y**] |
| **splm** | [–**a check**] [–**h**] [–**n**] [–**t** *table*] |
| **splm** | [–**a gather**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**k** *type*] [–**l cfs**] [–**o** *loc*] [–**r**] [–**s**] [–**t** *table*] [–**y**] |
| **splm** | [–**a service**] [–**cny**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**p** *ipts*] [–**t** *table*] |
| **splm** | [–**a service**] [–**d** *dir*] [–**f** *fanout*] [–**h**] [–**n**] [–**r**] [–**t** *table*] [–**y**] |
| **splm** | [–**a view**] [–**h**] [–**n**] [–**t** *table*] |

### Flags

The **splm** command requires the –**a** flag and an argument to select a function to execute. It also requires a log table that contains records specifying target nodes and files or commands.

| | |
|---|---|
| –**a** *action* | Specifies the function to perform: **archive**, **check**, **gather**, **service**, or **view**. |
| –**c** | Creates a compressed tar file. For the **archive** function, this will be a tar starting the –**d** *dir* flag. For **service** collections, **/usr/sbin/snap** –**c** –**d** *dir* will be called to create the tar file. The tar file will be named **node.tar.Z**. |
| –**d** *dir* | Specifies the path where the **archive** or **service** collection will be stored on each node. The **archive** default is **/var/adm/archives**. The **service** default is **/tmp**. |
| –**f** *fanout* | Sets the maximum fanout value which determines how many nodes will execute in parallel. The default is 32. Any number between 1–32 can be used. |
| –**h** | Displays usage information. |
| –**k** *type* | For the **gather** function only, this flag indicates whether a **service** collection or **archive** is being collected. |
| –**l cfs** | Specifies the path on the local node where the **archive** or **service** collections should be gathered. |
| –**n** | Ignores the node designation in the input table and executes all entries on the local node only. |
| –**o** *loc* | Specifies the device or mail location where to direct tar files. |
| –**p** *opts* | Accepts a string of characters representing option parameters for calling snap collection. Each character relates to a category of system data to be collected. Valid characters are: a, A, D, f, g, G, k, l, n, p, s, S, t. |

| | |
|---|---|
| **–r** | Removes **archive** or **service** on each node. Exclusive with **–s** flag. |
| **–s** | Staggers collection to a mail location or device. |
| **–t** | Specifies the input table of nodes and commands. |
| **–y** | Appends the *yymmdd* timestamp subdirectory to the per node directory. |

## Operands

None.

## Description

Use this command to execute a number of log management functions on a single host or a set of hosts in parallel. The **splm** functions are driven by a log table file that contains the target node designations and associated files, and the commands to execute. The table format follows:

```
# Comment line
target nodes:  file or command
```

The target node portion of a table stanza is a comma-delimited list with no blanks. The values in the list are interpreted as:

1. If the value begins with a slash (/), it is a file containing a list of node names, one per line.
2. If the value is an exclamation point (!), it refers to the local host.
3. Any string not matching 1 or 2, is interpreted as a node name.

The **–n** flag ignores the target node portion of the table and only executes on the local node. The file or command portion of the stanza specifies either a command to execute that displays information to standard output, or a file that will be archived, collected, or viewed. File specification can take advantage of Tcl file globbing (similar to csh file globbing). If the file or command portion of the stanza refers to a command to be executed, it must be followed by a redirection and a path or file name. The information generated by the command will be redirected to the path or file name under the **–d** top level directory. Use > or >> following the command to redirect the output. The view option ignores the file or command destinations and displays the file's contents or command output to the executing node.

1. To specify the local node, nodes listed in the **/tmp/nodelist** file and node k47n10, and archive or collect errpt output from those nodes to the **errpt.output** file under the top level directory, enter:

   ```
   !,/tmp/nodelist,k47n10:  /bin/errpt -a > errpt.output
   ```

2. To archive or collect **/etc/filesystems** file to a subdirectory on nodes k47n10 and k47n15, enter:

   ```
   k47n10,k47n15: /etc/filesystems  etc/filesystems
   ```

   This copies the file to the **/etc** subdirectory under the **–d** top level directory.

   **Note:** The **–d** top level directory is always appended with a subdirectory named **arch**_*table_name* for archives, or **srvc**_*table_name* for service collections.

### splm Functions

**Archive:** The **archive** function copies files and redirects command output as specified in the input table to the top level directory on each node. The −**c** flag then creates a compressed tar file of the data named **/topdirectory/***node_name***.tar.Z.** The −**r** flag removes an archive by removing all files starting from the top level directory down.

**Check:** The **check** function can be used to check a table for errors.

**Gather:** The **gather** function moves **archive** or **service** tar files to a central location on the executing node. The −**r** option removes the **archive** or **service** collection on each remote node only after the tar file was successfully copied to the central location. If the *node***.tar.Z** file is not found, the **gather** function will attempt to create one. Gathered tar files can be mailed or copied to a tape or disk device using the −**o** flag. If mailed, the files are first uuencoded. The −**l** flag specifies the file system on the local node where the tar files are to be gathered. The −**l** flag must be specified if the −**s** stagger flag is not used. The **gather** function makes two passes, if necessary. On the first pass, it allows each node to take up an equal amount of the central file system. If any nodes encounter errors, the **gather** function retries those nodes, one at a time, until the file system is full or all the nodes are copied. If **gather** is unsuccessful on any node, but a *node***.tar.Z** file exists for that node in the central location, it is moved to *node***.tar.Z.old**, and not sent to the output location. The −**s** stagger flag forces the fanout to 1, gathers the tar files one at a time, attempts to send the tar to the output location, then removes it from the local node. The −**r** flag cannot be used with −**s**. The default central location directory for stagger is **/tmp**.

**Service:** The **service** function first calls the AIX **snap** command to gather system data to the top level directory if the −**p** flag is used. The **snap** command creates a set of subdirectories based on the −**p** arguments. The additional data defined in the table data is then collected under the “other” subdirectory created by **snap**. If the −**p** flag is not used, the data will still be collected under the “other” subdirectory. If the −**c** flag is used, **splm** uses the **snap** −**c** command to create the **tar.Z** file. The −**r** flag can be used to remove service collections. **splm** calls **snap** −**r** which removes the tar file and all files under each **snap** subdirectory.

**View:** The **view** function displays the output of the command or contents of file entries in the input table to the local host.

## Files

**/etc/splm.allow**
Restricts table commands that can be executed.

**/etc/logmgt.acl**
Acl file for **archive**, **gather**, and **service** functions.

**/spdata/sys1/logtables/***
Contains sample tables for service collections.

## Security

The **archive**, **gather**, and **service** functions of **splm** require that you have a Kerberos principal defined in the **/etc/logmgt.acl file**. The command then runs as root on all target nodes. The viewing function requires that you be Kerberos authenticated to a valid user ID on the nodes that you are executing on. The server switches IDs from root to your authenticated ID before executing.

## Location

**/usr/lpp/ssp/bin/splm**

## Related Information

AIX commands: **compress snap**, **tar**, **uuencode**

The *PSSP: Administration Guide*

## Examples

1. To create an **archive** based on the entries in the **/etc/tables/logs.tab** table and to create a compressed tar file and have the **archive** under directory **/var/adm/archives/arch_logs.tab**, enter:

   ```
   splm -a archive -c -d /var/adm/archives -t /etc/tables/archive.tab
   ```

2. To create a **service** collection of entries in the **/spdata/sys1/logtables/amd.tab** table and have **snap** include general system information, enter:

   ```
   splm -a service -c -t /spdata/sys1/logtables/amd.tab -p g
   ```

3. To **gather** the **service** collections in Example 2, remove the collection on each node, and copy the gathered data to tape device rmt0, enter:

   ```
   splm -a gather -k service -t /spdata/sys1/logtables/amd.tab \
   -l /tmp/amdproblem -o /dev/rmt0 -r
   ```

## splogd Daemon

### Purpose

**splogd** – Reports error logging, writes state changes, and calls user exits.

### Syntax

**splogd** [–**d**] [–**b**] [–**f** *file_name*]

### Flags

–**d**
Turns debugging on.

–**b**
Starts the daemon in the background from the command line.

–**f** *file_name*
Names an input file to use to define what logging is to be done and what user exits should be called. The default file is **/spdata/sys1/spmon/hwevents**.

### Operands

None.

### Description

The SP logging daemon has the following functions:

**error logging**
Reports SP hardware errors to both the syslog and the AIX error log.

**state change logging**
Writes SP hardware state changes to a file.

**user exits**
Calls a user exit when a state change occurs.

The **hwevents** file contains state change actions that are to be performed by the **splogd** logging daemon. The fields are:

**frame**
Specifies a frame number (1–*n*) or * for all frames.

**slot**
Specifies the following:

- A number from 0–17
- One of:
  - NODES_ONLY (addresses 1 through 16)
  - SWITCH (address 17)
  - FRAME (address 0)
  - * (all addresses)
  - NODES_AND_SWITCH (addresses 1–17)
  - FRAME_AND_NODES (addresses 0–16)
  - FRAME_AND_SWITCH (addresses 0 and 17)

**variable**
Specifies a hardware variable (for example, nodePower, temp, LED7SegA).

**operator**
Specifies how to compare the value. Acceptable values are: **=**, <, >, and **!=**.

**value** Specifies the value of the variable to match with the operator wildcard (*), or a partial match with the wildcard at the end (23*).

**time** Specifies if the function should be called at startup, when the state changes, or both times. Valid options are **startup**, **change**, or **both**.

**function** Specifies the program to call when an event occurs.

There are two special keywords for function. If function is SP_ERROR_LOG, error logging is performed provided that syslog is set up and AIX error logging is set up to perform SP logging. Refer to the **setup_logd** command for details.

If function is SP_STATE_LOG, these state changes that meet the statement's criteria are logged to **/var/adm/SPlogs/spmon/splogd.state_changes.***timestamp*.

**Note:** To close the current **state_changes.***timestamp* and open a new one, send a SIGHUP signal to **splogd**. For example,

```
kill -HUP {splogd pid}
```

**User Exit Arguments**

When a user exit is called by **splogd**, the following arguments are passed:

1. A **c** or **s** depending on whether this call is for a change of state or to provide the startup values for the variables being monitored.
2. For each variable being reported, the following arguments are passed:
   a. Frame number.
   b. Node number.
   c. Variable name. Refer to the "System Monitor Variables, Display Types, and Attributes Appendix" of *PSSP: Administration Guide* for a list of variables.
   d. Value of the variables. Boolean variables are expressed as TRUE or FALSE, integers as decimal strings, and floating-point values as floating-point strings.

**Starting and Stopping the splogd Daemon**

The **splogd** daemon is under System Resource Controller (SRC) control. It uses the signal method of communication in SRC. The **splogd** daemon is a single subsystem and not associated with any SRC group. The subsystem name is **splogd**. To start the **splogd** daemon, use the **startsrc –s splogd** command. This starts the daemon with the default arguments and SRC options. The **splogd** daemon is setup to be respawnable and be the only instance of the **splogd** daemon running on a particular node or control workstation. Do **not** start the **splogd** daemon from the command line without using the **startsrc** command to start it.

To stop the **splogd** daemon, use the **stopsrc –s splogd** command. This stops the daemon and does not allow it to respawn.

To display the status of the **splogd** daemon, use the **lssrc –s splogd** command.

If the default startup arguments need to be changed, use the **chssys** command to change the startup arguments or the SRC options. Refer to *AIX Version 4*

*Commands Reference* and *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for more information about daemons under SRC control and how to modify daemon arguments when under SRC.

To view the current SRC options and daemon arguments, use the **odmget −q 'subsysname=splogd' SRCsubsys** command.

## Files

**/spdata/sys1/spmon/hwevents**
File that describes what logging is performed and what user exits are called.

**/var/adm/SPlogs/spmon/splogd.state_changes.***timestamp*
File where state changes are recorded.

## Location

**/usr/lpp/ssp/bin/splogd**

## Related Information

Command: **setup_logd**

The “System Monitor Variables, Display Types, and Attributes Appendix” in *PSSP: Administration Guide*

## Examples

1. To start the **splogd** daemon, enter:

   ```
   startsrc -s splogd
   ```

2. To stop the **splogd** daemon, enter:

   ```
   stopsrc -s splogd
   ```

3. To display the status of the **splogd** daemon, enter:

   ```
   lssrc -s splogd
   ```

4. To display the status of all the daemons under SRC control, enter:

   ```
   lssrc -a
   ```

5. To display the current SRC options and daemon arguments for the **splogd** daemon, enter:

   ```
   odmget -q 'subsysname=splogd' SRCsubsys
   ```

## splst_syspars

### Purpose

**splst_syspars** – Returns the list of defined system partitions.

### Syntax

**splst_syspars** [–**n**]

### Flags

**–n** Returns a list of host names instead of addresses.

### Operands

None.

### Description

This command returns the list of the system partitions. The system partition names are in dotted decimal format unless –**n** is specified.

### Location

**/usr/lpp/ssp/bin/splst_syspars**

### Examples

1. To display the IP addresses associated with all the defined system partitions on the SP, enter:

   ```
   splst_syspars
   ```

   You should receive output similar to the following:

   ```
   129.40.127.122
   129.40.127.47
   ```

2. To display the names of all the defined system partitions on the SP, enter:

   ```
   splst_syspars -n
   ```

   You should receive output similar to the following:

   ```
   k47sp1
   k47s
   ```

## splst_versions

### Purpose

**splst_versions** – Returns information about the PSSP code version installed on nodes in the SP system.

### Syntax

**splst_versions** [–**G**] [–**l**] [–**e**] [–**n** *node_num*] [–**N** *node_group*] [–**t**] [–**h**]

### Flags

–**G** Causes the command to look at all system partitions rather than just the current system partition (but not the control workstation).

–**l** Returns the latest PSSP version for the nodes that are the target of the command.

–**e** Returns the earliest PSSP version for the nodes that are the target of the command.

–**n** *node_num*
Returns the PSSP code version for *node_num*. Use *node_num* 0 to specify the control workstation.

–**N** *node_group*
Returns a list of PSSP versions for *node_group*. If –**G** is supplied, a global node group is used. Otherwise, a partitioned-bound node group is used.

–**t** Returns the node number and PSSP version in two columns.

–**h** Displays usage information.

### Operands

None.

### Description

Use this command to return a list of PSSP code versions that are installed on the
| nodes in the current system partition. The PSSP version and release numbers, and
| modification level are included in the output. The fix level is not returned in the
| output. Node number 0 (zero) is considered the control workstation and is **not** evaluated as part of any system partition. The output is sorted in ascending order by version.

If the –**t** flag is omitted, there will be only one record for each version present. If the –**t** flag is used, there will be a record for each node.

### Location

**/usr/lpp/ssp/bin/splst_versions**

## Examples

1. To list each PSSP version represented in the current system partition, enter:

   ```
   prompt> splst_versions
   PSSP-2.4
   PSSP-3.1
   ```

2. To list each node in the system partition and its PSSP code version, enter:

   ```
   prompt> splst_versions -t
   1 PSSP-2.4
   5 PSSP-2.4
   6 PSSP-2.4
   9 PSSP-3.1
   ```

3. To list the earliest and latest PSSP code versions in a system partition, enter:

   ```
   prompt> splst_versions -l -e
   PSSP-2.4                      /* this case has mixed partitions */
   PSSP-3.1
   ```

   The following will be the output if only PSSP-2.4 exists in the system partition:

   ```
   prompt> splst_versions -l -e
   PSSP-2.4                      /* this case has only 2.4 in partition */
   ```

## splstadapters

### Purpose

**splstadapters** – Use this command to list information about adapters to standard output.

### Syntax

**splstadapters** [–**h**] [–**x**] [–**G**] [–**d** *delimiter*] [–**p** *str*] [–**s** *attr*]
[–**t** {**standard | dependent**}] [*attr==value* ...] [*attr* ...]

### Flags

**–h** Displays usage information.

**–G** Removes system partition boundaries for this invocation. This flag causes the command to consider all nodes regardless of system partition.

**–x** Inhibits the output of the header record.

**–d** *delimiter* Forces the delimiter between tokens to be *delimiter*, where *delimiter* is any string value. If this flag is used, only one copy of the delimiter is used between tokens, even if the delimiter is a blank.

**–p** *str* Prints the *str* string in place of an attribute that does not apply to the object being output. The default is to print two double quotes ("").

**–s** *attr* Sorts the output by the value of the attribute *attr*.

**–t** Restricts the query to a specific node type. The node type can be one of the following:

**standard** Indicates that only adapters relating to SP nodes (nodes in a frame/slot) will be considered for output.

If the **–t** flag is not specified, the default is to consider adapters relating to both **standard** and **dependent** nodes for output.

**dependent** Indicates that only adapters on dependent nodes will be considered for output.

### Operands

*attr==value* Specifies certain adapter objects to be returned. The *attr* token must be a valid attribute of one of the adapter classes in the System Data Repository (SDR) (**Adapter** or **DependentAdapter**). If *attr* exists in both adapter classes, objects from each class will be considered unless that class is excluded with the **–t** flag. The token *value* is the value of *attr* that objects must have to be returned by this invocation of the command.

*attr* Specifies the attributes to be returned as output of the command. It does not limit the adapter objects that are considered for output. If an *attr* argument is not specified, the **node_number** and **adapter_type** attributes are returned.

## Description

Use this command to get configuration information about any adapter from the SDR. For a complete list of adapter attributes, see the Adapter and DependentAdapter classes in “The System Data Repository” appendix in *PSSP: Administration Guide*.

Not all of the attributes are applicable to each type of adapter.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit list_extadapters
```

## Environment Variables

The environment variable SP_NAME is used (if set) to direct this command to a system partition. The default is to use the default system partition when on the control workstation and the partition of the node when on a node.

## Standard Output

This command writes informational messages to standard output.

## Standard Error

This command writes all error messages to standard error.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **nonzero** | Indicates that an error occurred. |

## Implementation Specifics

You must specify an attribute in order for it to be displayed in the output. The attribute in the sort option (–**s** flag) and the attributes in the form *attr==value* must be repeated in order for them to be displayed.

## Location

**/usr/lpp/ssp/bin/splstadapters**

## Examples

1. To list the **node_number** and **adapter_type** attributes for all adapter objects in the current system partition, enter:

   ```
   splstadapters
   ```

   You should receive output similar to the following:

   ```
   node_number adapter_type
   1 en0
   1 css0
   5 en0
   5 css0
   ```

2. To list the **netmask** attribute of SP adapters along with their node numbers and have the output sorted by node number, enter:

   ```
   splstadapters -t standard -s node_number node_number netmask
   ```

   You should receive output similar to the following:

   ```
   1 255.255.255.192
   3 255.255.255.192
   ```

3. To list the “css0” adapters in the system, regardless of system partition, enter:

   ```
   splstadapters -G adapter_type==css0
   ```

   You should receive output similar to the following:

   ```
   node_number adapter_type
   1 css0
   5 css0
   7 css0
   9 css0
   19 css0
   23 css0
   ```

## splstdata

## Purpose

**splstdata** – Displays configuration data from the System Data Repository (SDR) or system information for each node.

## Syntax

| **splstdata** {–**A** │ –**n** │ –**s** │ –**t** │–**b** │ –**a** │ –**u** │ –**h** │ –**i** │ –**d** │ –**x**} [–**G**]
[{*start_frame start_slot* {*node_count* **|** **rest**} **|**
–**N** *node_group* **|** –**l** *node_list*}]

**OR**

**splstdata** {–**e** │ –**f** │ –**p**}

## Flags

One of the following flags must be specified with each invocation of **splstdata**:

–**A** Displays the following SDR accounting data:

node_number
host_name
acct_class_id
acct_enable
acct_excluse_enable
acct_job_charge

–**n** Displays the following SDR node data:

node_number
frame_number
slot_number
slots_used
host_name
rel_host
default_route
processor_type
| processors_installed

–**s** Displays the following SDR node and dependent node switch data in three lists:

node_number
host_name
switch_node_number
switch_protocol
switch_number
switch_chip_number
switch_port_number

| switch_number (common field)

frame_number
slot_number
switch_partition_number

switch_type
| clock_input

switch_partition_number (common field)

topology_filename
primary_name
arp_enabled
switch_node_nos._used

| **–t** Displays the following data from the ProcessorExtensionNode class.

| node_number
| frame_number
| slot_number
| host_name
| short_comment
| long_comment

**–b**

node_number
host_name
hdw.enet.addr
boot_server
bootp_response
install_disk
last_inst_image
last_inst_time
next_inst_image
lppsource_name
pssp_version

**–a** Displays the following SDR LAN data only for nodes in the current system partition:

node_number
adapter_type
netaddr
netmask
host_name
type
rate

**–u** Displays the following SDR **/usr** data:

node_number
host_name
usr_server_id
usr_gateway_id
usr_client_adapter
has_usr_clients

**–h** Displays hardware data for each node, as provided by the **lscfg** command.

**–i** Displays network adapter data for each node, as provided by the **netstat –n** command.

**–d** Displays file system data for each node, as provided by the **df** command.

**–x** Displays node expansion data for each node. If no nodes are specified, the command displays data for all node expansion units in the current system partition. The data is displayed in two lists:

expansion_number
frame_number
slot_number
slots
associated_node#
remote_port

node_number
expansion_list

The following flags are optional:

**–G** Allows the specification of nodes to include one or more nodes outside of the current system partition.

**–N** *node_group*
Specifies a node group to be used for this operation. If **–G** is supplied, a global node group is used. Otherwise, a partitioned-bound node group is used.

**–l** *node_list* Specifies a list of nodes to be used for this operation. Either specify a comma-delimited list of node numbers, or a file containing one line of data which is a comma-delimited list of node numbers. The file can also contain comment lines (preceded by a #) and lines that are all white space. If you use the *node_list* field, do not use the *start_frame*, *start_slot*, or *node_count* fields. (This is lowercase **l**, as in **l**ist.)

**–e** Displays SP object attributes and their values from the SDR.

**–f** Displays the following SDR frame data:

frame_number
tty
frame_type

**–p** Lists all information for the currently-applied system partition configuration (all active system partitions on the system). This includes the list of system partitions, plus information about each system partition.

## Operands

*start_frame* Frame number of first node to be used for this operation. Specify a value between 1 and 64 inclusive. If *start_frame*, *start_slot* and *node_count* are not specified, the default is **1 1 rest**. If *start_frame* is specified, *start_slot* and *node_count* must also be specified.

*start_slot* Slot number of first node to be used for this operation. Specify a value between 1 and 16 inclusive.

**Note:** *The start_frame* and *start_slot* must resolve to a node in the current system partition.

*node_count* Number of nodes to be used for this operation. Node information is provided for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame. Specify a value between 1 and 1024 inclusive. If **rest** is specified, all the nodes from *start_frame start_slot* to the end of your system are used.

**Note:** The *node_count* is considered to be within the current system partition.

## Description

You can use the System Management Interface Tool (SMIT) to run the **splstdata** command. To use SMIT, enter:

```
smit list_data
```

and select the System Data Repository option for the information you want to see. To see system information for each node, enter:

```
smit config_data
```

and select the option for the information you want.

**Note:** This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.

## Location

**/usr/lpp/ssp/bin/splstdata**

## Examples

1. To display SDR node data for all the nodes in the current SP system partition, enter:

   ```
   splstdata -n
   ```

   You should receive output similar to the following:

```
                  List Node Configuration Information

node frame slot
 #     #    #   slots initial_hostname reliable_hostname default_route
    processor_type processors_installed description
-----------------------------------------------------------------------
 1     1    1     2   k5n01.ppd.pok.ib  k5n01.ppd.pok.ib  129.40.85.126
         UP                  1       135_MHz_P2SC_Wide
 3     1    3     2   k5n03.ppd.pok.ib  k5n03.ppd.pok.ib  129.40.85.126
         UP                  1       77_MHz_PWR2_Wide-2
 5     1    5     1   k5n05.ppd.pok.ib  k5n05.ppd.pok.ib  129.40.85.126
         UP                  1       160_MHz_P2SC_Thin
 6     1    6     1   k5n06.ppd.pok.ib  k5n06.ppd.pok.ib  129.40.85.126
         UP                  1       120_MHz_P2SC_Thin
 7     1    7     1   k5n07.ppd.pok.ib  k5n07.ppd.pok.ib  129.40.85.126
         UP                  1       66_MHz_PWR2_Thin-2
 8     1    8     1   k5n08.ppd.pok.ib  k5n08.ppd.pok.ib  129.40.85.126
         UP                  1       66_MHz_PWR2_Thin
 9     1    9     4   k5n09.ppd.pok.ib  k5n09.ppd.pok.ib  129.40.85.126
         MP                  1       112_MHz_SMP_High
13     1   13     4   k5n13.ppd.pok.ib  k5n13.ppd.pok.ib  129.40.85.126
         MP                  1       200_MHz_SMP_High
⋮
```

2. To display SDR boot/install data for the first four nodes in the first frame, enter:

   ```
   splstdata -b 1 1 4
   ```

   You should receive output similar to the following:

```
                  List Node Boot/Install Information

node#     hostname  hdw_enet_addr srvr     response        install_disk
    last_install_image last_install_time next_install_image lppsource_name
                pssp_ver         selected_vg
------------------------------------------------------------------------
 1  k55n01.ppd.pok.i   0004AC493851    0       disk              hdisk0
                 default Sun_Jul_12_14:05:19           default     aix43k
               PSSP-3.1              rootvg
 3  k55n05.ppd.pok.i   0004AC4944FE    0       disk              hdisk0
                 default Sun_Jul_12_14:06:38           default     aix43k
               PSSP-3.1              rootvg
 5  k55n09.ppd.pok.i   0004AC493FA1    0       disk              hdisk0
                 default Sun_Jul_12_14:05:38           default     aix43k
               PSSP-3.1              rootvg
 6  k55n13.ppd.pok.i   0004AC493B66    0       disk              hdisk0
                 default Sun_Jul_12_14:05:27           default     aix43k
               PSSP-3.1              rootvg
```

3. To list system partition information, enter:

   ```
   splstdata -p
   ```

   You should receive output similar to the following:

```
List System Partition Information

System Partitions:
------------------
k55sp1
k55s

Syspar: k55sp1
-------------------------------------------------------------------------
syspar_name       k55sp1
ip_address        129.40.62.179
install_image     default
syspar_dir        /spdata/sys1/syspar_configs/2nsb0isb/config.4_28/
                  layout.8/syspar.1
code_version      PSSP-2.2
haem_cdb_version  852558375,501538560,0

Syspar: k55s
-------------------------------------------------------------------------
syspar_name       k55s
ip_address        129.40.62.55
install_image     default
syspar_dir        /spdata/sys1/syspar_configs/2nsb0isb/config.4_28/
                  layout.8/syspar.2
code_version      PSSP-2.2
haem_cdb_version  852558451,833611264,0
auth_install      k4:std
auth_root_rcmd    k4:std
auth_methods      k4:std
```

## splstnodes

### Purpose

**splstnodes** – Lists to standard output information about nodes.

### Syntax

**splstnodes** [–**h**] [–**x**] [–**G**] [–**d** *delimiter*] [–**p** *str*] [–**s** *attr*]
[–**t** {**standard | dependent**}] [–**N** *node_group*]
[*attr==value* ...] [*attr* ...]

### Flags

| | |
|---|---|
| –**h** | Displays usage information. |
| –**G** | Removes system partition boundaries for this invocation. This causes the command to consider all nodes regardless of system partition. |
| –**x** | Inhibits the output of the header record. |
| –**d** *delimiter* | Forces the delimiter between tokens to be *delimiter*, where *delimiter* is any string value. If this flag is used, only one copy of the delimiter is used between tokens, even if the delimiter is a blank. |
| –**p** *str* | Prints the *str* string in place of an attribute that does not apply to the object being output. The default is to print two double quotes ("). |
| –**s** *attr* | Sorts the output by the value of the *attr* attribute. |
| –**t** | Restricts the query to a specific node type. The node type can be one of the following: |

| | |
|---|---|
| **standard** | Only SP nodes (nodes in a frame/slot) are considered for output. |
| **dependent** | Only dependent nodes are considered for output. |

If the –**t** flag is not specified, the default is to consider both **standard** and **dependent** nodes for output.

–**N** *node_group*
Restricts the query to only the nodes in the node group specified by *node_group*. If *node_group* is a system node group, the –**G** flag is implied.

### Operands

| | |
|---|---|
| *attr==value* | Used to specify certain node objects to be returned. The *attr* token must be a valid attribute of one of the node classes in the System Data Repository (SDR) (**Node** or **DependentNode**). If *attr* exists in both node classes, the objects from each class will be considered, unless that class is excluded with the –**t** flag. The token *value* is the value of *attr* that objects must have to be returned by this invocation of the command. |
| *attr* | Used to specify the attributes to be returned as output of the command. It does not limit the node objects that are considered for output. If an *attr* argument is not specified, the *node_number* attribute is returned. |

## Description

Use this command to get configuration information about any node from the SDR. For a complete list of node attributes, see the Node and DependentNode classes in “The System Data Repository” appendix in *PSSP: Administration Guide*.

Not all of the attributes are applicable to each type of node.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit list_extnodes
```

## Environment Variables

The environment variable SP_NAME is used (if set) to direct this command to a system partition. The default is to use the default system partition when on the control workstation and the system partition of the node when on a node.

## Standard Output

This command writes informational messages to standard output.

## Standard Error

This command writes all error messages to standard error.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **nonzero** | Indicates that an error occurred. |

## Implementation Specifics

You must specify an attribute in order for it to be displayed in the output. The attribute in the sort option (**–s** flag) and the attributes in the form *attr==value* must be repeated in order for them to be displayed.

## Location

**/usr/lpp/ssp/bin/splstnodes**

## Examples

1. To list the node number of all wide node objects in the current system partition, enter:

   ```
   splstnodes slots_used==2
   ```

   You should receive results in the following output, if four wide nodes are in the system partition in slots 1, 3, 5, and 7:

   ```
   node_number
   1
   3
   5
   7
   ```

2. To list the **reliable_hostname** attribute of SP nodes along with their node numbers and have the output sorted by node number, enter:

```
splstnodes -t standard -s node_number node_number reliable_hostname
```

You should receive results in the following output:

```
node_number reliable_hostname
1 k22n1.ppd.pok.ibm.com
3 k22n3.ppd.pok.ibm.com
5 k22n5.ppd.pok.ibm.com
7 k22n7.ppd.pok.ibm.com
```

3. To list the "wide nodes" in the system, regardless of system partition, enter:

   ```
   splstnodes -G slots_used==2
   ```

   You should receive results in output similar to the following:

   ```
   node_number
   1
   3
   5
   7
   19
   21
   23
   ```

4. To list the **snmp_community_name** attribute of any SP dependent nodes along with their node numbers, enter:

   ```
   splstnodes -t dependent node_number snmp_community_name
   ```

   If you have dependent nodes, you should receive output similar to the following:

   ```
   node_number snmp_community_name
             8 mycomm
             2 yourcomm
   ```

## splsuser

### Purpose

**splsuser** – Lists the attributes of an SP user account.

### Syntax

**splsuser** [–**c** | –**f**] *name*

### Flags

| | |
|---|---|
| –**c** | Displays the attributes for the user in colon-separated records. |
| –**f** | Displays the attributes for the user in stanza format. |

### Operands

| | |
|---|---|
| *name* | Name of the user account you want to view. |

### Description

You can only list the information for one SP user at a time. Unlike the AIX **lsuser** command, the **ALL** option and the –**a** flag are not supported for this command.

If you specify this command with no flags, the information of the user appears in a sequential display of attribute and values.

You can use the System Management Interface Tool (SMIT) to run the **splsuser** command. To use SMIT, enter:

```
smit spusers
```

and select the Change/Show Characteristics of a User option.

### Location

**/usr/lpp/ssp/bin/splsuser**

### Examples

1. To display the attributes of the user account **rob** in a colon-separated list, enter:

   ```
   splsuser -c rob
   ```

   You should receive output similar to the following:

   ```
   #name:id:pgrp:groups:home:shell:gecos:login
   rob:16416:1::/u/rob on k46s.hpssl.kgn.ibm.com:/bin/ksh::true
   ```

2. To display the attributes of the user account **rob** in stanza format, enter:

   ```
   splsuser -f rob
   ```

   You should receive output similar to the following:

```
rob:
        id=16416
        pgrp=1
        groups=
        home=/u/rob on k46s.hpssl.kgn.ibm.com
        shell=/bin/ksh
        gecos=
        login=true
```

## spmgrd Daemon

### Purpose

**spmgrd** – Automates management and configuration required for extension nodes.

### Syntax

**spmgrd** [**–s | –l**] **–f** *filename* **–m** [*size* | **0**]

### Flags

**–s** Specifies that short tracing is to be turned on as part of initialization processing. This is used to capture trace events that occur during bring-up. The trace file is located in **/var/adm/SPlogs/spmgr/spmgrd.log** unless overridden. Short tracing does not include informational messages nor the content of messages exchanged with Simple Network Management Protocol (SNMP) agents. The default is for tracing to be turned off.

**–l** Specifies that long tracing is to be turned on as part of initialization processing. This is used to capture trace events that occur during bring-up. The trace file is located in **/var/adm/SPlogs/spmgr/spmgrd.log** unless overridden. Long tracing includes informational messages and the content of messages exchanged with SNMP agents in addition to error messages. The default is for tracing to be turned off.

**–f** Specifies the name of a trace file. The default trace file name is **/var/adm/SPlogs/spmgr/spmgrd.log**.

**–m** Specifies the maximum trace file size in bytes. When 0 is specified, there is no maximum size. The default is 0.

### Operands

None.

### Description

The **spmgrd** daemon is part of the **spmgr** subsystem and can only be controlled using the System Resource Controller (SRC). This daemon acts as an SNMP Manager monitoring SNMP *trap* messages received from SNMP agents supporting dependent nodes. A *trap* message may contain state information about an attached dependent node or may request the transfer of configuration data for a dependent node supported by the sending SNMP agent. When requested by a *trap* message, **spmgrd** transfers configuration data to the requesting SNMP agent. The data transfer is in the form of an SNMP *set-request* message containing the SNMP object instantiations representing configuration aspects of the dependent node and the values to which the aspects are to be set. When a *trap* message indicates that a dependent node previously fenced from the switch network with the "automatic rejoin" option is now active, **spmgrd** will automatically issue an **Eunfence** command to trigger the appropriate unfence processing.

The **spmgrd** daemon keeps log messages in a default file or in a file specified by the *filename* variable if the **–f** flag is specified. When the size of the log file exceeds an optional user-specified maximum log file size, the **spmgrd** daemon rotates the log file by moving the old log file to another file as follows:

```
*   LogFile.3 is deleted.

*   LogFile.2 is moved to LogFile.3.

*   LogFile.1 is moved to LogFile.2.

*   LogFile.0 is moved to LogFile.1.

*   LogFile is moved to LogFile.0.

*   LogFile continues in LogFile.
```

The **spmgrd** daemon only runs on the control workstation.

The **spmgrd** daemon is controlled using the SRC. The **spmgrd** daemon is a member of the **spmgr** system group. The **spmgrd** daemon is enabled by default and can be manipulated by SRC commands. Use the following SRC commands to manipulate the **spmgrd** daemon:

**startsrc** Starts a subsystem, group of subsystems, or a subserver. The **spmgrd** daemon is part of the **spmgr** subsystem. Issuing the **startsrc -s spmgr** command causes the **spmgrd** daemon to be activated. Any **spmgrd** switches must be set using the **startsrc** command **–a** switch and must be enclosed within double quotes (").

**stopsrc** Stops a subsystem, group of subsystems, or a subserver.

**traceson** Enables tracing of a subsystem, group of subsystems, or a subserver. Long tracing is specified by using the **–l** switch.

**tracesoff** Disables tracing of a subsystem, group of subsystems, or a subserver.

**lssrc** Gets the status of a subsystem, group of subsystems, or a subserver. When the long form of the subsystem's status is requested, information provided by the **spmgr** subsystem includes:

- Trace information:
  - Whether tracing is on or off
  - The name of the trace file
  - The mode of tracing (long or short)
  - The trace file size limit (if any)
- Summary information about the traps received
- The content and the completion status of the **snmpinfo** commands issued by the **spmgrd** daemon. The **snmpinfo** commands are issued internally to request the SNMP agent managing a dependent node to change the dependent node's administrative state (this occurs whenever an **enadmin** command is entered by a user). The **snmpinfo** command is also issued internally to send configuration data for a dependent node to the SNMP agent managing it (the request for configuration data is received in the form of a *trap* message).

## Files

**/var/adm/SPlogs/spmgr/spmgrd.log**
Is the **spmgrd** trace file.

**/usr/lpp/spp/config/spmgrd/ibmSPDepNode.my**
Is the Management Information Base (MIB) file containing the **ibmSPDepNode** object group that defines dependent node configuration objects.

**/usr/lpp/ssp/config/spmgrd/ibmSPDepNode.defs**
Is the compiled **ibmSPDepNode.my** object file.

**/etc/services** Contains a line, **spmgrd-trap**, that defines the User Datagram Protocol (UDP) port number over which trap messages are received from an SNMP agent supporting dependent nodes.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP) **ssp.spmgr** file set.

## Location

**/usr/lpp/ssp/bin/spmgrd**

## Related Information

Commands: **enadmin**, **lssrc**, **startsrc**, **stopsrc**, **tracesoff**, **traceson**

## Examples

1. To start the **spmgr** subsystem (for example, the **spmgrd** daemon with short tracing on), enter:

   ```
   startsrc -s spmgr -a'-s'
   ```

2. Use the **traceson** and **tracesoff** commands to control tracing after the **spmgrd** daemon is started.

   ```
   traceson -ls spmgr (to turn on long tracing)

   tracesoff -s spmgr (to stop tracing)
   ```

3. To stop the **spmgr** subsystem, enter:

   ```
   stopsrc -s spmgr
   ```

4. To obtain the trace status and a list of **snmpinfo** commands issued by the **spmgr** subsystem since it was last activated, enter:

   ```
   lssrc -ls spmgr
   ```

## spmkuser

### Purpose

**spmkuser** – Adds a new user account to the SP system.

### Syntax

**spmkuser** [*attribute*=*value* ... ] *name*

### Flags

None.

### Operands

| | |
|---|---|
| *attribute*=*value* | Pairs of the supported attributes and values as follows. |
| *name* | User login name. This name must follow the same rules enforced by the AIX **mkuser** command. |

**Supported Attributes and Values**

**id** ID of the user specified by the *name* operand.

**pgrp** Principle group of the user specified by the *name* operand.

**gecos** General information about the user.

**groups** The secondary groups to which the user specified by the *name* operand belongs.

**home** Host name of the file server where the home directory resides and the full path name of the directory. You can:

- Specify a host and directory in the format *host:path*, just specify the directory and have the host default to a value set in the SMIT site environment panel or the **spsitenv** command. If this value has not been set, then the host will default instead to the local machine name, if you do not specify host explicitly.

**login** Indicates whether the user specified by the *name* operand can log in to the system with the **login** command. This option does not change the **/etc/security/user** file. Instead, it alters the user password field in **/etc/security/passwd**.

**shell** Program run for the user specified by the *name* operand at the session initiation.

### Description

The –**a** flag is not supported. Except for **home**, the rules for the supported attributes and values correspond to those enforced by the AIX **mkuser** command.

All other attribute and value pairs are not supported.

The standard administrative AIX privileges do not apply to the SP users.

This command generates a random password for the user and stores it in **/usr/lpp/ssp/config/admin/newpass.log**. The root user has read and write permission to this file. It is the administrators responsibility to communicate this password to the new user and periodically delete the contents of this file.

You can use the System Management Interface Tool (SMIT) to run the **spmkuser** command. To use SMIT, enter:

```
smit spusers
```

and select the Add a User option.

**Note:** The home directory must be in an exported file system before you can run this command.

## Location

**/usr/lpp/ssp/bin/spmkuser**

## Examples

**Note**

The following examples assume that the SP automounter function is configured and the following defaults are specified:

| | |
|---|---|
| **spsitenv** command or SMIT panel | HOMEDIR_SERVER="svr1" |
| | HOMEDIR_PATH="/home/filesvr1" |
| **spmkuser.default** file | In the *user* stanza: group=staff groups=staff prog=/bin/ksh |

1. To create a user account for **baker** using the defaults specified in the **spmkuser.default** file and the home directory specified in the SMIT site environment panel or **spsitenv** command:

   ```
   spmkuser baker
   ```

2. To create a user account for **charlie** with a UID of 1234, a home directory of **/u/charlie** that is physically located at **/home/charlie** on **hostx**, the **staff** primary group and the **dev**, the **test** secondary groups, and the **/bin/ksh** default shell:

   ```
   spmkuser id=1234 groups=dev,test home=hostx:/home/charlie
   ```

## spmirrorvg

### Purpose

**spmirrorvg** – Initiates mirroring on a node or set of nodes.

### Syntax

| **spmirrorvg** [–**f**] {*start_frame start_slot node_count* **|** –**l** *node_list*}

### Flags

| | |
|---|---|
| –**f** | Forces the physical volume to be added to the specified volume group unless it is a member of another volume group in the Device Configuration Database or of a volume group that is active. If the volume group is varied off, and –**f** is specified, the physical volume will be added to the volume group. |
| –**l** *node_list* | Specifies a list of nodes for this operation. This list can be a single numeric node number, or a list of numeric node numbers separated by commas. |

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of the first node to be used for this operation. |
| *start_slot* | Specifies the slot number of the first node to be used for this operation. **Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition. |
| *node_count* | Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame. **Note:** The *node_count* is considered to be within the current system partition. |

### Description

The **spmirrorvg** command uses information found in the Volume_Group object to initiate mirroring on a node. If the number of requested copies is already achieved on the node, **spmirrorvg** exits. If the number of copies has not been achieved, the **spmirrorvg** command uses the *pv_list* attribute to extend the named volume group. After extending the volume group, **spmirrorvg** calls the AIX **mirrorvg** command to make two or three copies of the volume group. If not enough disks are specified in the *pv_list* to maintain strictness this command will exit. Quorum will be set according to the quorum attribute in the Volume_Group object. If the state of quorum changes as a result of running the **spmirrorvg** command, a message will be displayed that the node needs to be rebooted. The AIX commands **bosboot** and **bootlist** are run by **spmirrorvg**. **bosboot** updates the bootable image and **bootlist** sets the node bootlist to reflect multiple bootable logical volumes.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that a recoverable error occurred, some changes may have succeeded. |
| **2** | Indicates that an irrecoverable error occurred and no changes were made. |

## Security

A user must run as "root" and have a valid Kerberos ticket.

## Files

Log file created on node which contains AIX error messages if an error occurs during mirroring: **/var/adm/SPlogs/sysman/mirror.out**

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/spmirrorvg**

## Related Information

Commands: **spchvgobj**, **spunmirrorvg**

## Examples

1. To initiate mirroring on node 1 for the rootvg volume group enter:

   ```
   spmirrorvg  -l 1
   ```

2. To initiate mirroring on a list of nodes enter:

   ```
   spmirrorvg  -l 1,2,3
   ```

## spmkvgobj

### Purpose

**spmkvgobj** – Creates a new root volume group for a node or series of nodes in the System Data Repository (SDR).

### Syntax

| **spmkvgobj** –**r** *volume_group_name* [–**h** *pv_list*] [–**i** *install_image*]
| [–**p** *code_version*] [–**v** *lppsource_name*] [–**n** *boot_server*]
| [–**c** { **1** | **2** | **3**}] [–**q** {**true** | **false**}]
| {*start_frame start_slot node_count* **|** –**l** *node_list*}

### Flags

–**r** *volume_group*
Specifies the root volume group name to create.

–**h** *pv_list* Indicates the physical volumes to be used for installation for the volume group specified. The root volume group is defined on the disks indicated, and all data on the disks is destroyed. The physical volumes may be specified as logical names (for example, **hdisk0**), hardware location (for example, **00-00-00-0,0**), or connwhere (for example, **ssar//012345678912345**). If multiple physical volumes are specified, separate them by commas for logical names and by colons for hardware location and connwhere. The default value is **hdisk0**.

**Note:** IBM strongly suggests that you use the hardware location or connwhere format. It ensures that you install on the intended disk by targeting a specific disk at a specific location. The logical naming of physical volumes may change depending on hardware installed or possible hardware errors. This is especially true when there are external drives present, as the manner in which the device names are defined may not be obvious.

–**i** *install_image*
Specifies the name of the install image to be used for the volume group when they are next network-installed. Specify a file in the **/spdata/sys1/install/images** directory on the control workstation. At installation, the value for each volume group's install image name is default, which means that the default install image name for the system partition or the system is used for each node. The default install image name is found in the Syspar or the SP object in that order. The default value is "default".

–**p** *code_version*
Sets the volume group's code version. Use this to indicate the PSSP level to install on the node. The *code_version* value you choose must match the directory name that the PSSP installation files are placed under in the **/spdata/sys1/install/pssplpp** directory during installation. See the *PSSP: Installation and Migration Guide* for more details. The default value is "PSSP-3.1".

–**v** *lppsource_name*
Sets the volume group's lppsource name. Use this to indicate the AIX level to install on the node. The *lppsource_name* value you choose must

match the directory name you choose to place the lppsource files under in the **/spdata/sys1/install** directory during installation. See the *PSSP: Installation and Migration Guide* for more details. The default value is "default".

–**n** *boot_server*
: Identifies the boot/install server for the volume groups you have specified. The boot/install server is identifies by a node number. Node number 0 represents the control workstation. The value of the boot/install server at installation depends on how many frames are in your system. In a single frame system, the control workstation (node 0) is the default server for each node. In a multiple frame system, the default server for the first node in each frame is the control workstation, and the default server for the rest of the nodes in a frame is the first node in that frame. The default value for a new root volume group is "0".

–**c copies**
: Specifies the number of mirrors to create for the volume group. To enable mirroring, set this to 2 or 3. Setting this to 1 disables mirroring. When enabling mirroring, be sure that there are enough physical volumes to contain all the copies of the volume group. Each copy must have at least 1 physical volume. The default value is "1".

**-q true | false**
: Specifies whether quorum should be enabled. If quorum is enabled, a voting scheme will be used to determine if the number of physical volumes that are up is enough to maintain quorum. If quorum is lost, the entire volume group will be taken off line to preserve data integrity. If quorum is disabled, the volume group will remain on line as long as there is at least 1 running physical volume. The default value is "true".

**-l** *node_list*
: Specifies a list of nodes to be used for this operation. Specify a comma-delimited list of nodes numbers. If you use the **-l** flag, do not use the *start_frame*, *start_slot*, or *node_count* operands.

## Operands

*start_frame*
: Specifies the frame number of the first node to be used for this operation.

*start_slot*
: Specifies the slot number of the first node to be used for this operation.

  **Note:** The *node_count* is considered to be within the current system partition.

*node_count*
: Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame.

  **Note:** The *node_count* is considered to be within the current system partition.

## Description

Use the **spmkvgobj** command to create a new root volume group on a node or group of nodes in the System Data Repository (SDR). When this command is run and the SDR is changed, **setup_server** must be run on the affected boot/install servers and affected nodes may need to be customized or installed to apply the changes. Certain volume group information such as mirroring and the *pv_list* may be updated using the **spmirrorvg** or **spunmirrorvg** commands.

## Exit Values

| | |
|---|---|
| **0** | Indicates the command has run successfully. |
| **1** | A non-critical error occurred, some creations may have succeeded. |
| **2** | Indicates an irrecoverable error occurred and no changes were made. |

## Security

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets. If you do not have a ticket-granting-ticket, you must run **k4init**.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/spmkvgobj**

## Related Information

Commands: **spbootins**, **spchvgobj**, **spmirrorvg**, **sprmvgobj**, **spunmirrorvg**

## Examples

1. To create a new root volume group using 2 SSA physical volumes and taking all other default values on nodes 2 and 3, enter:

   ```
   spmkvgobj -r rootvg2 -h \
   ssar//567464736372821:ssar//67464736372821 -l 2,3
   ```

2. To create a new root volume group using 3 SCSI physical volumes, enabling mirroring and specifying a mksysb image on node 7, enter:

   ```
   spmkvgobj -r rootvg2 -h 00-00-00-0,0:00-00-00-1,0:00-00-00-2,0
         -c 2 -q false -i bos.obj.ssp.432 -l 7
   ```

## spmon

### Purpose

**spmon** – Operates the system controls and monitors system activity.

### Syntax

**spmon** [–**query** [–**Monitor**] [–**long**] | –**connect** *host_name* |
–**Global** | –**help** | –**key** {**normal** | **secure** | **service**} | –**Key**|
–**Led** | –**power** {**on** | **off**} |
–**reset** | –**mux** {**i** | **1** | **2** |**3**} |
–**open**| –**diagnostics**]
[[–**target**] *target_value*... ]

### Flags

All **spmon** commands require a –**target** parameter except those with –**diagnostics**, and –**help** parameters.

–**query** Queries the hardware variable specified as the target and returns the requested value. **query** is the default. If no other parameter is entered, a query is performed.

–**Monitor** Monitors the variables specified in the targets. If any of the specified variables change the state, the new state is written to standard output.

–**long** Applies only to –**query** and –**Monitor**. Returns the requested variables in fully qualified hierarchical format rather than the default format which is just the variable value.

–**connect** Connects to control workstation specified in *host_name* variable. Use this parameter with the –**key**, –**Key**, –**Led**, –**open**, –**power**, –**reset**, and –**mux** parameters.

–**Global** Allows targets that are outside of the current system partition. This parameter must be used for any query or command if specifying frames or switches.

–**help** Displays the usage information for the **spmon** command.

–**key** Choice of **normal | secure | service**. Changes the key mode switch position for the node specified as the target.

–**Key** Returns status of the key mode switch position for the node specified as the target.

–**Led** Displays the 3-digit display value.

–**power** Choice of **on | off**.

Turns the power on or off for the node, frame, or switch specified as the target. For example:

```
spmon -G -p off frame1
spmon -p off node16
spmon -G -p off frame2/switch
spmon -G -p off frame11/switch2
```

–**reset** Resets the node specified as the target.

–**mux** Choice of **i | 1 | 2 | 3**. Sets multiplexors that control the clocking of a switch to the value indicated. These values mean:

| | |
|---|---|
| **i** | Use internal oscillator (make this switch the master) |
| **1** | Use input 1 |
| **2** | Use input 2 |
| **3** | Use input 3 |

The **mux** setting must match the physical wiring of the switch clocks and requires a frame as its target. For a switch in node 17, use a frame as the target or frame/switch*N* for a switch in a switch-only frame.

–**open** Opens a **tty** connection to the node specified in the target flag. Press *Enter* to begin the session. Type **Ctrl-x** to close the connection. Refer to the **s1term** command for details.

–**diagnostics**
Performs the following diagnostics tests:

1. Checks if the server process is running
2. Tries to open a connection to the server
3. Queries the number of frames in system
4. If the –**G** parameter is specified, for each frame checks:
   - If the frame controller responding
   - If a switch is attached
   - The **mux** value
   - If the frame power supplies are ON or OFF
5. For each node in each frame, checks:
   - Node type
   - If power is on or off
   - hostResponds
   - switchResponds
   - The position of key switch
   - Environment problems
   - The values of the front panel LEDs.

   For each switch, checks:
   - Frame, slot
   - Node type
   - If power is on or off
   - Clock input
   - Environment problems

The tests are in dependent order. If any of these are unsuccessful, the subsequent tests do not run.

**[**–**target]** *target_value*
Specifies the target node, frame, variable, or attribute for the command as *target_value*.

The –**target** flag is optional. Any parameter without a flag is assumed to be the target. You can also have multiple target-flags (–**t**), which are optional.

## Operands

None.

## Description

Any unique abbreviation of flags and keywords is acceptable.

Specify *target_value* with the hierarchical format (or tree structure). The format is:

`/SP/frame/frameN/[nodeM|switchM]/variableX/value`

| | |
|---|---|
| *SP* | Is literally the string “SP”. |
| `frame` | Is the string **frame**. |
| `frameN` | Is frame1...frame*N* where *N* is the frame number in the SP system. |
| `nodeM` \| `switchM` | Is the node number or switch number within the specified frame. *M* is the slot number of that node or switch. When switch is specified without a number, it means switch 17. |
| *variableX* | Is a variable known to the SP System Monitor. Refer to the “System Monitor Variables, Display Types, and Attributes Appendix” of *PSSP: Administration Guide* for a list of variables. |
| *value* | Is literally the string “value.” |

You can use wildcards (*) to specify more than one target node or frame for the **query** command.

**Note:** Though they are not hardware variables, for compatibility with older systems, the variables *hostResponds* and *switchResponds* can be used as specific targets of the **spmon** command for both –**query** and –**Monitor** commands. However, the variable names must be entered explicitly. These two variables are not returned if the variable specified is a wildcard (*).

You can use aliases in place of fully qualified hierarchical target values. Aliases require less typing and may be more intuitive than the fully qualified targets. Leaving the leading slash (/) off the target indicates that it is an alias.

There are two formats for aliases:

- Format 1: `frameN/nodeM`

  | | |
  |---|---|
  | `frameN` | Is the target frame, where *N* is the frame number. |
  | `nodeM` | Is node1 to node16 within the specified frame. |

  You can include a variable and attribute after the alias.

- Format 2: `nodeM`

  | | |
  |---|---|
  | `nodeM` | Is node1 to node*M* where *M* is the node number of the node in the SDR node class. |

## Location

**/usr/lpp/ssp/bin/spmon**

## Related Information

The **sphardware** command launches a graphical user interface for monitoring and controlling an SP system.

## Examples

1. To query the key setting of node1 on frame1, enter:

   ```
   spmon -q -t /SP/frame/frame1/node1/keyModeSwitch/value
   0
   ```

2. To perform the same query using an alias (uses query flag default), enter:

   ```
   spmon node1/keyModeSwitch/value
   0
   ```

3. To query the LED settings of node1 on frame1, enter:

   ```
   spmon -L frame1/node1
   ```

   You should receive output similar to the following:

   ```
    ______________
   |              |
   |  _   _   _   |
   | |_| |_| |_|  | Frame 1, Node 1
   | |_| |_| |_|  |
   |______________|
   ```

4. To query the **mux** value on all switches in the system, enter:

   ```
   spmon -G -q -l frame*/switch*/mux/value
   /SP/frame/frame1/switch17/mux/value/0
   ```

5. To monitor the power LEDs on the nodes on frame1, enter:

   ```
   spmon -M frame1/node*/powerLED/value
   2
   1
   ```

6. To query the power LEDs on the nodes on frame1 and then monitor them and print the values in fully qualified hierarchical form, enter:

   ```
   spmon -M -q -l frame1/node*/powerLED/value
   /SP/frame/frame1/node1/powerLED/value/1
   /SP/frame/frame1/node3/powerLED/value/1
   /SP/frame/frame1/node5/powerLED/value/1
   /SP/frame/frame1/node7/powerLED/value/1
   /SP/frame/frame1/node9/powerLED/value/1
   /SP/frame/frame1/node10/powerLED/value/1
   /SP/frame/frame1/node11/powerLED/value/1
   /SP/frame/frame1/node12/powerLED/value/1
   /SP/frame/frame1/node13/powerLED/value/1
   /SP/frame/frame1/node14/powerLED/value/1
   /SP/frame/frame1/node15/powerLED/value/1
   /SP/frame/frame1/node16/powerLED/value/1
   /SP/frame/frame1/node1/powerLED/value/2
   /SP/frame/frame1/node1/powerLED/value/1
   ```

   **Note:** "node*" returns powerLED values on switches in slots 1—16 also.

7. To switch power off node3 on frame2, enter:

   ```
   spmon -p off frame2/node3
   ```

   If node3 on frame2 is outside the current system partition, enter:

```
spmon -G -p off frame2/node3
```

8. To switch power off node3 on frame2 using alias format2, enter:

   ```
   spmon -p off node19
   ```

9. To change key setting on node1 on frame1 to service, enter:

   ```
   spmon -k service node1
   ```

10. To switch power off frame1, (type 17 frame supervisor only), enter:

    ```
    spmon -G -p off frame1
    ```

11. To switch power off frame1, (SEPBU - type 18 frame supervisor), enter:

    ```
    spmon -G -p off frame1/A
    ```

12. To set the frame1 switch to be the master switch (use internal oscillator), enter:

    ```
    spmon -G -m i frame1
    ```

    or

    ```
    spmon -G -m i frame1/switch
    ```

13. To set frame 10, switch4 in a switch-only frame to be the master switch, enter:

    ```
    spmon -G -m i frame10/switch4
    ```

## spmon_ctest

### Purpose

**spmon_ctest** – Verifies that the System Monitor component is configured correctly.

### Syntax

**spmon_ctest** [–**l** *log_file*] [–**q**]

### Flags

| | |
|---|---|
| –**l** *log_file* | Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.) |
| –**q** | Specifies quiet mode; suppresses output to standard error. |

### Operands

None.

### Description

This command is designed to be run after installing the SP system to verify that the System Monitor is configured correctly. The test checks to make sure that the hardware is running, that it can be queried, and determines whether any node objects were created in the System Data Repository (SDR). The test also indicates whether the RS232 lines are connected properly.

A return code of zero indicates that the test completed as expected; otherwise it returns the number of errors. If you do not specify the –**q** flag, a message is displayed on standard output that indicates if the test was successful or not. In either case, the command returns 0 if successful, 1 if unsuccessful. If errors are detected, more detailed information is recorded in the log file. If you do not specify the –**l** flag, error messages are recorded in **/var/adm/SPlogs/spmon_ctest.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the System Monitor Configuration option.

You must run this test from a user who has monitor authority in **/spdata/sys1/spmon/hmacls**. The user must also have a nonexpired authentication ticket.

Refer to the "RS/6000 SP Files and Other Technical Information" section of *PSSP: Command and Technical Reference* for additional **Kerberos** information.

### Files

**/var/adm/SPlogs/spmon_ctest.log**
Default log file.

## Location

**/usr/lpp/ssp/bin/spmon_ctest**

## Related Information

Commands: **CSS_test**, **jm_install_verify**, **jm_verify**, **SDR_test**, **SYSMAN_test**, **spmon_itest**

## Examples

To verify installation of the SP System Monitor, saving error messages in **spmon.err** in the current working directory, enter:

```
spmon_ctest -l spmon.err
```

## spmon_itest

### Purpose

**spmon_itest** – Verifies that the System Monitor is installed and operational.

### Syntax

**spmon_itest** [–**l** *log_file*] [–**q**]

### Flags

| | |
|---|---|
| –**l** *log_file* | Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.) |
| –**q** | Specifies quiet mode; suppresses output to standard error. |

### Operands

None.

### Description

This command is designed to be run after installing the SP system to verify that the System Monitor is installed correctly.

A return code of zero indicates that the test completed as expected; otherwise it returns the number of errors. If you do not specify the –**q** flag, a message is displayed on standard output that indicates if the tests were successful or not. In either case, the command returns 0 if successful, 1 if unsuccessful. If errors are detected, more detailed information is recorded in the log file. If you do not specify the –**l** flag, error messages are recorded in **/var/adm/SPlogs/spmon_itest.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the System Monitor Installation option.

### Files

| | |
|---|---|
| | Path name of this command. |
| **/var/adm/SPlogs/spmon_itest.log** | Default log file. |

### Location

**/usr/lpp/ssp/bin/spmon_itest**

### Related Information

Commands: **CSS_test**, **jm_install_verify**, **jm_verify**, **SDR_test**, **SYSMAN_test**, **spmon_ctest**

## Examples

To verify installation of the SP System Monitor, saving error messages in **spmon.err** in the current working directory, enter:

```
spmon_itest -l spmon.err
```

## sppenode

### Purpose

**sppenode** – Enters configuration data for a processor extension node in the System Data Repository (SDR).

### Syntax

**sppenode** [–**h**]

**OR**

**sppenode** [–**n hostname**] [–**s short comment**] [–**l long comment**] *node_number*

### Flags

| | |
|---|---|
| –**h** | Displays usage information. |
| –**n hostname** | Hostname. |
| **–s short comment** | Allows you to enter a short description of the node. |
| **–l long comment** | Allows you to enter a long description of the node. |

### Operands

| | |
|---|---|
| *node_number* | Specifies the node number for this processor extension node. |

### Description

Execute this command during installation of the Netfinity server to add node specific data to the SDR for the ProcessorExtensionNode class. The *node_number* operand is required when entering this information.

### Standard Output

This command writes informational messages to standard out.

### Standard Error

This command writes all error messages to standard error.

Errors can result from causes that include:

- SDR access errors
- Insufficient user authorization for the command

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that an error occurred. The Netfinity node information was not updated. It is accompanied by one or more error messages that indicate the cause of the error. |

## Security

You must have root privilege, or be a member of the system group to run this command.

## Restrictions

This command may only be issued on the control workstation.

## Location

**/usr/lpp/ssp/bin/sppenode**

## Examples

Command for a processor extension node with a node number of 17:

```
sppenode-n SAPserver -s SAPserver -l Rack1FirstNode 17
```

## spperfmon

### Purpose

**spperfmon** – Directly launches the Performance Monitor Perspective graphical user interface (GUI).

### Syntax

**spperfmon** [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**]
[–**backgroundColor** *colorName*]
[–**foregroundColor** *colorName*] [–**fontFamily** *name*]
[–**fontSize** *size*] [–**fontBold**] [–**fontItalic**] [–**nosplash**] [–**h**]

### Flags

–**userProfile** *name*
Upon initialization, loads the specified user profile. If a user profile named "Profile" exists in the user's home directory, it will be loaded by default if the –**userProfile** flag is not specified.

–**systemProfile** *name*
Upon initialization, loads the specified system profile instead of the default system profile. The default system profile is named "Profile."

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**foregroundColor** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6—30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

–**nosplash** Does not display the splash screen before the Perspectives main window is displayed.

–**h** Displays usage information on the options available for the command.

**Note:** Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

## Operands

None.

## Description

Use this command to launch the SP Performance Monitor Perspective. This tool enables the user to monitor the performance of the SP in conjunction with other licensed products: Performance Toolbox for AIX (PTX), 5765-654.

From the Performance Monitor Perspective, you can perform most of the PTPE command set functions through point and click operations. For example, you can easily manipulate the PTPE monitoring hierarchy and save it to the System Data Repository (SDR).

The Performance Monitor Perspective window uses three panes to display SP system information:

1. *Hierarchy pane*: This shows the current monitoring hierarchy, displaying the central coordinator at the top, with data manager nodes below and reporter nodes at the bottom. By default, the monitoring hierarchy from the System Data Repository (SDR) is displayed when this Perspective is initialized.
2. *Syspar pane*: This shows how the SP is partitioned. The system partition selected in this pane is the one displayed in the Hierarchy and Nodes panes. If other partitions are defined by the SP, you can use this pane to select them.
3. *Nodes pane*: This shows the nodes in the SP system, organized by frame in the default display, but you can sort and filter them to suit your purposes.

When the command is invoked, preferences that define the look and layout of the SP Performance window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

## Files

- The Users Preferences are read from and saved to **$HOME/.spperfmon(User Profile Name)**.
- The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.spperfmon(System Profile name)**.

## Restrictions

Any user can run the **spperfmon** command. To get a read/write PTPE session requires root privilege and the user must be a member of the UNIX group 'perfmon'.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP) and the IBM Performance Toolbox Parallel Extensions for AIX separately priced feature.

## Prerequisite Information

For information on using **spperfmon** and SP Perspectives, see the online help and the “Using SP Perspectives” chapter in *PSSP: Administration Guide*.

## Location

**/usr/lpp/ssp/bin/spperfmon**

## Related Information

You can also access the Performance Monitor Perspective by using the SP Perspectives Launch Pad. The **perspectives** command invokes the Launch Pad. Other Perspectives can be launched by invoking the following commands: **spevent**, **sphardware**, **spsyspar**, and **spvsd**.

*IBM Performance Toolbox Parallel Extensions for AIX: Guide and Reference*

*IBM Performance Toolbox 1.2 and 2.1 for AIX: Guide and Reference*

## Examples

1. To invoke the spperfmon window, enter:

   ```
   spperfmon
   ```

2. To launch the SP Performance Monitor Perspective ignoring the preferences found in the system and user profile files, enter:

   ```
   spperfmon -noProfile
   ```

## sprestore_config

### Purpose

**sprestore_config** – Restores the system to a given system partitioning configuration as specified in the System Data Repository (SDR) which was previously archived.

### Syntax

**sprestore_config** *archive_file* [–**h**]

### Flags

| | |
|---|---|
| –**h** | Displays usage information. |

### Operands

| | |
|---|---|
| *archive_file* | Specifies the name of the archived SDR file to be restored. |

### Description

Use this command to restore the SDR from an archive file that was previously created with the **SDRArchive** command. In addition to restoring the SDR (using the **SDRRestore** command), the **sprestore_config** command also restores system partition-sensitive subsystems (for example, **hats**, **hb**, and **hr**) to their previous state. This command is most useful when recovering from an attempt to partition the SP (see the **spapply_config** command).

You can use the System Management Interface Tool (SMIT) to run the **sprestore_config** command. To use SMIT, enter:

```
smit syspar_restore
```

and enter (or select from a generated list) the name of the SDR archive from which to restore.

**Notes:**

1. This command should be run only on the control workstation.
2. Due to system partitioning changes, your SP_NAME environment variable may no longer be set to a valid system partition name. To get a list of valid system partition names, enter the **splst_syspars -n** command. Then verify that your SP_NAME environment variable is either unset or set to one of the system partition names in the list.

### Exit Values

| | |
|---|---|
| **0** | Indicates success. |
| **1** | Indicates that an error occurred while trying to restore the specified system partitioning configuration. |
| **2** | Indicates a usage error. |

## Related Information

Commands: **SDRArchive**, **SDRRestore**, **spapply_config**, **spcustomize_syspar**, **spdisplay_syspar**, **spverify_config**, **syspar_ctrl**

Files: **nodelist**, **topology**

## Examples

To restore the SDR and the system-partition sensitive subsystems (for example, **hats**, **hb**, and **hr**) from the archive **'backup.95110.1620'** which was previously created using the **SDRArchive** command, enter:

```
sprestore_config backup.95110.1620
```

## sprmuser

### Purpose

**sprmuser** – Removes a user account from the SP system.

### Syntax

**sprmuser** [–**i**] [–**p**] [–**r**] *name*

### Flags

| | |
|---|---|
| –**i** | Displays the current user information and enables interactive control. This allows you to quit before deleting the user account. |
| –**p** | Removes user password information from the **/etc/security/passwd** file. |
| –**r** | Removes the user's home directory specified in the **home** attribute. |
| *name* | Name of the user account you want to delete. |

### Operands

None.

### Description

The –**i** and –**r** options are unique to the SP system.

You can use the System Management Interface Tool (SMIT) to run the **sprmuser** command. To use SMIT, enter:

```
smit spusers
```

and select the Remove a User option.

### Location

**/usr/lpp/ssp/bin/sprmuser**

### Examples

1. To remove user account **charlie** without destroying the home directory, enter:

   ```
   sprmuser charlie
   ```

2. To remove user account **charlie**, remove any information about this user in the **/etc/security/passwd** file, and remove the home directory, enter:

   ```
   sprmuser -pr charlie
   ```

## sprmvgobj

### Purpose

**sprmvgobj** – Removes a root volume group for a node or series of nodes from the System Data Repository (SDR).

### Syntax

| **sprmvgobj** [–**r** *volume_group_name*]
| {*start_frame start_slot node_count* **|** –**l** *node_list*}

### Flags

| | |
|---|---|
| –**r** *volume_group* | Specifies the root volume group name to remove. |
| –**l** *node_list* | Specifies a list of nodes to be used for this operation. Specify a comma-delimited list of node numbers. If you use the **-l** flag, do not use the *start_frame*, *start_slot*, or *node_count* operands. |

### Operands

| | |
|---|---|
| *start_frame* | Specifies the frame number of the first node to be used for this operation. |
| *start_slot* | Specifies the slot number of the first node to be used for this operation.<br><br>**Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition. |
| *node_count* | Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame.<br><br>**Note:** The *node_count* is considered to be within the current system partition. |

### Description

The **sprmvgobj** command is used to remove a root volume group on a node or group of nodes from the System Data Repository (SDR).

### Exit Values

| | |
|---|---|
| **0** | Indicates the command has run successfully. |
| **2** | Indicates an irrecoverable error occurred and no changes were made. |

### Security

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets. If you do not have a ticket-granting-ticket, you must run **k4init**.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/sprmvgobj**

## Related Information

Commands: **spbootins**, **spchvgobj**, **sprmvgobj**

## Examples

1. To remove a root volume group on nodes 2 and 3, enter:

   ```
   sprmvgobj -r rootvg2 -l 2,3
   ```

## spseccfg

### Purpose

**spseccfg** – Displays host configuration information related to Security Services.

### Syntax

**spseccfg** [–h | *target_host*]

### Flags

| | |
|---|---|
| –**h** | Specifies that the command syntax is to be listed. |

### Operands

| | |
|---|---|
| *target_host* | The hostname or IP address of the host whose host configuration data is requested. When *target_host* is omitted, the information displayed is for the local host. |

### Description

The **spseccfg** command obtains the DCE-hostname that was assigned when DCE was configured on the target host and the system partition name. It prints each name to STDOUT as a separate line.

### Security

This command is available to all users.

### Standard Output

The first line of output is the DCE-hostname, if DCE is installed and configured, otherwise an empty line.

If the host is an SP node, the second line of output is the system partition name. If the host is the control workstation, It is the default system partition name. Otherwise, for an independent workstation, it is the (short) local hostname.

### Standard Error

Output consists of error messages, when the command cannot complete successfully.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that an error occurred. |

### Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP) (fileset ssp.clients).

## Prerequisite Information

The chapters on security in the *PSSP: Administration Guide*.

## Location

**/usr/lpp/ssp/bin/spseccfg**

## Examples

1. To display the configuration information for the local host, enter:

   ```
   $spseccfg
   hosts/sp3.xyz.com
   sp3
   ```

2. To display the configuration information for the host at address 120.14.89.10, enter:

   ```
   $spseccfg 120.14.89.10
   hosts/p16n9
   p16part3
   ```

## spsetauth

### Purpose

**spsetauth** – Sets the authentication methods to be installed on the control workstation and in the partition.

### Syntax

**spsetauth**
–**p** *partition name* [–**h**] –**d** k4 [std]

### Flags

–**p** *partition name*
Specifies the partition name for Syspar object.

–**h** Presents syntax to stdout.

–**d** Specifies authorization setup.

### Operands

Set of Authentication methods to be set in the Syspar object in the SDR (std is optional; k4 is required).

### Description

This command will only run on the control workstation. If the **-d** flag is set, it sets the set of authorization methods used for root access to remote commands. It writes this information to the SDR Syspar object for which an authorization method is to be used for root access to remote commands.

Values for either attribute are: k4 or std.

Currently k4 is required. If std is selected, it will be set last as an authorization method.

### Files

Log file created in **/var/adm/SPlogs/auth_inst/log**. This log file will be trimmed by using the **/usr/lpp/ssp/bin/cleanup.logs.ws** program.

### Exit Values

**0** Command completed successfully.

**1** Command was unsuccessful. Review any errors displayed on console or reported in the log file.

If anything but a return code of 0 occurs, the error must be corrected and the command executed again before proceeding to any further configuration or installation steps.

## Security

You will need root authorization to run this command.

## Location

**/usr/lpp/ssp/bin/spsetauth**

## Examples

1. To set partition "par_2" to have Kerberos 4 and Standard AIX as the set of authorization methods enter:

   ```
   /usr/lpp/ssp/bin/spsetauth -d -p par_2 k4 std
   ```

## spsitenv

### Purpose

**spsitenv** – Enters configuration parameters used by SP installation and system management scripts into the System Data Repository (SDR).

### Syntax

**spsitenv** [**acct_master** = *accounting_master*]
[**amd_config** = **true | false**]
[**cw_lppsource_name** = *lppsource_name*]
[**filecoll_config** = **true | false**]
[**homedir_path** = *home_directory_path*]
[**homedir_server** = *home_directory_server_host_name*]
[**install_image** = *default_network_install_image_name*]
[**ntp_config** = **consensus | internet | none | timemaster**]
[**ntp_server** = *ntp_server_host_name* ...]
[**ntp_version** = **3 | 1**]
[**passwd_file_loc** = *passwd_file_server_host_name*]
[**passwd_file** = *passwd_file_path*]
[**print_config** = **false | open | secure**]
[**print_id** = *secure_print_login_name*]
[**remove_image** = **true | false**]
[**spacct_enable** = **false | true**]
[**spacct_actnode_thresh** = *sp_accounting_active_node_threshold*]
[**spacct_excluse_enable** = **false | true**]
[**supfilesrv_port** = *port*]
[**supman_uid** = *supman_uid*]
[**usermgmt_config** = **false | true**]

### Flags

**acct_master**
Indicates which node is the accounting master, where **crunacct** runs. The initial value is *accounting_master*=0, specifying the control workstation.

**amd_config**
Indicates whether the automounter function should be configured and supported by the SP. Specify **true** if you want to have the automounter configured and the automounter daemon started on your SP. Automounter entries are created for your home directories if **usermgmt_config** is also **true**. Specify **false** if you do not want to have the SP manage the automounter. The initial value of **amd_config** is **true**.

**cw_lppsource_name**
Indicates the LPP source name to use when installing the NIM file sets on the control workstation. The name you specify must correspond to an LPP source directory on the control workstation. The directory must be named **/spdata/sys1/install/***LPP_source_name***/lppsource**, where *LPP_source_name* is the name that you have assigned to the LPP source. The default value is **default**.

| You must ensure that the AIX level on the LPP source
| (indicated by the **cw_lppsource_name**) matches the AIX level
| installed on your control workstation.

**filecoll_config**
Indicates whether the SP file collection management code should be installed.

Specify **true** if the code is to be installed. Specify **false** if the code is not to be installed. The initial value is **true**.

**homedir_path**
Specify an absolute path name for *home_directory_path* if you want to use a path other than the initial value of the control workstation path. The initial value is **/home/***cw*, where *cw* is the host name of the control workstation.

**homedir_server**
Indicates where user home directories physically reside. Specify a valid host name or IP address for *home_directory_server_host_name* if you want to use a server other than the initial value of the control workstation host name.

The initial value is the host name of the control workstation.

**install_image**
Indicates the location of the default network install image for your SP system. *default_network_install_image_name* should point to the install image which is used for a node if that node's install image field is not set. This should be a file in **/spdata/sys1/install/images**.

**ntp_config**
See **ntp_server**.

**ntp_server**
Indicates your choice for running NTP in your SP. To use your site's NTP time server, specify **ntp_config=timemaster** and specify the host name of your NTP time server with the **ntp_server** parameter.

To use an Internet NTP time server, your control workstation must be connected to the Internet. Specify **ntp_config=internet** and specify the full host name of an Internet time server with the **ntp_server** parameter.

To cause the control workstation and file servers to generate a consensus time based on their own date settings, specify **ntp_config=consensus** and specify **ntp_server=''**.

If you do not want to run NTP on the SP, specify **ntp_config=none** and **ntp_server=''**.

The initial value of **ntp_config** is **consensus** and the initial value of **ntp_server** is **''**. If **ntp_config** is specified as either **timemaster** or **internet**, the **ntp_server** value must be a valid host name.

**ntp_version** Indicates which version of NTP you are running. The initial value is **3**.

**passwd_file** Indicates the path of the password file where new user entries are placed. The initial value is **/etc/passwd**. If you change the value of **passwd_file** from **/etc/passwd** and are using NIS, be sure to modify your NIS Makefile to build the password map from the new password file.

This field is meaningful only if **usermgmt_config=true**.

**passwd_file_loc** Specifies the host name of the machine where your password file resides. The initial value of **passwd_file_loc** is the control workstation. The value of the **passwd_file_loc** cannot be one of the nodes in the SP system.

**print_config** Indicates how SP print management should be integrated into your system.

Specify **print_config = false** if you do not want to have the existing AIX print commands saved and have the file names linked to the SP print functions.

If you want to have the print management system integrated, specify **secure** if you do not want users to not have rsh privileges on the print host. Specify **open** if the users are to have rsh privileges on the print host.

The initial value of **print_config** is **false**.

**print_id** Specify the login name to be used for rsh for secure mode printing. This field is meaningful only if **print_config=secure** is specified.

The initial value is **''**. If the value remains at **''** and the value of **print_config** is **secure**, the installation script uses a print management ID of **prtid** as a default.

The SP Print Management System was removed from PSSP 2.3. That is, the SP Print Management System cannot be configured on nodes running PSSP 2.3 or later. We suggest the use of Printing Systems Manager (PSM) for AIX as a more general solution to managing printing on the SP system.

However, if you are running earlier versions of PSSP on some of your nodes, the SP Print Management System is still supported on those nodes. The **print_config** routine running on the control workstation will configure the SP Print Management System on nodes running versions of PSSP earlier than PSSP 2.3.

If you are running mixed levels of PSSP in a system partition, be sure to maintain and refer to the appropriate documentation for whatever versions of PSSP you are running.

**remove_image** Indicates whether install images are to be removed from the boot servers after an install has been completed.

Specify **remove_image=true** if the images are to be removed.

Specify **remove_image=false** if the images are not to be removed.

The initial value is **false**.

**spacct_actnode_thresh**
Indicates the percentage of nodes for which accounting data must be present in order for **crunacct** to continue processing that day. The initial value is **80**.

**spacct_enable**
Indicates whether accounting is enabled or disabled on all nodes that have an accounting enabled attribute set to default. The initial value is **false**, disabling accounting.

**spacct_excluse_enable**
Indicates if accounting start and end job records are generated for jobs having exclusive use of the node. A value of **true** indicates that exclusive use accounting is enabled and start and end job records are generated. A value of **false** indicates that exclusive use accounting is not enabled and start and end job records are not generated.

The initial value is **false**.

**supfilesrv_port**
Specifies the file collection daemon port. This is used in **/etc/services** for the file collection daemon. Pick a value that does not conflict with any other ports in use. It is meaningful only if **filecoll_config=true** is specified. The initial value is **8431**.

**supman_uid**
Specifies the uid for the file collection daemon. It is meaningful only if **filecoll_config=true** is specified. The initial value is **''**. If you are using login control, make this uid lower than the threshold ID you set in the **block_usr_sample** script.

**usermgmt_config**
Indicates whether SP user management scripts should be integrated into your system.

Specify **usermgmt=true** if you want to have the SP User Management scripts in the Security & Users SMIT menu. Specify **usermgmt=false** to remove the scripts from the SMIT menu.

The initial value is **true**.

## Operands

None.

## Description

Use this command during installation of the SP or at a later time to identify SP configuration parameters in use at your location.

You must have a ticket-granting-ticket to run this command. Refer to the chapter on security in *PSSP: Administration Guide* for additional information on ticket-granting-tickets.

If you do not have a ticket-granting-ticket, you must run **k4init**.

You can use the System Management Interface Tool (SMIT) to run the **spsitenv** command. To use SMIT, enter:

```
smit enter_data
```

and select the Site Environment Information option.

You cannot use SMIT if you are using AFS authentication services.

**Notes:**

1. This command should be run only on the control workstation. You must be logged into the control workstation as root to execute this command.
2. Any changes made will not take effect on the nodes until they are customized.

## Location

**/usr/lpp/ssp/bin/spsitenv**

## Examples

The following example enters site environment parameters into the System Data Repository. The NTP configuration is **consensus** and the file collection management code is to be installed:

```
spsitenv ntp_config=consensus filecoll_config=true
```

## spsvrmgr

### Purpose

**spsvrmgr** – SP Supervisor Manager.

### Syntax

**spsvrmgr** [–**G**] [–**f** *file_name*]
[[–**q rc | msg**] **|**
[–**r status | action**] **|**
[–**m status | action**] **|** [–**u**]] [*slot_spec* **| all**]

### Flags

**–G** Specifies Global mode. With this flag, commands can be sent to any hardware.

**–f** Uses *file_name* as the source of slot ID specifications.

**–q rc | msg** Checks the supervisor hardware configuration for supervisors that support microcode download, and that also require an action.

Action checks include:

**Install** Indicates that the Supervisor card has no supervisor installed. An install is required.

**Upgrade** Indicates that the Supervisor card has a supervisor installed, but it is not at the most current level. An upgrade is required.

**Reboot** Indicates that the Supervisor card has a supervisor installed and it is at the most current level, but it is not active. A reboot is required.

**Update Media**
Indicates that the Supervisor card has a supervisor installed, but the media that is the repository for microcode files does not contain the version that is installed on the card. A media update is required.

If **rc** is specified with the **–q** flag, the command will issue a return code indicating whether any of the hardware requires action. A return code of 0 indicates that no action is required. A return code of 1 indicates that at least one supervisor was found that required action.

If **msg** is specified with the **–q** flag, the command will issue a message indicating whether any of the hardware requires action. In this case, a return code of 0 is issued unless an error condition occurs.

**–r status | action**
Checks the supervisor hardware configuration for supervisors that support microcode download and displays status for those supervisors in “report” form.

If **status** is specified with the **–r** flag, the status is listed for all of the installed supervisors that support microcode download.

If **action** is specified with the −**r** flag, the status is listed for all of the installed supervisors that support microcode download and that also require an action.

In both cases, Status includes:

**Frame Number**
Indicates the number of the frame.

**Slot Number**
Indicates a number in the range of 0—17.

**Supervisor State**
Indicates either Active (supervisor is executing) or Inactive (supervisor is not executing).

**Media Versions**
Indicates the microcode files that are compatible with the supervisor installed in this frame/slot.

**Installed Version**
Indicates the microcode file installed as the supervisor.

**Required Action**
Can be one of the following: None, Install, Upgrade, Reboot, or Update Media.

−**m status | action**
Checks the supervisor hardware configuration for supervisors that support microcode download and displays status for those supervisors in "matrix" form.

If **status** is specified with the −**m** flag, the status is listed for all of the installed supervisors that support microcode download.

If **action** is specified with the −**m** flag, the status is listed for all of the installed supervisors that support microcode download and that also require an action.

In both cases, Status includes:

**Frame Number**
Indicates the number of the frame.

**Slot Number**
Indicates a number in the range of 0—17.

**Action Required**
Can be either Required or Not Required.

−**u**
Installs, upgrades, or reboots the hardware supervisors specified by the *slot_spec* option that support microcode download and that also requires an action.

**Note:** This flag starts an **hmcmds** process to perform the actual update. Refer to the **hmcmds** command specifically the **basecode**, **microcode**, and the **boot_supervisor** command options.

**Attention**

In most cases, the −**u** flag started processes powers off the target slots during the duration of the update.

## Operands

*slot_spec* **| all**
Specifies the addresses of the hardware components.

## Description

The design of the SP supervisor control system divides the microcode used in the frame supervisor, node supervisor, and switch supervisor into the following two types:

| | |
|---|---|
| **basecode** | Microcode that is loaded at the time of manufacture and gives the card the ability to load application microcode during system operation. |
| **application microcode** | Microcode that is loaded via basecode and contains the instruction that is the supervisor application. |

The **spsvrmgr** command controls the software level and state of the supervisor applications that reside on the SP supervisor hardware.

Normally, commands are only sent to the hardware components in the current system partition. A system partition contains only processing nodes. The switches and the frames themselves are not contained in any system partition. To access hardware components not in the current system partition or to any frame or switch, use the –**G** flag.

The *slot_spec* option is interpreted as slot ID specifications. A slot ID specification names one or more slots in one or more SP frames and has either of two forms:

```
fidlist:sidlist   or   nodlist
```

where:

| | |
|---|---|
| **fidlist** | = fval[,fval,...] |
| **sidlist** | = sval[,sval,...] |
| **nodlist** | = nval[,nval,...] |

The first form specifies frame numbers and slot numbers. The second form specifies node numbers. An *fval* is a frame number or a range of frame numbers of the form *a–b*. An *sval* is a slot number from the set 0 through 17 or a range of slot numbers of the form *a–b*. An *nval* is a node number or a range of node numbers of the form *a–b*.

The relationship of node numbers to frame and slot numbers is shown in the following formula:

```
node_number = ((frame_number – 1) × 16) + slot_number
```

**Note:** Node numbers can only be used to specify slots 1 through 16 of any frame.

Refer to the **hmcmds** command for examples of the *slot_spec*.

Optionally, slot ID specifications can be provided in a file rather than as command flags. The file must contain one specification per line. The command requires that slot ID specifications be provided. If the command is to be sent to all SP hardware, the keyword **all** must be provided in lieu of the *slot_spec* option. However, the **all** keyword can only be specified if the –**G** flag is specified.

## Files

The media that is the repository for the application microcode files is the **/spdata/sys1/ucode** directory structure.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Returned only in conjunction with the **–q rc** flag to indicate that at least one supervisor was found that required action. |
| **–1** | Indicates that the command was unsuccessful. This return value is always accompanied with an error message. |

## Restrictions

IBM suggests that you use this command through the RS/6000 SP Supervisor Manager option of the System Management Interface Tool (SMIT).

To access this command using SMIT, enter:

```
smit
```

and select the RS/6000 SP System Management option, then the RS/6000 SP Supervisor Manager option.

A list of options that correspond to the **spsvrmgr** command flags will be presented for selection.

You can also directly access this list of options using the following SMIT fast-path command:

```
smit supervisor
```

## Implementation Specifics

You must be authorized to access the Hardware Monitor subsystem to run the **spsvrmgr** command. In addition, for those frames specified to the command, you must have Virtual Front Operator Panel (VFOP) permission. Commands sent to frames for which you do not have VFOP permission are ignored. Since the Hardware Monitor subsystem uses SP authentication services, you must run the **k4init** command prior to running this command. Alternatively, site-specific procedures can be used to obtain the tokens that are otherwise obtained by **k4init**.

The **spsvrmgr** command, by design, only interacts with SP supervisor hardware that supports the ability to download application microcode. Commands sent to slots that do not support this ability are ignored.

## Location

**/usr/lpp/ssp/bin/spsvrmgr**

## Related Information

Commands: **hmcmds**

Refer to the “Installing and Configuring a New RS/6000 System” chapter in *PSSP: Installation and Migration Guide*.

## Examples

1. To perform a “quick check” of your configuration for supervisor hardware that requires action and to have a message issued, enter:

   ```
   spsvrmgr -G -q msg all
   ```

   You should receive output similar to the following:

   ```
   spsvrmgr: At least one occurrence of supervisor hardware was found to
             require attention.
             Enter "smit supervisor" for installation options.
   ```

2. To perform a “quick check” of your configuration for supervisor hardware that requires action and to have a status code returned, enter:

   ```
   spsvrmgr -G -q rc all
   echo $?
   ```

   Example usage in a script:

   ```
   spsvrmgr -G -q rc all
   if [[ $? = 1 ]]
   then
       echo "*** Attention*** One or more supervisors require action."
       echo "Enter \"smit supervisor\" for installation options."
   fi
   ```

3. To display status information in report form of all hardware that supports microcode download for frame 2, enter:

   ```
   spsvrmgr -G -r status 2:0-17
   ```

   You should receive report output similar to the following:

   ```
   spsvrmgr: Frame Slot Supervisor Media        Installed    Required
                        State      Versions     Version      Action
             _____ ____ __________ ____________ ____________ ____________
             2     1    Active     u_10.3a.0609 u_10.3a.060b None
                                   u_10.3a.060a
                                   u_10.3a.060b
                   ____ __________ ____________ ____________ ____________
                   5    Active     u_10.3a.0609 u_10.3a.060b None
                                   u_10.3a.060a
                                   u_10.3a.060b
                   ____ __________ ____________ ____________ ____________
                   9    Active     u_10.1a.0609 u_10.1a.060b None
                                   u_10.1a.060a
                                   u_10.1a.060b
                   ____ __________ ____________ ____________ ____________
                   13   Active     u_10.3a.0609 u_10.3a.060b None
                                   u_10.3a.060a
                                   u_10.3a.060b
   ```

4. To display status information in matrix form of all hardware that supports microcode download for in your configuration, enter:

   ```
   spsvrmgr -G -r status all
   ```

   You should receive matrix output similar to the following:

   ```
   spsvrmgr: Frame      Slots
             _____      _______________________________________________
             1          00 01 05 09 13 17
                (Action)  -  -  -  -  -  -
             _____      _______________________________________________
             2          01 05 09 13
                (Action)  +  +  +  -

             Action Codes:
                 +  -- Required
                 -  -- Not Required
   ```

5. To display status information in report form of all hardware that supports microcode download and requires an action for frame 1, enter:

   ```
   spsvrmgr -G -r action 1:0-17
   ```

   You should receive report output similar to the following:

   ```
   spsvrmgr: Frame Slot Supervisor Media        Installed    Required
                        State      Versions     Version      Action

             _____ ____ __________ ____________ ____________ ____________
             1     1    Active     u_10.3a.0609 u_10.3a.060a Upgrade
                                   u_10.3a.060a
                                   u_10.3a.060b

                   ____ __________ ____________ ____________ ____________
                   5    Inactive   u_10.3a.0609 u_10.3a.060b Reboot
                                   u_10.3a.060a
                                   u_10.3a.060b

                   ____ __________ ____________ ____________ ____________
                   9    Inactive   u_10.1a.0609 u_10.1a.060b Reboot
                                   u_10.1a.060a
                                   u_10.1a.060b
   ```

6. To update the hardware that supports microcode download in frame 1 slot 1, enter:

   ```
   spsvrmgr -u 1:1
   ```

   You should receive installation output similar to the following:

   ```
   spsvrmgr: Dispatched "microcode" process [24831] for frame 1 slot 1.
             Process will take approximately 12 minutes to complete.

   spsvrmgr: Process [24831] for frame 1 slot 1 completed successfully.
   ```

7. To update the hardware that supports microcode download in frame 1 slots 5 and 9, enter:

   ```
   spsvrmgr -u 1:5,9
   ```

   You should receive installation output similar to the following:

```
spsvrmgr: Dispatched "boot_supervisor" process [27956]
          for frame 1 slot 5.
          Process will take less than a minute to complete.

spsvrmgr: Dispatched "boot_supervisor" process [23606]
          for frame 1 slot 9.
          Process will take less than a minute to complete.

spsvrmgr: Process [27956] for frame 1 slot 5 completed successfully.
spsvrmgr: Process [23606] for frame 1 slot 9 completed successfully.
```

## spsyspar

### Purpose

**spsyspar** – Directly invokes the System Partitioning Aid Perspective graphical user interface (GUI).

### Syntax

**spsyspar** [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**]
[–**backgroundColor** *colorName*]
[–**foregroundColor** *colorName*] [–**fontFamily** *name*]
[–**fontSize** *size*] [–**fontBold**] [–**fontItalic**] [–**nosplash**] [–**h**]

### Flags

–**userProfile** *name*
Upon initialization, loads the specified user profile. If a user profile named "Profile" exists in the user's home directory, it will be loaded by default if the –**userProfile** flag is not specified.

–**systemProfile** *name*
Upon initialization, loads the specified system profile instead of the default system profile. The default system profile is named "Profile."

–**noProfile** Upon initialization, does not read either profile.

–**backgroundColor** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**foregroundColor** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6–30 points.

–**fontBold** Sets the font to bold.

–**fontItalic** Sets the font to italics.

–**nosplash** Does not display the splash screen before the Perspectives main window is displayed.

–**h** Displays usage information on the options available for the command.

**Note:** Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

## Operands

None.

## Description

Use this command to launch the System Partitioning Aid window of the SP Perspectives GUI. The System Partitioning Aid Perspective is used to view and manage the current system partitioning configuration. This tool can also be used to generate new configurations.

When the command is invoked, preferences which define the look and layout of the System Partitioning Aid window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

## Files

The users preferences are read from and saved to **$HOME/.spsyspar(User Profile Name)**. The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.spsyspar(System Profile name)**. If a new system partitioning configuration is created, the following files are created under the layout directory: **layout.desc**, **nodes.syspar** and a system partition directory for each system partition in the layout. For each system partition directory, a node list file and topology file are created.

## Restrictions

Any user can run the **spsyspar** command. Many actions require root privilege.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Prerequisite Information

For information on using the System Partitioning Aid Perspective and SP Perspectives, see the online help and the “Using SP Perspectives” chapter in the *PSSP: Administration Guide*.

Refer to the “Managing System Partitions” chapter in *PSSP: Administration Guide* for additional information on the System Partitioning Aid.

See also Appendix A, "The System Partitioning Aid – A Brief Tutorial" in IBM RS/6000 SP: Planning, Volume 2, Control Workstation and Software Environment.

## Location

**/usr/lpp/ssp/bin/spsyspar**

## Related Information

You can also access the System Partitioning Aid Perspective by using the SP Perspectives Launch Pad. The **perspectives** command invokes the Launch Pad. Other Perspectives windows may be launched by invoking the following commands: **spevent**, **sphardware**, **spperfmon**, and **spvsd**. The **sysparaid** command provides a command line interface into the System Partitioning Aid.

## Examples

1. To launch the Partitioning Aid Perspective, enter:

   ```
   spsyspar
   ```

2. To launch the Partitioning Aid Perspective with a pink background regardless of what is provided in the preference files, enter:

   ```
   spsyspar -backgroundColor pink
   ```

## sptg

### Purpose

**sptg** – Launches an SP TaskGuide.

### Syntax

**sptg** [–**h** | *name*]

### Flags

| | |
|---|---|
| –**h** | Displays usage. |

### Operands

| | |
|---|---|
| *name* | The name of the TaskGuide. The values allowed are *setsitenv* (Set Site Environment Information TaskGuide), *addframe* (Add Frames TaskGuide), *confnode* (Configure New Nodes TaskGuide), and *createim* (Create Node Image TaskGuide). |

### Description

The **sptg** command launches a specific SP TaskGuide. If no flags or operands are specified, the command launches a GUI that lists available TaskGuides and allows one to be selected and started.

### Restrictions

The **sptg** command can be run on the Control Workstation only.

### Location

**/usr/lpp/ssp/bin/sptg**

### Examples

1. To launch the Configure New Nodes TaskGuide, enter:

```
sptg confnode
```

## spunmirrorvg

### Purpose

**spunmirrorvg** – Initiates unmirroring on a node or a set of nodes.

### Syntax

**spunmirrorvg** {*start_frame start_slot node_count* **|** –**l** *node_list*}

### Flags

|  |  |
|---|---|
| –**l** *node_list* | Specifies a list of nodes for this operation. This list can be a single numeric node number, or a list of numeric node numbers separated by commas. |

### Operands

**start_frame** Specifies the frame number of the first node to be used for this operation.

**start_slot** Specifies the slot number of the first node to be used for this operation.

**Note:** The *start_frame* and *start_slot* must resolve to a node in the current system partition.

**node_count** Specifies the number of nodes to be used for this operation. The node information is added for successive nodes within a frame. If the count of nodes causes the nodes in a frame to be exhausted, the operation continues for nodes in the next sequential frame.

**Note:** The *node_count* is considered to be within the current system partition.

### Description

The **spunmirrorvg** command uses information found in the Volume_Group object to initiate unmirroring on a node or a list of nodes. If the number of desired copies is already achieved (the number of copies of a volume group equals the "copies" attribute in the Volume_Group object) the command exits. If the number is not yet achieved, **spunmirrorvg** will invoke the AIX **unmirror** command to reduce the number of copies. If **unmirrorvg** is successful, the volume group is reduced by any physical volumes that are part of the volume group, that are not listed in the *pv_list* attribute. If there are non-empty logical volumes on the physical volumes, the volume group will not be reduced by the physical volume. If reducing the volume group is unsuccessful, the command exits with an error. Quorum is set based on the value of the "quorum" attribute for the volume group in the Volume_Group object. If the state of quorum changes, a message is sent that the node requires rebooting. **spunmirrorvg** also issues the **bosboot** command to rebuild the bootable image, and the **bootlist** command, to remove any physical volumes from the bootlist that no longer contain bootable logical volumes.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that a recoverable error occurred, some changes may have succeeded. |
| **2** | Indicates that an irrecoverable error occurred and no changes were made. |

## Security

A user must run as "root" and have a valid Kerberos ticket.

## Files

Log file created on node which contains AIX error messages if an error occurs during unmirroring: **/var/adm/SPlogs/sysman/unmirror.out**

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/spunmirrorvg**

## Related Information

Commands: **spchvgobj**, **spmirrorvg**

## Examples

1. To initiate unmirroring on a node, enter:

   ```
   spunmirrorvg  -l 1
   ```

2. To initiate unmirroring on a list of nodes, enter:

   ```
   spunmirrorvg  -l 1,2,3
   ```

## spverify_config

### Purpose

**spverify_config** – Verifies the active system partition configuration information for the SP system.

### Syntax

**spverify_config**

### Flags

None.

### Operands

None.

### Description

This command is run by the **spapply_config** command after the System Data Repository (SDR) is updated. It can also be run by an administrator to verify that the SDR information is consistent (such as, after a system outage or a problem with the SDR). (This verification is only performed on a system which was partitioned beyond the initial single partition created at initial installation.)

### Exit Values

| | |
|---|---|
| **0** | Indicates that the SDR and corresponding layout directory are in agreement. |
| **1** | Indicates differences were found. |
| **2** | Indicates a usage error. |

### Location

**/usr/lpp/ssp/bin/spverify_config**

### Related Information

Commands: **spapply_config**, **spcustomize_syspar**, **spdisplay_config**

Files: **nodelist**, **topology**

### Examples

To verify that the information in the SDR matches the customization information previously supplied by the user, enter:

```
spverify_config
```

## spvsd

### Purpose

**spvsd** – Directly launches the IBM Virtual Shared Disk Perspective graphical user interface (GUI).

### Syntax

**spvsd** [–**userProfile** *name*] [–**systemProfile** *name*] [–**noProfile**] [–**backgroundColor** *colorName*] [–**foregroundColor** *colorName*] [–**fontFamily** *name*] [–**fontSize** *size*] [–**fontBold**] [–**fontItalic**] [–**nosplash**] [–**h**]

### Flags

–**userProfile** *name*
Upon initialization, loads the specified user profile. If a user profile named "Profile" exists in the user's home directory, it will be loaded by default if the –**userProfile** flag is not specified.

–**systemProfile** *name*
Upon initialization, loads the specified system profile instead of the default system profile. The default system profile is named "Profile."

–**noProfile**
Upon initialization, does not read either profile.

–**backgroundColor** *colorName*
Overrides the background color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**foregroundColor** *colorName*
Overrides the foreground color specified by any profile or default with the specified color. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid color names.

–**fontFamily** *name*
Overrides any font family with the specified font. The list of valid family names is dependent on the X server. Refer to Appendix A, “Perspectives Colors and Fonts” in *PSSP: Command and Technical Reference* for a list of valid fonts.

–**fontSize** *size*
Overrides any font point size with the specified size. Valid values are 6–30 points.

–**fontBold**
Sets the font to bold.

–**fontItalic**
Sets the font to italics.

–**nosplash**
Does not display the splash screen before the Perspectives main window is displayed.

–**h**
Displays usage information on the options available for the command.

**Note:** Most flags accepted by X will also be recognized. For example, –**display** *displayname*.

## Operands

None.

## Description

Use this command to launch the IBM Virtual Shared Disk Perspective. This Perspective allows the user to view and control the IBM Virtual Shared Disk subsystem.

By default, when the window is brought up, it displays:

- The control workstation system partitions pane
- The Nodes pane
- The IBM VSDs or IBM HSDs pane must be added for viewing.

The current system partition is indicated by a lightning bolt in the control workstation and system partitions pane. The Nodes pane displays all nodes in the current system partition. Other panes display virtual shared disks and hashed shared disks. You can control which panes are displayed by using the Add Pane and Delete Pane tool bar icons.

When the command is invoked, preferences that define the look and layout of the **spvsd** window are prioritized in the following order:

- Command line options
- User preferences profile
- System preferences profile
- Default values

## Files

The Users Preferences are read from and saved to **$HOME/.spvsd(User Profile Name)**. The System Preferences are read from and saved to **/usr/lpp/ssp/perspectives/profiles/.spvsd(System Profile name)**.

## Restrictions

Any user can run the **spvsd** command. Many actions require root privilege.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Prerequisite Information

For information on using the IBM Virtual Shared Disk Perspective and SP Perspectives, see the online help and the “Using SP Perspectives” chapter in *PSSP: Administration Guide*. For information about the IBM Virtual Shared Disk subsystem, see *PSSP: Managing Shared Disks*.

## Location

**/usr/lpp/ssp/bin/spvsd**

## Related Information

You can access the IBM Virtual Shared Disk Perspective by using the SP Perspectives Launch Pad. The **perspectives** command invokes the Launch Pad. Other Perspectives windows may be launched by invoking the following commands: **spevent**, **sphardware**, **spperfmon**, and **spsyspar**.

*PSSP: Managing Shared Disks*

## Examples

1. To invoke the **spvsd** window, enter:

   ```
   spvsd
   ```

2. To force **spvsd** to display bold text regardless of what is set in the preference files, enter:

   ```
   spvsd -fontBold
   ```

## st_clean_table

### Purpose

**st_clean_table** – Forces the unload of the job switch

resource table for a specified window on the specified node.

### Syntax

**st_clean_table** {**-h** | **-?** | *node_name*} [**-w** *window_id*] [**-k**]

### Flags

| | |
|---|---|
| –**h** | Prints out a short description of all of the flags. |
| –**?** | Prints out the usage statement. |
| –**w** *window_id* | Specifies the window id for which the unload and cleanup will be done. If no window id is specified, then the default window will be unloaded. |
| –**k** | Stops any job that is currently using the switch table on that *node_name* and unloads the table. The –**k** flag has the same function as the **ST_ALWAYS_KILL** option of the **swtbl_clean_table** API. |

### Operands

| | |
|---|---|
| *node_name* | Specifies the name of the node upon which the switch table window will be unloaded. |

### Description

Use this command to override user ID (uid) checking and to unload the job switch resource table window on the node specified.

Normal unloading of the job switch resource table by the **swtbl_unload_table** API checks that the user ID (uid) of the unload matches the uid specified during the load. The **st_clean_table** command ignores this check and allows the administrator to unload the window from a node. It is intended to be used for error recovery and not for normal unloading. Use this command when a parallel job has left a process in use and the window did not unload with the **swtbl_unload_table** API. If **-k** is not specified and a job is using the switch table, the unload will not be performed. The default window is defined within the **st_client.h** file. A single job switch resource table may contain more than one window. This command needs to be issued for every window within the table. Use the **st_status** command to obtain the current state of the windows. Additional error and information may be found in the **/var/adm/SPlogs/st/st_log** file.

### Files

| | |
|---|---|
| **/usr/lpp/ssp/include/st_client.h** | Path name of the client header file. |
| **/usr/lpp/ssp/lib/libswitchtbl.a** | Path name of the shared library containing APIs. |

## Standard Output

After the job switch resource table window is successfully unloaded, the status should be **ST_SWITCH_NOT_LOADED**.

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **nonzero** | Indicates that an error occurred. |

## Security

You must have root privilege to run this command.

## Location

**/usr/lpp/ssp/bin/st_clean_table**

## Related Information

Commands: **st_status**

## Examples

To stop the process currently using window 1 and unload the window from k10n10, enter:

```
st_clean_table -w 1 -k k10n10
```

This produces the result:

```
Node k1010 window 1 has been unloaded.
```

## st_status

### Purpose

**st_status** – Displays the status of every window within all job switch resource tables upon a node.

### Syntax

**st_status** [**-h** | **-?** | **-n** *node_group* | *nodelist*]

### Flags

| | |
|---|---|
| –**h** | Prints out a short description of all of the flags. |
| –**?** | Prints out the usage statement. |
| –**n** *node_group* | Reports status for all nodes within the specified *node_group*. This "node_group" is defined and managed by the "Node Grouping" commands. |

### Operands

| | |
|---|---|
| *nodelist* | Specifies a list of nodes separated by spaces for which status will be reported. If node names are not specified, all nodes defined within the current system partition will be reported. |

### Description

Use this command to report the current status of every window within all job switch resource tables. This command reports whether each window is loaded or unloaded but not whether it is currently in use.

### Files

| | |
|---|---|
| **/usr/lpp/ssp/include/st_client.h** | Path name of the header file containing return codes. |
| **/usr/lpp/ssp/lib/libswitchtbl.a** | Path name of the shared library containing the API interfaces. |

### Standard Output

The output of this command reports the following data when a window is loaded:

| | |
|---|---|
| **Status from node:** | Node for which the following data is reported. |
| **User:** | User name corresponding to the uid specified by the **swtbl_load_table** API. |
| **Load request from:** | Node upon which the load request was made. |
| **Pid:** | Pid of the process who issued the load, most likely a job management application. |
| **Uid:** | Uid specified by the **swtbl_load_table** API request. |
| **Job Description:** | String specified upon load request for the job who will be using the job switch resource table. |

| | |
|---|---|
| **Time of request:** | Timestamp of when load request was processed. |
| **Window id:** | Window for which the data is being reported. |

The output of this command reports the following data when a window is in some other state:

| | |
|---|---|
| **ST_SWITCH_NOT_LOADED** | Indicates that the switch table is not currently loaded. |
| **ST_LOADED_BYOTHER** | Indicates that the switch table is currently loaded but was not loaded via the Job Switch Resource Table Services. |

The output of this command reports the following data when a error occurred:

| | |
|---|---|
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_CANT_CONNECT** | Indicates that the connect request was unsuccessful. |
| **ST_INVALID_ADDR** | Indicates that the **st_addr** argument cannot be converted by **inet_ntoa**. |
| **ST_NO_SWITCH** | The open call of **/dev/css0** was unsuccessful. |

## Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **nonzero** | Indicates that an error occurred. |

## Location

**/usr/lpp/ssp/bin/st_status**

## Related Information

Commands: **ngcreate**, **ngfind**, **st_clean_table**, **st_verify**

## Examples

1. To show the status of all windows on k10n15, enter:

```
st_status k10n15
```

You should receive output similar to the following:

```
*****************************************
Status from node:k10n15  User: root
Load request from:k10n15 Pid: 12494 Uid:0
Job Description: No_job_description_given
Time of request:Wed_Jan_24_13:38:21_1998 Window_id:0
*****************************************
Node k10n15 Window 1 ST_SWITCH_NOT_LOADED
*****************************************
Node k10n15 Window 2 ST_SWITCH_NOT_LOADED
*****************************************
Node k10n15 Window 3 ST_SWITCH_NOT_LOADED
```

## st_verify

### Purpose

**st_verify** – Verifies that the installation of the Job Switch Resource Table Services component of the SP system completed successfully.

### Syntax

**st_verify** [–**h**] [–**q**] [–**l** *logfile*]

### Flags

| | |
|---|---|
| –**h** | Displays usage information. |
| –**q** | Specifies quiet mode and suppresses output to **stdout**. |
| –**l** *logfile* | Specifies the pathname of the logfile to which error messages are written. |

### Description

Use this command to perform various tests to determine whether the Job Switch Resource Table Services component of the SP system is completely installed. It checks that the necessary commands and files are installed correctly and checks for switchtbld entries in **/etc/services** and **/etc/inetd.conf** file. If this is executed on the control workstation and tests are successful, it will also check on each node within that system partition.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the Job Switch Resource Table Services Installation option.

### Files

| | |
|---|---|
| **/var/adm/SPlogs/st/st_verify.log** | Default log file. |

### Exit Values

| | |
|---|---|
| **0** | Indicates that the test completed as expected. |
| **nonzero** | Returns the number of errors. |

If you do not specify the **-q** flag, a message is displayed on stdout that indicates whether the tests were successful or not. In either case, the command returns 0 if successful, 1 if not. If errors are detected, more detailed information is recorded in the log file. If you do not specify the **-l** flag, error messages are recorded in **/var/adm/SPlogs/st/st_verify.log**.

### Security

You must have root privilege to run this command.

## Location

**/usr/lpp/ssp/bin/st_verify**

## Related Information

Commands: **CSS_test**, **SDR_test**, **spmon_ctest**, **spmon_itest**, **SYSMAN_test**

## Examples

To verify the installation of the Job Switch Resource Table Services, saving error messages in **st_install.out** in the current working directory, enter:

```
st_verify -l st_install.out
```

## startvsd

### Purpose

**startvsd** – Makes a virtual shared disk available and activates it.

### Syntax

**startvsd [**–**p | –b**] {–**a |** *vsd_name* ...}

### Flags

| | |
|---|---|
| –**p** | Specifies that the primary server node defined for the global volume group is to be the active server. This option is only used by the Recoverable Virtual Shared Disk subsystem. See the *PSSP: Managing Shared Disks*. |
| –**b** | Specifies that the secondary server node defined for the global volume group is to be the active server. **Note:** This flag is used only by the Recoverable Virtual Shared Disk subsystem. |
| –**a** | Specifies that all virtual shared disks that have been defined are to be started. |

### Operands

| | |
|---|---|
| *vsd_name* | Specifies a virtual shared disk. |

### Description

The **startvsd** command makes the specified virtual shared disks available and activates them. It is equivalent to running the **preparevsd** command followed by the **resumevsd** command on the specified virtual shared disk.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Start a Virtual Shared Disk option.

### Security

You must have root privilege to run this command.

### Restrictions

1. If you have the Recoverable Virtual Shared Disk software installed and operational, do not use this command. The results may be unpredictable.

   See *PSSP: Managing Shared Disks*
2. The –**b** flag is used only by the Recoverable Shared Disk subsystem.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/startvsd**

## Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **stopvsd**, **suspendvsd**, **ucfgvsd**

## Examples

To make available and activate the virtual shared disk **vsd1vg1n1**, enter:

```
startvsd vsd1vg1n1
```

## statvsd

### Purpose

**statvsd** – Displays IBM Virtual Shared Disk device driver statistics of a node.

### Syntax

**statvsd**

### Flags

None.

### Operands

None.

### Description

The **statvsd** command displays the level of IBM Virtual Shared Disk parallelism, several IBM Virtual Shared Disk IP device driver counters, nonzero sequence numbers and the nodes they are for, the node out cast status, and the *max_IP_msg_size*. If either the expected or outgoing sequence number for a node is nonzero, both are displayed along with the node number. If both the expected and outgoing sequence numbers are zero, no sequence numbers are displayed for that node.

The level of IBM Virtual Shared Disk parallelism defaults to **9** and is set via the **ctlvsd** **–p** command.

Sequence numbers are initially all zero at the first **cfgvsd**. They are incremented as requests are sent (outgoing) and received (expected), and reset via **ctlvsd** **–R |** **–r**.

The counters all start at zero at the first **cfgvsd**, then are incremented as the events they count occur. Use **suspendvsd** and **stopvsd** to ensure that there is no virtual shared disk activity; then use **ctlvsd** to reset the counters.

The requests queued waiting for a request block, pbuf, cache block, and buddy buffer counters indicate shortages of these resources. All four are tunable values on the **vsdnode** command. If a significant increase in these counters occurs during the running of a critical application, see *PSSP: Managing Shared Disks* for information about tuning virtual shared disk performance and how to respond to resource shortages.

When a user buffer address is not on a page boundary, two virtual shared disks can share a page in I/O requests. Typically, when a local virtual shared disk server is copying data to the user buffer, the DMA hides the page. If the client receives the data from a remote virtual shared disk server, network protocol interrupts the local I/O; however, the page is still hidden by the DMA. Therefore, the virtual shared disk places the remote request on a Rework_Q, swaps control to the local I/O, and later performs rework by copying data from the network protocol mbuf to the user buffer.

A request, or its corresponding response, may be lost due to transmission error or an error in allocating an mbuf. The current virtual shared disk communication protocol implements an exponential back-off retransmission strategy. A request is retransmitted to the server a fixed number of times. The IBM Virtual Shared Disk IP device driver waits about 2 seconds for a response after initially sending the request before retransmission. Thereafter, it waits about twice as long as the last time as it cycles through the fixed number of retries. If a response is not received after the timeout expires on the last retransmission attempt, the request returns an error with the ETIMEOUT errno value. Currently, the sum total of retransmission time is about 15 minutes. If a request is not responded to after about 10 transmissions of the request over a 15 minute period, the request is unsuccessful. **statvsd** displays the number of requests that were unsuccessful due to timeouts in the timeouts counter. The retries counters display the number of requests that have been retransmitted for each retry bucket value. The number of numbers on the retries line of **statvsd** output indicates the fixed number of times a request is retransmitted. The total retries is the sum of the retries bucket values, which is the total number of request retransmissions.

There is no tuning that can be performed to affect these values; the values are provided for information only. If a request to a virtual shared disk is being retransmitted, and the virtual shared disk is suspended and subsequently resumed, the request starts over with a fresh 15 minute retry period.

If a server gets heavily loaded, it may not be able to respond to a request fast enough to prevent the client from retransmitting the request. If the server responds after the client has retransmitted the request, the client rejects the response since its sequence number no longer matches the current sequence number of the request. The client records this event in the rejected response counter that **statvsd** displays.

If a server receives a request with an unexpected sequence number, a rejected request event is recorded in a counter that **statvsd** displays.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Related Information

Commands:

**netstat** −**m** for **mbuf** usage information.
**/etc/no** −**o thewall=16,384** to dynamically add more **mbufs**.
**/etc/no** −**a | grep thewall** to show your current **mbuf** setting.
**cfgvsd**, **statvsd**, **ucfgvsd**, **vsdnode**.
**ctlvsd** to cast nodes in and out, resetting sequence numbers and setting the *max_IP_msg_size*.

Refer to *PSSP: Managing Shared Disks* for information on tuning IBM Virtual Shared Disk performance and sequence numbers.

## Examples

The following example displays IBM Virtual Shared Disk device driver statistics:

```
VSD driver: IP interface

          9 vsd parallelism
      24576 vsd max IP message size
          0 requests queued waiting for a request block
          0 requests queued waiting for a pbuf
          0 requests queued waiting for a cache block
          0 requests queued waiting for a buddy buffer
        0.0 average buddy buffer wait_queue size
          0 rejected requests
          0 rejected responses
          0 rejected no buddy buffer
          0 requests rework
          0 indirect I/O
          0 64-byte unaligned reads
          9 comm. buf pool shortage
          0 timeouts
retries:  0 0 0 0 0 0 0 0 0
          0 total retries
Non-zero  Sequence numbers
 node#     expected     outgoing     outcast?     Incarnation: 0

 3 Nodes Up with zero sequence numbers: 256 1000 2000
```

## stopvsd

### Purpose

**stopvsd** – Makes a virtual shared disk unavailable.

### Syntax

**stopvsd** {–**a** | *vsd_name* ...}

### Flags

| | |
|---|---|
| –**a** | Specifies that all virtual shared disks in the suspended state are to be stopped. |

### Operands

| | |
|---|---|
| *vsd_name* | Specifies a virtual shared disk. If the virtual shared disk is not is the suspended state, you get an error message. |

### Description

The **stopvsd** command brings the specified virtual shared disks from the suspended state to the stopped state. They become unavailable. All applications that have outstanding requests for the virtual shared disk see these requests terminate with error. Read and write requests return errors with **errno** set to **ENODEV**. If the virtual shared disk is in the stopped state, this command leaves it in the stopped state.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Stop a Virtual Shared Disk option.

### Security

You must have root privilege to run this command.

### Restrictions

If you have the Recoverable Virtual Shared Disk software installed and operational, do not use this command. The results may be unpredictable.

See *PSSP: Managing Shared Disks*.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/stopvsd**

## Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **startvsd**, **suspendvsd**, **ucfgvsd**

## Examples

To bring the virtual shared disk **vsd1vg1n1** from the suspended state to the stopped state, enter:

```
stopvsd vsd1vg1n1
```

## supfilesrv Daemon

### Purpose

**supfilesrv** – Is the daemon that serves the file collections on the SP system.

### Syntax

**startsrc –s supfilesrv**

**stopsrc –s supfilesrv**

### Flags

There are no flags or options for this daemon. All arguments and options are defined to the System Resource Controller (SRC).

### Description

This daemon executes on the control workstation and the boot and install servers. It responds to requests executed on the nodes to update collections of files configured for File Collections. The **supfilesrv** daemon is under control of the System Resource Controller (SRC). It is normally started as part of the SP initialization scripts. Under normal circumstances, the administrator does have to start or stop the daemon.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that an error occurred. |

### Prerequisite Information

Refer to *PSSP: Administration Guide* for information on file collections.

### Location

**/var/sysman/etc**

### Related Information

Refer to *PSSP: Administration Guide* for information on file collections, the System Resource Controller, and user management.

### Examples

1. To start the **supfilesrv** daemon, enter:

   ```
   startsrc -s supfilesrv
   ```

2. To stop the **supfilesrv** daemon, enter:

   ```
   stopsrc -s supfilesrv
   ```

## supper

### Purpose

Manages the SP file collections.

### Syntax

**supper** [–**d**] [–**o** *options*] [–**v**] *subcommands*

### Flags

| | |
|---|---|
| –**d** | Turns on debug mode. |
| –**o** *options* | Lists SUP options to pass to SUP. See the SUP manual pages for more detail. |
| –**v** | Turns on verbose mode and echoes print of SUP messages. |

### Operands

Subcommands

| | |
|---|---|
| **activate** *volume* | Set the active volume group. The active volume group must be set before installing a collection that requires a file system. |
| **debug** | Choice of **on \| off**. Turn debug messages on or off. |
| **diskinfo** | Show available disk space and active volume. |
| **files** *collection* | Show all files associated with a resident collection. |
| **install** *collection* | Install a collection. |
| **log** | Show summary of last/current supper session. |
| **offline** *collection* | Disable updates of a collection. |
| **online** *collection* | Enable updates of a collection (this is the default). |
| **quit** | Exit the program. |
| **remove** *collection* | Remove a collection. |
| **reset** *collection* | Set the last update time of a collection to the epoch. |
| **rlog** | Show raw output of last/current supper session. |
| **scan** *collection* | Run a scan for a collection. |
| **serve** | List all collections this machine is able to serve. |
| **status** | Show the current status of all available collections. The status information includes the names of all collections, whether they are resident on the local machine, and the name and size of the file system associated with each collection. |
| **update** *collection* | Update a collection. |
| **verbose** | Choice of **on \| off**. Turn SUP output messages on or off. |
| **when** | Print the last update time of all resident collections. |
| **where** | Show current servers for collections. |

| | |
|---|---|
| **!** *command* | Shell escape. |

## Description

You can invoke **supper** as an interactive session by entering the **supper** command without any parameters or subcommands. This allows you to enter the subcommands in an interactive dialog.

## Examples

```
/var/sysman/supper rlog
```

```
/var/sysman/supper status
```

```
/var/sysman/supper scan user.admin
```

```
/var/sysman/supper update power_system
```

## suspendvsd

### Purpose

**suspendvsd** – Deactivates an available virtual shared disk.

### Syntax

**suspendvsd** {**–a** | *vsd_name*...}

### Flags

| | |
|---|---|
| **–a** | Specifies that all the virtual shared disks in the active state are to be suspended. |

### Operands

| | |
|---|---|
| *vsd_name* | Specifies a virtual shared disk. If the virtual shared disk is not in the active state, you get an error message. |

### Description

The **suspendvsd** command brings the specified virtual shared disks from the active state to the suspended state. They remain available. Read and write requests which were active while the virtual shared disk was active are suspended and held. Subsequent read and write operations are also held. If the virtual shared disk is in the suspended state, this command leaves it in the suspended state.

If you issue this command for the server node and not the client, retries occur unsuccessfully. An error occurs within 15 minutes.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Suspend a Virtual Shared Disk option.

### Security

You must have root privilege to run this command.

### Restrictions

If you have the Recoverable Virtual Shared Disk software installed and operational, do not use this command. The results may be unpredictable.

See *PSSP: Managing Shared Disks*.

### Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/suspendvsd**

## Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **startvsd**, **stopvsd**, **ucfgvsd**

## Examples

To bring the virtual shared disk **vsd1vg1n1** from the active state to the suspended state, enter:

```
suspendvsd vsd1vg1n1
```

## switch_stress

## Purpose

**switch_stress** – Tests the functionality of a specific switch chip.

**Attention**

**ATTENTION – READ THIS FIRST:** Do **not** activate the SP Switch advanced diagnostic facility until you have read this section completely, and understand this material. If you are not certain how to properly use this facility, or if you are not under the guidance of IBM Service, do **not** activate this facility.

Activating this facility may result in degraded performance of your system. Activating this facility may also result in longer response times, higher processor loads, and the consumption of system disk resources. Activating this facility may also obscure or modify the symptoms of timing-related problems.

## Syntax

**switch_stress** –**s** *switch_chip_id* [–**a** *allowed_nodes_list*]
[–**A** *allowed_nodes_file*] [–**f** *forbidden_nodes_list*]
[–**F** *forbidden_nodes_file*] [–**m** *model*] [–**z** *data_size*]
[–**p** *pattern_files_list*] [–**P** *pattern_files_file*] [–**g**] [–**h**]

## Flags

**-s** *switch_chip_id*
: Specifies the switch chip id of a suspicious switch chip.

**-a** *allowed_nodes_list*
: Specifies a list of nodes that the test can use. *allowed_nodes_list* is a blank-separated list of node identifiers. A node identifier can be a host name, IP address, frame, slot pair, or node number.

**-A** *allowed_nodes_file*
: Specifies a file containing a list of nodes that the test can use. *allowed_nodes_file* is a path to a file that contains a list of node identifiers.

**-f** *forbidden_nodes_list*
: Specifies a list of nodes that the test cannot use. *forbidden_nodes_list* is a blank-separated list of node identifiers.

**-F** *forbidden_nodes_file*
: Specifies a file containing a list of nodes that the test cannot use. *forbidden_nodes_file* is a path to a file that contains a list of node identifiers.

**-m** *model*
: Specifies a test model that will be used for testing. *model* is a name of the model to be used.

**-z** *data_size*
: Specifies amount of data in MB to be sent on every test iteration.

**-p** *pattern_files_list*
: Specifies a list of paths to the pattern files. *pattern_files_list* is a blank-separated list of paths. Each pattern file path is a full path to a file accessible from each participating node.

**-P** *pattern_files_file*
: Specifies the file containing a list of paths to the pattern files.

**-g** Requests to use SPD GUI.

**-h** Requests that usage information be displayed.

## Operands

None.

## Description

This command starts the switch chip stress test, which determines whether the specified switch chip is malfunctioning under stress. You are required to specify the switch chip ID that appeared in the primary node error reports.

The *model* argument lets you select a single test model. By default, all models will be executed.

You can specify the nodes that are allowed to participate in the test, or nodes that are not allowed to participate in the test. If the same node is present in both lists, it is not allowed to participate in the test. You must be aware that the selected nodes will not be able to run any application that uses a switch network during the test execution. By default all nodes are allowed to participate in the test. These nodes could be specified as a list of nodes or as a file that contains the list. The *data_size* argument allows you to control the amount of data that will be sent by every sender on every test iteration. By default this value is set to 360MB.

You can provide a path to a file that contains the data pattern to be used during the test. By default the output of the test is displayed on the command line. You can request to display the output on the SPD GUI.

## Location

**/usr/lpp/ssp/bin/spd/switch_stress**

## Examples

1. To test switch chip 16 using default settings, enter:
   ```
   switch_stress -s 16
   ```
2. To test switch chip 20 displaying the output on GUI, enter:
   ```
   switch_stress -s 20 -g
   ```
3. To test switch chip 25 specifying allowed nodes by host name, enter:
   ```
   switch_stress -s 25 -a n05 n06 n11
   ```
4. To test switch chip 25 specifying a forbidden node by frame,slot, enter:
   ```
   switch_stress -s 25 -f 2,9
   ```
5. To execute model A for switch chip 25, enter:

```
switch_stress -s 25 -m ModelA
```

6. To increase the amount of data sent through the switch chip under test, enter:

```
switch_stress -s 25 -z 1000
```

7. To use a different data pattern, create a data file, make it accessible to nodes (copy to every node or mount using the same name), and enter:

```
switch_stress -s 25 -p /tmp/spd/pattern1.dat
```

## sysctl

### Purpose

**sysctl** – Is the command interface to Sysctl remote command execution and monitoring server.

### Syntax

**sysctl** [–**c** *collection_of_nodes*] [–**f** *num*] [–**h** *host*] [–**L**] [–**m**]
[–**n**] [–**P** *port*] [–**q**] [–**r** *file* **|** –] [–**s**] [–**t** *sec*] [–**T** *sec*]
[–**v**] [–**x**] [*command ...*]

### Flags

**[–c** *collection_of_nodes***]**
Runs the command on the specified *collection_of_nodes*. If the *collection_of_nodes* argument contains a "/", it is assumed to name a file that holds the names of host machines. More than one *collection_of_nodes* can be specified with multiple –**c** options. Use – for standard input.

**[–f** *num***]** Controls maximum fan-out, when multiple hosts are specified. By default, a maximum of eight concurrent connections are used. The most allowed is 128. You may want to increase *num* to allow for greater parallelism when running commands that place low demand on system services.

**[–h** *host***]** The server on which to execute the command. More than one host can be specified using multiple –**h** options. If no –**h** (or –**c**) option is specified, the server is assumed to be the local host.

**[–L]** Provides an alternate way to delimit output from multiple servers. A host name, such as **helloworld.testcell.ibm.com** precedes each line of output from the host. This type of output is easier to parse in programs and scripts than the default. The default format is to separate the output in blocks labeled with the host name. This is compatible with the **dshbak** output filter.

**[–m]** Uses safe (message-authenticated) communication.

**[–n]** Sends no authentication information.

**[–P** *port***]** Connects to the server using a specified port number. If this flag is not specified, the port is obtained from file **/etc/services**.

**[–q]** Quick mode—do not wait for the result from the server. In this case, the server does not return the result to the client.

**[–r** *file* **|** –**]** Replays (drops) this file on the servers. Used to pass scripts to **sysctl** for execution. You can specify standard input as the source of input by entering a dash (–) instead of a file name; press <**Ctrl-D>** when you are finished entering script commands from standard input.

**[–s]** Tells the server to send results back via a TCP socket. The output from the server is demultiplexed into standard output and standard error, allowing you to separate the two types of output.

Using a socket also enables two-way communication between the client and a command procedure. Any commands that start a dialog requiring input from the client must be run with the **–s** option.

| | |
|---|---|
| **[–t** *sec***]** | Specifies the connection timeout in seconds. The default is 10 seconds. If the specified value falls outside the valid range of 1 to 60 seconds, the default value of 10 seconds is used. |
| **[–T** *sec***]** | Specifies the maximum time to wait in seconds for the result from a **sysctl** command. The default is 30 minutes. **–T0** is not valid; it will default to 30 minutes. The upper limit is the maximum positive integer. |
| **[–v]** | Prints the version number of **sysctl**, then exits. |
| **[–x]** | Sends a NULL RPC (such as a ping) to the servers. You can use this to check if a server is up before you try to run a command on that machine. |

## Operands

*command ...*
The command to pass to the server. You can enter multiple commands.

If you do not specify a command, **sysctl** runs in interactive mode.

## Description

The **sysctl** program provides a simple command line interface for communicating with the **Sysctl** server, **sysctld**. Together, the **Sysctl** client and server provide monitoring and execution abilities needed to remotely manage workstations on all nodes on the SP system. **Sysctl** connects to a remote host's **sysctld** using TCP/IP, passes keywords and commands to the server, and writes any output returned to standard output. **sysctl** does not interpret the commands it passes.

The **Sysctl** server uses the Tcl embeddable command language as the foundation for its built-in interpreter. The server is augmented with application-specific commands that may vary between servers. The **Sysctl** (Tcl) expressions that are passed to the server for execution can be anything from a single command to an entire script. The server uses SP authentication services for reliable third party authentication. The **Sysctl** request sent from the client optionally contains an authentication ticket that uniquely identifies the user that initiated the request. The server's built-in authorization mechanism controls the set of commands available to a user.

## Files

| | |
|---|---|
| **/etc/krb-srvtab** | Contains authentication key for services. |
| **/etc/sysctl.acl** | Default ACL file for the **sysctl** server |
| **/etc/sysctl.conf** | **sysctl** server configuration file. |

## Location

**/usr/lpp/ssp/bin/sysctl**

## Related Information

Commands: **dshbak**, **hostlist**, **sysctld**

The chapter on security in *PSSP: Administration Guide*.

## Examples

These examples show commands issued from the shell prompt. Unless noted otherwise, the syntax is the same (minus the word **sysctl**) when issuing the command from within an interactive session of **sysctl**:

1. To list the local file systems on server **ceti-alpha5**, enter:

   ```
   sysctl listfs -h ceti-alpha5
   ```

2. To list all the commands that you are authorized to run on **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 info commands
   ```

3. To add principal **arielle.admin** to the ACL of trusted users for the **sysctl** server on **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 acladd -p arielle.admin
   ```

   If you do not specify a realm, the local realm is assumed. If **ceti-alpha5** is in realm **TESTCELL.HAL.COM**, **sysctl** returns the following message:

   ```
   -principal arielle.admin@TESTCELL.HAL.COM
   ```

4. To add principal **arielle.admin** to ACL file **/test/data/mount.acl** on **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 acladd -f /test/data/mount.acl \
   -p arielle.admin
   ```

5. To see if **arielle** is in ACL file **/test/data/mount.acl** on **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 aclcheck -f /test/data/mount.acl arielle
   ```

   The server returns **1** if **arielle** is in the ACL, **0** if it is not.

6. To check to see if you are authorized to run the **test:mount_if** command on the SP node **ceti-alpha5**, enter:

   ```
   sysctl -h ceti-alpha5 checkauth -cmd test:mount_if
   ```

   The server returns **1** if you are authorized, **0** if not.

## sysctld Daemon

### Purpose

**sysctld** – Contains the remote execution and monitoring server.

### Syntax

**sysctld** [–**sd**] [–**a** *acl*] [–**f** *file*] [–**k** *keyfile*] [–**l** *log_file*] [–**P** *port*]

### Flags

| | |
|---|---|
| –**d** | Runs in debug mode. This causes a large amount of debugging information to be written to the log file as the server executes requests. |
| –**s** | Runs in security audit mode. An audit trace is written to the log file as the server executes each user request. |
| –**a** *acl* | Specifies the Access Control List (ACL) file. The default ACL file is **/etc/sysctl.acl**. |
| –**f** *file* | Specifies an alternate configuration file. The default is **/etc/sysctl.conf**. |
| –**k** *keyfile* | Specifies the server's key file. The default key file is **/etc/srvtab**. |
| –**l** *log_file* | Specifies the server log file. The default log file is **/var/adm/SPlogs/sysctl/sysctld.log**. (This is lowercase **l**, as in **l**ist.) |
| –**P** *port* | Specifies the server port number. If this flag is not specified, the port is obtained from **/etc/services** using **sysctl/tcp** as the key. |

### Operands

None.

### Description

The **sysctld** daemon is the server component for the Sysctl remote command execution facility. Security and performance characteristics of **sysctld** make it an ideal mechanism for managing a large, distributed computing environment. Typically, one instance of **sysctld** runs on every workstation. Commands are sent to a **sysctld** server via the **sysctl** client program. When a command request is received, **sysctld** parses and executes the request using an embedded Tcl command interpreter. Authorization to execute commands is determined by *authorization callbacks* that are attached to each command. These callbacks are pieces of Tcl code that implement the security policy for the **sysctld** server.

**Security and Access Control**

Sysctl uses the SP authentication service for reliable third-party authentication. Command requests sent by the client include an authentication ticket giving the identity of the client. As the server executes the commands sent by the client, the client's access to commands and variables is determined dynamically via *authorization callbacks*. In a typical command language, a *procedure* has a name, a set of arguments, a set of commands which form the body of the procedure, and a return value. With Sysctl procedures, an additional attribute is added, a policy for determining who is able to run the command. These policies are implemented using authorization callbacks. An authorization callback is a piece of Tcl code that is *attached* to a command and determines the access policy for that command. The authorization callback for a command is invoked whenever a client attempts to execute the command. If the callback returns a Tcl error, the command is not executed and the following message is returned to the caller:

```
Authorization Denied
```

If the callback returns a normal Tcl result, the command executes normally.

Sysctl variables also have an authorization callback attached to them which determines the read-access policy for the variable. Therefore, it is possible to create a “private” variable in which you restrict the set of clients who have access to its value.

The **sysctld** server defines a set of commands which are designed to be used as authorization callbacks. These commands provide a simple authorization policy; if more complex authorizations are required, you have the ability to code your own authorization callback procedures.

**NONE** Always returns a normal result. Any command it is attached to can be executed by any user.

**AUTH** Returns a normal result if the user is authenticated via SP authentication services.

**ACL [*file*]** Returns a normal result if the user is authenticated as the local host principal or if the user is listed in the ACL passed as an argument. If an ACL file is not supplied, the default system ACL (defined by the $ACL variable) is used. See **sysctl.acl** for more details about Sysctl ACLs.

**SYSTEM** Always returns a Tcl error. Any command it is attached to can never be executed directly by a user. Instead, these commands are intended to be executed from within a procedure registered with the Sysctl server by way of its configuration file.

**Bypassing Authorization Callbacks**

Under certain circumstances, the authorization callbacks are bypassed by the server. While the checks are bypassed, the user has access to all commands (and read/write access to variables) defined in the server. The authorization callbacks are bypassed while the server is:

- Reading configuration files
- Executing an authorization callback
- Executing the body of a procedure for which a user is authorized to run

Since authorization checks are bypassed during procedure execution, users are able to execute procedures that contain commands they are not authorized to run directly. For example, if a variable has the SYSTEM callback attached to it, a procedure for which a user is authorized can reference or modify the variable even though that user cannot modify or reference it.

**Authorization Variables**

Several variables, accessible as read-only variables in the interpreters or as environment variables, are set by the server prior to executing the client request. These variables provide a mechanism for external commands and procedures to perform their own authorization checking independent of the server's standard authorization checks. They are:

| | |
|---|---|
| **SCHOST** | Specifies the name of the host from which the request was issued. |
| **SCPRINCIPAL** | Indicates the authenticated identity of the issuer. If the user is unauthenticated, it is set to NULL. |
| **SCUSER** | Specifies the base name of the user, or NULL if unauthenticated. |
| **SCINSTANCE** | Specifies the instance of the user, or NULL if unauthenticated. |
| **SCREALM** | Specifies the realm of the user, or NULL if unauthenticated. |
| **SCLHOST** | Specifies the host name of the local server. |
| **SCLREALM** | Specifies the local realm of the server. |
| **SCLPRINCIPAL** | Specifies the principal name of the local **sysctld** server. |
| **SCMODE** | Specifies the communication mode between the client and server, either WAIT, NOWAIT, or SOCKET. NOWAIT indicates that the client is not interested in the result of the operation, and it is discarded by the server. SOCKET indicates that the result of the operation is returned to the client via a TCP/IP socket. |

**Determining Connection Authorization Policy**

Whenever a client connects to the server, the command **svcconnect** is invoked. If the user is not authorized to run this command or if the command returns a Tcl error, the result of the command is returned to the user and the connection is broken. Therefore, the **svcconnect** command determines the connection policy for the server. By default, **svcconnect** simply returns a normal result and its authorization callback is AUTH. This implies that any authenticated user can connect. This policy can be altered by changing the authorization for the **svcconnect** callback (via the **setauth** command), or by redefining the **svcconnect** procedure (via the **create proc** command).

**Server Configuration**

At startup, the server reads a configuration file. By default, this file is named **/etc/sysctl.conf**. The **–f** flag is used to specify an alternate configuration file. The server interprets the contents of the configuration file as Tcl commands. Typically, additional commands and variables are defined in this file. Also, commands are available that instruct the server to read additional configuration files or dynamically load in shared libraries. In this way, the set of commands available to a **sysctld** server is extendable. Refer to the **sysctl.conf** file for more details.

**Signals**

Sending a SIGHUP signal to the server causes it to close the log file, delete and re-create all interpreters, reread the configuration files, and reinitialize the log file. The **svcrestart** command performs the same function.

### Logging

The **sysctld** default log file is **/var/adm/SPlogs/sysctl/sysctld.log**. This default can be overridden with the **–l** command line option, or by by setting the LOG variable in the Sysctl configuration file. Each time a request is received, the **svclogevent** Sysctl command is invoked. By default, it writes a record to the log file giving the identity of the user who sent the request or “unknown” if the user is not authenticated. A different logging policy can be achieved by redefining the **svclogevent** procedure in the server's configuration file.

While running in security audit mode, each line written to the log file is tagged with a *Connection ID* field which is used to filter the audit trail for a particular connection in cases where multiple connections are processed simultaneously.

### Starting and Stopping the sysctld Daemon

The **sysctld** daemon is under System Resource Controller (SRC) control. It uses the signal method of communication in SRC. The **sysctld** daemon is a single subsystem and not associated with any SRC group. The subsystem name is **sysctld**. To start the **sysctld** daemon, use the **startsrc –s sysctld** command. This starts the daemon with the default arguments and SRC options. The **sysctld** daemon is setup to be respawnable and be the only instance of the **sysctld** daemon running on a particular node or control workstation. Do **not** start the **sysctld** daemon from the command line without using the **startsrc** command to start it.

To stop the **sysctld** daemon, use the **stopsrc –s sysctld** command. This stops the daemon and does not allow it to respawn.

To display the status of the **sysctld** daemon, use the **lssrc –s sysctld** command.

If the default startup arguments need to be changed, use the **chssys** command to change the startup arguments or the SRC options. Refer to *AIX Version 4 Commands Reference* and *AIX Version 4 General Programming Concepts: Writing and Debugging Programs* for more information about daemons under SRC control and how to modify daemon arguments when under SRC.

To view the current SRC options and daemon arguments, use the **odmget –q 'subsysname=sysctld' SRCsubsys** command.

## Files

| | |
|---|---|
| **/etc/sysctl.acl** | The default ACL used to assign base authorizations. |
| **/etc/sysctl.conf** | The default configuration file read by the server on startup. |
| **/var/adm/SPlogs/sysctl/sysctld.log** | The default log file. |

## Location

**/usr/lpp/ssp/bin/sysctld**

## Related Information

Command: **sysctl**

Files: **sysctl.acl**, **sysctl.conf**

## Examples

1. To start the **sysctld** daemon, enter:
   ```
   startsrc -s sysctld
   ```
2. To stop the **sysctld** daemon, enter:
   ```
   stopsrc -s sysctld
   ```
3. To display the status of the **sysctld** daemon, enter:
   ```
   lssrc -s sysctld
   ```
4. To display the status of all the daemons under SRC control, enter:
   ```
   lssrc -a
   ```
5. To display the current SRC options and daemon arguments for the **sysctld** daemon, enter:
   ```
   odmget -q 'subsysname=sysctld' SRCsubsys
   ```

## SYSMAN_test

### Purpose

**SYSMAN_test** – Verifies that the installation and customization of the Systems Management components of the SP system completed successfully.

### Syntax

**SYSMAN_test** [–**q** | –**v**] [–**l** *log_file*]

### Flags

| | |
|---|---|
| –**q** | Specifies quiet mode; suppresses all but summary output to standard output. |
| –**v** | Specifies verbose mode, includes informational messages to standard output. |
| –**l** *log_file* | Specifies the path name of the log file to which error messages are written. (This is lowercase **l**, as in **l**ist.) |

### Operands

None.

### Description

The **SYSMAN_test** command performs various tests to determine whether the systems management components of the SP system are completely installed and customized properly.

A return code of 0 indicates that the test completed as expected; otherwise it returns the number of errors. If you do not specify the –**q** flag, a message is displayed on standard output that indicates whether the tests were successful or not. In either case, the command returns 0 if successful, 1 if insuccessful. If errors are detected, more detailed information is recorded in the log file. If you do not specify the –**l** flag, error messages are recorded in **/var/adm/SPlogs/SYSMAN_test.log**.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit SP_verify
```

and select the RS/6000 SP System Management option.

### Files

**/var/adm/SPlogs/SYSMAN_test.log**
Default log file.

## Related Information

Commands: **CSS_test**, **jm_install_verify**, **jm_verify**, **SDR_test**, **spmon_ctest**, **spmon_itest**

## Location

**/usr/lpp/ssp/bin/SYSMAN_test**

## Examples

To verify systems management following customization, saving error messages in **sm.errors** in the current working directory, enter:

```
SYSMAN_test -l sm.errors
```

## syspar_ctrl

### Purpose

**syspar_ctrl** – Starts, stops, adds, deletes, and refreshes the system partition-sensitive subsystems installed on your SP system.

### Syntax

**syspar_ctrl** [–**G**] [–**V**] {–**a** | –**d** | –**s** | –**k** | –**t** | –**o** | –**c** | –**r** | –**h** | –**A** | –**D** | –**E** | –**R**} [*subsystem_name*]

### Flags

–**h** (help) Displays usage information. If a *subsystem_name* is specified, help is provided only for the specified subsystem's control script. Help is displayed as a syntax description and is written to standard output. Once help is displayed, no other action is taken even if other valid options are entered with the –**h** flag.

–**a** (add) Adds all subsystems. If a *subsystem_name* is specified, only the specified subsystem is added. The –**a** flag invokes each subsystem's control script. Typically, this causes each subsystem's control script to add itself to the System Resource Controller (SRC) subsystem, **/etc/inittab** and **/etc/services**. The actual function that is performed depends on whether the underlying control script runs on the control workstation or on a node.

–**A** (add and start) Adds and starts all subsystems. If a *subsystem_name* is specified, only the specified subsystem is added and started. Each subsystem's control script is invoked with the –**a** flag followed by the –**s** flag. This is a convenience option that provides the same function as first calling **syspar_ctrl** with the –**a** flag followed by the –**s** flag.

–**c** (clean) Cleans up after all of the subsystems. If a *subsystem_name* is specified, only the specified subsystem is cleaned up. Each subsystem's control script is invoked with the –**c** flag. Typically, this causes each subsystem's control script to stop any subsystem daemons that may be running and clean or remove all entries for this subsystem from the SRC, **/etc/inittab**, **/etc/services**. This flag is similar to the –**d** (delete) flag, but independent of system partitions. Cleaning up the subsystems is done in the reverse order of how the subsystems are listed in the Syspar Controller subsystems file. You can use this option to clean up subsystem information while trying to get back to some preexisting state, such as when an old System Data Repository (SDR) is restored and the old system partitioning needs to be restored.

–**d** (delete) Deletes all subsystems. If a *subsystem_name* is specified, the specified subsystem is deleted. Each subsystem's control script is invoked with the –**d** flag. Typically, this causes each subsystem's control script to delete itself from the SRC subsystem, **/etc/inittab** and **/etc/services**. Deleting subsystems is done in the reverse order of how the subsystems are listed in the Syspar Controller subsystems file. The actual function that is performed depends on whether the underlying control script runs on the control workstation or on a node.

| | |
|---|---|
| **–D** | (stop and delete) Stops and deletes all subsystems. If a *subsystem_name* is specified, that subsystem is stopped and deleted. Each subsystem's control script is invoked with the **–k** flag followed by the **–d** flag. This is a convenience option that provides the same function as first calling **syspar_ctrl** with the **–k** flag followed by the **–d** flag. |
| **–E** | (examine) Examines all subsystems. If a *subsystem_name* is specified, the specified subsystem is examined in the Syspar Controller subsystems file. Each subsystem name - control script pair in the subsystems file is examined and displayed. Entries that are not valid are noted. An entry is not valid when the control script for a particular subsystem does not exist at the specified location or does not have the correct read and execute permissions. |
| **–G** | (global) Invokes the appropriate underlying subsystem's control scripts for each system partition. If the **–G** flag is not specified, the appropriate underlying subsystem's control script is run only in the current system partition (SP_NAME). |
| **–k** | Stops all subsystems. If a *subsystem_name* is specified, only the specified subsystem is stopped. Each subsystem's control script is invoked with the **–k** flag. Typically, this causes each subsystem's control script to stop any daemons associated with this particular subsystem. Stopping subsystems is done in the reverse order of how the subsystems are listed in the Syspar Controller's subsystem file. The actual function that is performed depends on whether the underlying control script runs on the control workstation or on a node. |
| **–r** | (refresh) Refreshes all subsystems. If a *subsystem_name* is provided, only the specified subsystem is refreshed. Each subsystem's control script is invoked with the **–r** flag. Typically, this causes each subsystem's control script to rebuild configuration data and refresh any daemons associated with this particular subsystem. Subsystems may need to be refreshed when nodes are added to an existing system or the nodes PSSP version changes. The actual function that is performed depends on the subsystem. This option is only meaningful when run on the control workstation. |
| **–R** | (restore) Restores all subsystems. If a *subsystem_name* is specified, only the specified subsystem is restored. All subsystems are stopped and deleted before they are added and started. Each subsystem's control script is invoked with the **–k** flag followed by the **–d** flag, then the **–a** flag followed by the **–s** flag. This is a convenience option that provides the same function as first calling **syspar_ctrl** with the **–D** flag followed by the **–A** flag. |
| **–s** | (start) Starts all subsystems. If a *subsystem_name* is specified, only the specified subsystem is started. Each subsystem's control script is invoked with the **–s** flag. Typically, this causes each subsystem's control script to start any daemons associated with this particular subsystem. The actual function that is performed depends on whether the underlying control script runs on the control workstation or on a node. |

**–t** (trace on) Turns the trace option on for all subsystems. If a *subsystem_name* is specified, the trace option is turned on only for the specified subsystem. Each subsystem's control script is invoked with the **–t** flag.

**Note:** IBM suggests only turning on a particular subsystem's trace by providing a subsystem name. If the trace is turned on for all subsystems, the volume of data produced may quickly fill up **/var**.

**–o** (trace off) Turns the trace option off for all subsystems. If a *subsystem_name* is specified, the trace option is turned off only for the specified subsystem. Each subsystem's control script is invoked with the **–o** flag.

**–V** (verbose) Turns verbose mode on in the **syspar_ctrl** script which then prints out the actual calls it makes to the underlying subsystem control scripts. It also prints out additional information that is useful for debugging.

## Operands

*subsystem_name*
Specifies the subsystem name that you want the command to act on. If a *subsystem_name* is not provided, this command is run for all subsystems that are listed in the Syspar Controller subsystems file (syspar_subsystems). For example, if you only want this command to work with the Event Management subsystem, enter:

```
syspar_ctrl option haem
```

## Description

This command acts as an interface to the system partition-sensitive subsystems supporting the functions that are shared by all subsystems. This command is also referred to as the Syspar Controller. It can be used to add or delete, start or stop, refresh or restore the subsystems, and various other functions. When used on the control workstation, it works with the subsystems on the control workstation. When used on the nodes, it works with the subsystems on the nodes. The refresh option is an exception. To refresh some subsystems, the subsystem must be refreshed on both the control workstation and on the nodes. In this case, the refresh on the control workstation will **dsh** an appropriate refresh command from the control workstation to the appropriate nodes.

This command supports two types of options: primitive options and macro options. Primitive options are passed directly to the underlying control scripts, for example, **–a** (add), **–d** (delete), **–r** (refresh). Macro options conveniently group a commonly used set of primitive options into one option, for example, **–R** (restore). All of the subsystems and each subsystem's control script that are managed by the Syspar Controller are listed in the Syspar Controller subsystems file. By default, all of the control scripts listed in the Syspar Controller subsystems file will be called unless a *subsystem_name* is provided. In that case, the control script for just the specified subsystem will be called.

This command is automatically called when the system is partitioned (**spapply_config**) to first stop and delete the system partition-sensitive subsystems from system partitions that are being removed, and then to add and start the system partition-sensitive subsystems (for example, **hats**, **hb**, and **hr**) in new system partitions.

The Syspar Controller is also called when restoring the SDR with **sprestore_config** to first clean up and then add and start the system partition-sensitive subsystems (for example, **hats**, **hb** and **hr**) in each system partition.

The Syspar Controller also needs to be called with refresh flag (–**r**) by the System Administrator using the command line whenever a node is added or deleted from the system, or a node is migrated to a new level of PSSP.

## Files

**syspar_subsystems**
Lists all of the system partition sensitive subsystems and their control scripts that are controlled by the Syspar Controller. Only the **syspar_ctrl** command should read this file. This file is located in the directory **/usr/lpp/ssp/config/cmi**.

## Security

You must be running with an effective user ID of root.

## Environment Variables

**SP_NAME** **syspar_ctrl** sets the SP_NAME environment variable prior to calling the underlying subsystems. Typically, SP_NAME is set to the value returned from the **spget_syspar -n** command. However, when **syspar_ctrl** is called with the –**G** flag, **syspar_ctrl** sets SP_NAME in turn to each value returned by the **splst_syspars -n** command. The –**c** flag ignores system partition boundaries while all other options respect system partition boundaries.

## Exit Values

**0** Indicates the successful completion of the command.

**1** Indicates that the command was unsuccessful. Most likely a subsystem's control script returned a problem return code.

## Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

## Location

**/usr/lpp/ssp/bin/syspar_ctrl**

## Related Information

Commands: **emonctrl**, **hatsctrl**, **hbctrl**, **hrctrl**, **haemctrl**, **hagsctrl**, **pmanctrl**, **sp_configdctrl**, **spapply_config**, **spcw_apps**, **sprestore_config**

## Examples

1. To add and start all of the system partitions subsystems in each of the system partitions, enter:

   ```
   syspar_ctrl -G -A
   ```

2. To stop and delete all of the system partition subsystems in each of the system partitions, enter:

   ```
   syspar_ctrl -G -D
   ```

3. To refresh all of the system partition subsystems in the current system partition, enter:

   ```
   syspar_ctrl -r
   ```

4. To restore all of the system partition subsystems running in the current system partition, enter:

   ```
   syspar_ctrl -R
   ```

5. To stop all of the system partition subsystems running in the current system partition, enter:

   ```
   syspar_ctrl -k
   ```

6. To get help for the event manager subsystem (haem) control script, enter:

   ```
   syspar_ctrl -h haem
   ```

7. To display a list of all subsystems managed by the Syspar Controller, enter:

   ```
   syspar_ctrl -E
   ```

8. To see the state of the system partition subsystems controlled by the Syspar Controller for system partition spp1, enter the commands:

   ```
   lssrc -a | grep spp1
   lssrc -a | grep sp_configd
   ```

   **Note:** The SDR is not managed by the System Controller.

## sysparaid

### Purpose

**sysparaid** – Creates a layout for a new system partition configuration of an SP system.

### Syntax

**sysparaid** [–**h**] [–**i**] [–**s** *layout_name* **|** *a_fully_qualified_path*] [–**t** *tmpdir*] *input_file* [*topology_file*]

### Flags

–**h** Displays usage information. If the command is issued with the –**h** flag, the syntax description of the command and the startup guidelines are displayed to standard output and no other action is taken (even if other valid flags are entered along with the –**h** flag).

–**i** Creates a switch-map file (**spa.sysinfo**) for the current SP system if a System Data Repository (SDR) is present. If an SDR is not available, it will generate the file for the system described by the *topology_file*. If the –**s** flag is entered along with the –**i** flag, it will be ignored.

–**s** Saves the newly generated layout in the system partition directory tree (**/spdata/sys1/syspar_configs/...**) if a *layout_name* is specified; otherwise, it will be saved under the directory specified by the fully qualified path given.

–**t** Saves the snapshot, performance, and intermediate files under *tmpdir*.

### Operands

*input_file* Specifies the file containing the system partition configuration requirement.

*topology_file*
Specifies the topology file for the system to be partitioned. This operand is required only when the topology file for the system to be partitioned is not under **/spdata/sys1/syspar_configs/topologies**. It is also required with the –**i** flag when there is no SDR or when the switch-map file for a system not represented by the SDR is desired.

### Description

Use this command to invoke the System Partitioning Aid, a tool for generating new system partition configuration layouts. When invoked with the –**i** flag, it creates a switch-map file that will help the user to generate the *input_file* for creating a layout for a desired system partition configuration. When invoked with no flags or with the –**s** or –**t** flags, it attempts to partition the system according to the input requirement. If the attempt is unsuccessful, it will output appropriate error messages to the log and exit. If the attempt is successful and the –**s** flag is specified, the newly created layout will be saved at the desired location specified by the flag argument.

## Extended Description

The **sysparaid** command uses a set of built-in rules to create a layout for a desired system partition configuration. The following startup guidelines will help to generate an acceptable input to the command:

1. The nodes can be identified by either using *node_numbers* or *switch_port_numbers*. While both schemes are permitted for partitioning a system defined by an SDR, *switch_port_numbers* is the only allowed choice when running the tool without an SDR. Also, the numbering schemes cannot be mixed when both schemes are allowed.
2. Identify the four nodes linked to any switch chip to place them in the same system partition. If an SDR is present, the identity of the switch chips linked to the nodes in the system can be obtained by issuing the following command:

   ```
   sysparaid -i -t spa_dir
   ```

   This command places a **spa.sysinfo** file in the **spa_dir** if the –**t** flag is used; otherwise, it places it in the current directory. If an SDR is not present, issue the following command:

   ```
   sysparaid -i -t spa_dir topology_file
   ```

   where *topology_file* is the name of the topology file for the system to be partitioned.

   **Note:** In this case, only the *switch_port_numbers* are provided. No *node_numbers* are available.
3. The keyword “remaining_nodes” can be used for the last system partition provided all nodes or switch ports not in the last system partition were placed in other system partitions. Therefore, the keyword cannot be used with the *node_number* numbering scheme for systems with empty input switch ports.
4. Nodes on a switch board can be part of a maximum of two multichip system partitions.
5. The input file must be formatted according the the template provided in **/spdata/sys1/syspar_configs/bin/inpfile_template**.

## Standard Input

This command requires an input file when invoked with no flag or the –**s** flag. The template for the input file can be found in **/spdata/sys1/syspar_configs/bin**.

## Standard Output

Informational messages are written to standard output.

## Standard Error

Error messages are written to standard error.

## Output Files

This command creates **spa.snapshot** and **spa.metrics** under *tmpdir* (if specified) or under the current working directory. If the –**s** flag is specified and the attempt is successful, it creates the following under the layout directory:

**layout.desc**
**spa.snapshot**

**nodes.syspar** and a system partition directory for each system partition in the layout. Under each system partition directory, it creates the following:

node list
topology
**spa.snapshot**
**spa.metrics**

When invoked with the **–i** flag, the command creates **spa.sysinfo** under *tmpdir* (if specified), or under the current working directory.

## Security

Any user can run this command. Only users authorized to write to the system partitioning directory can save a generated layout under it.

## Location

**/usr/lpp/ssp/bin/sysparaid**

## Related Information

The **spsyspar** command provides the graphical user interface (GUI) for the System Partitioning Aid.

## Examples

1. The following is an example of an input file with the switch port number option (all switch ports linked to nodes):

```
Number of Nodes in System: 32
Number of Frames in System: 2
Frame Type: tall
Switch Type: HiPS
Number of Switches in Node Frames: 2
Number of Switches in Switch Only Frames: 0
Node Numbering Scheme: switch_port_number
Number of Partitions: 3
Partition Name: part1
Number of Nodes in Partition: 8
0 - 7
Partition Name: part2
Number of Nodes in Partition: 8
8 - 15
Partition Name: part3
Number of Nodes in Partition: 16
remaining_nodes
```

   To use **/tmp** as the working directory, enter:

```
sysparaid -t /tmp inpfile
```

   You should receive a message similar to the following:

```
A layout, for the desired system partition configuration or an
equivalent, can be created.
To save this layout, invoke the command again with -s option.
```

   To save the layout for this configuration under **/spdata/sys1/syspar_configs/2nsb0isb/config.8_8_16/layout.myconfig**, enter:

```
sysparaid -s myconfig inpfile
```

To save the layout for this configuration under **/tmp/custom/config1**, enter:

```
sysparaid -s /tmp/custom/config1 inpfile
```

2. The following is an example of an input file with the switch port number option (not all switch ports in the system are linked to nodes):

   ```
   Number of Nodes in System: 87
   Number of Frames in System: 6
   Frame Type: tall
   Switch Type: SP
   Number of Switches in Node Frames: 6
   Number of Switches in Switch Only Frames: 4
   Node Numbering Scheme: switch_port_number
   Number of Partitions: 2
   Partition Name: ProductionPartition
   Number of Nodes in Partition: 82
   0
   4
   16 - 95
   Partition Name: TestPartition
   Number of Nodes in Partition: 5
   2
   6
   8
   10
   12
   ```

   If you enter the **sysparaid -s myconfig inpfile** command, this configuration will be saved under **/spdata/sys1/syspar_configs/6nsb4isb/config.12_84/layout.myconfig**. Note that the nine unspecified switch port numbers have been allocated to one of the two system partitions.

3. The following is an example of an input file with the node number option (not all switch ports are linked to nodes):

   ```
   Number of Nodes in System: 8
   Number of Frames in System: 2
   Frame Type: tall
   Switch Type: SP
   Number of Switches in Node Frames: 1
   Number of Switches in Switch Only Frames: 0
   Node Numbering Scheme: node_number
   Number of Partitions: 3
   Partition Name: part1
   Number of Nodes in Partition: 2
   25
   29
   Partition Name: part2
   Number of Nodes in Partition: 4
   1
   5
   17
   21
   Partition Name: part3
   Number of Nodes in Partition: 2
   3
   7
   ```

This input file for a particular SP system returned the location of an existing layout:

```
The layout for the desired/equivalent system partition configuration is
under /spdata/sys1/syspar_configs/1nsb0isb/config.4_4_8/layout.2
```

4. The **spa.sysinfo** file for the system in Example 3 that was generated using the **sysparaid -i** command follows:

```
switch_number   switch_chip   switch_port_number   node_number
      1             4                 9                25
      1             4                13                29
      1             5                 0                 1
      1             5                 1                17
      1             5                 4                 5
      1             5                 5                21
      1             6                 2                 3
      1             6                 6                 7
```

5. The following is an example of an input file for a switchless system:

```
Number of Nodes in System: 32
Number of Frames in System: 2
Frame Type: tall
Switch Type: NA
Number of Switches in Node Frames: 0
Number of Switches in Switch Only Frames: 0
Node Numbering Scheme: switch_port_number
Number of Partitions: 2
Partition Name: part1
Number of Nodes in Partition: 14
2 - 5
10
11
13
15
19
24 - 25
29 - 31
Partition Name: partition2
Number of Nodes in Partition: 18
remaining_nodes
```

To save the layout for this configuration under **/spdata/sys1/syspar_configs/2nsb0isb/config.14_18/layout.myconfig**, enter:

```
sysparaid -s myconfig inpfile
```

## s1term

### Purpose

**s1term** – Opens a connection to an SP node's S1 serial port.

### Syntax

**s1term** [–**G**] [–**w**] *frame_ID slot_ID*

### Flags

| | |
|---|---|
| –**G** | Allows specification of nodes outside the current system partition. |
| –**w** | Opens the connection in read/write mode. |

### Operands

| | |
|---|---|
| *frame_ID* | Specifies the number of the frame containing the node. |
| *slot_ID* | Specifies the number of the slot containing the node. |

### Description

Use this command to open a connection to the S1 serial port of the SP node contained in the slot specified by the *frame_ID* and *slot_ID* operands. The specified node must be in the current system partition unless the –**G** flag is also specified. By default, the connection is read only. As data arrives from the serial port, it is written to standard output. When the connection is read/write and standard input is a terminal, the terminal is placed in *raw* mode, that is, canonical processing is turned off in the terminal driver. As data is read from standard input, it is sent to the S1 serial port. Standard input and output can be files or pipes.

When the connection is read only, the command terminates upon receipt of a signal, usually generated by the terminal Interrupt key. When in read/write mode, the command terminates when either the termination character or End-of-File is read from standard input. The termination character is **Ctrl-x** by default. Another termination character can be used by setting the S1TERMESC environment variable to the octal (denoted by leading 0), decimal or hexadecimal (denoted by leading 0x) value of the desired termination character.

**Note:** The termination character must only be one byte.

To execute this command, the user must be authorized to access the Hardware Monitor subsystem and, for the frame specified to the command, must be granted S1 permission. Since the Hardware Monitor subsystem uses SP authentication services, the user must execute the **k4init** command prior to executing this command. Alternatively, site-specific procedures can be used to obtain the tokens that are otherwise obtained by **k4init**.

### Location

**/usr/lpp/ssp/bin/s1term**

## Related Information

Commands: **hmcmds**, **hmmon**

## Examples

1. To open an interactive connection to the S1 serial port of the node in slot 8 in frame 12, enter:

   ```
   s1term -w 12 8
   ```

2. To write the output of the S1 serial port of the node in slot 2 in frame 9 to a file, enter:

   ```
   s1term 9 2 > s1term.output
   ```

## tecad_pssp

## Purpose

**tecad_pssp** – Forwards PSSP events.

## Syntax

**tecad_pssp** [–**l** *path/filename*] [–**Cc**]
[–**m** *text*] [–**a** *tiv_admin_name*]
[–**s** *severity*] [–**p** *port*]

## Flags

–**l** *path/filename*
Specifies the *path/filename* of the configuration file. The default value is **/usr/lpp/ssp/donfig/tecad_pssp.cfg**. The only required value in this file is the *ServerLocation* parameter, which should be one of the following:

- *ServerLocation* = **hostname.domain**, for secure communications
- *ServerLocation* = **@ServerName**, for a managed node over a TME channel
- *ServerLocation* = **@ServerName#RegionName**, for secure transport in connected TMRs

Consult the *TME 10 EIF User's Guide* for the correct values of the other configuration parameters.

–**C** Specifies the connection-oriented protocol.

–**c** Specifies a connectionless protocol (the default).

–**m** *text* Adds *text* to the message field of the event.

–**a** *tiv_admin_name*
Adds *admin* in the **T/EC_administrator** field of the event.

–**s** *severity* Sets the severity of the event to *severity*. The following strings are the legal values for *severity*:

FATAL

CRITICAL

WARNING

MINOR

HARMLESS

INDETERMINATE

If an incorrect value is used, the default UNKNOWN is used.

–**p** *port* Sets the communication port number to *port*. Note that you can also set the port number in the configuration file.

## Operands

None.

## Description

The **tecad_pssp** command was designed to be executed by the PSSP Problem Management subsystem. It should not be executed by any other subsystem, since it depends on environment variables that are exported by the Problem Management daemon, **pmand**. Therefore, to forward PSSP events using the **tecad_pssp** command, you need to make a Problem Management subscription using either the SP Event Perspective, or using Problem Management directly. In either case, you should select **tecad_pssp** as the command to run for that subscription, and provide the appropriate parameters.

## Prerequisite Information

*Integrating TME 10 on the RS/6000 SP*

## Related Information

Commands: **pmandef**, **wtdumprl**

## Examples

1. This example creates event subscriptions using the **pmandef** command:

```
pmandef –s example1
        –e "AnyResourceVariable;Any InstanceVector;AnyPredicate"
        –c "$AGENT_PATH/tecad_pssp –l $CONF_PATH/tecad_pssp.cfg"
        –r "AnyRearmPredicate"
        –C "$AGENT_PATH/tecad_pssp –l $CONF_PATH/tecad_pssp.cfg"
        –n 0
```

## ucfghsd

### Purpose

**ucfghsd** – Makes a hashed shared disk unavailable.

### Syntax

**ucfghsd** {–**a** | *hsd_name*...}

### Flags

| | |
|---|---|
| –**a** | Specifies that all the hashed shared disks defined are to be unconfigured. |

### Operands

| | |
|---|---|
| *hsd_name* | Specifies the name of a specific hashed shared disk that is unconfigured. |

### Description

This command unconfigures the already defined hashed shared disks. This command does not change the definition of the hashed shared disks; it makes the hashed shared disks unavailable on one node.

### Security

You must have root privilege to run this command.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/ucfghsd**

### Related Information

Commands: **cfghsd**, **defhsd**, **hsdatalst**, **lshsd**, **undefvsd**

### Examples

To unconfigure the hashed shared disk *hsd1*, enter:

```
ucfghsd hsd1
```

## ucfghsdvsd

### Purpose

**ucfghsdvsd** – Stops the virtual shared disks that comprise a hashed shared disk and makes the hashed shared disk and the virtual shared disks unavailable.

### Syntax

**ucfghsdvsd** –**a** | {*hsd_name*...}

### Flags

| | |
|---|---|
| –**a** | Specifies that all the hashed shared disks defined on this system or system partition are to be unconfigured. |
| *hsd_name* | Specifies the names of defined hashed shared disks that are to be unconfigured. This command unconfigures the underlying virtual shared disks as well. |

### Operands

None.

### Description

Use this command to unconfigure hashed shared disks and their underlying virtual shared disks. This command does not change the definition of the hashed shared disks and virtual shared disks; it just makes them unavailable to the node on which this command is run. The underlying virtual shared disks do not have to be in the stopped state for this command to work. The virtual shared disks will be stopped and then unconfigured.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit hsd_mgmt
```

and select the Unconfigure an HSD and its Underlying Virtual Shared Disks option.

### Security

You must have root privilege or **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/ucfghsdvsd**

## Related Information

Commands: **cfghsdvsd**, **ucfghsd**, **ucfgvsd**

## Examples

To unconfigure the hashed shared disk *hsd1* and the virtual shared disks that comprise it, enter:

```
ucfghsdvsd hsd1
```

## ucfgvsd

### Purpose

**ucfgvsd** – Makes a virtual shared disk unavailable.

### Syntax

**ucfgvsd** {**–a** | *vsd_name* ...}

### Flags

| | |
|---|---|
| **–a** | Specifies that all virtual shared disks in the stopped state are to be unconfigured. |

### Operands

| | |
|---|---|
| *vsd_name* | Specifies a virtual shared disk. |

### Description

The **ucfgvsd** command unconfigures the specified virtual shared disks. This command does not change any virtual shared disk definitions. It moves virtual shared disks from the stopped state to the defined state.

If a configured hashed shared disk is using this virtual shared disk, you must first unconfigure the hashed shared disk before you unconfigure the virtual shared disk.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Unconfigure a Virtual Shared Disk option.

### Security

You must have root privilege to run this command.

### Restrictions

If you have the Recoverable Virtual Shared Disk software installed and operational, do not use this command. The results may be unpredictable.

See *PSSP: Managing Shared Disks*.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/ucfgvsd**

## Related Information

Commands: **cfgvsd**, **ctlvsd**, **lsvsd**, **preparevsd**, **resumevsd**, **startvsd**, **stopvsd**, **suspendvsd**

## Examples

To unconfigure the virtual shared disk **vsd1vg1n1** in the stopped state, enter:

```
ucfgvsd vsd1vg1n1
```

## unallnimres

### Purpose

**unallnimres** – Deallocates Network Installation Management (NIM) resources from a NIM master to one or more NIM clients.

### Syntax

**unallnimres** **–h** | **–l** *node_list*

### Flags

| | |
|---|---|
| **–h** | Displays usage information. If the command is issued with the **–h** flag, the syntax description is displayed to standard output and no other action is taken (even if other valid flags are entered along with the **–h** flag). |
| **–l** *node_list* | Indicates by *node_list* the SP nodes to which to unallocate installation resources. The *node_list* is a comma-separated list of node numbers. |

### Operands

None.

### Description

Use this command to unallocate all NIM resources from a NIM client.

### Standard Error

This command writes error messages (as necessary) to standard error.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **–1** | Indicates that an error occurred. |

### Security

You must have root privilege to run this command.

### Implementation Specifics

This command is part of the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP).

### Location

**/usr/lpp/ssp/bin/unallnimres**

## Related Information

Commands: **allnimres**, **setup_server**

## Examples

To unallocate boot/installation resources to boot/install client nodes 1, 3, and 5 from their respective boot/install servers, enter:

```
unallnimres -l 1,3,5
```

## undefhsd

### Purpose

**undefhsd** – Undefines a hashed shared disk.

### Syntax

**undefhsd** *hsd_name*...

### Flags

None.

### Operands

| | |
|---|---|
| *hsd_name* | Specifies the unique name defined in the SDR names that you want to delete. |

### Description

This command is used to remove a hashed shared disk by removing its definition from the system, including the special device files in **/dev**. The hashed shared disks must be unconfigured and in the defined state on all nodes in the system partition.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Undefine a Hashed Shared Disk option.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/undefhsd**

### Related Information

Commands: **defhsd**, **hsdatalst**

### Examples

To delete the information associated with the hashed shared disk *hsd1* from the SDR, enter:

```
undefhsd hsd1
```

## undefvsd

### Purpose

**undefvsd** – Undefines a virtual shared disk.

### Syntax

**undefvsd** *vsd_name* ...

### Flags

None.

### Operands

*vsd_name* Specifies the virtual shared disk whose underlying logical volume you no longer want to be globally accessed by any virtual shared disk nodes.

### Description

This command is used to remove virtual shared disk definition data from the System Data Repository (SDR) and any special device files from **/dev** for the given *vsd_names* on all the virtual shared disk nodes. The virtual shared disks must be unconfigured and in the defined state on all the virtual shared disk nodes.

You can use the System Management Interface Tool (SMIT) to run the **undefvsd** command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Undefine a Virtual Shared Disk option.

### Security

You must be in the **bin** group to run this command.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/undefvsd**

### Related Information

Command: **defvsd**

### Examples

To delete the information associated with the virtual shared disk **vsd1vg2n1** from the SDR, enter:

```
undefvsd vsd1vg2n1
```

## unfencevsd

### Purpose

**unfencevsd** – Gives applications running on a node or group of nodes access to a virtual shared disk or group of virtual shared disks that were previously fenced from applications running on those nodes.

### Syntax

**unfencevsd** [–**v**] *vsd_name_list* {–**n** *node_list* [–**f**] **|** –**r**}

### Flags

–**v** Specifies one or more virtual shared disk names, separated by commas.

–**n** Specifies one or more node numbers separated by commas.

–**f** Allows a fenced node to unfence itself.

–**r** Removes records associated with virtual shared disks listed in *vsd_name_list* from the SDR.

**Note:** Use **unfencevsd** –**v** –**n** to unfence nodes. Only use –**r** to remove an IBM Virtual Shared Disk fence record from the SDR while no IBM Virtual Shared Disk is configured on any node.

### Operands

None.

### Description

Under some circumstances, the system may believe a node has become inoperable and may begin recovery procedures when the node is actually operational, but is cut off from communication with other nodes running the same application. In this case, the problem node must not be allowed to serve requests for the virtual shared disks it normally manages until recovery is complete and the other nodes running the application recognize the problem node as operational. The **fencevsd** command prevents the problem node from filling requests for its virtual shared disks. The **unfencevsd** command allows fenced nodes to regain access to the virtual shared disks.

This command can be run from any node.

**Note:** This command will be unsuccessful if you do not specify a current server (primary or backup) to a virtual shared disk with the –**v** flag.

**Note:** This command changes SDR attributes when issued with the –**r** flag. Specify –**r** only when disks have already been removed from a fenced virtual shared disk.

## Security

You must have root privilege to run this command.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/unfencevsd**

## Related Information

Commands: **fencevsd**, **lsfencevsd**, **lsvsd**, **updatevsdtab**, **vsdchgserver**

Refer to *PSSP: Managing Shared Disks* for information on how to use this command in writing applications.

## Examples

1. To unfence node 5 from the virtual shared disks vsd1 and vsd2, enter:

   ```
   unfencevsd -v vsd1,vsd2 -n 5
   ```

2. To unfence node 7 from the virtual shared disks vsd1 and vsd2 when the **unfencevsd** command must be entered from node 7, enter:

   ```
   unfencevsd -v vsd1,vsd2 -n 7 -f
   ```

## updatehsd

### Purpose

**updatehsd** – Lets you change the option in the System Data Repository (SDR) that prevents overwriting the Logical Volume Control Block (LVCB) for specified hashed shared disks.

### Syntax

**updatehsd** {–**d** *hsd_names* **|** –**a}** [–**f**]
–**o** {**protect_lvcb | not_protect_lvcb**}

### Flags

–**d** Specifies the names of the hashed shared disks that are the targets of this command.

–**a** Updates the option on all hashed shared disks defined in the system or system partition.

–**o protect_lvcb | not_protect_lvcb**
Specifies whether to skip the first stripe (the LVCB) up to a maximum of 128KB of every virtual shared disk that constitutes the hashed shared disk. **protect_lvcb** specifies skipping the LVCB; **not_protect_lvcb** specifies not skipping it.

–**f** Forces the SDR changes by reconfiguring one or more virtual shared disks on all nodes in the current partition on which those virtual shared disks are currently configured.

### Operands

None.

### Description

Use this command only on the control workstation.

**Note:** This utility is very powerful. Misuse can destroy the contents of a database. Only the superuser should be allowed to run it. If a database has been loaded on a configured hashed shared disk, modifying the *protect_lvcb* or *not_protect_lvcb* option will destroy the database.

The hashed shared disk name must be specified. You must choose either **protect_lvcb** or **not_protect_lvcb**. The hashed shared disk must be defined in the System Data Repository (SDR).

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit set_HSDdd_parms
```

and select the Update Hashed Shared Disk Options. option or

```
smit hsd_mgmtd_parms
```

and select the Set/Show HSD Device Driver Operational Parameters. option or the Update Hashed Shared Disk Options option.

## Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/updatehsd**

## Related Information

Commands: **defhsd**, **lshsd**, **hsdatalst**

## Examples

To set the **protect_lvcb** option for hsdcont01 and hsdcont02, enter:

```
updatehsd -d hsdcont01,hsdcont02 -o protect_lvcb
```

## updatevsdnode

### Purpose

**updatevsdnode** – Changes IBM Virtual Shared Disk subsystem options in the System Data Repository (SDR).

### Syntax

**updatevsdnode** –**n** {**ALL** | *node_number* [,node_number ...]}
{[–**a** {*VSD_adapter* | **none**}]
[–**i** *init_cache_buffer_count*]
[–**m** *max_cache_buffer_count*] [–**r** *vsd_request_count*]
[–**p** *rw_request_count*] [–**b** *min_buddy_buffer_size*]
[–**x** *max_buddy_buffer_size*] [–**s** *max_buddy_buffers*]
[–**M** *vsd_max_ip_packet_size*]}
[–**f**]

### Flags

–**n** Specifies the node numbers of the nodes whose SDR information you want this command to update, or **ALL** nodes in the system or system partition. You can issue the command **/usr/lpp/ssp/install/bin/node_number** to find out the node number of the node you are running on.

–**a** Specifies the adapter name to be used for IBM Virtual Shared Disk communications with this node or nodes. IBM suggests using the switch (adapter name **css0**) for the virtual shared disk for best performance.

–**i** The IBM Virtual Shared Disk device driver implements an optional write-through cache of pinned kernel memory with a block size of 4KB. When the first cached virtual shared disk is configured on a node, the cache is created, and it contains the number of blocks specified in this field of the SDR. The minimum value is 1. IBM suggests using a value of 64, which results in a 256K cache. If you use the switch as the virtual shared disk adapter, no cache buffer is allocated.

–**m** The number of buffers in the cache can be increased up to *max_cache_buffer_count*. You cannot decrease the number; you must unconfigure all the virtual shared disks and start over. IBM suggests using the value of 256, which results in a 1MB cache. If you use the switch as the virtual shared disk adapter, no cache buffer is allocated.

–**r** Specifies the maximum number of outstanding virtual shared disk requests originating on each node. If the number is too small, local requests will queue up waiting for a request block to become available. IBM suggests using the value of 256. The size of the block is approximately 76 bytes.

–**p** Specifies the maximum number of outstanding read/write requests the virtual shared disk will make to each underlying logical volume. The minimum value is 1. IBM suggests using a value of 48. **WARNING: generally this value should not exceed 192. For details see the information on pufs in the chapter on "Performance and Tuning Considerations for Virtual Shared Disks" in** *PSSP: Managing Shared Disks*.

**–b** Specifies the smallest buddy buffer a server uses to satisfy a remote request to a virtual shared disk. This value must be a power of 2 and greater than or equal to 4096. IBM suggests using the value of 4096 (4KB).

**–x** The largest buddy buffer a server will use to satisfy a remote request. This value must be a power of 2 and greater than or equal to the *min_buddy_buffer_size*. IBM suggests using the maximum value of 65536 (64KB). This value must be the same on all nodes within a system partition.

**–s** The size of the buddy buffer affects the number of remote requests the virtual shared disk server node can handle at one time. Remote requests can queue waiting for a buddy buffer. **statvsd** reports this queuing as buddy buffer shortages. Use the output from **statvsd** to select a buddy buffer size for your environment. When the switch is used as the virtual shared disk adapter, IBM suggests using a value of at least 4, which results in a 256KB combined buddy buffer size in combination with the default *max_buddy_buffer_size* of 64KB.

**–M** Specifies the maximum IP message size for virtual shared disks, in bytes. The default value is 24KB (24,576). If you are using the switch as your virtual shared disk adapter, use a value of 60KB (61,440).

**–f** Specifies that this command will force the SDR changes by reconfiguring one or more virtual shared disks on all nodes in the current partition on which those virtual shared disks are currently configured.

## Operands

None.

## Description

Use **updatevsdnode** to change the specified values in the SDR for all nodes in *node_list*.

**Note:** This command only changes the information in the SDR. To effectively configure the virtual shared disks, you must first unconfigure all the virtual shared disks and then reconfigure them.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Set/Show Virtual Shared Disk Device Driver Operational Parameters option or the Update IBM Virtual Shared Disk Device Driver Node Parameters option.

## Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/updatevsdnode**

## Related Information

Commands: **lsvsd**, **vsdatalst**, **vsdnode**

## Examples

To change buddy buffer options after you've installed a new SP Switch Adapter, enter:

```
updatevsdnode -n ALL -b 4096 -x 4 -s 5
```

This command leaves *min_buddy_buffer_size* at 4KB, the default, lowers the *max_buddy_buffer_size* to 4KB as well, and allocates a maximum of 5 buddy buffers.

## updatevsdtab

### Purpose

**updatevsdtab** – Changes the IBM Virtual Shared Disk subsystem option to set cache or nocache in the System Data Repository (SDR).

### Syntax

**updatevsdtab** {–**v** *vsd_names* **|** –**a**} {[–**o** {**cache | nocache**}] [–**s**]} [–**f**]

### Flags

–**v** *vsd_names*
: Specifies a list of virtual shared disk names to be updated.

–**a**
: Specifies that the option is to be changed on all nodes of the system or system partition.

–**o cache | nocache**
: Specifies either the **cache** or the **nocache** option. The default is **cache**.

–**s**
: Updates the virtual shared disk size after the associated logical volume size is changed.

–**f**
: Forces SDR changes by reconfiguring a virtual shared disk on all nodes in the current system partition on which the virtual shared disk is configured.

### Operands

None.

### Description

| Use this command to update the SDR, if necessary. When a feature of the virtual shared disk in the SDR (such as cache/nocache option or the virtual shared disk size), is changed using the **updatevsdtab** command, the change will not take effect until the virtual shared disk is unconfigured and configured again.

If the –**f** flag is specified, the virtual shared disks involved will be reconfigured (using **sysctl**) on all nodes that are up and initially had these virtual shared disks configured.

You can use the System Management Interface Tool (SMIT) to run this command. To use SMIT, enter:

```
smit vsd_mgmt
```

and select the Set/Show IBM Virtual Shared Disk Device Driver Operational Parameters option or the Update IBM Virtual Shared Disk Options option.

### Security

You must have **sysctl** and **sysctl.vsd** access and authorization from your system administrator to run this command.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/updatevsdtab**

## Related Information

Commands: **defvsd**, **updatevsdnode**

## Examples

1. To change the cache default for all virtual shared disks on a system or system partition, enter:

   ```
   updatevsdtab -a -o nocache
   ```

2. To reset the size of the virtual shared disk named USER1n3, enter:

   ```
   updatevsdtab -v USER1n3 -s
   ```

## updauthfiles

### Purpose

**updauthfiles** – Updates and creates if necessary, **.rhost** files on the control workstation and all nodes in the system for which Standard AIX is defined as an Authentication Method.

### Syntax

**updauthfiles** [–**h**]

### Flags

| | |
|---|---|
| –**h** | Presents syntax message. |

### Operands

None.

### Description

**updauthfiles** will determine the authorization methods for the local machine and create or update the appropriate authorization file. The control workstation's authorization files will be updated according to the union of all authentication methods in the system.

### Environment Variables

SDR attribute **auth_root_rcmd** for the Syspar class, will have an instance per Syspar object defining which authorization methods should be used.

### Security

You must have root user authority to run this command.

### Standard Input

This command will read the **.rhost** files.

### Standard Output

This command will update the **.rhost** file.

A Log file is created in **/var/adm/SPlogs/auth_inst/log**.

### Exit Values

| | |
|---|---|
| **0** | Indicates successful completion of the command. |
| **1** | Indicates a problem running the command. Review the **log** file for the specific problem. |

## Consequences of Error

One or more nodes will not have correct authorization files. This could possibly cause errors when remote commands are issued causing problems with some PSSP administrative tasks.

## Related Information

Commands: **rsh**

Files: **.rhost**

## Location

**/usr/lpp/ssp/bin/updauthfiles**

## Examples

1. To create appropriate authorization file (**.rhost**) on the control workstation, enter:

   ```
   /usr/lpp/ssp/bin/updauthfiles
   ```

## verparvsd

### Purpose

**verparvsd** – Verifies IBM Virtual Shared Disk system partitioning.

### Syntax

**verparvsd** [–**F**] [–**o** *output_file*] *layout_directory* [*new_partition* ...]

### Flags

–**F**
Returns success if correctable virtual shared disk errors are found in the system partitioning operation. This flag is the same as **spapply_config** –**F** and is used only when **spapply_config** invokes **verparvsd** when it is invoked with –**F**.

–**o**
Specifies the file where the System Data Repository (SDR) commands are placed to load the IBM Virtual Shared Disk data in the new system partitions. If –**o** is not specified, the output is placed in the **/spdata/sys1/vsd/partitionVSDdata** file.

### Operands

*layout_directory*
Specifies the *layout_directory* that describes the new system partitions that the user wants to apply, and wants **verparvsd** to verify for IBM Virtual Shared Disk system partitioning. This operand is used as the first argument in the invocation of the **spdisplay_config** command. Refer to the **spdisplay_config** command for more details.

*new_partition* **...**
Specifies the list of new system partitions to be processed. If some system partitions are going to be unaffected by the system partitioning operation implied by the *layout_directory*, and you do not want **verparvsd** to look at them, do not list them here, but list only the system partitions being affected. The **verparvsd** command only verifies and processes the system partitions passed as arguments. If no new system partitions are given as arguments, **all** system partitions in *layout_directory* are processed and analyzed. The **spapply_config** command invokes **verparvsd** listing only the new, changing system partitions.

### Description

Use this command to verify that the system partition proposed in the *layout_directory* will work for all the existing IBM Virtual Shared Disk data. The **spapply_config** command invokes this command to partition the IBM Virtual Shared Disk data during a system partition operation. The **verparvsd** command extracts all IBM Virtual Shared Disk data from nodes involved in the system partitioning and writes SDR commands to the output file that will reload the IBM Virtual Shared Disk SDR data into the correct new system partitions. This file is executed during the system partitioning process to partition the IBM Virtual Shared Disk data.

The **spapply_config** command invokes this command and its output to effect IBM Virtual Shared Disk system partitioning. You can also invoke the command prior to invoking the **spapply_config** command to see how well suited the desired layout is for the existing IBM Virtual Shared Disk configuration as defined in the SDR.

This command only checks and processes the new system partitions listed on the command line. If some existing system partitions are to be unchanged in the system partitioning operation, do not list those system partition names on the command line. If no new system partitions are listed, the default is to process all system partitions in the layout directory.

This command checks to see if the IBM Virtual Shared Disk data can be partitioned as specified by the layout directory without any problems. The command reports any problems it identifies, as well as reports how it would *fix* the problem.

The **verparvsd** command places global volume groups (GVGs) in the system partition containing their primary server node. Virtual shared disks are placed in the system partition of their GVG. HSDs are placed in the system partition containing their first virtual shared disk.

The **verparvsd** command looks for the following types of errors in each new system partition:

1. Inconsistent VSD_adapter Node attributes. If any are found, the VSD_adapter field is set to en0 for all virtual shared disk nodes in the system partition.
2. Inconsistent VSD_max_buddy_buffer_size Node attributes. The **verparvsd** command sets the VSD_max_buddy_buffer_size field for all virtual shared disk nodes in the system partition to the largest value of any node in the system partition, and adjusts the VSD_max_buddy_buffers so that the buddy buffer is still the same size, or just minimally larger than it was before on each node.
3. A twin-tailed GVG with primary and secondary server nodes in different system partitions. GVGs are placed in the system partition of the primary server. If the secondary is in a different system partition, the **verparvsd** command will set the secondary server to NULL, making the GVG have only one server, the primary.
4. An HSD with virtual shared disks in more than one system partition. The **verparvsd** command appends .BAD to the HSD's name. These HSDs would be unusable if the new system partition were applied and the VSD_adapter was **css0**.

   As a corollary, if an HSD with .BAD at the end of its name is found in the new system partition to have all its virtual shared disks in the system partition, the .BAD will be removed from its name.
5. Any duplicate GVG, virtual shared disk, or HSD name. The **verparvsd** command keeps the original name for the first name it encounters, but makes up unique names for any subsequent duplicate names encountered. New names follow the following suggested naming conventions:

| | |
|---|---|
| **GVG** | vg01n01 for single tailed GVG on node 1.<br>vg01p0ss02 for twin tailed GVGs, primary server node 1, secondary server node 2. |
| **VSD** | vsd01vg01n01 (for example, vsdnn*GVG name*) |
| **HSD** | hsd01 (for example, hsdnn) |

## Files

**/spdata/sys1/vsd/partitionVSDdata**
The default location of the output file containing all the SDR commands to correctly system partition the IBM Virtual Shared Disk data.

## Exit Values

The **verparvsd** command looks for error types (described previously) in each new system partition and corrects them as specified:

- Without –**F**, any single error of the preceding type for even a single system partition causes **verparvsd** to return an error code upon completion.
- With –**F**, all errors of all of the types described previously are corrected as described, and **verparvsd** returns a zero (successful) return code.

In either case, **verparvsd** processes all the IBM Virtual Shared Disk data, and generates a complete list of errors on standard error, and a complete SDR command list to the output file.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Related Information

Commands: **defhsd**, **defvsd**, **spapply_config**, **spdisplay_config**, **vsdnode**, **vsdvg**

## Examples

To see how well suited the configuration specified in the **config.4_4_8/layout.6** layout directory is to your IBM Virtual Shared Disk configuration, enter:

```
verparvsd config.4_4_8/layout.6
```

## vhostname

### Purpose

**vhostname** – Sets or displays the virtual host name.

### Syntax

**vhostname** [–**s**] [*host_name*]

### Flags

| | |
|---|---|
| –**s** | Trims any domain information from the printed name. |

### Operands

| | |
|---|---|
| *host_name* | Sets the virtual host name to *host_name*. **Note:** You must have root authority to use the *host_name* operand. |

### Description

Use this command to display or set the virtual host name of the local host. Only users with root authority can set the virtual host name. The *host_name* is stored in the **/etc/vhostname** file.

If displaying the virtual host name and the virtual host name has not been set and the **/etc/vhostname** file does not exist, **vhostname** will return the real host name from the kernel variable.

When setting the virtual host name, if the **/etc/vhostname** file does not exist, it will be created. If it does exist, the file contents will be overwritten by the new virtual host name.

To clear the virtual host name so that the virtual host name no longer exists, remove the **/etc/vhostname** file.

**Note:** You must have root authority to remove the **/etc/vhostname** file.

The virtual host name is used in fail over situations when an application has associated the host name in the kernel of a particular machine to the service it is providing. When the application is restarted on the fail over node that has a different host name, the application may not work or work incorrectly. If the application needs to associate a host name with a particular service and it cannot handle having multiple host names, a virtual host name can be provided. The application can call **vhostname** instead of **hostname** and get the host name of the node it normally runs on. This eliminates the need to change the real host name in the kernel on the fail over node. It should be noted that changing the real host name in the kernel can cause problems with other applications that rely on the real host name in the kernel to identify the *physical machine*.

**Note:** The High Availability Cluster Multiprocessing (HACMP) event scripts provided with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP) set and clear the virtual host name in the HACMP pre- and post-event scripts. The administrator normally should not have to set or clear the virtual host name.

## Files

**/etc/vhostname**
Contains the virtual host name.

## Exit Values

| | |
|---|---|
| **0** | Indicates that if a parameter was used, a virtual host name was successfully set. If a parameter was not used, either a virtual or real host name was printed out. |
| **1** | Indicates that an error occurred. |

## Related Information

Subroutines: **getvhostname**, **setvhostname**

AIX command: **hostname**

AIX Subroutines: **gethostname**, **sethostname**

## Examples

1. To display the virtual host name, enter:

   ```
   vhostname
   ```

2. To set the virtual host name to **spcw_prim**, enter:

   ```
   vhostname spcw_prim
   ```

3. To display the virtual host name and trim domain information for host **donald.ibm.com**, enter:

   ```
   vhostname -s
   ```

   A vhostname of **donald** prints out.

4. To clear the virtual host name so it no longer exists, enter:

   ```
   rm /etc/vhostname
   ```

   **Note:** You must have root authority to remove the **/etc/vhostname** file.

## vsdatalst

### Purpose

**vsdatalst** – Displays IBM Virtual Shared Disk subsystem definition data from the System Data Repository (SDR).

### Syntax

**vsdatalst** [–**G**] {–**g** | –**n** | –**v**}

### Flags

–**G** Displays information for all system partitions on the SP, not only the current system partition.

Only one of the following flags can be specified with each invocation of **vsdatalst**:

–**g** Displays the following SDR virtual shared disk global volume group data:

*global_group_name*,
*local_group_name*,
*primary_server_node*,
*secondary_server_node*. (This is only enabled with the Recoverable Virtual Shared Disk subsystem.)
*eio_recovery*
*recovery*

–**n** Displays the following SDR virtual shared disk Node data:

*node_number*,
*host_name*,
*adapter_name*,
*init_cache_buffer_count*,
*max_cache_buffer_count*,
*rw_request_count*,
*vsd_request_count*,
*min_buddy_buffer_size*,
*max_buddy_buffer_size*,
*max_buddy_buffers*.

–**v** Displays the following SDR virtual shared disk definition data:

*vsd_name*,
*logical_volume_name*,
*global_group_name*,
*minor_number*,
*option (cache|nocache)*.

### Operands

None.

## Description

Use this command to display one of several kinds of information to standard output.

You can use the System Management Interface Tool (SMIT) to run the **vsdatalst** command. To use SMIT, enter:

```
smit list_vsd
```

and select the option for the kind of IBM Virtual Shared Disk SDR information you want to see.

## Security

You must be in the **bin** group to run this command.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Related Information

Commands: **lsvsd**, **updatevsdnode**, **vsdnode**

## Examples

1. To display SDR virtual shared disk global volume group data, enter:

   ```
   vsdatalst -g
   ```

   The system displays a message similar to the following:

   **Note:** *backup* or *secondary_server_node* is only enabled with the Recoverable Virtual Shared Disk subsystem.

   ```
   VSD Global Volume Group Information
   Global Volume     Local     Server Node  Numbers:  eio_
   Group name        VG name   primary      backup    recovery  Recovery
   ----------------  --------  -----------  --------  --------  --------
   hunter-rileysvg   rileysvg       1          0          0         0
   ppstest1-rootvg   rootvg         3          0          0         0
   tattooine-rootvg  rootvg         2          0          0         0
   ```

2. To display SDR virtual shared disk node data, enter:

   ```
   vsdatalst -n
   ```

   The system displays a message similar to the following:

   ```
   VSD Node Information
                             Initial Maximum  VSD    rw    Buddy Buffer:
   node           VSD    cache   cache    req.   req.  min.  max.    size: #
   #    host_name adapt. buffers buffers  count  count size  size    maxbufs
   ---- --------- ------ ------- -------  -----  ----- ---- ----------------
    1   hunter     tr0     64      256     256    48   4096  65536     4
    2   tattooine  tr0     64      256     256    48   4096  65536     4
    3   ppstest1   tr0     64      256     256    48   4096  65536     4
   ```

3. To display SDR virtual shared disk definition data, enter:

   ```
   vsdatalst -v
   ```

   The system displays a message similar to the following:

```
VSD Table

VSD name          logical volume  Global Volume Group     minor# option
----------------- --------------- ----------------------- ------ ------
vsd.rlv01         rlv01           hunter-rileysvg              2 cache
vsd.rlv02         rlv02           hunter-rileysvg              3 cache
vsd.vsd1          vsd1            tattooine-rootvg             1 nocache
vsd.vsdp1         vsd1            ppstest1-rootvg              4 nocache
```

## vsdchgserver

### Purpose

**vsdchgserver** – Switches the server function for one or more virtual shared disks from the node that is currently acting as the server node to the other.

### Syntax

**vsdchgserver** –**g** *vsd_global_volume_group_name* –**p** *primary_node*
[–**b** *secondary_node*] [–**o** *EIO_recovery*]

### Flags

–**g** Specifies the Global Volume Group name for the volume group that represents all the virtual shared disks defined on a particular node.

–**p** Specifies the node number defined as the primary server node for the global volume group specified with the –**g** flag. The value of the –**p** option must be the same as the current acting server of the global volume group.

–**b** Specifies the node number defined as the secondary server node for the global volume group specified with the –**g** flag. If the –**b** flag is not specified, it will set the *secondary_node* to undefined in the System Data Repository (SDR). If the current *secondary_node* in the SDR is not defined and the –**b** flag is specified, the **vsdchgserver** command will set the *secondary_node* for the global volume group specified in the –**g** flag.

–**o** Specified as **0**, for no recovery on an EIO error, or **1**, for recovery on an EIO error. The default is the current value defined in the SDR.

### Operands

None.

### Description

The **vsdchgserver** command allows the serving function for a global volume group defined on a primary node to be taken over by the secondary node, or to be taken over by the primary node from the secondary node. This allows an application to continue to use virtual shared disks in situations where the cable or adapter between the physical disks and one of the attached nodes is not working.

The Recoverable Virtual Shared Disk subsystem automatically updates the virtual shared disk devices if, and only if, the **vsdchgserver** command is used to flip the currently-defined primary node and secondary node in the global volume group specified in the –**g** flag.

### Security

You must have root privilege to run this command.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Related Information

Refer to *PSSP: Managing Shared Disks* for information on how to use this command in writing applications.

## Location

**/usr/lpp/csd/bin/vsdchgserver**

## Examples

To change the primary server node for the global volume group node12vg to node 1 and the secondary node to node 2, with EIO recovery, enter:

```
vsdchgserver -g node12vg -p 1 -b 2 -o 1
```

## vsddiag

### Purpose

**vsddiag** – Displays information about the status of virtual shared disks.

### Syntax

**vsddiag**

### Flags

None.

### Operands

None.

### Description

This command displays information about virtual shared disks that can help you determine their status and collect information that helps IBM service representatives diagnose system problems.

**Note:** The **vsddiag** command can only be used when no virtual shared disk I/O is in progress.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/vsddiag**

### Related Information

Commands: **vsdatalst**, **vsdsklst**

### Examples

To display information about the virtual shared disks in your system or system partition, enter:

```
vsddiag
```

If all virtual shared disks are created and configured correctly, the output is:

```
Checking server vsds
Checking VSD request sequence number.
Checking device drivers.
end of vsdl1diag:checkvsdl1 program.
```

If there are no virtual shared disks defined, the output is:

```
k5n02.ppd.pok.ibm.com
VSD_ERROR:3:No IBM Virtual Shared Disks are configured on this node.

k5n01.ppd.pok.ibm.com
VSD_ERROR:3:No IBM Virtual Shared Disks are configured on this node.

Checking server vsds
Checking VSD request sequence number.
Checking device drivers.
end of vsdl1diag:checkvsdl1 program.
```

If there is something wrong with the virtual shared disks, the output is:

```
k5n02.ppd.pok.ibm.com
VSD_ERROR:3:No IBM Virtual Shared Disks are configured on this node.

k5n01.ppd.pok.ibm.com
VSD_ERROR:3:No IBM Virtual Shared Disks are configured on this node.

Checking server vsds
Checking VSD request sequence number.
Checking device drivers.
vsdl1diag:checkvsdl1: 0034-619 Device driver on node 14 is not at the
same level as others on this SP system or system partition.
vsdl1diag:checkvsdl1: 0034-620 VSD Maximum IP Message Size on node 14 is
not at the same level as others on this SP system or system partition.
```

## vsdelnode

### Purpose

**vsdelnode** – Removes IBM Virtual Shared Disk information for a node or series of nodes from the System Data Repository (SDR).

### Syntax

**vsdelnode** *node_number* ...

### Flags

None.

### Operands

| | |
|---|---|
| *node_number* | Specifies the number attribute assigned to a node in the SDR. |

### Description

This command is used to remove IBM Virtual Shared Disk data for a node or series of nodes from the SDR.

The **vsdelnode** command makes the listed nodes no longer virtual shared disk nodes so that no virtual shared disks can be accessed from them. This command is unsuccessful for any nodes that are servers for any global volume groups.

You can use the System Management Interface Tool (SMIT) to run the **vsdelnode** command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Delete Virtual Shared Disk Node Information option.

### Security

You must be in the **bin** group to run this command.

### Restrictions

If you have the Recoverable Virtual Shared Disk software installed and operational, do not use this command. The results may be unpredictable.

See *PSSP: Managing Shared Disks*.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Related Information

Commands: **vsdatalst**, **vsdnode**

## Examples

To delete virtual shared disk node information for nodes **3** and **6**, enter:

```
vsdelnode 3 6
```

## vsdelvg

### Purpose

**vsdelvg** – Removes virtual shared disk global volume group information from the System Data Repository (SDR).

### Syntax

**vsdelvg** [–**f**] *global_group_name* ...

### Flags

| | |
|---|---|
| –**f** | Forces the removal of any virtual shared disks defined on this global volume group. |

### Operands

| | |
|---|---|
| *global_group_name* | Specifies the volume group that you no longer want to be global to the system. |

### Description

Use this command to remove virtual shared disk global volume group information from the SDR. If any virtual shared disks are defined on a global volume group, the **vsdelvg** command is unsuccessful unless –**f** is specified. If –**f** is specified, any such virtual shared disks must be unconfigured and in the defined state on all the virtual shared disk nodes to be deleted.

You can use the System Management Interface Tool (SMIT) to run the **vsdelvg** command. To use SMIT, enter:

```
smit delete_vsd
```

and select the Delete Virtual Shared Disk Global Volume Group Information option.

### Security

You must be in the **bin** group to run this command.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/vsdelvg**

### Related Information

Commands: **undefvsd**, **vsdatalst**, **vsdvg**

## Examples

To delete the virtual shared disk information associated with global volume group **vg1n1** from the SDR, enter:

```
vsdelvg vg1n1
```

## vsdnode

### Purpose

**vsdnode** – Enters IBM Virtual Shared Disk information for a node or series of nodes into the System Data Repository (SDR).

### Syntax

**vsdnode** *node_number... adapter_name init_cache_buffer_count max_cache_buffer_count vsd_request_count rw_request_count min_buddy_buffer_size max_buddy_buffer_size max_buddy_buffers vsd_max_ip_msg_size*

### Flags

None.

### Operands

| | |
|---|---|
| *node_number* | Specifies the node or nodes whose virtual shared disk information is to be set as identified by the *node_number* attribute of the SDR node class. |
| *adapter_name* | Specifies the adapter name to be used for virtual shared disk communications for the nodes specified. The adapter name must already be defined to the nodes. Note that the nodes involved in IBM Virtual Shared Disk support must be fully connected so that proper communications can take place. Use **css0** to specify that the IBM Virtual Shared Disk device driver transmits data requests over the SP Switch. The **css0** adapter will be used the next time the IBM Virtual Shared Disk device driver is loaded. |
| *init_cache_buffer_count* | Specifies the number of 4KB blocks you want to assign to an optoinal cache if you do not use the switch as your adapter. The recommended value is 256. |
| *max_cache_buffer_count* | Specifies the maximum number of buffers to be used for virtual shared disk caching for the nodes specified. The recommended initial value is 256. If you use the switch as your adapter, no cache buffer is allocated. |
| *vsd_request_count* | Specifies the number of outstanding virtual shared disk requests for the nodes specified. The recommended value is 256. |
| *rw_request_count* | Specifies the number of outstanding read and write requests the virtual shared disk issues at one time to each logical volume for the nodes specified. The recommended value for a server node is 48. |
| *min_buddy_buffer_size* | Specifies the smallest buddy buffer a server uses to satisfy a remote request to a virtual shared disk. This value must be a power of 2 and greater than or equal to 4096. IBM suggests using a value of 4096 (4KB). For a 512 byte request, 4KB is excessive. However, recall that |

| | |
|---|---|
| | a buddy buffer is only used for the short period of time while a remote request is being processed at the server node. |
| *max_buddy_buffer_size* | Specifies the largest buddy buffer a server uses to satisfy a remote noncached request. This value must be a power of 2 and greater than or equal to the *min_buddy_buffer_size*. IBM suggests using a value of 262144 (256KB). This value depends on the I/O request size of applications using the virtual shared disks and the network used by the IBM Virtual Shared Disk software. |
| *max_buddy_buffers* | This value should be the same on all nodes in a system partition. The buddy buffer is pinned kernel memory allocated when the IBM Virtual Shared Disk device driver is loaded the first time a virtual shared disk is configured (and freed when the last virtual shared disk is unconfigured). The size of the buddy buffer affects the number of remote requests the virtual shared disk server node can handle at one time. Remote requests can queue while waiting for a buddy buffer. **statvsd** reports this queuing as buddy buffer shortages. Use this number to select the buddy buffer size for your environment. IBM suggests using a value of 2 initially, resulting in a 512KB buddy buffer for the switch. **Note:** If the application issues requests larger than 64KB, the number of buddy buffers can be increased, depending on the size of the requests. |
| *vsd_max_ip_msg_size* | Specifies the maximum message size in bytes for virtual shared disks. If you use SMIT to define the virtual shared disk node, the default is 24576 (24KB); otherwise, the default is none. If you are using the switch as your adapter, the recommended value is 61440 (61KB). |

## Description

Use this command to make the specified nodes virtual shared disk nodes and to assign their IBM Virtual Shared Disk operational parameters. The operational parameters are: adapter name, initial cache buffer count, maximum cache buffer count, read/write request count, virtual shared disk request count, and buddy buffer parameters. If this information is the same for all nodes, run this command once. If the information is different for the nodes, run this command once for each block of nodes that should have the same virtual shared disk information.

You can use the System Management Interface Tool (SMIT) to run the **vsdnode** command. To use SMIT, enter:

```
smit vsd_data
```

and select the IBM Virtual Shared Disk Node Information option.

## Security

You must be in the **bin** group to run this command.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/vsdnode**

## Related Information

Commands: **updatevsdnode**, **vsdatalst**, **vsdelnode**

Refer to *PSSP: Managing Shared Disks* for defining virtual shared disk information in the SDR.

## Examples

The following example adds SDR information for a **css0** network and nodes 1 through 8 with **css0** being the adapter name used for virtual shared disk communications. Initially, this is accomplished with 64 cache buffers and with a maximum of 256 cache buffers to be used for IBM Virtual Shared Disk support, with allowance for 48 outstanding read/write requests per virtual shared disk and 256 outstanding virtual shared disk requests per node with two 256KB *max_buddy_buffers* and a 4KB *min_buddy_buffer_size*.

```
vsdnode 1 2 3 4 5 6 7 8 css0 64 256 256 48 4096 65536 2 262144
```

## vsdsklst

### Purpose

**vsdsklst** – Produces output that shows you the disk resources used by the IBM Virtual Shared Disk subsystem across a system or system partition.

### Syntax

**vsdsklst** [–**v**] [–**d**] {–**a** | –**n** *node_number*[, node_number2, ...]} [–**G**]

### Flags

–**v** Displays only disk utilization information about volume groups and the virtual shared disks associated with them.

–**d** Displays only disk utilization information about volume groups and the physical disks associated with them.

–**a** Displays specified information for all nodes in the system or system partition.

–**n** *node_number*
Lists one or more node numbers for which information is to be displayed.

–**G** Displays global disk information (across system partitions).

### Operands

None.

### Description

Use this command to check disk utilization across a system or system partition.

### Prerequisite Information

*PSSP: Managing Shared Disks*

### Location

**/usr/lpp/csd/bin/vsdisklist**

### Related Information

Command: **vsdatalst**

### Examples

This command:

```
vsdsklst -dv -a
```

displays the following information on a system that has volume groups and virtual shared disks defined on nodes 1, 3, 5, 7, 10, and 12. Node 5 is temporarily inactive.

```
k7n12.ppd.pok.ibm.com
Node Number:12; Node Name:k7n12.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:315
        Physical Disk:hdisk0; Total:537; Free:315
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD8n12{lv1HsD8n12}; Size:2
        VSD Name:1HsD20n12{lv1HsD20n12}; Size:2


k7n01.ppd.pok.ibm.com
Node Number:1; Node Name:k7n01.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:210
        Physical Disk:hdisk0; Total:537; Free:210
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD1n1{lv1HsD1n1}; Size:2
        VSD Name:1HsD13n1{lv1HsD13n1}; Size:2


k7n05.ppd.pok.ibm.com
No response


k7n10.ppd.pok.ibm.com
Node Number:10; Node Name:k7n10.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:303
        Physical Disk:hdisk0; Total:537; Free:303
        VSD Name:vsdn10v1{lvn10v1}; Size:4
        VSD Name:vsdn10v2{lvn10v2}; Size:4
        VSD Name:vsdn10v3{lvn10v3}; Size:4
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD6n10{lv1HsD6n10}; Size:2
        VSD Name:1HsD18n10{lv1HsD18n10}; Size:2


k7n03.ppd.pok.ibm.com
Node Number:3; Node Name:k7n03.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:269
        Physical Disk:hdisk0; Total:537; Free:269
        VSD Name:vsdn03v1{lvn03v1}; Size:4
        VSD Name:vsdn03v2{lvn03v2}; Size:4
        VSD Name:vsdn03v3{lvn03v3}; Size:4
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD2n3{lv1HsD2n3}; Size:2
        VSD Name:1HsD14n3{lv1HsD14n3}; Size:2
```

```
k7n07.ppd.pok.ibm.com
Node Number:7; Node Name:k7n07.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:300
        Physical Disk:hdisk0; Total:537; Free:300
        VSD Name:vsdn07v1{lvn07v1}; Size:4
        VSD Name:vsdn07v2{lvn07v2}; Size:4
        VSD Name:vsdn07v3{lvn07v3}; Size:4
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD4n7{lv1HsD4n7}; Size:2
        VSD Name:1HsD16n7{lv1HsD16n7}; Size:2
```

To view the output for a specific node, type:

```
vsdsklst -n 12
```

The output is:

```
k7n07.ppd.pok.ibm.com
Node Number:7; Node Name:k7n07.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:300
        Physical Disk:hdisk0; Total:537; Free:300
        VSD Name:vsdn07v1{lvn07v1}; Size:4
        VSD Name:vsdn07v2{lvn07v2}; Size:4
        VSD Name:vsdn07v3{lvn07v3}; Size:4
    Volume group:vsdvg; Partition Size:4; Total:537; Free:533
        Physical Disk:hdisk1; Total:537; Free:533
        VSD Name:1HsD4n7{lv1HsD4n7}; Size:2
        VSD Name:1HsD16n7{lv1HsD16n7}; Size:2
```

If both the rootvg and testvg volume groups are varied on, the system displays output similar to the following:

```
Node Number:12; Node Name:k21n12.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:47
        Physical Disk:hdisk0; Total:537; Free:47
        VSD Name:1HsD1n12[lv1HsD1n12]; Size:5
        VSD Name:1HsD2n12[lv1HsD2n12]; Size:5
        VSD Name:vsd4n12[lvvsd4n12]; Size:4
        VSD Name:vsd5n12[lvvsd5n12]; Size:4
        VSD Name:vsd6n12[lvvsd6n12]; Size:4
    Volume group:testvg; Partition Size:4; Total:537; Free:313
        Physical Disk:hdisk1; Total:537; Free:313
        VSD Name:vsd14n12[lvvsd14n12]; Size:4
```

If the testvg volume group is not varied on, the system displays output similar to the following:

```
Node Number:12; Node Name:k21n12.ppd.pok.ibm.com
    Volume group:rootvg; Partition Size:4; Total:537; Free:47
        Physical Disk:hdisk0; Total:537; Free:47
        VSD Name:1HsD1n12[lv1HsD1n12]; Size:5
        VSD Name:1HsD2n12[lv1HsD2n12]; Size:5
        VSD Name:vsd4n12[lvvsd4n12]; Size:4
        VSD Name:vsd5n12[lvvsd5n12]; Size:4
        VSD Name:vsd6n12[lvvsd6n12]; Size:4
    Volume group:testvg is not varied on.
        Physical Disk:hdisk1;
```

Instead of issuing this command directly, you should use the appropriate SMIT panels to view it in the best format. To view information about volume groups, type:

```
smit lsvg
```

To view information about logical volumes, type:

```
smit lslv
```

To view information about physical volumes, type:

```
smit lspv
```

## vsdvg

### Purpose

**vsdvg** – Defines a virtual shared disk global volume group.

### Syntax

**vsdvg** [–**g** *global_group_name* ] *local_group_name primary_server_node* [*secondary_server_node*] [*eio_recovery*]

### Flags

| | |
|---|---|
| –**g** *global_group_name* | Specifies a unique name for the new global volume group. This name must be unique across the system partition. It should be unique across the SP, to avoid any naming conflicts during future system partitioning operations. The suggested naming convention is **vg***xx***n***yy*, where *yy* is the node number, and *xx* uniquely numbers the volume groups on that node. If this is not specified, the local group name is used for the global name. The length of the name must be less than or equal to 31 characters. |

### Operands

*local_group_name*
: Specifies the name of a volume group that you want to indicate as being used for virtual shared disks. This name is local to the host upon which it resides. The length of the name must be less than or equal to 15 characters.

*primary_server_node*
: Specifies the primary server node on which the volume group resides. The length of the name must be less than or equal to 31 characters. This can be specified in four different ways:

- frame,slot
- node number
- host name
- IP address

*secondary_server_node*
: Specifies the secondary server node on which the volume group resides. The length of the name must be less than or equal to 31 characters.

This can be specified in four different ways:

- frame,slot
- node number
- host name
- IP address

**Note:** This operand is used only by the Recoverable Virtual Shared Disk subsystem.

| | |
|---|---|
| *eio_recovery* | If a *secondary_server_node* is not specified, the default is 0 for no recovery. If the *secondary_server_node* is specified, the default is 1. |

## Description

Use this command to define volume groups for use by the IBM Virtual Shared Disk subsystem. This is done by specifying the local volume group name, the node on which it resides, and the name by which the volume group will be known throughout the cluster.

If *eio_recovery* is set (to a value of 1) due to disk error (EIO error), the IBM Recoverable Virtual Shared Disk system will perform a full recovery by flipping the current primary node and the secondary node and doing one more retry on the new primary node.

You can use the System Management Interface Tool (SMIT) to run the **vsdvg** command. To use SMIT, enter:

```
smit vsd_data
```

and select the Virtual Shared Disk Global Volume Group Information option.

## Security

You must be in the **bin** group to run this command.

## Restrictions

The *secondary_server_node* operand is used only by the Recoverable Shared Disk subsystem.

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Location

**/usr/lpp/csd/bin/vsdvg**

## Related Information

Command: **vsdelvg**

## Examples

1. The following example adds SDR information indicating that the volume group known as **vg2n17** on node **17** is available for global access and is known to the cluster as **vg2n17**. Node **17** is the primary and only server.

   ```
   vsdvg vg2n17 17
   ```

2. The following example with the Recoverable Virtual Shared Disk subsystem adds SDR information indicating that the volume group known as **vg1p3s15** on nodes **3** and **15** is available for global access and is known to the cluster as **vg1p3s15**. **3** is the primary server node and **15** is the secondary server node.

   ```
   vsdvg vg1p3s15 3 15
   ```

## vsdvgts

### Purpose

**vsdvgts** – Reads the timestamp from the volume group descriptor area (VGDA) of the physical disks and sets the value in the System Data Repository (SDR).

### Syntax

**vsdvgts** [–**a**] [*volgrp*]

### Flags

| | |
|---|---|
| –**a** | Specifies that the timestamps for this volume group for both primary and secondary nodes should be updated. If this flag is not specified, the timestamp is updated on the local node only. |

### Operands

| | |
|---|---|
| *volgrp* | Specifies a volume group. If this operand is not specified, the timestamps for all the volume groups on this node are updated. |

### Description

Use this command to update the timestamp that the Recoverable Virtual Shared Disk subsystem uses to determine if a twin-tailed volume group has changed. When the subsystem detects a change, the recovery scripts export the volume group and then import the volume group.

This command can be used to avoid exporting the volume group and then importing the volume group during recovery in situations where the export and import operations are not really necessary. This command should be used very carefully.

### Exit Values

| | |
|---|---|
| **0** | Indicates the successful completion of the command. |
| **1** | Indicates that the program was unable to read one or more timestamps. |

### Security

You must have root privilege to issue the **vsdvgts** command.

### Implementation Specifics

This command is part of the Recoverable Virtual Shared Disk optional component of PSSP.

### Prerequisite Information

See *PSSP: Managing Shared Disks*.

## Location

**/usr/lpp/csd/bin/vsdvgts**

## Examples

To update the timestamp associated with the virtual shared disk volume group vsdvg1 for just this node, enter:

```
vsdvgts vsdvg1
```

## vsdvts

### Purpose

**vsdvts** – Verifies that the IBM Virtual Shared Disk component works.

### Syntax

**vsdvts** [–**b** *block_size*] [–**n** *number_of_blocks*] *vsd_name* [*file*]

### Flags

| | |
|---|---|
| –**b** | Specifies the *block_size* used on the read and write calls to the virtual shared disk. Because the virtual shared disk raw device is used, the block size must be a multiple of 512. The default block size is 4096. |
| –**n** | Specifies the number of blocks of the file to read. The default is to read 1MB of data from the file, so 1MB divided by *block_size* is the default number of blocks. Specifying 0 means to read as many full blocks of data as there are in the file. If more blocks are specified than are in the file, only the number of full blocks that exist will be used. |

### Operands

| | |
|---|---|
| *vsd_name* | Specifies the virtual shared disk to be verified (for example, that will be written and read with the data from the file). The virtual shared disk should be in the active state. Ensure that the virtual shared disk is large enough to hold all the data you plan to write to it. A virtual shared disk on a logical volume with one physical system partition is large enough if all the **vsdvts** defaults are taken. |
| *file* | Specifies the file to be written to the virtual shared disk to verify its operation. The data is then read from the virtual shared disk and compared to this file to ensure the virtual shared disk read and write operations are successful. The default file is **/unix**. |

### Description

**Attention**

Data on *vsd_name* and its underlying logical volume is overwritten and, therefore, destroyed. Use this command after you have defined a virtual shared disk (including its underlying logical volume), but **before** placing application data on it.

Use this command to verify that the *vsd_name* is in the active state and then to write the specified part of *file* to the raw *vsd_name* device, **/dev/rvsd_name**. This command reads the data back from the virtual shared disk, then compares it to *file*. If the data is the same, the test is successful and **vsdvts** succeeds. Otherwise, **vsdvts** is unsuccessful. The **dd** command is used for all I/O operations.

Try **vsdvts** on both a server and client node (for example, on both the node with a logical volume and one without it).

## Prerequisite Information

*PSSP: Managing Shared Disks*

## Related Information

Commands: **vsdnode**, **vsdvg**, **defvsd**, **cfgvsd**, **startvsd**, **dd**

The preceding commands are listed in their order of use.

## Examples

To verfiy that the IBM Virtual Shared Disk component works, choose a newly created vsd that has no application data on it, say vsd1, and enter:

```
vsdvts vsd1
```

## wrap_test

## Purpose

**wrap_test** – Checks the functionality of a link.

**Attention**

**ATTENTION – READ THIS FIRST:** Do **not** activate the SP Switch advanced diagnostic facility until you have read this section completely, and understand this material. If you are not certain how to properly use this facility, or if you are not under the guidance of IBM Service, do **not** activate this facility.

Activating this facility may result in degraded performance of your system. Activating this facility may also result in longer response times, higher processor loads, and the consumption of system disk resources. Activating this facility may also obscure or modify the symptoms of timing-related problems.

## Syntax

**wrap_test** {[–**j** *jack*] [–**s***switch_chip_id* –**p** *switch_chip_port*]}
[–**c** *cable_length*] [–**h**]

## Flags

| | |
|---|---|
| –**j** *jack* | Specifies the Frame-Switch-BulkHead-Jack connected to the suspected link. |
| **-s** *switch_chip_id* | Specifies the ID of a switch chip connected to the suspected link. |
| **-p** *switch_chip_port* | Specifies the number of the switch chip port connected to the suspected link. |
| **-c** *cable_length* | Specifies the length of the cable in meters. The flag is applicable only to the links connecting two switches. The default value is 10 m. |
| **-h** | Displays usage information. |

## Operands

None.

## Description

The **wrap_test** command checks the functionality of a suspected link, and points to the faulty part of the link that should be replaced. You must specify either the Frame-Switch-BulkHead-Jack, or the *switch_chip_id* and *switch_chip_port* number that identify the switch chip port connected to the link. If the suspected link connects two switches, you might also specify the *cable_length* parameter. This will help the wrap test to choose the correct technique for the cable testing.

If the link under test connects a switch to a node, you are required to fence the node before running the test. If the link under test connects two switches, the link will be disabled during the test.

## Location

**/usr/lpp/ssp/bin/spd/wrap_test**

## Examples

1. To test the link connected to jack 6 of switch 17 of frame 1 with a 15 m cable (switch - switch link), enter:

   ```
   wrap_test -j E01-S17-BH-J6 -c 15
   ```

2. To test the link connected to port 3 of switch chip 23 enter:

   ```
    wrap_test -s 23 -p 3
   ```

# Part 2. Technical Reference

This part of the book contains RS/6000 SP Files and Other Technical Information and SP Subroutines.

# Chapter 2. RS/6000 SP Files and Other Technical Information

# auto.master File

## Purpose

**auto.master** – Specifies the master input file to the AIX **automount** daemon defining the file systems to be controlled and their associated map files.

## Description

The **auto.master** file is the master map file for the AIX **automount** daemon. It identifies the file systems that are to be controlled by the automounter and the directory map file associated with each file system. It may also contain default mount information for specific file systems. The **auto.master** file may reference a Network Information Service (NIS) configuration map that is to be used by the automounter. Entries in the master map file use one of the following formats:

```
+NIS_map

Directory_path Automount_map_name [default_mount_options]
```

The first form specifies an NIS configuration map that contains mount point and automounter map information. The second form directly identifies the directory path of the file system that is to be controlled by the automounter, along with the map file containing entries for the supported directories within that file system. Default options to be used when the file system is mounted can be optionally specified.

## Files

**/etc/auto.master**
Specifies map files for the AIX **automount** daemon.

## Related Information

AIX Daemons: **automount**

The “Managing the Automounter” chapter in *PSSP: Administration Guide*

The “Network File System (NFS)” and “Network Information Service (NIS)” chapters in *AIX Version 4 System Management Guide: Communications and Networks*

## Examples

The following example is a copy of the default **auto.master** file shipped with the SP modified to support an NIS configuration map:

```
# The following entry will use the NIS configuration map if NIS is

# defined for this system and the database exists

+auto_master

# The following entry provides automount support for users' home

# directories

/u /etc/auto/maps/auto.u -rw,hard,retry=3,timeo=40,rsize=4096, \
wsize=4096
```

## bootptab.info File

### Purpose

**/etc/bootptab.info** – Specifies the hardware ethernet addresses of SP nodes.

### Description

The **/etc/bootptab.info** file contains a list of nodes with their hardware ethernet addresses. It is used by the **sphrdwrad** command during node conditioning to save time in obtaining the hardware ethernet addresses. Each line of the **bootptab.info** file contains two tokens. The first token represents the node, and the second token represents the hardware ethernet address. This information may be input in one of two formats:

```
node_number   hardware_ethernet_address
```

or

```
frame,slot    hardware_ethernet_address
```

### Related Information

Commands: **sphrdwrad**

### Examples

The following is an example of a **bootptab.info** file for a single framed system that contains four high nodes:

```
1 02608CE8764D
5 02608CE87174
1,9 02608CE87180
13 02608CE8618B
```

# haemloadlist File

## Purpose

**haemloadlist** File – Event Management configuration data that is to be loaded into the System Data Repository (SDR)

## Description

RS/6000 Cluster Technology (RSCT), which is included with the IBM Parallel System Support Programs (PSSP) Licensed Program Product (LPP), contains a number of resource monitors that use the Resource Monitor Application Programming Interface (RMAPI) to supply information about system resources to the Event Management subsystem. Information about the monitors and the resources is defined in the **haemloadlist** file and includes:

- The resource variables from which events may be generated
- The resource IDs for each resource variable
- The classes in which the resource variables are grouped
- The resource monitors that supply the variables.

Before the Event Management subsystem can use this information, it must first be loaded into the System Data Repository (SDR) and then compiled into a binary Event Management Configuration Database (EMCDB) file. The **haemloadcfg** command is used to load the data from the **haemloadlist** file into the SDR. The **haemcfg** command is then used to compile the data from the SDR into the binary EMCDB file. Both of these commands are issued automatically by the **haemctrl** command when it is used to start the Event Management subsystem on the control workstation.

For resource monitors other than those supplied by RSCT, you must create one or more separate files in load list format to contain their Event Management configuration data. You can specify the name of any file in load list format on the **haemloadcfg** command.

You can use SDR commands to work with objects in the Event Management classes directly. However, IBM suggests that you use load list files and the **haemloadcfg** command to work with objects in these classes.The **haemloadcfg** command provides validation, unique to the Event Management configuration data, not available in the SDR commands.

You cannot use System Management Interface Tool (SMIT) panels to work with objects in these classes.

**The Format of an Event Management Load List File**

Source information for the EMCDB is kept in the several partitioned classes in the SDR, each of which has a set of associated attributes. The format of an Event Management load list file, which can be used as input to the **haemloadcfg** command, follows the structure of the Event Management data in the SDR.

This man page describes the format of the Event Management load list file. For detailed information about its content (the Event Management SDR classes, attributes, and their values), see the description of the EMCDB in *PSSP: Event Management Programming Guide and Reference*.

The **haemloadlist** file consists of stanzas, each of which describes an object in one of the Event Management SDR classes. The stanzas may appear in any order in the file.

Each stanza begins with the SDR class name to which the object belongs, followed by one or more lines that define the attributes of the object. The SDR class name must begin in column 1.

An attribute line consists of leading whitespace (blanks or tabs) followed by the attribute name followed by an = (equal sign) followed by a data field. The = (equal sign) may not be surrounded by blank spaces. The data field consists of a single value for the attribute. If the field contains blanks, it must be surrounded by double quotes (**"**). If the field contains a double quote character, it should be preceded by a backslash (**\"**). If the field contains a backslash character, it should be preceded by a second backslash (**\\**).

The data field must be on the same line as the attribute. The attribute lines for a stanza stop at either the start of a new stanza or at the end of the file.

Comments may be present in the file. Any line in which the first nonwhite space character is a pound sign (**#**) is a comment. Blank lines are also considered comment lines and are ignored.

**The EM_Resource_Variable Stanza**

The **EM_Resource_Variable** SDR class contains one object for each resource variable defined in the database. Accordingly, there is one **EM_Resource_Variable** stanza for each resource variable that is defined.

Here is an example of a stanza that defines a resource variable of type Quantity:

```
EM_Resource_Variable
        rvName="IBM.PSSP.aixos.PagSp.totalsize"
        rvDescription="99"
        rvValue_type="Quantity"
        rvData_type="long"
        rvInitial_value="0"
        rvClass="IBM.PSSP.aixos.PagSp"
        rvPTX_name="PagSp/totalsize"
        rvLocator="NodeNum"
        rvDynamic_instance="0"
```

Here is an example of a stanza that defines a resource variable of type State:

```
EM_Resource_Variable
        rvName="IBM.PSSP.Prog.pcount"
        rvDescription="1009"
        rvValue_type="State"
        rvData_type="SBS"
        rvClass="IBM.PSSP.Prog"
        rvLocator="NodeNum"
        rvDynamic_instance="1"
```

**The EM_Structured_Byte_String Stanza**

The **EM_Structured_Byte_String** SDR class contains one object for each structured field that is defined in a structured byte string. Accordingly, there is one **EM_Structured_Byte_String** stanza for each structured field that is defined.

The **IBM.PSSP.Prog.pcount** resource variable is an SBS that has three structured fields. Here is an example of the stanzas that define the fields:

```
EM_Structured_Byte_String
        sbsVariable_name="IBM.PSSP.Prog.pcount"
        sbsField_name="CurPIDCount"
        sbsField_type="long"
        sbsField_SN="0"

EM_Structured_Byte_String
        sbsVariable_name="IBM.PSSP.Prog.pcount"
        sbsField_name="PrevPIDCount"
        sbsField_type="long"
        sbsField_SN="1"

EM_Structured_Byte_String
        sbsVariable_name="IBM.PSSP.Prog.pcount"
        sbsField_name="CurPIDList"
        sbsField_type="cstring"
        sbsField_SN="2"
```

### The EM_Resource_ID Stanza

The **EM_Resource_ID** SDR class contains one object for each resource ID element that is defined for a resource and all of its resource variables. Accordingly, there is one **EM_Resource_ID** stanza for each resource ID element that is defined.

Here are some examples of stanzas that define resource IDs. The resource ID for the resource named **IBM.PSSP.aixos.PagSp** contains only one resource ID element. The resource ID for the resource named **IBM.PSSP.Prog** contains three elements.

```
EM_Resource_ID
        riResource_name="IBM.PSSP.aixos.PagSp"
        riElement_name="NodeNum"
        riElement_description="701"

EM_Resource_ID
        riResource_name="IBM.PSSP.Prog"
        riElement_name="UserName"
        riElement_description="701"

EM_Resource_ID
        riResource_name="IBM.PSSP.Prog"
        riElement_name="ProgName"
        riElement_description="701"

EM_Resource_ID
        riResource_name="IBM.PSSP.Prog"
        riElement_name="NodeNum"
        riElement_description="701"
```

### The EM_Resource_Class Stanza

The **EM_Resource_Class** SDR class contains one object for each resource variable class that is defined in the database. Accordingly, there is one **EM_Resource_Class** stanza for each resource variable class that is defined.

Here are two examples of stanzas that define resource classes:

```
EM_Resource_Class
        rcClass="IBM.PSSP.aixos.PagSp"
        rcResource_monitor="aixos"
        rcObservation_interval="30"
        rcReporting_interval="0"

EM_Resource_Class
        rcClass="IBM.PSSP.Prog"
        rcResource_monitor="IBM.PSSP.harmpd"
        rcObservation_interval="0"
        rcReporting_interval="0"
```

### The EM_Resource_Monitor Stanza

The **EM_Resource_Monitor** SDR class contains one object for each resource monitor that is defined in the database. Accordingly, there is one **EM_Resource_Monitor** stanza for each resource monitor that is defined.

Here are some examples of stanzas that define resource monitors:

```
EM_Resource_Monitor
        rmName="IBM.PSSP.harmpd"
        rmPath="/usr/lpp/ssp/bin/haemRM/harmpd"
        rmMessage_file="PEM.cat"
        rmMessage_set="2"
        rmConnect_type="server"
        rmPTX_prefix="0"
        rmPTX_description="0"
        rmPTX_asnno="0"

EM_Resource_Monitor
        rmName="IBM.PSSP.harmld"
        rmPath="/usr/lpp/ssp/bin/haemRM/harmld"
        rmMessage_file="harm_des.cat"
        rmMessage_set="1"
        rmConnect_type="server"
        rmPTX_prefix="IBM/PSSP.harmld"
        rmPTX_description="1,2"
        rmPTX_asnno="2"
```

## Related Information

Commands: **haemcfg**, **haemloadcfg**

For a general overview of configuring Event Management, see “The Event Management Subsystem” chapter of *PSSP: Administration Guide*.

For a description of the SDR classes and attributes that are related to the EMCDB, see *PSSP: Event Management Programming Guide and Reference*.

## hmacls File

### Purpose

**/spdata/sys1/spmon/hmacls** – Defines the Access Control Lists (ACLs) used by the Hardware Monitor daemon.

### Description

The **/spdata/sys1/spmon/hmacls** file contains permission specifications for users to execute the various Hardware Monitor operations. Each line in the file consists of three white-space separated tokens in the following format:

```
obj    user_name     permissions
```

The *obj* token is either a frame ID or the host name of the control workstation (known as the Monitor and Control Node (MACN) by the Hardware Monitor). The *user_name* token is the user's principal name or principal name and instance.

The *permissions* token specifies which operations the user specified by the *user_name* token can execute against the object specified by the *obj* token.

If the *obj* token is a frame ID, the *permissions* token is one or more characters taken from the set **v**, **s**, and **m**. A definition of each follows:

| | |
|---|---|
| **v** | Specifies Virtual Front Operator Panel (VFOP) permission |
| **s** | Specifies S1 permission |
| **m** | Specifies Monitor permission |

The VFOP permission implies the monitor permission. These permissions are described in the various commands. If the *obj* token is the MACN host name, the *permissions* token must be the character **a**. The character **a** is the administrative permission required to use the **hmadm** command.

Users are authorized to use the Hardware Monitor by virtue of having their names in the **/spdata/sys1/spmon/hmacls** file and have issued the **k4init** command with that name.

### Files

**/spdata/sys1/spmon/hmacls**
Specifies ACLs for the Hardware Monitor daemon.

### Related Information

Commands: **hmadm**, **hmcmds**, **hmmon**, **nodecond**, **s1term**

### Examples

The following is an example of a **hmacls** file:

```
workstn3.kgn.ibm.com  john.admin  a
1  john.admin  vsm
2  john.admin  vsm
3  john.admin  vsm
1  mary        m
2  mary        m
3  mary        m
```

## hmthresholds File

### Purpose

**hmthresholds** – provides a software mechanism for applying threshold values to SP Frame, Node, and Switch environmental conditions such as voltage, amperage, and temperature. The Hardware Monitor (**hardmon**), with the aid of the Logging Daemon (**splogd**), provides notification to the ERRPT when an applied threshold value is crossed. This software checking, if enabled, is done in parallel with the internal self-checking done by the hardware itself.

### Description

The **/spdata/sys1/spmon/hmthresholds** file contains threshold values in the form of low/high warning conditions, and low/high shutdown conditions. Each line in the file, excluding comment lines, which begin with a # (pound sign) in column one, consists of at least three white-space separated tokens in the following format:

**node_type low_value high_value [ [low_value high_value] [... ...] ]**

The **node_type** token is the value of the Hardware Monitor variable "type" which is the definitive identifier for the SP Supervisor card that is installed in the hardware for any given slot. Use the **hmmon** command to obtain the value of the variable "type" for a given node.

For example, using an SP2 wide node with 4.0 volt power:

```
$ hmmon -G -Q -v type 1:1
frame 001, slot 01:
supervisor type 0x0051
```

**low_value** and **high_value** are token pairs that are the low/high values for the environmental condition you want to threshold.

For example, examine the following excerpt from the **/spdata/sys1/spmon/hmthresholds** file.

```
#
# #### SP2 wide node w/4.0 volt power:
#       VOLTP5M    VOLTP12    VOLTN12    VOLTP5I     TEMP      VOLTP4
#      low  high  low  high  low  high  low  high  low  high  low  high
# Warning thresholds
# 0x51 4.75 5.4   11.0 13.0  11.0 13.0  4.75 5.4   0x00 45.0  3.46 4.2
# Shutdown thresholds
# 0x51 4.50 5.75  10.2 13.8  10.2 13.8  4.50 5.75  0x00 60.0  3.28 4.4
# Software thresholding disabled
  0x51 0x00 0xff  0x00 0xff  0x00 0xff  0x00 0xff  0x00 0xff  0x00 0xff
```

The node_type is 0x51. It is an SP2 wide node w/4.0 volt power. There are three sets of token pairs defined (only one is uncommented). Each of the three sets contain six token pairs. The six token pairs are the low/high threshold values for, in order from left to right, +5 volts memory, +12 volts, -12 volts, +5 volts I/O, temperature, and +4 volts. The first set establishes warning thresholds, the second establishes shutdown thresholds. The last set, which is the one used by the Hardware Monitor since its the one that's uncommented, effectively disables environmental checking since the low/high value pairs are set to maximum low and maximum high.

For hardware that does not report environmental conditions there is a dummy entry in the **/spdata/sys1/spmon/hmthresholds** file. Recall that at least three white-space separated tokens are required for each SP hardware type supported by the Hardware Monitor.

For example, examine the following excerpt from the **/spdata/sys1/spmon/hmthresholds** file for an SP Switch and Twin-Tail Frame hardware in basecode, or non-active, mode:

```
#
# #### SP Switch - base code only; no supervisor:
#     DUMMY (cardtype has no A/D registers)
#     low  high
# Software thresholding disabled
0x80 0x00 0xff
#
# #### twin-tail frame - base code only; no supervisor:
#      DUMMY (cardtype has no A/D registers)
#      low  high
# Software thresholding disabled
0x10 0x00 0xff
```

**Enabling Software Thresholding**

Software thresholding can be enabled in one of following ways:

- **Use the Released Values:** Since the **/spdata/sys1/spmon/hmthresholds** file is released with nominal warning and shutdown values you would need to simply decide which environmental condition you wish to monitor (warning or shutdown) and uncomment the appropriate line for that hardware type. The released values coincide with the values that the hardware itself uses to perform hardware out-of-spec thresholding, or said another way, internal self-checking.
- **Make Up Your Own:** You could code your own low/high values. If you code your own values, it's recommended that, instead of modifying the warning or shutdown line itself, you add a new line thereby preserving the released values.

  Low/high threshold values (token pairs) are coded as either decimal (base 10), or hex. Temperatures are in Celsius. Volts are volts. Amps are amps. You can enable any number of threshold low/high pairs. In other words, you need not enable all of the value pairs if it is only a subset of them that you are interested in.

  For example, assume that you suspect the room temperature is causing SP2 wide nodes w/4.0 volt power to overheat and you want to be notified in advance of the released warning threshold.

```
#
# #### SP2 wide node w/4.0 volt power:
#       VOLTP5M    VOLTP12    VOLTN12    VOLTP5I     TEMP      VOLTP4
#      low  high  low  high  low  high  low  high  low  high  low  high
# Warning thresholds
# 0x51 4.75 5.4   11.0 13.0  11.0 13.0  4.75 5.4   0x00 45.0  3.46 4.2
# Shutdown thresholds
# 0x51 4.50 5.75  10.2 13.8  10.2 13.8  4.50 5.75  0x00 60.0  3.28 4.4
# Software thresholding disabled
# 0x51 0x00 0xff  0x00 0xff  0x00 0xff  0x00 0xff  0x00 0xff  0x00 0xff
# Software thresholding enabled
  0x51 0x00 0xff  0x00 0xff  0x00 0xff  0x00 0xff  0x00 35.0  0x00 0xff
```

In the example, note that a temperature limit of 35 degrees Celsius has been coded (it's the uncommented line and again the only one that the Hardware Monitor will read). All other threshold values on that same line remain disabled. This does not mean that the node will never get hotter than 35 degrees. It means that you will be notified when this temperature is reached.

**Deciding What Threshold Value To Use**

Using the **hmmon** command you can query the Hardware Monitor for the nodes' environmental values. If you query more than once, with a 5 second lapse between queries (the Hardware Monitor refreshes itself with hardware state data every 5 seconds), you should begin to see a pattern develop. The information returned will provide insight into what values to use when coding your own low/high threshold pairs.

For example, using an SP2 wide node with 4.0 volt power:

```
$ hmmon -Q -s -v temp 1:1
1   1  temp                 36.480  0x9034  temperature
$ hmmon -Q -s -v temp 1:1
1   1  temp                 38.916  0x9034  temperature
$ hmmon -Q -s -v temp 1:1
1   1  temp                 39.212  0x9034  temperature
$ hmmon -Q -s -v temp 1:1
1   1  temp                 38.882  0x9034  temperature
```

In the example, you can see that the nodes' temperature seems to be hovering around 38-39 degrees. This is only 6 or 7 degrees below the established nominal warning condition. You may choose to establish a high warning threshold value of approximately 40 degrees.

**Note:** When using the **hmmon** command to obtain environmental values be sure to use the "**-s** (symbolic)" flag. This flag causes **hmmon** to apply a scalar to the environmental value before displaying it. With the scalar applied, the values can be interpreted as "real-life" values. Again, with temperatures in Celsius, volts as volts, and amps as amps.

**hmmon** "**-r** (raw)" and "default" formats do not cause **hmmon** to apply the scalar value. These two formats display information exactly as it's obtained from the Hardware Monitor - to be more specific, from the hardware itself which holds the values in a "normalized" format.

## Environmental Condition Notification

Environmental condition checking and notification comes in two flavors, **Hardware Out-Of-Spec Notification** and **Software Out-Of-Range Notification**.

- **Hardware Out-Of-Spec Notification:** A hardware out-of-spec condition occurs when the Frame, Node, or Switch Supervisor determines that the hardware is operating outside the scope of its established warning or shutdown limits. These established limits are kept internal to the hardware and cannot be modified through **/spdata/sys1/spmon/hmthresholds**.

  If a hardware out-of-spec environmental condition occurs the Frame, Node, or Switch Supervisor will alert the Hardware Monitor. **hardmon** will in turn report the out-of-spec state to interested parties, such as the Logging Daemon. **splogd** will cut an ERRPT entry. This all happens regardless of whether software thresholding is enabled or disabled. That is, it is a function of the hardware itself.

  The following is an example of the hardware out-of-spec environmental states for an SP2 wide node w/4.0 volt power:

```
$ hmmon -G -Q -s 1:1
  1   1  warningN12Low        FALSE    0x9073  -12 volt low warning
  1   1  shutdownN12Low       FALSE    0x9074  -12 volt low shutdown
  1   1  warningN12High       FALSE    0x9075  -12 volt high warning
  1   1  shutdownN12High      FALSE    0x9076  -12 volt high shutdown
  1   1  warningP12Low        FALSE    0x9064  +12 volt low warning
  1   1  shutdownP12Low       FALSE    0x9065  +12 volt low shutdown
  1   1  warningP12High       FALSE    0x9062  +12 volt high warning
  1   1  shutdownP12High      FALSE    0x9063  +12 volt high shutdown
  1   1  warningP5mLow        FALSE    0x907b  +5m volt low warning
  1   1  shutdownP5mLow       FALSE    0x907c  +5m volt low shutdown
  1   1  warningP5mHigh       FALSE    0x907d  +5m volt high warning
  1   1  shutdownP5mHigh      FALSE    0x907e  +5m volt high shutdown
  1   1  warningP5iLow        FALSE    0x907f  +5i volt low warning
  1   1  shutdownP5iLow       FALSE    0x9080  +5i volt low shutdown
  1   1  warningP5iHigh       FALSE    0x9081  +5i volt high warning
  1   1  shutdownP5iHigh      FALSE    0x9082  +5i volt high shutdown
  1   1  fanwarning5          FALSE    0x9055  fan 5 warning
  1   1  fanfail5d            FALSE    0x9070  fan 5 delayed shutdown
  1   1  warningTemp          FALSE    0x9058  temperature warning
  1   1  shutdownTemp         FALSE    0x9059  temperature shutdown
  1   1  warningP4Low         FALSE    0x9087  +4 volt low warning
  1   1  shutdownP4Low        FALSE    0x9088  +4 volt low shutdown
  1   1  warningP4High        FALSE    0x9089  +4 volt high warning
  1   1  shutdownP4High       FALSE    0x908a  +4 volt high shutdown
  1   1  fanwarning1          FALSE    0x904d  fan 1 warning
  1   1  fanfail1d            FALSE    0x906b  fan 1 delayed shutdown
  1   1  fanwarning2          FALSE    0x904f  fan 2 warning
  1   1  fanfail2d            FALSE    0x906c  fan 2 delayed shutdown
  1   1  fanwarning3          FALSE    0x9051  fan 3 warning
  1   1  fanfail3d            FALSE    0x906e  fan 3 delayed shutdown
  1   1  fanwarning4          FALSE    0x9053  fan 4 warning
  1   1  fanfail4d            FALSE    0x906f  fan 4 delayed shutdown
```

  In the example, there are no hardware out-of-spec conditions since the environmental states are not asserted. That is, they are all FALSE.

- **Software Out-Of-Range Notification:** A software out-of-range assertion occurs when the Hardware Monitor detects that the nodes' environmental condition has either fallen below the low threshold value coded in **/spdata/sys1/spmon/hmthresholds**, or has risen higher than the high threshold value coded in **/spdata/sys1/spmon/hmthresholds**. For both of these cases the Hardware Monitor will transition the software out-of-range environmental state to TRUE.

The software out-of-range environmental state will transition to FALSE when the Hardware Monitor detects that the nodes' environmental condition has settled between the low/high threshold value coded in **/spdata/sys1/spmon/hmthresholds**. That is, has either crossed the low or high value toward the norm, or average value.

The following is an example of the software out-of-range environmental states for an SP2 wide node w/4.0 volt power.

```
$ hmmon -G -Q -s 1:1
  1  1   voltP5mRange         FALSE    0x9092  +5 mem volts out of range
  1  1   voltP12Range         FALSE    0x908e  +12 voltage out of range
  1  1   voltN12Range         FALSE    0x9091  -12 voltage out of range
  1  1   voltP5iRange         FALSE    0x9090  +5 I/O volts out of range
  1  1   tempRange            FALSE    0x9084  temperature out of range
  1  1   voltP4Range          FALSE    0x9093  +4 voltage out of range
```

In the example, there are no software out-of-range conditions since the environmental states are not asserted. That is, they are all FALSE. Note that the environmental states end with the suffix "Range". This is true of all software out-of-range environmental state variables.

If a software out-of-range environmental condition were to occur the Hardware Monitor would report the out-of-range state to interested parties, such as the Logging Daemon, by transitioning the state to TRUE. **splogd** will cut an ERRPT entry.

The following is an example of an ERRPT entry for a temperature out-of-range condition:

```
LABEL:          SPMON_INFO103_TR
IDENTIFIER:     0D1620A8

Date/Time:       Thu Jul  9 12:15:04
Sequence Number: 47977
Machine Id:      000044587000
Node Id:         k4s
Class:           H
Type:            UNKN
Resource Name:   sphwlog
Resource Class:  NONE
Resource Type:   NONE
Location:        NONE

Description
THRESHOLD HAS BEEN EXCEEDED

Probable Causes
POWER SUBSYSTEM
COOLING FAN
THERMAL DETECTOR

Failure Causes
POWER SUBSYSTEM
COOLING FAN
THERMAL DETECTOR

        Recommended Actions
        NONE

Detail Data
DETECTING MODULE
LPP=PSSP,Fn=splogd.c,SID=1.16.1.16,L#=925,
DIAGNOSTIC EXPLANATION
Information; Node 1:1; tempRange; Value out of range.
```

If the software out-of-range environmental condition was to transition toward the norm, or average value, the Hardware Monitor would transition the state to FALSE and notify interested parties such as the Logging Daemon. **splogd** will cut an ERRPT entry.

The following is an example of an ERRPT entry for a temperature out-of-range condition that transitioned to FALSE:

```
LABEL:          SPMON_INFO104_TR
IDENTIFIER:     E91A5929

Date/Time:       Thu Jul  9 12:15:09
Sequence Number: 47978
Machine Id:      000044587000
Node Id:         k4s
Class:           H
Type:            TEMP
Resource Name:   sphwlog
Resource Class:  NONE
Resource Type:   NONE
Location:        NONE

Description
PROBLEM RESOLVED

Probable Causes
POWER SUBSYSTEM
COOLING FAN
THERMAL DETECTOR
UNDETERMINED

Failure Causes
POWER SUBSYSTEM
COOLING FAN
THERMAL DETECTOR

        Recommended Actions
        NONE

Detail Data
DETECTING MODULE
LPP=PSSP,Fn=splogd.c,SID=1.16.1.16,L#=925,
DIAGNOSTIC EXPLANATION
Information; Node 1:1; tempRange; Condition cleared.
```

## Implementation Specifics

This file is part of the PSSP **ssp.basic** fileset.

## Files

**/spdata/sys1/spmon/hmthresholds**

## Related Information

Commands: **hmmon**

Subsystems: **Hardware Monitor (hardmon), Logging Daemon (splogd)**

## Examples

The following is an example of the software out-of-range environmental states for an SP2 wide node w/4.0 volt power.

```
$ hmmon -G -Q -s 1:1
  1  1   voltP5mRange         FALSE    0x9092  +5 mem volts out of range
  1  1   voltP12Range         FALSE    0x908e  +12 voltage out of range
  1  1   voltN12Range         FALSE    0x9091  -12 voltage out of range
  1  1   voltP5iRange         FALSE    0x9090  +5 I/O volts out of range
  1  1   tempRange            FALSE    0x9084  temperature out of range
  1  1   voltP4Range          FALSE    0x9093  +4 voltage out of range
```

## hwevents File

### Purpose

**hwevents** – Contains state change actions that are to be performed by the logging daemon (**splogd**). The Hardware Monitor (**hardmon**) detects these state changes, and notifies the logging daemon. The Logging Daemon (**splogd**) can perform three different operations, based on information in **hwevents**. It can 1) write state changes to the ERRPT, 2) write state changes to a log file, 3) run user exits when specified state changes occur.

### Description

The **/spdata/sys1/spmon/hwevents** file contains lines that specify one of the three operations described in the Purpose section. Each line in the file, excluding comment lines, which begin with a # (pound sign) in column one, consists of at least seven white-space separated tokens in the following format:

```
frame slot variable operator value time function  [argument [...  ...]]
```

Following is a description of these tokens:

**frame** Specifies a frame number (1–*n*) or * for all frames.

**slot** Specifies the following:

- A number from 0–17
- One of:
  - NODES_ONLY (addresses 1 through 16)
  - SWITCH (address 17)
  - FRAME (address 0)
  - * (all addresses)
  - NODES_AND_SWITCH (addresses 1–17)
  - FRAME_AND_NODES (addresses 0–16)
  - FRAME_AND_SWITCH (addresses 0 and 17)

**variable** Specifies a hardware variable, as returned from the Hardware Monitor (for example, nodePower, temp, LED7SegA). For a list of all variables, issue the command **hmmon –V**.

**operator** Specifies how to compare the value. Acceptable values are: **=**, <, >, and **!=**.

**value** Specifies the value of the variable to match with the operator wildcard (*), or a partial match with the wildcard at the end (23*).

**time** Specifies if the function should be called at startup, when the state changes, or at both times. Valid options are **startup**, **change**, or **both**.

**function** Specifies the program to call when an event occurs. Any 'argument' tokens will be passed as arguments to the program.

There are two special keywords for function. If function is SP_ERROR_LOG, error logging is performed provided that syslog is set up and AIX error logging is set up to perform SP logging. Refer to the **setup_logd** command for details.

If function is SP_STATE_LOG, these state changes that meet the statement's criteria are logged to **/var/adm/SPlogs/spmon/splogd.state_changes.timestamp**.

## Files

**/spdata/sys1/spmon/hwevents**

**/var/adm/SPlogs/spmon/splogd.state_changes.timestamp**

## Related Information

SP Commands: **hmmon**

Subsystems: Logging Daemon (**splogd**), Hardware Monitor (**hardmon**)

## Examples

The following is an example of an hwevents file:

```
# frame slot   variable        operator value time  function
   *     *      *                 =      *     b   SP_ERROR_LOG
   *     *      type              =      *     b   /usr/lpp/ssp/install/bin/SDR_config -l
   *     *      hostHWFrameID    !=      0     c   /usr/lpp/ssp/install/bin/SDR_config -l
   *     *      hostHWSlotID     !=      0     c   /usr/lpp/ssp/install/bin/SDR_config -l
   *     *      mux               =      *     c   /usr/lpp/ssp/bin/Eclock.state_change -s
   *     *      portClkMissing    =      1     c   /usr/lpp/ssp/bin/Eclock.state_change -s
   *     *      chipClkMissing    =      1     c   /usr/lpp/ssp/bin/Eclock.state_change -s
   *     17     nodePower         =      *     c   /usr/lpp/ssp/bin/Eclock.state_change -p
   * NODES_ONLY PowerOnRequest    =      1     b   /usr/lpp/ssp/install/bin/spexpon
#
#
# Uncomment the SP_STATE_LOG line to turn state change logging on.
# If you run with state change logging on, be sure you have a large enough /var.
# The state change logs will be closed each night on the control workstation
# if they exist and any that are older than a week will be removed by
# cleanup.logs.ws.
#
#  *      *     *            =       *     c    SP_STATE_LOG
```

## .klogin File

### Purpose

**.klogin** – Specifies remote principals that can use a local user account.

### Description

The **$HOME/.klogin** file defines which users and services on any remote hosts (computers in a network) within an authentication realm are allowed to invoke commands on the local user account.

The format of the **$HOME/.klogin** file is:

```
principal.instance@realm
```

A typical **.klogin** file, if present, looks similar to the following:

```
harry@KGN.IBM.COM
beverly.root@KGN.IBM.COM
root.admin@KGN.IBM.COM
user.wkst3@KGN.IBM.COM
user1@KGN.IBM.COM
rcmd.wkst3@KGN.IBM.COM
```

The **principal** name is the name of the remote user using SP authentication to execute remote commands on a local user account.

The **instance** is used to distinguish among variations on the principal name and may be used to indicate special privileges such as root authorization. In many environments, the **instance** is used with a service to indicate the workstation the service is running on. The service entries are usually found only in the root directory **.klogin** file.

The realm is the authentication realm. This may be different if there are several authentication realms, each with a different realm name, in your environment.

If the originating remote user is authenticated to one of the principals named in the **.klogin** file, access is granted to the account. The owner of the account is granted access if there is no **.klogin** file. If a **.klogin** file is present, the owner must also be listed to gain access to his or her account from a remote host.

For security reasons, any **$HOME/.klogin** file must be owned by either the local user or root, and only the owner should have read and write access.

### Files

**$HOME/.klogin**
Specifies remote users that can use a local user account.

## Prerequisite Information

- Refer to *AIX Version 4 System Management Guide: Communications and Networks* for a network overview.
- Refer to the chapter on security in *PSSP: Administration Guide* for an overview and for additional **Kerberos** information.

## Related Information

SP Commands: **k4init**, **rcp**, **rsh**

## Examples

The following examples assume both Jeff and Anna have principal names in the authentication database (anna@KGN.IBM.COM, jeff@KGN.IBM.COM) and have issued a **k4init** to be authenticated on their local host. In addition, there is one authentication realm signified by KGN.IBM.COM.

1. To allow only user Anna on host wkst3 to rsh into her own account on host wkst7, a **.klogin** file is not required.
2. To allow user Jeff on host wkst3 to rsh into Anna's account on host wkst7, the **.klogin** file in Anna's account on wkst7 must have the following entries:

   ```
   anna@KGN.IBM.COM
   jeff@KGN.IBM.COM
   ```

   Anna's entry must be present in the **.klogin** file since the **.klogin** file exists. Jeff can now use the **–l** flag on the rsh or rcp to access Anna's account.

## Kerberos

## Purpose

**Kerberos** – Contains an introduction to SP authentication services.

## Description

**Kerberos** authenticates individual users in a network environment. After authenticating your identity, you can use facilities such as Sysctl, the SP System Monitor, and the authenticated versions of network utilities **rsh** and **rcp**, without having to present passwords to remote hosts and without having to use **.rhosts** files.

Before you can use the SP authenticated commands, you must make sure that you are added to the authentication database. You can use the **k4init** command to find out. This command tries to authenticate your identity in the system. The **k4init** command prompts you for a principal name and password. Enter your principal name and password. If the command accepts your authentication information without sending you a message, you are already registered. If you enter your user name and **k4init** responds with the following message:

```
Kerberos principal unknown
```

contact your system administrator.

A principal identifier contains three parts. The first is the user or service name. The second is the instance, which in the case of a user is usually null. Some users can have privileged instances, however, such as root or admin. In the case of a service, the instance is the name of the machine on which it runs (for example, there can be a **rcmd** (**sysctld** daemon) service running on the machine abc, which is different from the kshd service running on the machine xyz). The third part of the name is the realm. The realm corresponds to the service providing authentication for the principal. For example, computing resources within an enterprise can be partitioned into multiple administrative units for convenience or other business reasons.

When writing a principal identifier, the user or service name is separated from the instance (if not null) by a period, and the realm (if not the local realm) follows, preceded by an at sign (@).

When you authenticate your identity using the **k4init** command, you are given an initial ticket (which is an encrypted protocol message that provides authentication). This ticket is used to request other tickets from the authentication service for SP authenticated services such as SP **sysctl**. The ticket transactions are done transparently, so you do not have to worry about their management.

Be aware that tickets expire. Some tickets, such as admin instance tickets, may expire in a few minutes, while other tickets may be good for hours, days, or weeks depending on the installation's policy. If your login session extends beyond the time limit, you have to reauthenticate your identity using the **k4init** command to get new tickets.

For more information about the **k4init** and **k4destroy** commands, see the **k4init** and **k4destroy** command.

## Related Information

Commands: **kadmin**, **k4destroy**, **k4init**, **k4list**, **kpasswd**

# krb.conf File

## Purpose

**/etc/krb.conf** – Contains the SP authentication configuration file.

## Description

The **krb.conf** file contains configuration information describing the local authentication realm and the location of authentication servers for known realms.

The first line of the **krb.conf** file contains the name of the local authentication realm. Additional lines specify the location of an authentication server for a realm. The **krb.conf** file must contain at least one entry for each realm used by the local system. Each line specifying the location of an authentication server must be of the form:

```
REALM_NAME host_name
```

or

```
REALM_NAME host_name admin server
```

When “admin server” follows the host name, it indicates that the hosts also provides an administrative database server.

## Related Information

SP File: **krb.realms**

Refer to the chapter on security in *PSSP: Administration Guide* for additional **Kerberos** information.

## Examples

The following example of an **/etc/krb.conf** shows a simple configuration consisting a single realm with two servers, the primary and one secondary:

```
EAST.COAST
EAST.COAST master.authent.abc.com admin server
EAST.COAST backup.authent.abc.com
```

Here, “admin server” identifies the system whose full host name is “master.authent.abc.com” as the primary server, responsible for administration of the master database. Note that, in this case, there would have to be information in the **/etc/krb.realms** file to map the two host names or the domain name **authent.abc.com** to the local realm name, “EAST.COAST”. See the Example section of the **krb.realms** file.

## krb.realms File

### Purpose

**/etc/krb.realms** – Specifies the translations from host names to authentication realms.

### Description

The **krb.realms** file provides a translation from a host name or a network domain name to the authentication realm name for the services provided by that host. Each line of the translation file is in one of the following forms (domain names should begin with a period (.)):

```
host_name    realm_name
domain_name  realm_name
```

If a host name exactly matches the *host_name* field in a line of the first form, the corresponding realm is the realm of the host. If a host name does not match any *host_name* in the file but its domain exactly matches a domain name, the corresponding realm is the realm of the host.

If no translation entry applies, the host's realm is considered to be the host name's domain portion converted to uppercase. If the host name does not contain a domain name, the host's realm is considered to be the host name converted to uppercase.

### Related Information

SP File: **krb.conf**

Refer to the chapter on security in *PSSP: Administration Guide* for additional **Kerberos** information.

### Examples

The following example of an **/etc/krb.realms** shows the entries that could be used to map a host name or a domain name to a realm. These names correspond to those used in the **krb.conf** file example.

```
master.authent.abc.com EAST.COAST
.authent.abc.com EAST.COAST
```

The first line maps a specific host name to the realm “EAST.COAST”.  If the host name were “master.east.coast”, no entry would have been required. The second entry maps all host names whose domain portion is “authent.abc.com” to the same domain. The default mapping for this realm is:

```
.east.coast EAST.COAST
```

This type of mapping is always assumed, even if the **/etc/krb.realms** file is empty.

## SDR_dest_info File

### Purpose

**SDR_dest_info** – Provides connection addresses for System Data Repository (SDR) clients.

### Description

The **SDR_dest_info** file provides connection addresses for System Data Repository (SDR) clients. It contains the following fields:

| | |
|---|---|
| **primary** | Specifies the IP address of the system partition to which the node belongs. On the control workstation, this is the IP address of the default system partition. |
| **default** | Specifies the IP address of the default system partition. |
| **nameofprimary** | Specifies the host name of the system partition to which the node belongs. |
| **nameofdefault** | Specifies the host name of the default system partition. |

### Related Information

Refer to the *PSSP: Administration Guide* for additional information on the **SDR_dest_info** file.

### Examples

This example shows the contents of an **SDR_dest_info** file for a node. The node is a member of system partition 129.40.127.46. The default system partition on this system is 129.40.127.47. The host name of the node's system partition is k99sp1. The host name of the default system partition is k99s.

```
default:129.40.127.47
primary:129.40.127.46
nameofprimary:k99sp1
nameofdefault:k99s
```

## sysctl.acl File

### Purpose

**sysctl.acl** – Contains the Access Control List (ACL) for the Sysctl ACL authorization callback.

### Description

The **sysctl.acl** file contains the list of users authorized to access objects which are assigned the ACL authorization callback. Each line of the file must conform to one of the following formats:

**Comment** A line which begins with a number sign (#) is considered a comment.

**Principal name** A line of the form of "_PRINCIPAL *principal_name*" defines a principal on the ACL. If the *principal_name* is not fully qualified (for example, it does not contain an instance or realm separator), the realm defaults to the local realm and the instance defaults to NULL.

**ACL file** A line in the form "_ACL_FILE *file_name*" references another ACL file the contents of which is included in this file.

The first line of this (or any Sysctl ACL file) must begin with the line:

```
#acl#
```

**Note:** Including a principal in the **sysctl.acl** file gives that principal the same level of authority as having the root password.

### Prerequisite Information

See the descriptions of SP authentication and Sysctl in *PSSP: Administration Guide*.

### Related Information

SP Command: **sysctl**

SP Daemon: **sysctld**

### Examples

The following **sysctl.acl** file defines two principals and includes the contents of another ACL file:

```
#acl#
#
# This is the Sysctl ACL file for the system.  It defines two
# principals: "root.@HPSSL.KGN.IBM.COM" and "rcr.@HPSSL.KGN.IBM.COM"
# It also includes the contents of file /etc/my.acl.file
#
_PRINCIPAL root.@HPSSL.KGN.IBM.COM
_PRINCIPAL rcr.@HPSSL.KGN.IBM.COM
_ACL_FILE /etc/my.acl.file
```

# sysctl.conf File

## Purpose

**sysctl.conf** – Configures the Sysctl server (**sysctld**) running on the local SP node.

## Description

The **sysctl.conf** file configures the local Sysctl server daemon by optionally creating variables, procedures and classes, setting variables, loading shared libraries, and executing Sysctl commands. These items are used by the server in the processing of requests from local and remote Sysctl clients.

The default location of this file is **/etc/sysctl.conf**. An alternate configuration file location can be specified by using the **–f** flag when starting the server (see the **sysctld** command).

The **/etc/sysctl.conf** file contains Sysctl commands which are read and executed by the Sysctl server during its initialization. The following commands are supported:

**create var** *var_name var_value* **[***auth_callback***]**
The **create var** statement creates a variable, assigns it a value, and assigns it an authorization callback. This variable can then be referenced from within other Sysctl procedures and commands. If the *auth_callback* parameter is not supplied, a value of NONE is assumed. For example:

```
create var buildTop /usr/lpp/ssp AUTH
```

creates the variable **buildTop**, assigns it a value of **/usr/lpp/ssp**, and assigns it an authorization callback of AUTH. The variable **buildTop** can be referenced within Sysctl commands and procedures.

Another example:

```
create var STARTTIME [exec /bin/date] NONE
```

creates the variable **STARTTIME**, assigns it the value returned from the execution of the **/bin/date** command at server initialization, and assigns it an authorization callback of NONE. This variable contains the date and time at which the server was started on the node.

**create proc** *proc_name* **{***parameters***}** *auth_callback* **{***procedure***}**
The **create proc** statement creates a new procedure in the Sysctl server. This new procedure can be invoked from a client by supplying its name along with any defined parameters. For example:

```
create proc mydate {} AUTH {exec /bin/date}
```

creates the procedure **mydate** which has no parameters. This procedure has an authorization callback of AUTH. The procedure is comprised of a single statement (**exec /bin/date**).

**create class** *class_name class_file_name* **[***auth_callback***]**
The **create class** statement creates a class of commands in the Sysctl server. An optional authorization callback can be supplied. The authorization callback assigned to each object in the class is the logical OR of the class' callback (if supplied) and the object's callback. Thus, access to a class' object is granted if either the object's or the class' authorization callback allows access. For example:

```
create class sys $buildTop/samples/sysctl/sys.cmds
```

creates the class **sys**. If **$buildTop** is defined as **/usr/lpp/ssp**, the file **/usr/lpp/ssp/samples/sysctl/sys.cmds** contains the definition of the **sys** class.

**include**
The **include** statement includes the contents of another file in the configuration file. This provides a way of organizing the Server configuration statements into manageable groupings. For example:

```
include $buildTop/samples/sysctl/pdfpfck.cmds
```

causes the Sysctl server to read the contents of the specified file at initialization time.

**set** The **set** statement sets the value of the server variables ACL, LOG, or KEY, or sets values in the **env**() array. The default values for the Sysctl server's ACL file, log file, and service key file can be overridden by assigning values to the ACL, LOG, and KEY variables in the configuration file. For example, the following line overrides the default value for the log file name:

```
set LOG /var/sysctld.log_file
```

The values assigned to the ACL, LOG, and KEY variables are overridden by the optional command line arguments **–a**, **–l**, and **–k**.

Environment variables (such as the default PATH) can also be set within the configuration file by assigning values to the **env**() array. For example:

```
set env(PATH) /usr/bin:/bin:/usr/etc:/etc
```

sets the PATH for the Sysctl server. The **env**() array is assigned an authorization callback of SYSTEM which prevents its modification from outside the Sysctl server by a request sent by a Sysctl client.

**load** *lib_path* **[init_proc]**
The **load** command dynamically loads the shared library at *lib_path* into memory. If the **init_proc** parameter is given, it is used as the library's initialization procedure. Otherwise, the name of the initialization function is derived from the library name as follows:

```
library name    init function
------------    -------------
libxxx.sl       xxx_Init()
libxxx.a        xxx_Init()
```

The Sysctl server exports an API which the library uses to define commands, variables, authorization callbacks, and interpreter deletion callbacks. See the **load**() help page for details.

**Other Sysctl Commands**
The configuration file can also contain other Sysctl commands to modify the behavior of the server. For example, the following command in the configuration file causes the authorization callback for the **svcconnect** command to be changed from the default value of AUTH to NONE. This would allow nonauthenticated clients to connect to the server.

```
setauth -cmd svcconnect NONE
```

## Prerequisite Information

See the description of Sysctl in *PSSP: Administration Guide*.

## Related Information

SP Command: **sysctl**

SP Daemon: **sysctld**

## tuning.commercial File

### Purpose

**tuning.commercial** – Contains initial performance tuning parameters for a typical commercial SP environment.

### Description

This file is a Korn shell script file containing commands to set network performance tuning parameters. It can be copied to the **/tftpboot/tuning.cust** file on the control workstation for propagation to the nodes.

### Related Information

SP Commands: **cptuning**

AIX Commands: This file contains invocations of the **no** command.

SP Files: **tuning.default**, **tuning.development**, **tuning.scientific**

*PSSP: Installation and Migration Guide*

## tuning.default File

### Purpose

**tuning.default** – Contains initial (default) performance tuning parameters for an SP environment.

### Description

This file is a Korn shell script file containing commands to set network performance tuning parameters. In the absence of explicit administrator action, this file is copied to the **/tftpboot/tuning.cust** file on the control workstation for propagation to the nodes.

### Related Information

SP Commands: **cptuning**

AIX Commands: This file contains invocations of the **no** command.

SP Files: **tuning.commercial**, **tuning.development**, **tuning.scientific**

*PSSP: Installation and Migration Guide*

## tuning.development File

### Purpose

**tuning.development** – Contains initial performance tuning parameters for a typical software development/interactive SP environment.

### Description

This file is a Korn shell script file containing commands to set network performance tuning parameters. It can be copied to the **/tftpboot/tuning.cust** file on the control workstation for propagation to the nodes.

### Related Information

SP Commands: **cptuning**

AIX Commands: This file contains invocations of the **no** command.

SP Files: **tuning.commercial**, **tuning.default**, **tuning.scientific**

*PSSP: Installation and Migration Guide*

## tuning.scientific File

### Purpose

**tuning.scientific** – Contains initial performance tuning parameters for a typical engineering or scientific SP environment.

### Description

This file is a Korn shell script file containing commands to set network performance tuning parameters. It can be copied to the **/tftpboot/tuning.cust** file on the control workstation for propagation to the nodes.

### Related Information

Commands: **cptuning**

AIX Commands: This file contains invocations of the **no** command.

SP Files: **tuning.commercial**, **tuning.default**, **tuning.development**

*PSSP: Installation and Migration Guide*

# Chapter 3. SP Subroutines

## getvhostname Subroutine

### Purpose

**getvhostname** – Gets the virtual host name.

### Library

Availability Library (libavail.a)

### Syntax

```
#include <vhost.h>
int getvhostname (name, name_length);
char *name;
int name_length;
```

| | |
|---|---|
| *name* | Specifies the virtual host name. |
| *name_length* | Specifies the length of the *name* array. |

### Description

Use this subroutine to retrieve the virtual host name of a host machine. This routine is similar to the **gethostname** system call with the exception that it retrieves the virtual host name from the **/etc/vhostname** file instead of using a kernel variable. The **getvhostname** subroutine is a library call and **gethostname** is a system call.

The *name* is retrieved from the **/etc/vhostname** file. If the file does not exist, the **gethostname** system call is used and the real host name is returned. If excess space is provided, the returned *name* parameter is null terminated. If insufficient space is provided, the returned *name* parameter is truncated to fit in the given space. Virtual host names are limited to MAX_VHOSTNAME_LEN bytes (255), not including the terminating null character. The MAX_VHOSTNAME_LEN macro is defined in the **vhost.h** header file. To guarantee sufficient buffer space to hold the virtual host name, the *name_length* parameter should be MAX_VHOSTNAME_LEN + 1 or 256.

To clear the virtual host name so that the virtual host name no longer exists, remove the **/etc/vhostname** file.

**Note:** You must have root authority to remove the **/etc/vhostname** file.

The virtual host name is used in fail over situations when an application has associated the host name in the kernel of a particular machine to the service it is providing. When the application is restarted on the fail over node that has a different host name, the application may not work or work incorrectly. If the application needs to associate a host name with a particular service and it cannot handle having multiple host names, a virtual host name can be provided. The application can call **getvhostname** instead of **gethostname** and get the host name of the node it normally runs on. This eliminates the need to change the real host name in the kernel on the backup node. It should be noted that changing the real host name in the kernel can cause problems with other applications that rely on the real host name to identify the *physical machine*.

**Note:** The High Availability Cluster Multiprocessing (HACMP) event scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set and clear the virtual host name in the supplied HACMP pre- and post-event scripts. The administrator should not normally have to set the virtual host name.

## Return Values

Upon successful completion, the **getvhostname** subroutine returns a value of 0. Otherwise, a value of -1 is returned, the global variable **errno** is set to identify the error, and the contents of the buffer pointed to by the *name* parameter are indeterminate.

## Error Values

The **getvhostname** subroutine is unsuccessful if the following error occurs:

| | |
|---|---|
| **EFAULT** | Indicates that either the *name* or *name_length* parameter gave an address that is not valid. |
| **EINVAL** | Indicates that the *name_length* parameter is less than 0. |

If one of the system calls used to retrieve the virtual host name from the **/etc/vhostname** file encounters and error(for example, open or read), **errno** is set by the system call that encountered an error.

## Related Information

Commands: **vhostname**

Subroutines: **setvhostname**

AIX Commands: **hostname**

AIX Subroutines: **gethostname**, **sethostname**

Header Files: **vhost.h**

## Examples

1. To clear the virtual host name so that it no longer exists, enter:

   ```
   rm /etc/vhostname
   ```

   **Note:** You must have root authority to remove the **/etc/vhostname** file.

2. To get the virtual host name from the **/etc/vhostname** file, enter:

   ```
   #include <vhost.h>
   main ( )
   {
   char name [MAX_VHOSTNAME_LEN + 1];
   getvhostname (name, (MAX_VHOSTNAME_LEN + 1));
   }
   ```

## hacws_set Subroutine

### Purpose

**hacws_set** – Sets the state of the control workstation.

### Library

Availability Library (libavail.a)

### Location

**/usr/lib/libavail.a**

### Syntax

```
#include <hacws.h>
int hacws_set (state);
int state;
```

### Parameters

| | |
|---|---|
| *state* | Specifies the state of the control workstation. Valid values are: 0, 1, 2, 16, 32. |

### Description

Use this subroutine to set the current state of the control workstation. It is valid only when issued on the control workstation. When the subroutine is called and the calling process is not on a control workstation, an error occurs.

**Note:** The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set the control workstation state. The state is changed during fail over or reintegration in the HACWS supplied pre- and post-event scripts for HACMP. The administrator should not normally have to set the control workstation state.

### Return Values

Upon successful completion, the **hacws_set** subroutine returns a value of 0. Otherwise, a value of -1 is returned and the global variable **errno** is set to identify the error.

### Error Values

The **hacws_set** subroutine is unsuccessful if any of the following errors occur:

| | |
|---|---|
| **EINVAL** | Indicates that the value of the *state* parameter is not one of the valid values contained in **hacws.h**. |
| **ENODEV** | Indicates that the calling process is not on a control workstation. |
| **ENOENT** | Indicates that the **hacws_set** subroutine could not determine whether the calling process is on a control workstation. |
| **EPERM** | Indicates that the calling process did not have root's effective user ID. |

If one of the system calls used to store the HACWS state value into the **/etc/hacws.state** file encounters an error (for example, open, write, rename), **errno** is set by the system call that encountered and error.

## Macros

The **/usr/include/hacws.h** header file defines the following macros as valid input values for the **hacws_set** subroutine:

**0** or **HACWS_NOT_AN_HACWS**
Indicates that this control workstation is not in an HACWS configuration.

**1** or **HACWS_PRIM_INACT_CWS**
Indicates that this control workstation is the primary control workstation, but not the currently active control workstation.

**2** or **HACWS_PRIM_ACT_CWS**
Indicates that this control workstation is the primary control workstation and is the currently active control workstation.

**16** or **HACWS_BACK_INACT_CWS**
Indicates that this control workstation is the backup control workstation, but not the currently active control workstation.

**32** or **HACWS_BACK_ACT_CWS**
Indicates that this control workstation is the backup control workstation and is the currently active control workstation.

## Prerequisite Information

Refer to *PSSP: Administration Guide* for information on the HACWS option.

## Related Information

Commands: **lshacws**, **sethacws**

Subroutines: **hacws_set**

Header Files: **hacws.h**

## hacws_stat Subroutine

### Purpose

**hacws_stat** – Gets the state of the control workstation.

### Library

Availability Library (libavail.a)

### Location

**/usr/lib/libavail.a**

### Syntax

```
#include <hacws.h>
int hacws_stat (void);
```

### Description

Use this subroutine to return the current state of the control workstation. It returns an integer that indicates the state of the primary or backup control workstation and specifies whether the control workstation is a high availability configuration. This subroutine is valid only when issued on the control workstation. When the subroutine is called and the calling process is not on a control workstation, an error occurs.

**Note:** The High Availability Cluster Multiprocessing (HACMP) event scripts and installation scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set the control workstation state. The state is changed during fail over or reintegration in the HACWS supplied pre- and post-event scripts for HACMP. The administrator should not normally have to set the control workstation state.

### Return Values

Upon successful completion, the **hacws_stat** subroutine returns a nonnegative value. If the **hacws_stat** subroutine is unsuccessful, a value of -1 is returned and the global variable **errno** is set to identify the error.

### Error Values

The **hacws_stat** subroutine is unsuccessful if any of the following errors occur:

| | |
|---|---|
| **ENODEV** | Indicates that the calling process is not on a control workstation. |
| **ENOENT** | Indicates that the **hacws_stat** subroutine could not determine whether the calling process is on a control workstation. |
| **ERANGE** | Indicates that the **/etc/hacws.state** file does not contain a valid HACWS state. |

If one of the system calls used to retrieve the HACWS state value from the **/etc/hacws.state** file encounters an error (for example, open or read), **errno** is set by the system call that encountered an error.

## Macros

The **/usr/include/hacws.h** header file defines the following macros for the nonnegative return values for the **hacws_stat** subroutine:

**0** or **HACWS_NOT_AN_HACWS**
: Indicates that this control workstation is not in an HACWS configuration.

**1** or **HACWS_PRIM_INACT_CWS**
: Indicates that this control workstation is the primary control workstation, but not the currently active control workstation.

**2** or **HACWS_PRIM_ACT_CWS**
: Indicates that this control workstation is the primary control workstation and is the currently active control workstation.

**16** or **HACWS_BACK_INACT_CWS**
: Indicates that this control workstation is the backup control workstation, but not the currently active control workstation.

**32** or **HACWS_BACK_ACT_CWS**
: Indicates that this control workstation is the backup control workstation and is the currently active control workstation.

## Prerequisite Information

Refer to *PSSP: Administration Guide* for information on the HACWS option.

## Related Information

Commands: **lshacws**, **sethacws**

Subroutines: **hacws_set**

Header Files: **hacws.h**

## LAPI_Address Subroutine

### Purpose

**LAPI_Address** – Gets an unsigned integer value for a specified address.

### C Syntax

```
#include <lapi.h>

int LAPI_Address(my_addr, ret_addr)
void *my_addr;
uint *ret_addr;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_ADDRESS(my_addr, ret_addr, ierror)
INTEGER my_addr;
INTEGER ret_addr;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *my_addr* | Specifies the address to save. This parameter cannot be NULL. (IN) |
| *ret_addr* | Stores the *my_addr* address for later use. This is especially useful in FORTRAN programs. This parameter cannot be NULL. (OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine in FORTRAN programs when specified addresses need to be stored in an array. In FORTRAN, the concept of address ('&') does not exist as it does in C. This function gives that ability to FORTRAN.

### Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

### Prerequisite Information

Refer to the "Understanding and Using the Communications Low-Level Application Programming Interface" chapter in *PSSP: Administration Guide* for additional LAPI information.

## LAPI_Address_init Subroutine

### Purpose

**LAPI_Address_init** – Exchanges virtual addresses for non-Single Program Multiple Data (SPMD) programs and for dynamically allocated data.

### C Syntax

```
#include <lapi.h>

int LAPI_Address_init(hndl, my_addr, add_tab)
lapi_handle_t hndl;
void *my_addr;
void *add_tab[];
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_ADDRESS_INIT(hndl, my_addr, add_tab, ierror)
INTEGER hndl;
<type> my_addr(*);
INTEGER add_tab(*);
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-Level Application Programming Interface (LAPI) context. (IN) |
| *my_addr* | Specifies the entry supplied by each process. This parameter can be NULL. (IN) |
| *add_tab* | Specifies the address table containing the addresses supplied by all processes. This parameter can be NULL. (IN/OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to exchange virtual and dynamically allocated addresses. *add_tab* is an array of pointers of **size >= LAPI_Qenv(,NUM_TASKS,)**. This function is a collective call over the LAPI context *hndl* which fills the table *add_tab* with the entries supplied by each task. Upon completion of this call, *add_tab*[**i**] will contain the entry provided by task **i**.

### Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## LAPI_Amsend Subroutine

### Purpose

**LAPI_Amsend** – Invokes a user-provided Active Message (AM) handler to run on a remote (target) process.

### C Syntax

```
#include <lapi.h>

typedef void (compl_hndlr_t)(hndl, user_info);
lapi_handle_t *hndl;           LAPI context passed in from LAPI_Amsend.
void *       user_info; Buffer (user_info) pointer passed in from
                               header handler (void * (hnd_hndlr_t)).
typedef void * (hdr_hndlr_t)(hndl, uhdr, uhdr_len, msg_len,
comp_h, user_info);

lapi_handle_t    *hndl;      LAPI context passed in from LAPI_Amsend.
void *           uhdr;       uhdr passed in from LAPI_AMsend.
uint             *uhdr_len;  uhdr_len passed in from LAPI_Amsend.
uint *           msg_len;    udata_len passed in from LAPI_Amsend.
compl_hndlr_t ** comp_h;     Function address of completion handler
                             (void (compl_hndlr_t)) that needs to be
                             filled out by this header handler function.
void **          user_info;  Buffer pointer (user_info) that is
                             provided by this head handler function
                             to pass to the completion handler.

int LAPI_Amsend(hndl, tgt, hdr_hdl, uhdr, uhdr_len, udata,
   udata_len, tgt_cntr, org_cntr, cmpl_cntr)
lapi_handle_t hndl;
uint tgt;
void *hdr_hdl;
void *uhdr;
uint uhdr_len;
void *udata;
uint udata_len;
lapi_cntr_t *tgt_cntr;
lapi_cntr_t *org_cntr;
lapi_cntr_t *cmpl_cntr;
```

## FORTRAN Syntax

```
include 'lapif.h'

COMPL_H(hndl, user_info);
INTEGER hndl;
INTEGER user_info;

INTEGER FUNCTION HDR_HDL(hndl, uhdr, uhdr_len, msg_len, comp_h,
user_info)
INTEGER hndl;
INTEGER uhdr;
INTEGER uhdr_len;
INTEGER msg_len;
INTEGER comp_h;
INTEGER user_info;

LAPI_AMSEND(hndl, tgt, hdr_hdl, uhdr, uhdr_len, udata, udata_len,
tgt_cntr
   org_cntr, cmpl_cntr, ierror)
INTEGER hndl;
INTEGER tgt;
<type> hdr_hdl(*);
INTEGER uhdr;
INTEGER uhdr_len;
INTEGER udata;
INTEGER udata_len;
<type> tgt_cntr(*);
INTEGER org_cntr;
INTEGER cmpl_cntr;
INTEGER ierror;
```

## Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-Level Application Programming Interface (LAPI) context. (IN) |
| *tgt* | Specifies the target task number. This parameter is valid from **0** <**= tgt** < **LAPI_Qenv(,NUM_TASKS,)**. (IN) |
| *hdr_hdl* | Specifies the pointer to the remote header handler function to be invoked at the target. This parameter cannot be NULL. (IN) |
| *uhdr* | Specifies the pointer to the local header (parameter list) that is passed to the handler function. This parameter can be NULL if *uhdr_len* is equivalent to 0. (IN) |
| *uhdr_len* | This parameter is valid from **0** <**=** *uhdr_len* <**= LAPI_Qenv(,MAX_UHDR_SZ,)**. (IN) |
| *udata* | Specifies the pointer to the user data. This parameter can be NULL if *udata_len* is equivalent to 0. (IN) |
| *udata_len* | Specifies the length of the user data in bytes. This parameter is valid from **0** <**e** *udata_len* <**= LAPI_Qenv(,MAX_DATA_SZ,)**. (IN) |
| *tgt_cntr* | Specifies the target counter address. The target counter is incremented after data arrives at the target and after the completion handler completes. If the parameter is NULL, this counter will not be updated. (IN) |

| | |
|---|---|
| *org_cntr* | Specifies the origin counter address. The origin counter is incremented after data is copied out of the origin address. If the parameter is NULL, this counter will not be updated. (IN/OUT) |
| *cmpl_cntr* | Specifies the counter at the origin that signifies completion of the completion handler. It is updated once the completion handler completes. If the parameter is NULL, the counter will not be updated. (IN/OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

## Description

Use this subroutine to transfer the *hdr_hdl* function pointer along with the contents of *uhdr* and *udata* from the origin to the *tgt* target process. When the message arrives at the target process, the *hdr_hdl* header handler is invoked at the *tgt* target with the pointer to *uhdr* as one of the parameters.

The user-supplied header handler is expected to return a buffer pointer (*user_info* as the return value) in which *udata* is to be copied. The header handler is also expected to save any information that will be required later by the completion handler. The header handler returns (through reference parameters) the completion handler and a pointer to the saved information (*user_info*).

**Note:** The header handler should be nonblocking because no progress on the messages associated with *hndl* can be made until control is returned to the communications library from the header handler.

After the header handler returns, the *udata* (if any) is copied into the user-specified buffer (*user_info*). When all of the *udata* is copied into the user buffer, the completion handler you specified through the header handler is enqueued.

After the parameters (including the contents of *uhdr* and *udata*) are copied out of the memory at the origin, the *org_cntr* is incremented. After the completion handler finishes running at the *tgt*, the *tgt_cntr* is incremented. If the completion handler specified is NULL, the *tgt_cntr* is incremented after all of the *udata* is copied into the user-specified buffers. If the user-specified buffer is NULL and the completion handler is also NULL, the *tgt_cntr* will be incremented in some implementation-specific manner. Either counter addresses may be NULL.

This is a nonblocking call. The calling process cannot change the *uhdr* origin header and *udata* data until completion at the origin is signaled by the *org_cntr* being incremented. Similarly, you can assume that the specified AM handler has run at the *tgt* only after the *tgt_cntr* target counter is incremented. The *cmpl_cntr* and *tgt_cntr* counters are incremented after the AM handler has completed execution at the target. When the AM handler has both a *hdr_hdl* header handler and a *comp_h* completion handler, the *cmpl_cntr* and *tgt_cntr* counters are incremented after the completion handler has completed execution. If the AM handler has only a *hdr_hdl* header handler, the *cmpl_cntr* and *tgt_cntr* counters will be incremented after the header handler has completed execution. This call can be made synchronous if the origin waits for the *cmpl_cntr* update to complete.

The length (*uhdr_len*) of the user-specified header is constrained by an implementation specified maximum value **(LAPI_Qenv(,MAX_UHDR_SZ,))**. In the current implementation, the amount of *udata* sent per packet is

**LAPI_Qenv(,MAX_UHDR_SZ,) -** *uhdr_len*. To get the best bandwidth, *uhdr_len* should be as small as possible.

## Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Fence**, **LAPI_Getcntr**, **LAPI_Qenv**, **LAPI_Waitcntr**

## LAPI_Fence Subroutine

### Purpose

**LAPI_Fence** – Enforces order on Low-Level Application Programming Interface (LAPI) calls.

### C Syntax

```
#include <lapi.h>

int LAPI_Fence(hndl)
lapi_handle_t hndl;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_FENCE(hndl, ierror)
INTEGER hndl;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular LAPI context. (IN) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to enforce order on LAPI calls. If a process calls **LAPI_Fence()**, all the LAPI operations that were initiated by that process, before the fence using the LAPI context *hndl*, are guaranteed to complete at the target processes. This occurs before any of its communication operations using *hndl*, initiated after the fence, complete at the target processes. This is a data fence which means that the data movement is complete. This is not an operation fence which would need to include Active Message completion handlers completing on the target.

### Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

### Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Amsend**, **LAPI_Get**, **LAPI_Gfence**, **LAPI_Put**, **LAPI_Rmw**

## LAPI_Get Subroutine

### Purpose

**LAPI_Get** – Copies data from a remote process to the local address on a local process.

### C Syntax

```
#include <lapi.h>

int LAPI_Get(hndl, tgt, len, tgt_addr, org_addr, tgt_cntr, org_cntr)
lapi_handle_t hndl;
uint tgt;
uint len;
void *tgt_addr;
void *org_addr;
lapi_cntr_t *tgt_cntr;
lapi_cntr_t *org_cntr;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_GET(hndl, tgt, len, tgt_addr, org_addr, tgt_cntr, org_cntr,
ierror)
INTEGER hndl;
INTEGER tgt;
INTEGER len;
<type> tgt_addr(*);
INTEGER org_addr;
<type> tgt_cntr(*);
INTEGER org_cntr;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-level Applications Programming Interface (LAPI) context. (IN) |
| *tgt* | Specifies the target task that is the source of the data. This parameter is valid from **0** <= *tgt* < **LAPI_Qenv(,NUM_TASKS,)**. (IN) |
| *len* | Specifies the number of bytes of data to be copied. This parameter is valid from **0** <= *len* <= **LAPI_Qenv(,MAX_DATA_SZ,)**. (IN) |
| *tgt_addr* | Specifies the target buffer address of the data source. This parameter can be NULL only if *len* is equivalent to 0. (IN) |
| *org_addr* | Specifies the local buffer address that the received data is copied into. This parameter can be NULL only if *len* is equivalent to 0. (IN/OUT) |
| *tgt_cntr* | Specifies the target counter address. The target counter is incremented after data arrives at the target. If the parameter is NULL, this counter will not be updated. (IN) |
| *org_cntr* | Specifies the origin counter address. The origin counter is incremented after data arrives at the origin. If the parameter is NULL, the counter will not be updated. (IN/OUT) |

| | |
|---|---|
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

## Description

Use this subroutine to transfer the *len* number of bytes from the *tgt_addr* address at the target process to the *org_addr* virtual address at the origin process over the port identified by *hndl*. After the data is copied out of the memory at the *tgt_addr*, the *tgt_cntr* is incremented. After the data arrives at the origin, the *org_cntr* is incremented. If either counter address is NULL, the data transfer occurs, but the corresponding counter increment does not take place.

This is a nonblocking call in that the calling program cannot assume that the target buffer can be changed, nor that the contents of the memory pointed to by the *org_addr* on the origin is ready for use. However, after the origin waits for the *org_cntr* update to complete, the origin can use the *org_addr* data. Similarly, the target can reuse the target buffer *tgt_addr* only after it has waited for the *tgt_cntr* update to complete at the target.

## Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Fence**, **LAPI_Getcntr**, **LAPI_Gfence**, **LAPI_Put**, **LAPI_Qenv**, **LAPI_Waitcntr**

## LAPI_Getcntr Subroutine

### Purpose

**LAPI_Getcntr** – Gets the integer value of the counter.

### C Syntax

```
#include <lapi.h>

int LAPI_Getcntr(hndl, cntr, val)
lapi_handle_t hndl;
lapi_cntr_t *cntr;
int *val;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_GETCNTR(hndl, cntr, val, ierror)
INTEGER hndl;
INTEGER cntr;
INTEGER val;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-Level Application Programming Interface (LAPI) context. (IN) |
| *cntr* | Specifies the address of the counter. This parameter cannot be NULL. (IN) |
| *val* | Stores the integer value of the counter. This parameter cannot be NULL. (OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to get the integer value of *cntr*. It can be used to find how much progress is being made in LAPI context *hndl*. In conjunction, **LAPI_Probe()** can be used to make progress in LAPI context *hndl* if **LAPI_Getcntr()** is called inside a loop.

### Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Amsend**, **LAPI_Get**, **LAPI_Probe**, **LAPI_Put**, **LAPI_Setcntr**, **LAPI_Waitcntr**

## LAPI_Gfence Subroutine

### Purpose

**LAPI_Gfence** – Enforces order on Low-Level Application Programming Interface (LAPI) calls on all processes.

### C Syntax

```
#include <lapi.h>

int LAPI_Gfence(hndl)
lapi_handle_t hndl;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_GFENCE(hndl, ierror)
INTEGER hndl;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular LAPI context. (IN) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

This is a collective call. On completion of this call, it is assumed that all LAPI communications associated with *hndl* from all processes has quiesced.

**Note:** Although *hndl* is a local variable, it has a set of nodes that were associated with it at **LAPI_Init** all of which have to participate in this operation in order for it to complete.

This is a data fence which means that the data movement is complete. This is not an operation fence which would need to include Active Message completion handlers completing on the target.

### Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

### Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Fence**

## LAPI_Init Subroutine

### Purpose

**LAPI_Init** – Initializes the Low-Level Application Programming Interface (LAPI) subsystem.

### C Syntax

```
#include <lapi.h>

int LAPI_Init(hndl, lapi_info)
lapi_handle_t *hndl;
lapi_info_t *lapi_info;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_INIT(hndl, lapi_info, ierror)
INTEGER hndl;
INTEGER lapi_info(10);
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular LAPI context. This parameter cannot be NULL. (OUT) |
| *lapi_info* | Specifies a structure that provides the parallel job information that this LAPI context is associated with. This parameter cannot be NULL. (IN/OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to instantiate a new context of the LAPI subsystem and to initialize it. A handle to the newly-created LAPI context is returned in *hndl*. All subsequent LAPI calls can use *hndl* to specify the context of the LAPI operation. The *lapi_info* structure (**lapi_info_t**) needs to be filled in:

```
typedef struct        {        /* Not in use currently */
        lapi_dev_t      protocol;     /* OUT - Which protocol is
                                         initialized */
        int             info2         /* Future support */
        int             info3;        /* Future support */
        int             info4;        /* Future support */
        int             info5;        /* Future support */
        int             info6;        /* Future support */
        LAPI_err_hndlr  *err_hndlr;   /* IN - User registered error
                                         handler */
        void            *info_info2;  /* Future support */
        void            *info_info3;  /* Future support */
        void            *info_info4;  /* Future support */
}      lapi_info_t;
```

**lapi_dev_t** is defined as follows:

```
typedef enum {NULL_DEV=0, TB2_DEV, TB3_DEV,
              UDP_DEV, VIRTUAL_DEV, LAST_DEV} lapi_dev_t;
```

**Note:** Only the **TB3_DEV lapi_dev_t** type is supported at this time.

You can register an error handler through the *lapi_info* structure.

To create a function, you need the following parameters:

```
void  (User func name)   (lapi_handle_t *hndl,   /* LAPI handle */
                          int *error_code,       /* Error code */
                          lapi_err_t *err_type,  /* GET/PUT/RMW/AM/
                                                    INTERNAL */
                          int *task_id,          /* Current node */
                          int *src);             /* Source node */
```

The error code (**error_code*) of **LAPI_ERR_TIMEOUT** is a recoverable error if you choose to ignore it in your error handler. All other error codes are currently terminal and you should do clean-up processing of your process and terminate the process (for example, exit()).

An error occurs if any LAPI calls are made before calling **LAPI_Init()**, except for **LAPI_Address()** and **LAPI_Msg_string()**.

## Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**CSS_KE_INTERNAL_ERROR**
Is a system error indicating that the Kernel extension internal memory management encountered an error.

**CSS_KE_UCODE_ERROR**
Is a system error indicating that the adapter microcode is not responding.

**EBUSY** Is a system error indicating that the previous job is still running.

**EINVAL** Is a system error indicating that the specified argument was not valid.

**ENODEV** Is a system error indicating that the adapter type and library do not match.

**ENOSPC** Is a system error indicating that you cannot attach to bus memory (either out of memory or segment register).

**EPERM** Is a system error indicating that the caller is not authorized to perform the specified action.

**ETIMEDOUT** Is a system error indicating that the switch network is not available.

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

**LAPI_ERR_INIT_FAILED**
Indicates that initialization was unsuccessful.

**LAPI_ERR_NOMORE_PORTS**
Indicates that no more communication ports are available.

**LAPI_ERR_OPEN_FAILED**
Indicates that the opening of the communication device was unsuccessful.

**LAPI_ERR_UNKNOWN_DEVICE**
Indicates that the specified device is not supported.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Term**

## LAPI_Msg_string Subroutine

### Purpose

**LAPI_Msg_string** – Gets the Low-Level Application Programming Interface (LAPI) and system message string.

### C Syntax

```
#include <lapi.h>

LAPI_Msg_string(error_code, buf)
int error_code;
void * buf;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_MSG_STRING(error_code, buf, ierror)
INTEGER error_code;
INTEGER buf(40);
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *error_code* | Specifies the return value of a previous LAPI call. (IN) |
| *buf* | Specifies the buffer to store the message string. (OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to return the message string representation of the return value for a specific LAPI call.

### Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

### Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Init**, **LAPI_Term**

## LAPI_Probe Subroutine

### Purpose

**LAPI_Probe** – Transfers control to the communications subsystem to check for arriving messages and to make progress in polling mode.

### C Syntax

```
#include <lapi.h>

int LAPI_Probe(hndl)
lapi_handle_t hndl;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_PROBE(hndl, ierror)
INTEGER hndl;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-Level Application Programming Interface (LAPI) context. (IN) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to transfer control to the communications subsystem to make progress on messages associated with the context *hndl*.

**Note:** There is no guarantee about receipt of messages on the return from this function.

### Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

### Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Getcntr**, **LAPI_Setcntr**, **LAPI_Waitcntr**

## LAPI_Put Subroutine

### Purpose

**LAPI_Put** – Puts data into the target address on a target process.

### C Syntax

```
#include <lapi.h>

int LAPI_Put(hndl, tgt, len, tgt_addr, org_addr, tgt_cntr,
   org_cntr, cmpl_cntr)
lapi_handle_t hndl;
uint tgt;
uint len;
void *tgt_addr;
void *org_addr;
lapi_cntr_t *tgt_cntr;
lapi_cntr_t *org_cntr;
lapi_cntr_t *cmpl_cntr;
```

### FORTRAN Syntax

```
include 'lapif.h'

int LAPI_PUT(hndl, tgt, len, tgt_addr, org_addr, tgt_cntr,
   org_cntr, cmpl_cntr, ierror)
INTEGER hndl;
INTEGER tgt;
INTEGER len;
<type> tgt_addr(*);
INTEGER org_addr;
<type> tgt_cntr(*);
INTEGER org_cntr;
INTEGER cmpl_cntr;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-Level Application Programming Interface (LAPI) context. (IN) |
| *tgt* | Specifies the target task number. This parameter is valid from **0** <**=** *tgt* < **LAPI_Qenv(,NUM_TASKS,)**. (IN) |
| *len* | Specifies the number of bytes to be transferred. This parameter is valid from **0** <**=** *len* <**= LAPI_Qenv(,MAX_DATA_SZ,)**. (IN) |
| *tgt_addr* | Specifies the address on the target process where data is to be copied into. This parameter can be NULL only if *len* is equivalent to 0. (IN) |
| *org_addr* | Specifies the address on the origin process where data is to be copied from. This parameter can be NULL only if *len* is equivalent to 0. (IN) |
| *tgt_cntr* | Specifies the target counter address. The target counter is incremented after data arrives at the target. If the parameter is NULL, this counter will not be updated. (IN) |

| | |
|---|---|
| *org_cntr* | Specifies the origin counter address. The origin counter is incremented after data is copied out of the origin address. If the parameter is NULL, this counter will not be updated. (IN/OUT) |
| *cmpl_cntr* | Specifies the address of the completion counter that is a reflection of the *tgt_cntr*. This counter is incremented at the origin after the *tgt_cntr* is incremented. If the parameter is NULL, the counter will not be updated. (IN/OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

## Description

Use this subroutine to transfer the *len* number of bytes from the *org_addr* virtual address on the origin to the *tgt* target process at the *tgt_address* address over the port identified by *hndl*. After the data is copied out of the memory at *org_addr*, the *org_cntr* is incremented. After the data arrives at the *tgt*, the *tgt_cntr* is incremented. If either counter address is NULL, the data transfer occurs, but the corresponding counter increment does not take place.

This is a nonblocking call in that the calling program cannot assume that the origin buffer can be changed, nor that the contents of the memory pointed to by *tgt_addr* on *tgt* is ready for use. However, after the origin waits for the *org_cntr* update to complete, the origin can modify the *org_addr* origin buffer. Similarly, the target can modify the data in the *tgt_addr* target buffer after it has waited for the *tgt_cntr* update to complete on the target. This call can be made synchronous if the origin waits for the *cmpl_cntr* update to complete.

## Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Fence**, **LAPI_Get**, **LAPI_Getcntr**, **LAPI_Gfence**, **LAPI_Qenv**, **LAPI_Waitcntr**

## LAPI_Qenv Subroutine

### Purpose

**LAPI_Qenv** – Queries the Low-Level Application Programming Interface (LAPI) interface for parallel job information.

### C Syntax

```
#include <lapi.h>

int LAPI_Qenv(hndl, query, ret_val)
lapi_handle_t hndl;
lapi_query_t query;
int *ret_val;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_QENV(hndl, query, ret_val, ierror)
INTEGER hndl;
INTEGER query;
INTEGER ret_val;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular LAPI context. (IN) |
| *query* | Specifies the type of query requested as defined by **lapi_query_t** in **lapi.h**. (IN) |
| *ret_val* | Specifies the integer value of the query request. This parameter cannot be NULL. (OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to query the LAPI interface for information about a specific LAPI instance. **lapi_query_t** defines the types of LAPI queries available.

```
typedef enum {  TASK_ID=0,      /* Query task id of current task in
                                   job */
                NUM_TASKS,      /* Query number of tasks in job */
                MAX_UHDR_SZ,    /* Query max. user header size for
                                   AM */
                MAX_DATA_SZ,    /* Query max. data length that can
                                   be sent */
                ERROR_CHK,      /* Query & Set parameter checking
                                   on(1)/off(0) */
                TIMEOUT,        /* Query & Set current comm.
                                   timeout setting in seconds */
                MIN_TIMEOUT,    /* Query minimum comm. timeout
                                   setting */
                MAX_TIMEOUT,    /* Query maximum comm. timeout
                                   setting */
                INTERRUPT_SET,  /* Query & Set interrupt
                                   on(1)/off(0) */
                MAX_PORTS,      /* Query max. available comm. ports */
                MAX_PKT_SZ,     /* This is the payload size of 1
                                   packet */
                NUM_REX_BUFS,   /* Number of retransmission buffers */
                REX_BUF_SZ,     /* Size of Each retransmission buffer
                                   in bytes */
                LAST_QUERY} lapi_query_t;
```

## Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Amsend**, **LAPI_Get**, **LAPI_Put**, **LAPI_Senv**

## LAPI_Rmw Subroutine

### Purpose

**LAPI_Rmw** – Provides the synchronization primitives.

### C Syntax

```
#include <lapi.h>

int LAPI_Rmw(hndl, op, tgt, tgt_var, in_val, prev_tgt_val, org_cntr)
lapi_handle_t hndl;
RMW_ops_t op;
uint tgt;
int *tgt_var;
int *in_val;
int *prev_tgt_val;
lapi_cntr_t *org_cntr;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_RMW(hndl, op, tgt, tgt_var, in_val, prev_tgt_val, org_cntr,
ierror)
INTEGER hndl;
INTEGER op;
INTEGER tgt;
<type> tgt_var(*);
INTEGER in_val;
INTEGER prev_tgt_val;
INTEGER org_cntr;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-Level Application Programming Interface (LAPI) context. (IN) |
| *op* | Specifies the operation to be performed. (IN) |
| *tgt* | Specifies the target task where the Read-Modify-Write (RMW) variable resides. This parameter is valid from **0** <**=** *tgt* < **LAPI_Qenv(,NUM_TASKS,)**. (IN) |
| *tgt_var* | Specifies the target RMW variable address. This parameter cannot be NULL. (IN) |
| *in_val* | Specifies the value input to the *op*. This parameter cannot be NULL. (IN) |
| *prev_tgt_val* | Specifies the location at the origin in which the previous *tgt_var* on the target process is stored before the RMW *op* is executed. This parameter can be NULL. (IN/OUT) |
| *org_cntr* | Specifies the origin counter address. The origin counter is incremented after data is copied out of the origin address. If the parameter is NULL, this counter will not be updated. (IN/OUT) |

|  |  |
|---|---|
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

## Description

Use this subroutine to synchronize two independent operations, such as two processes sharing a common data structure. The operation is performed at the *tgt* target process and is atomic. The operation takes an *in_val* from the origin and performs one of four selected *op* operations on a *tgt_var* variable at the *tgt* target, and then replaces the *tgt_var* target variable with the results of the *op* operation. The *prev_tgt_val* original value of the *tgt_var* target variable is returned to the origin.

The valid operations for *op* are:

- **COMPARE_AND_SWAP**
- **FETCH_AND_ADD**
- **FETCH_AND_OR**
- **SWAP**

The operations are performed over the context referred to by *hndl*. The outcome of the execution of these calls is as if the following code was executed atomically:

```
*prev_tgt_val = *tgt_var;
*tgt_var = f(*tgt_var, *in_val);
```

where:

f(a,b) = a + b for **FETCH_AND_ADD**

f(a,b) = a | b for **FETCH_AND_OR** (bitwise or)

f(a,b) = b for SWAP

For **COMPARE_AND_SWAP**, *in_val* is treated as a pointer to an array of two integers, and the *op* is the following atomic operation:

```
if(*tgt_var == in_val[0])  {
   *prev_tgt_val = TRUE;
*tgt_var = in_val[1];
} else {
   *prev_tgt_val = FALSE;
}
```

All the calls are nonblocking. To test for completion, use the **LAPI_Getcntr** and **LAPI_Waitcntr** functions. There is no *tgt_cntr* on RMW calls and they do not provide any indication of completion on the *tgt* process.

## Return Values

|  |  |
|---|---|
| **LAPI_SUCCESS** | Indicates successful completion. |

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that addresses were passed in that were not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Getcntr**, **LAPI_Probe**, **LAPI_Qenv**, **LAPI_Setcntr**, **LAPI_Waitcntr**

## LAPI_Senv Subroutine

### Purpose

**LAPI_Senv** – Sets the Low-level Application Programming Interface (LAPI) environment for the specified context.

### C Syntax

```
#include <lapi.h>

int LAPI_Senv(hndl, query, set_val)
lapi_handle_t hndl;
lapi_query_t query;
int set_val;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_SENV(hndl, query, set_val, ierror)
INTEGER hndl;
INTEGER query;
INTEGER set_val;
INTEGER ierror;
```

### Parameters

*hndl* Contains a handle that specifies a particular LAPI context. (IN)

*query* Specifies the LAPI set environment type as defined by **lapi_query_t** in **lapi.h**. (IN)

*set_val* Specifies the integer value to set the LAPI environment. (IN)

*ierror* Specifies a FORTRAN return code. It is always the last argument. (OUT)

### Description

Use this subroutine to set the LAPI environment for a specific LAPI instance. **lapi_query_t** defines the types of LAPI set environment variables.

```
typedef enum { ...
               ERROR_CHK,          /* Query & Set parameter checking
                                      on(1)/off(0) */
               TIMEOUT,            /* Query & Set current
                                      communications timeout setting
                                      in seconds */
               INTERRUPT_SET,      /* Query & Set interrupt
                                      on(1)/off(0) */
               ... }  lapi_query_t;
```

To obtain the default values of the settings, use the **LAPI_Qenv** function.

**Note:** If **ERROR_CHK** is set to 0 for all LAPI calls, parameter error checking is ignored (for example, **LAPI_ERR_BAD_PARAMETER** is not returned).

## Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Qenv**

## LAPI_Setcntr Subroutine

### Purpose

**LAPI_Setcntr** – Sets a counter to a specified value.

### C Syntax

```
#include <lapi.h>

int LAPI_Setcntr(hndl, cntr, val)
lapi_handle_t hndl;
lapi_cntr_t *cntr;
int val;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_SETCNTR(hndl, cntr, val, ierror)
INTEGER hndl;
INTEGER cntr;
INTEGER val;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-Level Application Programming Interface (LAPI) context. (IN) |
| *cntr* | Specifies the address of the counter to be set. This parameter cannot be NULL. (IN/OUT) |
| *val* | Specifies the value the counter needs to be set to. (IN) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to set the *cntr* to the appropriate value. The LAPI context associated with *hndl* may or may not be polled for incoming messages.

### Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

### Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Getcntr**, **LAPI_Probe**, **LAPI_Waitcntr**

## LAPI_Term Subroutine

### Purpose

**LAPI_Term** – Terminates and cleans up the Low-Level Application Programming Interface (LAPI) subsystem.

### C Syntax

```
#include <lapi.h>

int LAPI_Term(hndl)
lapi_handle_t hndl;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_TERM(hndl, ierror)
INTEGER hndl;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular LAPI context. (OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to terminate the LAPI context specified by *hndl*. Any LAPI notification threads associated with this context are terminated. An error occurs when any LAPI calls are made using *hndl* after **LAPI_Term()** is called, except for **LAPI_Msg_string()** and **LAPI_Address()**.

### Return Values

| | |
|---|---|
| **LAPI_SUCCESS** | Indicates successful completion. |

The following is returned when an error occurs:

| | |
|---|---|
| **EINVAL** | Is a system error indicating that the specified argument was not valid. |
| **EPERM** | Is a system error indicating that the caller is not authorized to perform the specified action. |
| **LAPI_ERR_BAD_PARAMETER** | Indicates that a parameter was passed in that was not valid. |
| **LAPI_ERR_CLOSE_FAILED** | Indicates the close of the communication device was unsuccessful. |

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Init**

## LAPI_Waitcntr Subroutine

### Purpose

**LAPI_Waitcntr** – Waits until a specified counter reaches the value specified.

### C Syntax

```
#include <lapi.h>

int LAPI_Waitcntr(hndl, cntr, val, cur_cntr_val)
lapi_handle_t hndl;
lapi_cntr_t *cntr;
int val;
int *cur_cntr_val;
```

### FORTRAN Syntax

```
include 'lapif.h'

LAPI_WAITCNTR(hndl, cntr, val, cur_cntr_val, ierror)
INTEGER hndl;
INTEGER cntr;
INTEGER val;
INTEGER cur_cntr_val;
INTEGER ierror;
```

### Parameters

| | |
|---|---|
| *hndl* | Contains a handle that specifies a particular Low-Level Application Programming Interface (LAPI) context. (IN) |
| *cntr* | Specifies the address of the counter to be waited on. This parameter cannot be NULL. (IN) |
| *val* | Specifies the value the counter needs to reach. (IN) |
| *cur_cntr_val* | Specifies the integer value of the current counter. This parameter can be NULL. (OUT) |
| *ierror* | Specifies a FORTRAN return code. It is always the last argument. (OUT) |

### Description

Use this subroutine to wait until the *cntr* reaches or exceeds the specified *val*. Once the *cntr* reaches the *val*, the *cntr* is decremented by that value. (We say decremented rather than set to zero since the *cntr* could have had a value greater than the specified *val* when the call was made.) This call may or may not check for message arrivals over the LAPI context *hndl*; **LAPI_Probe** will always check for message arrivals.

## Return Values

**LAPI_SUCCESS** Indicates successful completion.

The following is returned when an error occurs:

**LAPI_ERR_BAD_PARAMETER**
Indicates that a parameter was passed in that was not valid.

## Prerequisite Information

Refer to the “Understanding and Using the Communications Low-Level Application Programming Interface” chapter in *PSSP: Administration Guide* for additional LAPI information.

## Related Information

Subroutines: **LAPI_Amsend**, **LAPI_Get**, **LAPI_Getcntr**, **LAPI_Probe**, **LAPI_Put**, **LAPI_Rmw**, **LAPI_Setcntr**

## setvhostname Subroutine

### Purpose

**setvhostname** – Sets the virtual host name.

### Library

Availability Library (libavail.a)

### Syntax

**#include <vhost.h>**

**int setvhostname** (*name*, *name_length*); **char** **name*; **int** *name_length*;

### Parameters

| | |
|---|---|
| *name* | Specifies the virtual host name. |
| *name_length* | Specifies the length of the *name* array. |

### Description

Use this subroutine to set the virtual host name of a host machine. Only programs with a root user ID can use this subroutine. This routine is similar to the **sethostname** system call with the exception that it stores the virtual host name in the **/etc/vhostname** file instead of using a kernel variable. The **setvhostname** subroutine is a library call and **sethostname** is a system call.

The *name* is stored in the **/etc/vhostname** file. If the file does not exist, it will be created. If it does exist, the file contents will be overwritten by the new virtual host name. Virtual host names are limited to MAX_VHOSTNAME_LEN bytes (255), not including the terminating null character. The MAX_VHOSTNAME_LEN macro is defined in the **vhost.h** header file. The *name_length* parameter does not have to allow for the terminating null character, therefore, the largest allowable value for *name_length* is MAX_VHOSTNAME_LEN.

To clear the virtual host name so that the virtual host name no longer exists, remove the **/etc/vhostname** file.

**Note:** You must have root authority to remove the **/etc/vhostname** file.

The virtual host name is used in fail over situations when an application has associated the host name in the kernel of a particular machine to the service it is providing. When the application is restarted on the fail over node that has a different host name, the application may not work or work incorrectly. If the application needs to associate a host name with a particular service and it cannot handle having multiple host names, a virtual host name can be provided. The application can call **getvhostname** instead of **gethostname** and get the host name of the node it normally runs on. This eliminates the need to change the real host name in the kernel on the backup node. It should be noted that changing the real host name in the kernel can cause problems with other applications that rely on the real host name to identify the *physical machine*.

**Note:** The High Availability Cluster Multiprocessing (HACMP) event scripts supplied with the High Availability Control Workstation (HACWS) option of the IBM Parallel System Support Programs for AIX (PSSP), set and clear the virtual host name in the HACMP pre- and post-event scripts. The administrator should not normally have to set the virtual host name.

## Return Values

Upon successful completion, the **setvhostname** subroutine returns a value of 0. Otherwise, a value of -1 is returned and the global variable **errno** is set to identify the error.

## Error Values

The **setvhostname** subroutine is unsuccessful if the following error occurs:

| | |
|---|---|
| **EINVAL** | Indicates that the *name_length* parameter is greater than MAX_VHOSTNAME_LEN or less than 0. |
| **EPERM** | Indicates that the calling process does not have an effective root user ID. |

If one of the system calls used to store the virtual host name into the **/etc/vhostname** file encounters an error (for example, open, write, rename), **errno** is set by the system call that encountered an error.

## Examples

1. To clear the virtual host name so that it no longer exists, enter:

   ```
   rm /etc/vhostname
   ```

   **Note:** You must have root authority to remove the **/etc/vhostname** file.

2. To set the virtual host name to **spcw_prim**, enter:

   ```
   #include <string.h>
   #include <vhost.h>
   main ( )
   {
   char name[]='spcw_prim';
   setvhostname(name, strlen(name));
   }
   ```

## Related Information

Commands: **vhostname**

Subroutines: **getvhostname**

AIX Commands: **hostname**

AIX Subroutines: **gethostname**, **sethostname**

## swclockGetIncrement Subroutine

### Purpose

**swclockGetIncrement** – Returns the hertz frequency at which the switch clock operates.

### Library

Switch Clock Library (libswclock.a)

### Location

**/usr/lib/libswclock.a**

### Syntax

```
#include <swclock.h>
int swclockGetIncrement(swclock_handle_t swclock_handle);
```

### Parameters

*swclock_handle*   Specifies the handle returned by the **swclockInit** subroutine.

### Description

Use this thread-safe subroutine to obtain the hertz frequency at which the switch clock operates. Switch clock frequency can be used to convert the switch clock value returned by the **swclockRead** subroutine.

### Return Values

Upon successful completion, the **swclockGetIncrement** subroutine returns the hertz frequency at which the switch clock operates. Otherwise, a value of -1 is returned and the global variable **errno** is set to identify the error.

### Error Values

| | |
|---|---|
| **EINVAL** | Indicates that the switch clock read interface was not initialized by the current thread. |
| **EPERM** | Indicates that a handle that was not valid was passed to the **swclockGetIncrement** subroutine. |

### Related Information

Subroutines: **swclockInit**, **swclockRead**, **swclockReadSec**, **swclockTerm**

Header File: **swclock.h**

## swclockInit Subroutine

### Purpose

**swclockInit** – Initializes the switch clock read interface for a thread.

### Library

Switch Clock Library (libswclock.a)

### Location

**/usr/lib/libswclock.a**

### Syntax

```
#include <swclock.h>
swclock_handle_t swclockInit(void);
```

### Description

Use this thread-safe subroutine to initialize the switch clock read interface for the current thread. It returns a handle which must be passed as an input parameter to all other switch clock library subroutines.

Usage Notes:

1. This subroutine **must** be called on a per-thread basis.
2. This subroutine allocates a segment register (one per process) that might otherwise be used by shared memory, memory-mapped files or the extended heap.

### Return Values

Upon successful completion, the **swclockInit** subroutine returns a handle that must be passed as input to all other switch clock library subroutines. Otherwise, a value of 0 is returned and the global variable **errno** is set to identify the error.

### Error Values

**EAGAIN** Indicates that the node is not active on the switch. Since this may be a temporary condition, the **swclockInit** subroutine should be retried; IBM suggests a number of retry attempts of **SWCLOCK_RETRY**. If retry attempts are unsuccessful, refer to the “Using a Switch” chapter in the *PSSP: Administration Guide* for information on determining switch connectivity.

**EBUSY** Indicates that the switch clock lock for the process could not be obtained. The **swclockInit** subroutine should be retried; IBM suggests a number of retry attempts of **SWCLOCK_RETRY**.

**ENOENT, ENXIO**
Indicates that either the switch adapter does not exist or the adapter did not configure successfully.

**ENOMEM** Indicates that the **swclockInit** subroutine could not obtain sufficient system resources to satisfy the request. The **swclockInit** subroutine should be retried; IBM suggests a number of retry attempts of **SWCLOCK_RETRY**.

**ETXTBSY**
Indicates that diagnostics is running on the switch adapter. The **swclockInit** subroutine should be retried after adapter diagnostics have completed.

## Related Information

*PSSP: Administration Guide*

Subroutines: **swclockGetIncrement**, **swclockRead**, **swclockReadSec**, **swclockTerm**

Header File: **swclock.h**

## swclockRead Subroutine

### Purpose

**swclockRead** – Returns the current switch clock value.

### Library

Switch Clock Library (libswclock.a)

### Location

**/usr/lib/libswclock.a**

### Syntax

```
#include <swclock.h>
long64_t swclockRead(swclock_handle_t swclock_handle);
```

### Parameters

| | |
|---|---|
| *swclock_handle* | Specifies the handle returned by the **swclockInit** subroutine. |

### Description

Use this thread-safe subroutine to read the switch clock. The switch clock value can be converted using the frequency returned by the **swclockGetIncrement** subroutine. The **swclockRead** subroutine can be called as many times as needed once the switch clock read interface is initialized for the current thread.

The switch clock is synchronous across all nodes active on a switch. Its value is set to zero when the primary node powers on, and can be reset during switch operation and management.

Usage Notes:

1. IBM suggests that a SIGBUS signal handler be established before this subroutine is called since adapter problems can result in a bus error when the switch clock is accessed. The SIGBUS signal handler should regard such an error as permanent. For information on diagnosing adapter problems, refer to the "Diagnosing Switch Problems" chapter in the *PSSP: Diagnosis Guide*.
2. Callers of this subroutine may want to check for a switch clock value that has regressed since the previous call due to a wrap or resetting condition. The High Performance Switch communications adapter wraps after approximately 81 days, while the SP Switch communications adapter effectively has no wrap. For events that can reset the switch clock, refer to the "Using a Switch" chapter in the *PSSP: Administration Guide*.

### Return Values

Upon successful completion, the **swclockRead** subroutine returns the current value of the switch clock. Otherwise, a value of -1 is returned and the global variable **errno** is set to identify the error.

## Error Values

| | |
|---|---|
| **EAGAIN** | Indicates that the node is not active on the switch. Since this may be a temporary condition, the **swclockRead** subroutine should be retried; IBM suggests a number of retry attempts of **SWCLOCK_RETRY**. If retry attempts are unsuccessful, refer to the “Using a Switch” chapter in the *PSSP: Administration Guide* for information on determining switch connectivity. |
| **EINVAL** | Indicates that the switch clock read interface was not initialized by the current thread. |
| **EPERM** | Indicates that a handle that was not valid was passed to the **swclockRead** subroutine. |

## Related Information

*PSSP: Diagnosis Guide*

*PSSP: Administration Guide*

Subroutines: **swclockGetIncrement**, **swclockInit**, **swclockReadSec**, **swclockTerm**

Header File: **swclock.h**

## swclockReadSec Subroutine

### Purpose

**swclockReadSec** – Returns the current switch clock value in seconds.

### Library

Switch Clock Library (libswclock.a)

### Location

**/usr/lib/libswclock.a**

### Syntax

```
#include <swclock.h>
double   swclockReadSec(swclock_handle_t swclock_handle);
```

### Parameters

*swclock_handle*   Specifies the handle returned by the **swclockInit** subroutine.

### Description

Use this thread-safe subroutine to read the switch clock. The **swclockReadSec** subroutine returns switch clock value converted to seconds. It can be called as many times as needed to read the switch clock once the switch clock read interface is initialized for the current thread.

The switch clock is synchronous across all nodes active on a switch. Its value is set to zero when the primary node powers on, and can be reset during switch operation and management.

Usage Notes:

1. IBM suggests that a SIGBUS signal handler be established before this subroutine is called since adapter problems can result in a bus error when the switch clock is accessed. The SIGBUS signal handler should regard such an error as permanent. For information on diagnosing adapter problems, refer to the “Diagnosing Switch Problems” chapter in the *PSSP: Diagnosis Guide*.
2. Callers of this subroutine may want to check for a switch clock value that has regressed since the previous call due to a wrap or resetting condition. The High Performance Switch communications adapter wraps after approximately 81 days, while the SP Switch communications adapter effectively has no wrap. For events that can reset the switch clock, refer to the “Using a Switch” chapter in the *PSSP: Administration Guide*.

### Return Values

Upon successful completion, the **swclockReadSec** subroutine returns the current value of the switch clock converted to seconds. Otherwise, a value of -1 is returned and the global variable **errno** is set to identify the error.

## Error Values

**EAGAIN** Indicates that the node is not active on the switch. Since this may be a temporary condition, the **swclockReadSec** subroutine should be retried; IBM suggests a number of retry attempts of **SWCLOCK_RETRY**. If retry attempts are unsuccessful, refer to the “Using a Switch” chapter in the *PSSP: Administration Guide* for information on determining switch connectivity.

**EINVAL** Indicates that the switch clock read interface was not initialized by the current thread.

**EPERM** Indicates that a handle that was not valid was passed to the **swclockReadSec** subroutine.

## Related Information

*PSSP: Diagnosis Guide*

*PSSP: Administration Guide*

Subroutines: **swclockGetIncrement**, **swclockInit**, **swclockRead**, **swclockTerm**

Header File: **swclock.h**

# swclockTerm Subroutine

## Purpose

**swclockTerm** – Terminates the switch clock read interface for a thread.

## Library

Switch Clock Library (libswclock.a)

## Location

**/usr/lib/libswclock.a**

## Syntax

```
#include <swclock.h>
int swclockTerm(swclock_handle_t swclock_handle);
```

## Parameters

|  |  |
|---|---|
| *swclock_handle* | Specifies the handle returned by the **swclockInit** subroutine. |

## Description

Use this thread-safe subroutine to terminate the switch clock read interface for the current thread. Switch clock library subroutines called subsequent to the **swclockTerm** subroutine will encounter an error unless the thread reinitializes the interface. If the **swclockTerm** subroutine is not called, the switch clock read interface will be terminated when the thread itself terminates.

## Return Values

Upon successful completion, the **swclockTerm** subroutine returns a value of 0. Otherwise, a value of -1 is returned and the global variable **errno** is set to identify the error.

## Error Values

|  |  |
|---|---|
| **EBUSY** | Indicates that the switch clock lock for the process could not be obtained. The **swclockTerm** subroutine should be retried; IBM suggests a number of retry attempts of **SWCLOCK_RETRY**. |
| **EINVAL** | Indicates that the switch clock read interface was not initialized by the current thread. |
| **EPERM** | Indicates that a handle that was not valid was passed to the **swclockTerm** subroutine. |

## Related Information

Subroutines: **swclockGetIncrement**, **swclockInit**, **swclockRead**, **swclockReadSec**

Header File: **swclock.h**

## swtbl_clean_table Subroutine

### Purpose

**swtbl_clean_table** – Forces the unload of the job switch resource table window on the node from which it is invoked.

### Library

Job Switch Resource Table Services library (libswitchtbl.a)

### Syntax

```
#include <st_client.h>
int swtbl_clean_table(int version,char *adapter,int option,int window_id);
```

### Parameters

*version*
: Specifies the version of header file used. Should be **ST_VERSION** defined in **st_client.h**.

*adapter*
: Specifies the adapter to be opened upon which the window resides. Only **css0** is supported.

*option ST_OPTION*
: enumeration value which indicates the following action:

| | |
|---|---|
| *ST_LEAVE_INUSE:* | If a process is currently using the switch table then don't force the unload. |
| *ST_ALWAYS_KILL:* | If a process is currently using the switch table, stop the process and unload the table. |

*window_id*
: Specifies the window id for which the unload and cleanup will be done. If *window_id* equals **-1**, then the default window will be unloaded.

### Description

Use this subroutine to override user ID (uid) checking and to unload the job switch resource table window on the node from which it is invoked.

Normal unloading of the job switch resource table window by the **swtbl_unload_table** API verifies that the uid of the unload matches the uid specified during the load. The **swtbl_clean_table** API ignores this verification and allows the caller to unload the window from a node.

This subroutine should be used for error recovery and not for normal switch table unloading. For example, when a parallel job has left a process in use and the window did not unload by the **swtbl_unload_table** API, the caller can use **swtbl_clean_table** to clean up that node.

If the **ST_LEAVE_INUSE** option is specified and a process is using the switch, the **swtbl_clean_table** will not unload the window.

If the **ST_ALWAYS_KILL** option is specified, and a process is using the switch then the process will be stopped by a **SIGKILL** signal.

To invoke this interface, the effective user id of the calling process must be the root user id.

The default window is specified within the **st_client.h** file.

Any information or error messages are recorded in the **/var/adm/SPlogs/st/st_log** file.

## Return Values

Upon successful completion, the **swtbl_clean_table** subroutine returns a value of **ST_SUCCESS**. Otherwise, it returns an error value defined by the **ST_RETURN_CODE** enumerator found in **st_client.h**.

## Error Values

| | |
|---|---|
| **ST_INVALID_PARAM** | Indicates that a specified parameter was not valid. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_NOT_AUTHOR** | Caller did not have effective user id of root. |
| **ST_NO_SWITCH** | The open call of the specified adapter was unsuccessful. |
| **ST_SWITCH_NOT_LOADED** | Indicates that the switch table is not currently loaded. |
| **ST_SWITCH_IN_USE** | Indicates that the switch table is currently in use and the **ST_LEAVE_INUSE** option was specified. |
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |

## Related Information

Commands: **st_clean_table**, **st_status**

Subroutines: **swtbl_load_job**, **swtbl_load_table**, **swtbl_unload_job**, **swtbl_unload_table**, **swtbl_status**, **swtbl_status_node**

Files: **/usr/lpp/ssp/include/st_client.h**, **/var/adm/SPlogs/st/st_log**

## Examples

To clean up a running process and unload the table from this node:

```
#include <st_client.h>

  main() {
    int rc;
    enum ST_OPTION option = ST_ALWAYS_KILL;
    int window_id = 1;

    rc = swtbl_clean_table(ST_VERSION,"css0",option,window_id);
    fprintf(stdout,"swtbl_clean_table returned %d\n");
  }
```

## swtbl_load_job Subroutine

### Purpose

**swtbl_load_job** – Loads the job switch resource table on the indicated nodes for a single job.

### Library

Job Switch Resource Table Services library (libswitchtbl.a)

### Syntax

```
#include <st_client.h>
int swtbl_load_job(int version,uid_t uid,int job_key,int num_tasks, \
char *job_desc,char *cws, ST_NODE_INFO *nodeinfo);
```

### Parameters

*version*
: Specifies the version of header file used. Should be **ST_VERSION** defined in **st_client.h**.

*uid*
: Specifies the real user ID of the user for whom the tables are being loaded and who will be authenticated to run the user space job.

*job_key*
: Specifies a globally unique job key. This job key is then used later by the user space job to access the table. Must be greater than 0.

*num_tasks*
: Specifies the number of tasks that will be accessing the job switch resource table during user space execution.

*job_desc*
: Specifies the string describing the job that will be using the job switch resource table. Maximum length of 50 characters, truncated after maximum. Defaults to "no_job_description_given".

*cws*
: Specifies the string representing the control workstation or system partition where the specified nodes reside. It is used to determine the switch node numbers defined within the System Data Repository (SDR). The default is the **SP_NAME** environment variable.

*nodeinfo*
: Specifies the pointer to the **ST_NODE_INFO** structure that was allocated and defined by the caller. A nodeinfo structure must exist for every **num_tasks** defined. **ST_NODE_INFO** members:

  *char st_node_name*
  : Indicates the name or dotted decimal IP string of the node for which this data pertains. It works in combination with **st_addr**, if **st_node_name** is NULL, the **st_addr** must contain the struct **in_addr** address of the node. The **st_node_name** is checked first and if non-NULL, the **st_addr** is ignored. The maximum length is 256 characters.

  *struct in_addr st_addr*
  : Indicates the network address of the node for which this data pertains. It works in combination with

**st_node_name**, the **st_node_name** must be NULL to use this.

*int st_virtual_task_id*
Specifies an integer between 0 and (num_tasks-1), which represents this task. It must be sequential with other task ids for this job switch resource table.

*int st_window_id*
Specifies the window id to be loaded and used by the corresponding task id. The window id must be between 0 and 3 inclusive. Each window is dedicated to a task and cannot be shared. If -1 is specified, then the default window will be used.

## Description

Use this subroutine to load all of the job switch resource tables needed for a single parallel job. A switch table needs to be loaded upon every node used by the job for communication in user space mode over the switch.

An **ST_NODE_INFO** structure needs to be defined for every node that the parallel job will execute on. The **swtbl_load_job** subroutine interfaces with the **switchtbld** daemon over the network to call the local **swtbl_load_table** API which loads the table on that node.

The effective user id of the calling process must be the root user id.

Any information or error messages are recorded in the **/var/adm/SPlogs/st/st_log** file on the calling node and for all of the nodes within the request.

The default window is defined within the **st_client.h** header file.

## Return Values

Upon successful completion, the **swtbl_load_job** subroutine returns a value of **ST_SUCCESS**. Otherwise, it returns an integer value defined by the **ST_RETURN_CODE** enumerator found in **st_client.h**.

The **ST_NODE_INFO** member **st_return_code** is updated to reflect any errors that may have occurred on the individual nodes. This return code corresponds to the return code of the **swtbl_load_table** API called on that node.

The **ST_ALREADY_CONNECTED** return code does not indicate an error. It means that the load request was already sent to that node because the **st_node_name** was defined within a previous nodeinfo structure.

The **ST_UNLOADED** return code indicates that the switch table was unloaded upon that node after another node within the same request encountered an error.

## Error Values

| | |
|---|---|
| **ST_CANT_CONNECT** | Indicates that the connect request was unsuccessful. |
| **ST_INVALID_ADDR** | Indicates that the **st_addr** argument cannot be converted by **inet_ntoa**. |
| **ST_INVALID_PARAM** | Indicates that a specified parameter was not valid. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_INVALID_TASK_ID** | The virtual task id was negative, greater than the number of tasks requested or not sequentially defined. |
| **ST_NOT_AUTHOR** | Caller did not have effective user id of root. |
| **ST_SDR_ERROR** | Indicates an error occurred during interaction with the SDR. |
| **ST_SWITCH_NOT_LOADED** | Indicates that the load request was not issued due to another error. |
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |

## Related Information

Commands: **st_clean_table**, **st_status**

Subroutines: **swtbl_clean_table**, **swtbl_load_table**, **swtbl_status**, **swtbl_status_node**, **swtbl_unload_job**, **swtbl_unload_table**

Files: **/usr/lpp/ssp/include/st_client.h**, **/var/adm/SPlogs/st/st_log**

## Examples

Code fragment to load the job switch resource table for a job that will be executing on two nodes:

```
#include <st_client.h>
  main() {
    int rc;
    int job_key;
    int num_tasks = 2;
    struct ST_NODE_INFO node[num_tasks];
    struct hostent *hp;
    uid_t uid;

     uid = set_user_id();     /* User defined routine to get uid */
     job_key = set_unique_key(); /* User defined routine to get job_key */

     strcpy(node[0].st_node_name,"Node1");
     node[0].st_window_id = 0;
     node[0].st_virtual_task_id = 0;

     hp = gethostbyname("Node2");
     bcopy(hp->h_addr_list[0],&node[1].swtbl_addr,hp->h_length);
     node[1].st_window_id = 1;
     node[1].st_virtual_task_id = 1;

     rc = swtbl_load_job(uid,job_key,num_tasks,"User1_job",NULL,node);
     fprintf(stdout,"swtbl_load_job returned %d\n",rc);
  }
```

## swtbl_load_table Subroutine

### Purpose

**swtbl_load_table** – Loads a job switch resource table on the node from which it is invoked.

### Library

Job Switch Resource Table Services library (libswitchtbl.a)

### Syntax

```
#include <st_client.h>
int swtbl_load_table(int version,uid_t uid,pid_t pid,int job_key, \
char *requester_node, int num_tasks,char *job_desc,\
ST_NODE_INFO *nodeinfo);
```

### Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. Should be **ST_VERSION** defined in **st_client.h**. |
| *uid* | Specifies the real user ID of the user for whom the table is being loaded and who will be authenticated to run the user space job. |
| *pid* | Specifies the process ID (pid) of the calling process, typically a job scheduler. |
| *job_key* | Specifies a globally unique job key. This job key is then used later by the user space job to access the table. Must be greater than 0. |
| *requester_node* | Specifies the name of the node from which the request is being made. It is used in conjunction with **swtbl_load_job** API to determine where the load was made from. The default is "no_request_node_given" |
| *num_tasks* | Specifies the number of tasks that are accessing the job switch resource table during user space execution. |
| *job_desc* | Specifies the string that describes the job that is using the job switch resource table. Maximum length of 50 characters, truncated after maximum. Defaults to "no_job_description_given". |
| *nodeinfo* | Specifies the pointer to the **ST_NODE_INFO** structure that was allocated and defined by the caller. A nodeinfo structure must exist for every num_tasks defined. **ST_NODE_INFO** members: |

*int st_virtual_task_id*
: Specifies an integer between 0 and (num_tasks-1), which represents this task.

*int st_switch_node_num*
: Specifies the switch node number from the System Data Repository (SDR) for the node defined by this structure.

*int st_window_id*
: Specifies the window id to be loaded and used by the corresponding task id. The window id must be between 0 and 3 inclusive. Each window is dedicated to task and cannot be shared. If -1 is specified, then the default window will be used.

## Description

Use this subroutine to load the job switch resource table on the node from which it is invoked. This switch table is used by parallel jobs running user space over the switch.

An **ST_NODE_INFO** structure needs to be defined for every node that the parallel job will execute on.

The effective user id of the calling process must be the root user id.

Any information or error messages are recorded in the **/var/adm/SPlogs/st/st_log** file.

The default window is defined within the **st_client.h** header file.

## Return Values

Upon successful completion, the **swtbl_load_table** subroutine returns a value of **ST_SUCCESS**. Otherwise, it returns an integer value defined by the **ST_RETURN_CODE** enumerator found in st_client.h.

The **ST_NODE_INFO** member **st_return_code** is not updated by this subroutine.

## Error Values

| | |
|---|---|
| **ST_DOWNON_SWITCH** | Indicates that one or more nodes in switch table list are down on the switch or outside the switch boundaries. |
| **ST_INVALID_PARAM** | Indicates that a specified parameter was not valid. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_INVALID_TASK_ID** | The virtual task id was negative or greater than the number of tasks requested. |
| **ST_NO_SWITCH** | The open call of **/dev/css0** was unsuccessful. |
| **ST_NOT_AUTHOR** | Caller did not have effective user id of root. |
| **ST_INVALID_ADDR** | Indicates the node name or node address specified could not be resolved. |
| **ST_SWITCH_IN_USE** | Indicates a switch table is already loaded or the switch table is loaded and in use. |
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |

## Related Information

Commands: **st_clean_table**, **st_status**

Subroutines: **swtbl_clean_table**, **swtbl_load_job**, **swtbl_status**, **swtbl_status_node**, **swtbl_unload_job**, **swtbl_unload_table**

Files: **/usr/lpp/ssp/include/st_client.h**, **/var/adm/SPlogs/st/st_log**

## Examples

Code fragment to load the job switch resource table for a job that will be running on 2 nodes:

```
#include <st_client.h>
  main() {
    int i,rc;
    struct ST_NODE_INFO node[2];
    uid_t uid;
    pid_t pid;
    int job_key;
    int num_tasks=2;
    char hostname[MAXHOSTNAMELEN];

    uid = set_user_id();      /* User defined routine to get uid */
    pid = getpid();
    gethostname(hostname,MAXHOSTNAMELEN);
    job_key = set_unique_key(); /* User defined routine to get job_key */

    node[0].st_virtual_task_id = 0;
    node[0].st_window_id = 0;
    node[0].st_switch_node_num = get_num_from_SDR(); /* get switch num */

    node[1].st_virtual_task_id = 1;
    node[1].st_window_id = 1;
    node[1].st_switch_node_num = get_num_from_SDR(); /* get switch num */

    rc = swtbl_load_table(ST_VERSION,uid,pid,job_key,hostname,num_tasks,
     "User1_job",node);
    fprintf(stdout,"swtbl_load_table returned %d\n",rc);
  }
```

## swtbl_query_adapter Subroutine

### Purpose

**swtbl_query_adapter** – Returns the status of the adapter specified on the node from which it is invoked.

### Library

Job Switch Resource Table Services library (libswitchtbl.a)

### Syntax

```
#include <st_client.h>
int swtbl_query_adapter(int version,char *adapter,\
enum ST_ADAPTER_STATUS *status)
```

### Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. Should be **ST_VERSION** defined in **st_client.h**. |
| *adapter* | Specifies the adapter to obtain status for. Only **css0** is supported. |
| *status* | Specifies the address of the **ST_ADAPTER_STATUS** enum to contain the status. |

### Description

Use this subroutine to obtain the current status of the adapter specified on the node from which it is invoked. The **ST_ADAPTER_STATUS** variable will contain **ADAPTER_READY** or **ADAPTER_NOTREADY**.  **ADAPTER_READY** indicates that the node has connectivity over the switch for sending and receiving data. **ADAPTER_NOTREADY** indicates that the node is not connected to other nodes over the switch.

The effective user id of the calling process must be the root user id.

Any information or error messages are recorded in the **/var/adm/SPlogs/st/st_log** file.

### Return Values

Upon successful completion, the **swtbl_query_adapter** subroutine returns a value of **ST_SUCCESS**. Otherwise, it returns an integer value defined by the **ST_RETURN_CODE** enumerator found in **st_client.h**.

### Error Values

| | |
|---|---|
| **ST_INVALID_PARAM** | Indicates that a specified parameter was not valid. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_NO_SWITCH** | The open call of the specified adapter was unsuccessful. |
| **ST_NOT_AUTHOR** | Indicates that the caller did not have an effective user id of root. |

| | |
|---|---|
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |

## Related Information

Commands: **st_clean_table**, **st_status**

Subroutines: **swtbl_clean_table**, **swtbl_load_job**, **swtbl_load_table**, **swtbl_status**, **swtbl_status_node**, **swtbl_unload_job**

Files: **/usr/lpp/ssp/include/st_client.h**, **/var/adm/SPlogs/st/st_log**

## Examples

Code fragment to query **css0** on the current node:

```
#include <st_client.h>
  main() {
    int rc;
    enum ST_ADAPTER_STATUS status;

    rc = swtbl_query_adapter(ST_VERSION,"css0",&status);
    fprintf(stdout,"swtbl_query_adapter returned %d\n",rc);
    fprintf(stdout,"status = %d\n",status);
  }
```

## swtbl_status Subroutine

### Purpose

**swtbl_status** – Returns the status of all job switch resource table windows upon a node.

### Library

Job Switch Resource Table Services library (libswitchtbl.a)

### Syntax

```
#include <st_client.h>
int swtbl_status(int version,int num_nodes,ST_STATUS *status_info);
```

### Parameters

*version*
: Specifies the version of header file used. Should be **ST_VERSION** defined in **st_client.h**.

*num_nodes*
: Specifies the number of nodes on which status is reported.

*status_info*
: Specifies the pointer to the array of **ST_STATUS** structures that was allocated by the caller. A **status_info** structure must exist for every **num_nodes** defined. **ST_STATUS** members include:

*char st_node_name*
: Indicates the name or dotted decimal IP string of the node for which this data pertains. It works in combination with **st_addr**, if **st_node_name** is NULL, then **st_addr** must contain the struct **in_addr** address of the node. The **st_node_name** is checked first and if non-NULL, the **st_addr** is ignored. The maximum length is 256 characters.

*struct in_addr st_addr*
: Indicates the network address of the node for which this data pertains. Works in combination with **st_node_name**, the **st_node_name** must be NULL to use this.

### Description

Use this subroutine to return the status of all job switch resource table windows on the nodes specified within each **ST_STATUS** structure. This subroutine interfaces with the **switchtbld** daemon to call the local **swtbl_status_node** API on each node. Each node returns a linked list of **ST_STATUS** structures representing the windows defined upon that node. The caller is responsible for freeing the memory allocated for the linked list. A **NULL** *next pointer indicates the end.

If a window is loaded, the corresponding **ST_STATUS** structure will contain information about who made the load request, which user was designated to use the table and when the request was made. If a window is not loaded or an error occurred, the **st_return_code** contains the corresponding error value.

The **swtbl_status** API gives no information about whether the switch table is in use or not.

Any information or error messages are recorded in the **/var/adm/SPlogs/st/st_log** file on the calling node and for all of the nodes within the request.

## Return Values

Upon successful completion, the **swtbl_status** subroutine returns a value of **ST_SUCCESS**. Otherwise, it returns an integer value defined by the **ST_RETURN_CODE** enumerator found in **st_client.h**.

The **ST_STATUS** member **st_return_code** is updated to reflect any errors that may have occurred on the individual nodes. This return code corresponds to the return code of the **swtbl_status_node** API called on that node.

The following data members are returned when the switch table is loaded:

| | |
|---|---|
| **st_user_name** | The name corresponding to the uid given during the load request. |
| **st_node_name** | Node for which this status is reported. |
| **st_req_node** | Node from which the load request was issued. |
| **st_description** | String given during load request describing the job using the switch table. |
| **st_time_loaded** | Timestamp of when the load request was processed. |
| **st_client_pid** | Pid of the process who made the load request. |
| **st_uid** | Uid given during the load request for the user who will be using the switch table. |
| **st_window_id** | Window for which the data is being reported. |

## Error Values

| | |
|---|---|
| **ST_CANT_CONNECT** | Indicates that the connect request was unsuccessful. |
| **ST_INVALID_ADDR** | Indicates that the **st_addr** argument cannot be converted by **inet_ntoa**. |
| **ST_INVALID_PARAM** | Indicates that a specified parameter was not valid. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_SWITCH_NOT_LOADED** | Indicates that the switch table is not currently loaded. |
| **ST_LOADED_BYOTHER** | Indicates that the switch table is currently loaded but was not loaded via the Job Switch Resource Table Services. |
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_NO_SWITCH** | The open call of **/dev/css0** was unsuccessful. |

## Related Information

Commands: **st_clean_table**, **st_status**

Subroutines: **swtbl_clean_table**, **swtbl_load_job**, **swtbl_load_table**, **swtbl_status_node**, **swtbl_unload_job**, **swtbl_unload_table**

Files: **/usr/lpp/ssp/include/st_client.h**, **/var/adm/SPlogs/st/st_log**

## Examples

Code fragment to get the status for three nodes:

```
#include <st_client.h>

  main() {
    int i,rc;
    int num_nodes = 3;
    struct ST_STATUS status[num_nodes];

    strcpy(status_info[0].st_node_name,"k10n09");
    strcpy(status_info[1].st_node_name,"129.40.161.74");
    strcpy(status_info[2].st_node_name,"k10n11");

    rc = swtbl_status(ST_VERSION,num_nodes,status_info);
    fprintf(stdout,"swtbl_status returned %d\n",rc);
    print_header();
    for (i=0; i<num_tasks; i++) {
      /* Print out each member of the struct */
      print_status(status_info[i]);
    }
  }
```

## swtbl_status_node Subroutine

### Purpose

**swtbl_status_node** – Returns the status of all job switch resource table windows on the node from which it is invoked.

### Library

Job Switch Resource Table Services library (libswitchtbl.a)

### Syntax

```
#include <st_client.h>
swtbl_status_node(int version, ST_STATUS *status_info);
```

### Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. Should be **ST_VERSION** defined in **st_client.h**. |
| *status_info* | Specifies the pointer to the **ST_STATUS** structure that was allocated by the caller. |

### Description

Use this subroutine to return the status of all job switch resource table windows on the node from which it is invoked. The caller must supply the first **ST_STATUS** structure but the API will allocate any remaining structures depending on the number of windows defined for that node. These structures will be linked to the first **ST_STATUS** by the next pointer. The caller is responsible for freeing all allocated memory. If a window is loaded, the corresponding **ST_STATUS** structure will contain information about who made the load request, which user was designated to use the table and when the request was made. If a window is not loaded or an error occurred, the **st_return_code** contains the corresponding error value.

The **swtbl_status_node** API gives no information about whether the switch table is in use or not.

The effective user id of the calling process must be the root user id.

Any information or error messages are recorded in the **/var/adm/SPlogs/st/st_log** file.

### Return Values

Upon successful completion, the **swtbl_status_node** subroutine returns a value of **ST_SUCCESS**. Otherwise, it returns an integer value defined by the **ST_RETURN_CODE** enumerator found in **st_client.h**.

The following data members are returned when the switch table is loaded:

| | |
|---|---|
| **st_user_name** | The name corresponding to the uid given during the load request. |
| **st_node_name** | Node for which this status is reported. |
| **st_req_node** | Node from which the load request was issued. |

| | |
|---|---|
| **st_description** | String given during load request describing the job using the switch table. |
| **st_time_loaded** | Timestamp of when the load request was processed. |
| **st_client_pid** | Pid of the process who made the load request. |
| **st_uid** | Uid given during the load request for the user who will be using the switch table. |
| **st_window_id** | Window for which the data is being reported. |

## Error Values

| | |
|---|---|
| **ST_NOT_AUTHOR** | Indicates that the caller did not have an effective user id of root. |
| **ST_INVALID_PARAM** | Indicates that a specified parameter was not valid. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_NO_SWITCH** | The open call of **/dev/css0** was unsuccessful. |
| **ST_SWITCH_NOT_LOADED** | Indicates that the switch table is not currently loaded. |
| **ST_LOADED_BYOTHER** | Indicates that the switch table is currently loaded but was not loaded via the Job Switch Resource Table Services. |
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |

## Related Information

Commands: **st_clean_table**, **st_status**

Subroutines: **swtbl_clean_table**, **swtbl_load_job**, **swtbl_load_table**, **swtbl_status**, **swtbl_unload_job**, **swtbl_unload_table**

Files: **/usr/lpp/ssp/include/st_client.h**, **/var/adm/SPlogs/st/st_log**

## Examples

Code fragment to get the status for this node:

```
#include <st_client.h>
  main() {
    int rc;
    struct ST_STATUS status[1];

    rc = swtbl_status_node(ST_VERSION,status_info);
    fprintf(stdout,"swtbl_status_node returned %d\n",rc);
    print_header();
    print_status(status_info);  /* Print out each member of the struct */
  }
```

## swtbl_unload_job Subroutine

### Purpose

**swtbl_unload_job** – Unloads the job switch resource tables on the indicated nodes for a single job.

### Library

Job Switch Resource Table Services library (libswitchtbl.a)

### Syntax

```
#include <st_client.h>
int swtbl_unload_job(int version,uid_t uid,int num_tasks,\
ST_NODE_INFO *nodeinfo);
```

### Parameters

*version*
: Specifies the version of header file used. Should be **ST_VERSION** defined in **st_client.h**

*uid*
: Specifies the real user ID of the user for whom the unload is done. This uid must match the uid provided during the **swtbl_load_job** API.

*num_tasks*
: Specifies the number of tasks that accessed the job switch resource table during execution. Represents the size of the nodeinfo structure list.

*nodeinfo*
: Specifies the pointer to the **ST_NODE_INFO** structure that was allocated and defined by the caller. A nodeinfo structure must exist for every **num_tasks** defined. ST_NODE_INFO members include:

    *char st_node_name*
    Indicates the name or dotted decimal IP string of the node for which this data pertains. It works in combination with **st_addr**, if **st_node_name** is NULL, then **st_addr** must contain the **struct in_addr** address of the node. The **st_node_name** is checked first and if non-NULL, the **st_addr** is ignored. The maximum length is 256 characters.

    *struct in_addr st_addr*
    Indicates the network address of the node for which this data pertains. It works in combination with **st_node_name**, the **st_node_name** must be NULL to use this.

    *int st_window_id*
    Specifies the window id to be unloaded. The window id must be between 0 and 3 inclusive. If -1 is specified, then the default window will be used.

## Description

Use this subroutine to unload all of the job switch resource tables needed for a single parallel job. This switch table is used by parallel jobs running user space over the switch.

An **ST_NODE_INFO** structure needs to be defined for every node that needs to have the switch table unloaded. The **swtbl_unload_job** subroutine interfaces with the **switchtbld** daemon over the network to call the local **swtbl_unload_table** API which unloads every window within the table on that node.

The effective user id of the calling process must be the root user id.

The default window is defined within the **st_client.h** header file.

Any information or error messages are recorded in the **/var/adm/SPlogs/st/st_log** file on the calling node and for all of the nodes within the request.

## Return Values

Upon successful completion, the **swtbl_unload_job** subroutine returns a value of **ST_SUCCESS**. Otherwise, it returns an integer value defined by the **ST_RETURN_CODE** enumerator found in **st_client.h**. The **ST_NODE_INFO** member **st_return_code** is updated to reflect any errors that may have occurred on the individual nodes. This return code corresponds to the return code of the **swtbl_unload_table** API called on that node.

## Error Values

| | |
|---|---|
| **ST_CANT_CONNECT** | Indicates that the connect request was unsuccessful. |
| **ST_INVALID_ADDR** | Indicates that the **st_addr** argument cannot be converted by **inet_ntoa**. |
| **ST_INVALID_PARAM** | Indicates that a specified parameter was not valid. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_NOT_AUTHOR** | Indicates that the caller did not have effective user id of root. |
| **ST_NOT_UNLOADED** | Indicates the unload request was not issued due to another error. |
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |

## Related Information

Commands: **st_clean_table**, **st_status**

Subroutines: **swtbl_clean_table**, **swtbl_load_job**, **swtbl_load_table**, **swtbl_status**, **swtbl_status_node**, **swtbl_unload_table**

Files: **/usr/lpp/ssp/include/st_client.h**, **/var/adm/SPlogs/st/st_log**

## Examples

Code fragment to unload the job switch resource table windows for a job that was previously loaded for two nodes:

```
#include <st_client.h>
  main() {
    int rc;
    int num_tasks = 2;
    struct ST_NODE_INFO node[num_tasks];
    uid_t uid;

    uid = set_user_id(); /* User defined routine to get uid */

    strcpy(node[0].st_node_name,"Node1");
    node[0].st_window_id = 1;

    strcpy(node[1].st_node_name,"129.40.161.74");
    node[1].st_window_id = 2;

    rc = swtbl_unload_job(ST_VERSION,uid,num_tasks,node);
    fprintf(stdout,"swtbl_unload_job returned %d\n",rc);
  }
```

## swtbl_unload_table Subroutine

### Purpose

**swtbl_unload_table** – Unloads the job switch resource table window on the node from which it is invoked.

### Library

Job Switch Resource Table Services library (libswitchtbl.a)

### Syntax

```
#include <st_client.h>
int swtbl_unload_table(int version, char *adapter, \
uid_t uid, int window_id);
```

### Parameters

| | |
|---|---|
| *version* | Specifies the version of header file used. Should be **ST_VERSION** defined in **st_client.h** |
| *adapter* | Specifies the adapter to be opened upon which the window resides. Only **css0** is supported. |
| *uid* | Specifies the real user ID of the user for whom the unload is done. This uid must match the uid provided during the **swtbl_load_table** API. |
| *int st_window_id* | Specifies the window id to be unloaded. The window id must be between 0 and 3 inclusive. If -1 is specified, then the default window will be used. |

### Description

Use this subroutine to unload the job switch resource table window on the node from which it is invoked. This switch table is used by parallel jobs running user space over the switch. This subroutine checks that the uid provided matches the one that is stored during the **swtbl_load_table** call. Each window within the job switch resource table must be unloaded individually.

The effective user id of the calling process must be the root user id.

The default window is defined within the **st_client.h** header file.

Any information or error messages are recorded in the **/var/adm/SPlogs/st/st_log** file.

### Return Values

Upon successful completion, the **swtbl_unload_table** subroutine returns a value of **ST_SUCCESS**. Otherwise, it returns an integer value defined by the **ST_RETURN_CODE** enumerator found in **st_client.h**.

## Error Values

| | |
|---|---|
| **ST_INVALID_PARAM** | Indicates that a specified parameter was not valid. The **/var/adm/SPlogs/st/st_log** file will have further details. |
| **ST_NO_SWITCH** | The open call of the specified adapter was unsuccessful. |
| **ST_NOT_AUTHOR** | Indicates that the caller did not have an effective user id of root or the input uid does not match the uid of the corresponding **st_datafile**. |
| **ST_SWITCH_IN_USE** | Indicates that the currently loaded switch table window is in use by a process. |
| **ST_SWITCH_NOT_LOADED** | Indicates that the switch table window is not currently loaded. |
| **ST_SYSTEM_ERROR** | Indicates a system error occurred. The **/var/adm/SPlogs/st/st_log** file will have further details. |

## Related Information

Commands: **st_clean_table**, **st_status**

Subroutines: **swtbl_clean_table**, **swtbl_load_job**, **swtbl_load_table**, **swtbl_status**, **swtbl_status_node**, **swtbl_unload_job**

Files: **/usr/lpp/ssp/include/st_client.h**, **/var/adm/SPlogs/st/st_log**

## Examples

Code fragment to unload window 1 on the current node:

```
#include <st_client.h>
  main() {
    int i,rc;
    uid_t uid;
    int window = 1;

    uid = set_user_id(); /* User defined routine to get uid */

    rc = swtbl_unload_table(ST_VERSION,"css0",uid,window);
    fprintf(stdout,"swtbl_unload_table returned %d\n",rc);
  }
```

# Part 3. Appendixes

# Appendix A. Perspectives Colors and Fonts

## Perspectives Colors with Red, Green, and Blue (RGB) Triplets

The following list contains valid color names that can be supplied as optional arguments to the –**backgroundColor** and –**foregroundColor** flags. Colors may vary depending on the type of display you are using.

| Color | Red | Green | Blue |
|---|---|---|---|
| aquamarine | 127 | 255 | 212 |
| azure | 240 | 255 | 255 |
| beige | 245 | 245 | 220 |
| bisque | 255 | 228 | 196 |
| black | 0 | 0 | 0 |
| blue | 0 | 0 | 255 |
| brown | 165 | 42 | 42 |
| burlywood | 222 | 184 | 135 |
| chartreuse | 127 | 255 | 0 |
| chocolate | 210 | 105 | 30 |
| coral | 255 | 127 | 80 |
| cornsilk | 255 | 248 | 220 |
| cyan | 0 | 255 | 255 |
| firebrick | 178 | 34 | 34 |
| gold | 255 | 215 | 0 |
| goldenrod | 218 | 165 | 32 |
| gray | 190 | 190 | 190 |
| green | 0 | 255 | 0 |
| honeydew | 240 | 255 | 240 |
| ivory | 255 | 255 | 240 |
| khaki | 240 | 230 | 140 |
| lavender | 230 | 230 | 250 |
| linen | 250 | 240 | 230 |
| magenta | 255 | 0 | 255 |
| maroon | 176 | 48 | 96 |
| moccasin | 255 | 228 | 181 |
| oldlace | 253 | 245 | 230 |
| orange | 255 | 165 | 0 |
| orchid | 218 | 112 | 214 |
| peru | 205 | 133 | 63 |
| pink | 255 | 192 | 203 |
| plum | 221 | 160 | 221 |
| purple | 160 | 32 | 240 |
| red | 255 | 0 | 0 |
| salmon | 250 | 128 | 114 |
| seashell | 255 | 245 | 238 |
| sienna | 160 | 82 | 45 |
| snow | 255 | 250 | 250 |
| tan | 210 | 180 | 140 |
| thistle | 216 | 191 | 216 |
| tomato | 255 | 99 | 71 |
| turquoise | 64 | 224 | 208 |
| violet | 238 | 130 | 238 |
| wheat | 245 | 222 | 179 |
| white | 255 | 255 | 255 |
| yellow | 255 | 255 | 0 |

## Perspectives Fonts

**Note:** Fonts will vary depending on the type of Xmachine or Xstation you are using.

The following list contains font names that can be supplied as optional arguments to the –**fontFamily** flag:

application
block
charter
clean
courier
ergonomic
fixed
helvetica
lucida
lucida bright
lucida typewriter
new century schoolbook
roman
sans serif
serif
special
terminal
times
times new roman
type
typewriter
utopia

# Glossary of Terms and Abbreviations

This glossary includes terms and definitions from:

- The *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.
- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *ANSI/EIA Standard - 440A: Fiber Optic Terminology* copyright 1989 by the Electronics Industries Association (EIA). Copies can be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue N.W., Washington, D.C. 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

The following cross-references are used in this glossary:

**Contrast with.** This refers to a term that has an opposed or substantively different meaning.
**See.** This refers the reader to multiple-word terms in which this term appears.
**See also.** This refers the reader to terms that have a related, but not synonymous, meaning.
**Synonym for.** This indicates that the term has the same meaning as a preferred term, which is defined in the glossary.

This section contains some of the terms that are commonly used in the SP publications.

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard *Vocabulary for Information Processing* (Copyright 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of the American National Standards Committee X3. ANSI definitions are preceded by an asterisk (*).

Other definitions in this glossary are taken from *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems* (SC20-1699) and *IBM DATABASE 2 Application Programming Guide for TSO Users* (SC26-4081).

## A

**adapter**. An adapter is a mechanism for attaching parts. For example, an adapter could be a part that electrically or physically connects a device to a computer or to another device. In the SP system, network connectivity is supplied by various adapters, some optional, that can provide connection to I/O devices, networks of workstations, and mainframe networks. Ethernet, FDDI, token-ring, HiPPI, SCSI, FCS, and ATM are examples of adapters that can be used as part of an SP system.

**address**. A character or group of characters that identifies a register, a device, a particular part of storage, or some other data source or destination.

**AFS**. A distributed file system that provides authentication services as part of its file system creation.

**AIX**. Abbreviation for Advanced Interactive Executive, IBM's licensed version of the UNIX operating system. AIX is particularly suited to support technical computing applications, including high function graphics and floating point computations.

**Amd**. Berkeley Software Distribution automount daemon.

**API**. Application Programming Interface. A set of programming functions and routines that provide access between the Application layer of the OSI seven-layer model and applications that want to use the network. It is a software interface.

**application**. The use to which a data processing system is put; for example, a payroll application, an airline reservation application.

**application data**. The data that is produced using an application program.

**ARP**. Address Resolution Protocol.

**ATM**. Asynchronous Transfer Mode. (See *TURBOWAYS 100 ATM Adapter*.)

**Authentication**. The process of validating the identity of a user or server.

**Authorization**. The process of obtaining permission to perform specific actions.

## B

**batch processing**. * (1) The processing of data or the accomplishment of jobs accumulated in advance in such a manner that each accumulation thus formed is processed or accomplished in the same run. * (2) The processing of data accumulating over a period of time. * (3) Loosely, the execution of computer programs serially. (4) Computer programs executed in the background.

**BMCA**. Block Multiplexer Channel Adapter. The block multiplexer channel connection allows the RS/6000 to communicate directly with a host System/370 or System/390; the host operating system views the system unit as a control unit.

**BOS**. The AIX Base Operating System.

## C

**call home function**. The ability of a system to call the IBM support center and open a PMR to have a repair scheduled.

**CDE**. Common Desktop Environment. A graphical user interface for UNIX.

**charge feature**. An optional feature for either software or hardware for which there is a charge.

**CLI**. Command Line Interface.

**client**. * (1) A function that requests services from a server and makes them available to the user. * (2) A term used in an environment to identify a machine that uses the resources of the network.

**Client Input/Output Sockets (CLIO/S)**. A software package that enables high-speed data and tape access between SP systems, AIX systems, and ES/9000 mainframes.

**CLIO/S**. Client Input/Output Sockets.

**CMI**. Centralized Management Interface provides a series of SMIT menus and dialogues used for defining and querying the SP system configuration.

**connectionless**. A communication process that takes place without first establishing a connection.

**connectionless network**. A network in which the sending logical node must have the address of the receiving logical node before information interchange can begin. The packet is routed through nodes in the network based on the destination address in the packet. The originating source does not receive an acknowledgment that the packet was received at the destination.

**control workstation**. A single point of control allowing the administrator or operator to monitor and manage the SP system using the IBM AIX Parallel System Support Programs.

**css**. Communication subsystem.

## D

**daemon**. A process, not associated with a particular user, that performs system-wide functions such as administration and control of networks, execution of time-dependent activities, line printer spooling and so forth.

**DASD**. Direct Access Storage Device. Storage for input/output data.

**DCE**. Distributed Computing Environment.

**DFS**. distributed file system. A subset of the IBM Distributed Computing Environment.

**DNS**. Domain Name Service. A hierarchical name service which maps high level machine names to IP addresses.

## E

**Error Notification Object**. An object in the SDR that is matched with an error log entry. When an error log entry occurs that matches the Notification Object, a user-specified action is taken.

**ESCON**. Enterprise Systems Connection. The ESCON channel connection allows the RS/6000 to communicate directly with a host System/390; the host operating system views the system unit as a control unit.

**Ethernet**. (1) Ethernet is the standard hardware for TCP/IP local area networks in the UNIX marketplace. It is a 10-megabit per second baseband type LAN that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by collision detection (CSMA/CD). (2) A passive coaxial cable whose interconnections contain devices or components, or both, that are all active. It uses CSMA/CD technology to provide a best-effort delivery system.

**Ethernet network**. A baseband LAN with a bus topology in which messages are broadcast on a coaxial cabling using the carrier sense multiple access/collision detection (CSMA/CD) transmission method.

**event**. In Event Management, the notification that an expression evaluated to true. This evaluation occurs each time an instance of a resource variable is observed.

**expect**. Programmed dialogue with interactive programs.

**expression**. In Event Management, the relational expression between a resource variable and other elements (such as constants or the previous value of an instance of the variable) that, when true, generates an event. An example of an expression is `X < 10` where `X` represents the resource variable `IBM.PSSP.aixos.PagSp.%totalfree` (the percentage of total free paging space). When the expression is true, that is, when the total free paging space is observed to be less than 10%, the Event Management subsystem generates an event to notify the appropriate application.

## F

**failover**. Also called fallover, the sequence of events when a primary or server machine fails and a secondary or backup machine assumes the primary workload. This is a disruptive failure with a short recovery time.

**fall back**. Also called fallback, the sequence of events when a primary or server machine takes back control of its workload from a secondary or backup machine.

**FDDI**. Fiber Distributed Data Interface.

**Fiber Distributed Data Interface (FDDI)**. An American National Standards Institute (ANSI) standard for 100-megabit-per-second LAN using optical fiber cables. An FDDI local area network (LAN) can be up to 100 km (62 miles) and can include up to 500 system units. There can be up to 2 km (1.24 miles) between system units and concentrators.

**file**. * A set of related records treated as a unit, for example, in stock control, a file could consist of a set of invoices.

**file name**. A CMS file identifier in the form of 'filename filetype filemode' (like: TEXT DATA A).

**file server**. A centrally located computer that acts as a storehouse of data and applications for numerous users of a local area network.

**File Transfer Protocol (FTP)**. The Internet protocol (and program) used to transfer files between hosts. It is an application layer protocol in TCP/IP that uses TELNET and TCP protocols to transfer bulk-data files between machines or hosts.

**foreign host**. Any host on the network other than the local host.

**FTP**. File transfer protocol.

## G

**gateway**. An intelligent electronic device interconnecting dissimilar networks and providing protocol conversion for network compatibility. A gateway provides transparent access to dissimilar networks for nodes on either network. It operates at the session presentation and application layers.

## H

**HACMP**. High Availability Cluster Multi-Processing for AIX.

**HACWS**. High Availability Control Workstation function, based on HACMP, provides for a backup control workstation for the SP system.

| **HAL**. Hardware Abstraction Layer, a communication
| device interface that provides communication channels
| for processes.

**Hashed Shared Disk (HSD)**. The data striping device for the IBM Virtual Shared Disk. The device driver lets application programs stripe data across physical disks in multiple IBM Virtual Shared Disks, thus reducing I/O bottlenecks.

**help key**. In the SP graphical interface, the key that gives you access to the SP graphical interface help facility.

**High Availability Cluster Multi-Processing**. An IBM facility to cluster nodes or components to provide high availability by eliminating single points of failure.

**HiPPI**. High Performance Parallel Interface. RS/6000 units can attach to a HiPPI network as defined by the ANSI specifications. The HiPPI channel supports burst rates of 100 Mbps over dual simplex cables; connections can be up to 25 km in length as defined by the standard and can be extended using third-party HiPPI switches and fiber optic extenders.

**home directory**. The directory associated with an individual user.

**host**. A computer connected to a network, and providing an access method to that network. A host provides end-user services.

## I

**instance vector**. Obsolete term for resource identifier.

**Intermediate Switch Board**. Switches mounted in the Sp Switch expansion frame.

**Internet**. A specific inter-network consisting of large national backbone networks such as APARANET, MILNET, and NSFnet, and a myriad of regional and campus networks all over the world. The network uses the TCP/IP protocol suite.

**Internet Protocol (IP)**. (1) A protocol that routes data through a network or interconnected networks. IP acts as an interface between the higher logical layers and the physical network. This protocol, however, does not provide error recovery, flow control, or guarantee the reliability of the physical network. IP is a connectionless protocol. (2) A protocol used to route data from its source to it destination in an Internet environment.

**IP address**. A 32-bit address assigned to devices or hosts in an IP internet that maps to a physical address. The IP address is composed of a network and host portion.

**ISB**. Intermediate Switch Board.

## K

**Kerberos**. A service for authenticating users in a network environment.

**kernel**. The core portion of the UNIX operating system which controls the resources of the CPU and allocates them to the users. The kernel is memory-resident, is said to run in “kernel mode” and is protected from user tampering by the hardware.

## L

**LAN**. (1) Acronym for Local Area Network, a data network located on the user's premises in which serial transmission is used for direct data communication among data stations. (2) Physical network technology that transfers data a high speed over short distances. (3) A network in which a set of devices is connected to another for communication and that can be connected to a larger network.

**local host**. The computer to which a user's terminal is directly connected.

**log database**. A persistent storage location for the logged information.

**log event**. The recording of an event.

**log event type**. A particular kind of log event that has a hierarchy associated with it.

**logging**. The writing of information to persistent storage for subsequent analysis by humans or programs.

## M

**mask**. To use a pattern of characters to control retention or elimination of portions of another pattern of characters.

**menu**. A display of a list of available functions for selection by the user.

**Motif**. The graphical user interface for OSF, incorporating the X Window System. Also called OSF/Motif.

**MTBF**. Mean time between failure. This is a measure of reliability.

**MTTR**. Mean time to repair. This is a measure of serviceability.

## N

**naive application**. An application with no knowledge of a server that fails over to another server. Client to server retry methods are used to reconnect.

**network**. An interconnected group of nodes, lines, and terminals. A network provides the ability to transmit data to and receive data from other systems and users.

**NFS**. Network File System. NFS allows different systems (UNIX or non-UNIX), different architectures, or vendors connected to the same network, to access remote files in a LAN environment as though they were local files.

**NIM**. Network Installation Management is provided with AIX to install AIX on the nodes.

**NIM client**. An AIX system installed and managed by a NIM master. NIM supports three types of clients:

- Standalone
- Diskless
- Dataless

**NIM master**. An AIX system that can install one or more NIM clients. An AIX system must be defined as a NIM master before defining any NIM clients on that system. A NIM master managers the configuration database containing the information for the NIM clients.

**NIM object**. A representation of information about the NIM environment. NIM stores this information as objects in the NIM database. The types of objects are:

- Network
- Machine
- Resource

**NIS**. Network Information System.

**node**. In a network, the point where one or more functional units interconnect transmission lines. A computer location defined in a network. The SP system can house several different types of nodes for both serial and parallel processing. These node types can include thin nodes, wide nodes, 604 high nodes, as well as other types of nodes both internal and external to the SP frame.

**Node Switch Board**. Switches mounted on frames that contain nodes.

**NSB**. Node Switch Board.

**NTP**. Network Time Protocol.

## O

**ODM**. Object Data Manager. In AIX, a hierarchical object-oriented database for configuration data.

## P

**parallel environment**. A system environment where message passing or SP resource manager services are used by the application.

**Parallel Environment**. A licensed IBM program used for message passing applications on the SP or RS/6000 platforms.

**parallel processing**. A multiprocessor architecture which allows processes to be allocated to tightly coupled multiple processors in a cooperative processing environment, allowing concurrent execution of tasks.

**parameter**. * (1) A variable that is given a constant value for a specified application and that may denote the application. * (2) An item in a menu for which the operator specifies a value or for which the system provides a value when the menu is interpreted. * (3) A name in a procedure that is used to refer to an argument that is passed to the procedure. * (4) A particular piece of information that a system or application program needs to process a request.

**partition**. See system partition.

**Perl**. Practical Extraction and Report Language.

**perspective**. The primary window for each SP Perspectives application, so called because it provides a unique view of an SP system.

**pipe**. A UNIX utility allowing the output of one command to be the input of another. Represented by the | symbol. It is also referred to as filtering output.

**PMR**. Problem Management Report.

**POE**. Formerly Parallel Operating Environment, now Parallel Environment for AIX.

**port**. (1) An end point for communication between devices, generally referring to physical connection. (2) A 16-bit number identifying a particular TCP or UDP resource within a given TCP/IP node.

**predicate**. Obsolete term for expression.

**Primary node or machine**. (1) A device that runs a workload and has a standby device ready to assume the primary workload if that primary node fails or is taken out of service. (2) A node on the SP Switch that initializes, provides diagnosis and recovery services, and performs other operations to the switch network. (3) In IBM Virtual Shared Disk function, when physical disks are connected to two nodes (twin-tailed), one node is designated as the primary node for each disk and the other is designated the secondary, or backup, node. The primary node is the server node for IBM Virtual Shared Disks defined on the physical disks under normal conditions. The secondary node can become the server node for the disks if the primary node is unavailable (off-line or down).

**Problem Management Report**. The number in the IBM support mechanism that represents a service incident with a customer.

**process**. * (1) A unique, finite course of events defined by its purpose or by its effect, achieved under defined conditions. * (2) Any operation or combination of operations on data. * (3) A function being performed or waiting to be performed. * (4) A program in operation. For example, a daemon is a system process that is always running on the system.

**protocol**. A set of semantic and syntactic rules that defines the behavior of functional units in achieving communication.

## R

**RAID**. Redundant array of independent disks.

**rearm expression**. In Event Management, an expression used to generate an event that alternates with an original event expression in the following way: the event expression is used until it is true, then the rearm expression is used until it is true, then the event expression is used, and so on. The rearm expression is commonly the inverse of the event expression (for example, a resource variable is on or off). It can also be used with the event expression to define an upper and lower boundary for a condition of interest.

**rearm predicate**. Obsolete term for rearm expression.

**remote host**. *See foreign host*.

**resource**. In Event Management, an entity in the system that provides a set of services. Examples of resources include hardware entities such as processors, disk drives, memory, and adapters, and software entities such as database applications, processes, and file systems. Each resource in the system has one or more attributes that define the state of the resource.

**resource identifier**. In Event Management, a set of elements, where each element is a name/value pair of the form `name=value`, whose values uniquely identify the copy of the resource (and by extension, the copy of the resource variable) in the system.

**resource monitor**. A program that supplies information about resources in the system. It can be a command, a daemon, or part of an application or subsystem that manages any type of system resource.

**resource variable**. In Event Management, the representation of an attribute of a resource. An example of a resource variable is `IBM.AIX.PagSp.%totalfree`, which represents the percentage of total free paging space. `IBM.AIX.PagSp` specifies the resource name and `%totalfree` specifies the resource attribute.

**RISC**. Reduced Instruction Set Computing (RISC), the technology for today's high performance personal computers and workstations, was invented in 1975. Uses a small simplified set of frequently used instructions for rapid execution.

**rlogin (remote LOGIN)**. A service offered by Berkeley UNIX systems that allows authorized users of one machine to connect to other UNIX systems across a network and interact as if their terminals were connected directly. The rlogin software passes information about the user's environment (for example, terminal type) to the remote machine.

**RPC**. Acronym for Remote Procedure Call, a facility that a client uses to have a server execute a procedure call. This facility is composed of a library of procedures plus an XDR.

**RSH**. A variant of RLOGIN command that invokes a command interpreter on a remote UNIX machine and passes the command line arguments to the command interpreter, skipping the LOGIN step completely. See also *rlogin*.

## S

**SCSI**. Small Computer System Interface.

**Secondary node**. In IBM Virtual Shared Disk function, when physical disks are connected to two nodes (twin-tailed), one node is designated as the primary node for each disk and the other is designated as the secondary, or backup, node. The secondary node acts as the server node for the IBM Virtual Shared disks defined on the physical disks if the primary node is unavailable (off-line or down).

**server**. (1) A function that provides services for users. A machine may run client and server processes at the same time. (2) A machine that provides resources to the network. It provides a network service, such as disk storage and file transfer, or a program that uses such a service. (3) A device, program, or code module on a network dedicated to providing a specific service to a network. (4) On a LAN, a data station that provides facilities to other data stations. Examples are file server, print server, and mail server.

**shell**. The shell is the primary user interface for the UNIX operating system. It serves as command language interpreter, programming language, and allows foreground and background processing. There are three different implementations of the shell concept: Bourne, C and Korn.

**Small Computer System Interface (SCSI)**. An input and output bus that provides a standard interface for the attachment of various direct access storage devices (DASD) and tape drives to the RS/6000.

**Small Computer Systems Interface Adapter (SCSI Adapter)**. An adapter that supports the attachment of various direct-access storage devices (DASD) and tape drives to the RS/6000.

**SMIT**. The System Management Interface Toolkit is a set of menu driven utilities for AIX that provides functions such as transaction login, shell script creation, automatic updates of object database, and so forth.

**SNMP**. Simple Network Management Protocol. (1) An IP network management protocol that is used to monitor attached networks and routers. (2) A TCP/IP-based protocol for exchanging network management information and outlining the structure for communications among network devices.

**socket**. (1) An abstraction used by Berkeley UNIX that allows an application to access TCP/IP protocol functions. (2) An IP address and port number pairing. (3) In TCP/IP, the Internet address of the host computer on which the application runs, and the port number it uses. A TCP/IP application is identified by its socket.

**standby node or machine**. A device that waits for a failure of a primary node in order to assume the identity of the primary node. The standby machine then runs the primary's workload until the primary is back in service.

**subnet**. Shortened form of subnetwork.

**subnet mask**. A bit template that identifies to the TCP/IP protocol code the bits of the host address that are to be used for routing for specific subnetworks.

**subnetwork**. Any group of nodes that have a set of common characteristics, such as the same network ID.

**subsystem**. A software component that is not usually associated with a user command. It is usually a daemon process. A subsystem will perform work or provide services on behalf of a user request or operating system request.

**SUP**. Software Update Protocol.

| **switch capsule**. A group of SP frames consisting of a
| switched frame and its companion non-switched frames.

**Sysctl**. Secure System Command Execution Tool. An authenticated client/server system for running commands remotely and in parallel.

**syslog**. A BSD logging system used to collect and manage other subsystem's logging data.

**System Administrator**. The user who is responsible for setting up, modifying, and maintaining the SP system.

**system partition**. A group of nonoverlapping nodes on a switch chip boundary that act as a logical SP system.

## T

**tar**. Tape ARchive, is a standard UNIX data archive utility for storing data on tape media.

| **TaskGuides**. SP TaskGuides are a form of advanced
| online assistance designed to walk you through
| complex or infrequently performed tasks. Each
| TaskGuide does not simply list the required steps. It
| actually performs the steps for you, automating the
| steps to the highest degree possible and prompting you
| for input only when absolutely necessary. You might
| recognize them as *wizards*.

**Tcl**. Tool Command Language.

**TclX**. Tool Command Language Extended.

**TCP**. Acronym for Transmission Control Protocol, a stream communication protocol that includes error recovery and flow control.

**TCP/IP**. Acronym for Transmission Control Protocol/Internet Protocol, a suite of protocols designed to allow communication between networks regardless of the technologies implemented in each network. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It assumes that the underlying protocol is the Internet Protocol.

**Telnet**. Terminal Emulation Protocol, a TCP/IP application protocol that allows interactive access to foreign hosts.

**Tk**. Tcl-based Tool Kit for X Windows.

**TMPCP**. Tape Management Program Control Point.

**token-ring**. (1) Network technology that controls media access by passing a token (special packet or frame) between media-attached machines. (2) A network with a ring topology that passes tokens from one attaching device (node) to another. (3) The IBM Token-Ring LAN connection allows the RS/6000 system unit to participate in a LAN adhering to the IEEE 802.5 Token-Passing Ring standard or the ECMA standard 89 for Token-Ring, baseband LANs.

**transaction**. An exchange between the user and the system. Each activity the system performs for the user is considered a transaction.

**transceiver (transmitter-receiver)**. A physical device that connects a host interface to a local area network, such as Ethernet. Ethernet transceivers contain electronics that apply signals to the cable and sense collisions.

**transfer**. To send data from one place and to receive the data at another place. Synonymous with move.

**transmission**. * The sending of data from one place for reception elsewhere.

**TURBOWAYS 100 ATM Adapter**. An IBM high-performance, high-function intelligent adapter that provides dedicated 100 Mbps ATM (asynchronous transfer mode) connection for high-performance servers and workstations.

## U

**UDP**. User Datagram Protocol.

**UNIX operating system**. An operating system developed by Bell Laboratories that features multiprogramming in a multiuser environment. The UNIX operating system was originally developed for use on minicomputers, but has been adapted for mainframes and microcomputers. **Note:** The AIX operating system is IBM's implementation of the UNIX operating system.

**user**. Anyone who requires the services of a computing system.

**User Datagram Protocol (UDP)**. (1) In TCP/IP, a packet-level protocol built directly on the Internet Protocol layer. UDP is used for application-to-application programs between TCP/IP host systems. (2) A transport protocol in the Internet suite of protocols that provides unreliable, connectionless datagram service. (3) The Internet Protocol that enables an application programmer on one machine or process to send a datagram to an application program on another machine or process.

**user ID**. A nonnegative integer, contained in an object of type *uid_t*, that is used to uniquely identify a system user.

## V

**Virtual Shared Disk, IBM**. The function that allows application programs executing at different nodes of a system partition to access a raw logical volume as if it were local at each of the nodes. In actuality, the logical volume is local at only one of the nodes (the server node).

## W

**workstation**. * (1) A configuration of input/output equipment at which an operator works. * (2) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

## X

**X Window System**. A graphical user interface product.

# Bibliography

This bibliography helps you find product documentation related to the RS/6000 SP hardware and software products.

You can find most of the IBM product information for RS/6000 SP products on the World Wide Web. Formats for both viewing and downloading are available.

PSSP documentation is shipped with the PSSP product in a variety of formats and can be installed on your system. The man pages for public code that PSSP includes are also available online.

You can order hard copies of the product documentation from IBM. This bibliography lists the titles that are available and their order numbers.

Finally, this bibliography contains a list of non-IBM publications that discuss parallel computing and other topics related to the RS/6000 SP.

## Finding Documentation on the World Wide Web

Most of the RS/6000 SP hardware and software books are available from the IBM RS/6000 Web site at:

http://www.rs6000.ibm.com

You can view a book or download a Portable Document Format (PDF) version of it. At the time this manual was published, the Web address of the "RS/6000 SP Product Documentation Library" page was:

http://www.rs6000.ibm.com/resource/aix_resource/sp_books

However, the structure of the RS/6000 Web site can change over time.

## Accessing PSSP Documentation Online

On the same medium as the PSSP product code, IBM ships PSSP man pages, HTML files, and PDF files. In order to use these publications, you must first install the **ssp.docs** file set.

To view the PSSP HTML publications, you need access to an HTML document browser such as Netscape. The HTML files and an index that links to them are installed in the **/usr/lpp/ssp/html** directory. Once installed, you can also view the HTML files from the RS/6000 SP Resource Center.

If you have installed the SP Resource Center on your SP system, you can access it by entering the **/usr/lpp/ssp/bin/resource_center** command. If you have the SP Resource Center on CD-ROM, see the **readme.txt** file for information about how to run it.

To view the PSSP PDF publications, you need access to the Adobe Acrobat Reader 3.0.1. The Acrobat Reader is shipped with the AIX Version 4.3 Bonus Pack.

To successfully print a large PDF file (approximately 300 or more pages) from the Adobe Acrobat reader, you may need to select the “Download Fonts Once” button on the Print window.

## Manual Pages for Public Code

The following manual pages for public code are available in this product:

| | |
|---|---|
| **SUP** | /usr/lpp/ssp/man/man1/sup.1 |
| **NTP** | /usr/lpp/ssp/man/man8/xntpd.8 |
| | /usr/lpp/ssp/man/man8/xntpdc.8 |
| **Perl (Version 4.036)** | /usr/lpp/ssp/perl/man/perl.man |
| | /usr/lpp/ssp/perl/man/h2ph.man |
| | /usr/lpp/ssp/perl/man/s2p.man |
| | /usr/lpp/ssp/perl/man/a2p.man |
| **Perl (Version 5.003)** | Man pages are in the /usr/lpp/ssp/perl5/man/man1 directory |

Manual pages and other documentation for **Tcl**, **TclX**, **Tk**, and **expect** can be found in the compressed **tar** files located in the **/usr/lpp/ssp/public** directory.

## RS/6000 SP Planning Publications

This section lists the IBM product documentation for planning for the IBM RS/6000 SP hardware and software.

*IBM RS/6000 SP:*

- *Planning, Volume 1, Hardware and Physical Environment*, GA22-7280
- *Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281

## RS/6000 SP Hardware Publications

This section lists the IBM product documentation for the IBM RS/6000 SP hardware.

*IBM RS/6000 SP:*

- *Planning, Volume 1, Hardware and Physical Environment*, GA22-7280
- *Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281
- *Maintenance Information, Volume 1, Installation and Relocation*, GA22-7375
- *Maintenance Information, Volume 2, Maintenance Analysis Procedures*, GA22-7376
- *Maintenance Information, Volume 3, Locations and Service Procedures*, GA22-7377
- *Maintenance Information, Volume 4, Parts Catalog*, GA22-7378

## RS/6000 SP Switch Router Publications

The RS/6000 SP Switch Router is based on the Ascend GRF switched IP router product from Ascend Communications, Inc.. You can order the SP Switch Router as the IBM 9077.

The following publications are shipped with the SP Switch Router. You can also order these publications from IBM using the order numbers shown.

- *Ascend GRF Getting Started*, GA22-7368
- *Ascend GRF Configuration Guide*, GA22-7366
- *Ascend GRF Reference Guide*, GA22-7367
- *IBM SP Switch Router Adapter Guide*, GA22-7310.

## RS/6000 SP Software Publications

This section lists the IBM product documentation for software products related to the IBM RS/6000 SP. These products include:

- IBM Parallel System Support Programs for AIX (PSSP)
- IBM LoadLeveler for AIX (LoadLeveler)
- IBM Parallel Environment for AIX (Parallel Environment)
- IBM General Parallel File System for AIX (GPFS)
- IBM Engineering and Scientific Subroutine Library (ESSL) for AIX
- IBM Parallel ESSL for AIX
- IBM High Availability Cluster Multi-Processing for AIX (HACMP)
- IBM Client Input Output/Sockets (CLIO/S)
- IBM Network Tape Access and Control System for AIX (NetTAPE)

### PSSP Publications

*IBM RS/6000 SP:*

- *Planning, Volume 2, Control Workstation and Software Environment*, GA22-7281

*PSSP:*

- *Installation and Migration Guide*, GA22-7347
- *Administration Guide*, SA22-7348
- *Managing Shared Disks*, SA22-7349
- *Performance Monitoring Guide and Reference*, SA22-7353
- *Diagnosis Guide*, GA22-7350
- *Command and Technical Reference*, SA22-7351
- *Messages Reference*, GA22-7352

*RS/6000 Cluster Technology (RSCT):*

- *Event Management Programming Guide and Reference*, SA22-7354
- *Group Services Programming Guide and Reference*, SA22-7355

As an alternative to ordering the individual books, you can use SBOF-8587 to order the PSSP software library.

### LoadLeveler Publications

*LoadLeveler:*

- *Using and Administering*, SA22-7311
- *Diagnosis and Messages Guide*, GA22-7277

### GPFS Publications

*GPFS:*

- *Installation and Administration Guide*, SA22-7278

### Parallel Environment Publications

*Parallel Environment:*

- *Installation Guide*, GC28-1981

- *Hitchhiker's Guide*, GC23-3895
- *Operation and Use, Volume 1*, SC28-1979
- *Operation and Use, Volume 2*, SC28-1980
- *MPI Programming and Subroutine Reference*, GC23-3894
- *MPL Programming and Subroutine Reference*, GC23-3893
- *Messages*, GC28-1982

As an alternative to ordering the individual books, you can use SBOF-8588 to order the PE library.

**Parallel ESSL and ESSL Publications**

- *ESSL Products: General Information*, GC23-0529
- *Parallel ESSL: Guide and Reference*, SA22-7273
- *ESSL: Guide and Reference*, SA22-7272

**HACMP Publications**

*HACMP:*

- *Concepts and Facilities*, SC23-4276
- *Planning Guide*, SC23-4277
- *Installation Guide*, SC23-4278
- *Administration Guide*, SC23-4279
- *Troubleshooting Guide*, SC23-4280
- *Programming Locking Applications*, SC23-4281
- *Programming Client Applications*, SC23-4282
- *Master Index and Glossary*, SC23-4285
- *HANFS for AIX Installation and Administration Guide*, SC23-4283
- *Enhanced Scalability Installation and Administration Guide*, SC23-4284

**CLIO/S Publications**

*CLIO/S:*

- *General Information*, GC23-3879
- *User's Guide and Reference*, GC28-1676

**NetTAPE Publications**

*NetTAPE:*

- *General Information*, GC23-3990
- *User's Guide and Reference*, available from your IBM representative

## AIX and Related Product Publications

For the latest information on AIX and related products, including RS/6000 hardware products, see *AIX and Related Products Documentation Overview*, SC23-2456. You can order a hard copy of the book from IBM. You can also view it online from the “AIX Online | Publications and Books” page of the RS/6000 Web site at:

| http://www.rs6000.ibm.com/resource/aix_resource/Pubs

## Red Books

IBM's International Technical Support Organization (ITSO) has published a number of redbooks related to the RS/6000 SP. For a current list, see the ITSO Web site at: http://www.redbooks.ibm.com.

## Non-IBM Publications

Here are some non-IBM publications that you may find helpful.

- Almasi, G., Gottlieb, A., *Highly Parallel Computing*, Benjamin-Cummings Publishing Company, Inc., 1989.
- Foster, I., *Designing and Building Parallel Programs*, Addison-Wesley, 1995.
- Gropp, W., Lusk, E., Skjellum, A., *Using MPI*, The MIT Press, 1994.
- Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard, Version 1.1*, University of Tennessee, Knoxville, Tennessee, June 6, 1995.
- Message Passing Interface Forum, *MPI-2: Extensions to the Message-Passing Interface, Version 2.0*, University of Tennessee, Knoxville, Tennessee, July 18, 1997.
- Ousterhout, John K., *Tcl and the Tk Toolkit*, Addison-Wesley, Reading, MA, 1994, ISBN 0-201-63337-X.
- Pfister, Gregory, F., *In Search of Clusters*, Prentice Hall, 1998.

# Index

# Communicating Your Comments to IBM

Parallel System Support Programs for AIX
Command and Technical
Reference, Volume 2
Version 3 Release 1.1

Publication No. SA22-7351-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a reader's comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - FAX: (International Access Code)+1+914+432-9405
- If you prefer to send comments electronically, use one of these network IDs:
  - IBM Mail Exchange: USIB6TC9 at IBMMAIL
  - Internet e-mail: mhvrcfs@us.ibm.com

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies

Optionally, if you include your telephone number, we will be able to respond to your comments by phone.

# Reader's Comments — We'd Like to Hear from You

**Parallel System Support Programs for AIX**
**Command and Technical**
**Reference, Volume 2**
**Version 3 Release 1.1**

**Publication No. SA22-7351-01**

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

**Note:** Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today's date: ____________________________________

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

| | | | |
|---|---|---|---|
| [ ] | As an introduction | [ ] | As a text (student) |
| [ ] | As a reference manual | [ ] | As a text (instructor) |
| [ ] | For another purpose (explain) | | |

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

Name

Address

Company or Organization

Phone No.

**Reader's Comments — We'd Like to Hear from You**
SA22-7351-01

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie NY 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

SA22-7351-01

Cut or Fold
Along Line

Program Number: 5765-D51

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SA22-7351-01