Solaris™ Security Toolkit 4.1 Man Page Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 819-0111-10
October 2004, Revision A

Submit comments about this document at: `http://www.sun.com/hwdocs/feedback`

DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

LA DOCUMENTATION EST FOURNIE "EN L’ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L’APTITUDE A UNE UTILISATION PARTICULIERE OU A L’ABSENCE DE CONTREFAÇON.

Contents

Preface

Both novice users and those familiar with the SunOS operating system can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question “What does it do?” In general, man pages comprise a reference manual. They are not intended to be a tutorial.

Overview

The following contains a brief description of each section in the man pages and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character-set tables.
- Section 6 contains available games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.

- Section 9 provides reference information needed to write device drivers in the kernel operating systems environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).
- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer may include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man`(1) for more information about man pages in general.

NAME

This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS

This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

`[ ]` Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.

… Ellipses. Several values may be provided for the previous argument, or the previous argument can be specified multiple times, for example "`filename...`".

| Separator. Only one of the arguments separated by this character can be specified at one time.

`{ }` Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

PROTOCOL

This section occurs only in subsection 3R to indicate the protocol description file.

DESCRIPTION

This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, functions and such, are described under USAGE.

IOCTL

This section appears on pages in Section 7 only. Only the device class which supplies appropriate parameters to the `ioctl`(2) system call is called `ioctl` and generates its own heading. `ioctl` calls for a specific device are listed alphabetically (on the man page for that specific device). `ioctl` calls are used for a particular class of devices all of which have an `io` ending, such as `mtio`(7I)

OPTIONS

This lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.

OPERANDS

This section lists the command operands and describes how they affect the actions of the command.

OUTPUT

This section describes the output – standard output, standard error, or output files – generated by the command.

RETURN VALUES

If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.

ERRORS

On failure, most functions place an error code in the global variable `errno` indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.

USAGE

This section lists special rules, features and commands that require in-depth explanations. The subsections listed below are used to explain built-in functionality:

Commands
Modifiers
Variables
Expressions
Input Grammar

EXAMPLES

This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command line entry and machine response is shown. Whenever an example is given, the prompt is shown as `example%` or if the user must be superuser, `example#`. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS and USAGE sections.

ENVIRONMENT VARIABLES

This section lists any environment variables that the command or function affects, followed by a brief description of the effect.

EXIT STATUS

This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion and values other than zero for various error conditions.

FILES

This section lists all filenames referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.

ATTRIBUTES

This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes`(5) for more information.

SEE ALSO

This section lists references to other man pages, in-house documentation and outside publications.

DIAGNOSTICS

This section lists diagnostic messages with a brief explanation of the condition causing the error.

WARNINGS

This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.

NOTES

This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.

BUGS

This section describes known bugs and wherever possible, suggests workarounds.

NAME

Intro - Solaris Security Toolkit Administration.

DESCRIPTION

This section describes the commands executed in the Solaris Security Toolkit (also known as JASS) environment.

Sun support for Solaris Security Toolkit software is available only for its use in the Solaris 8 and Solaris 9 Operating Systems. While the software can be used in the Solaris 2.5.1, Solaris 2.6 and Solaris 7 Operating Systems, Sun support is not available for its use in those operating systems.

The Solaris Security Toolkit software automatically detects which version of the Solaris Operating System software is installed, then runs tasks appropriate for that operating system version.

LIST OF COMMANDS

The following commands, functions and drivers are supported:

`Intro`	Solaris Security Toolkit Administration.
`audit_public_funcs`	Change Solaris Security Toolkit audit behavior.
`common_log_funcs`	Control all logging and reporting Solaris Security Toolkit functions
`common_misc_funcs`	Miscellaneous framework Solaris Security Toolkit functions.
`drivers_funcs`	Functions for the Solaris Security Toolkit drivers.
`jass-check-sum`	Identify file changes made since the last Security Toolkit backup.
`jass-execute`	Create Solaris Security Toolkit package stream file.
`make-jass-pkg`	Configure the Solaris Security Toolkit application.
`rm-client`	Remove JumpStart client.
security_drivers	Solaris Security Toolkit drivers.

NAME

add-client - install JumpStart client for the Solaris Security Toolkit (JASS)

SYNOPSIS

add-client `-c` *client-host-name* [`-i` *install-server*] [`-m` *client-mach-class*] [`-o` *solaris-os-instance*] [`-s` *sysidcfg-dir*]

add-client `-?| -h`

add-client `-v`

DESCRIPTION

`add-client` installs the JumpStart client and configuration information needed by the Solaris Security Toolkit (also known as JASS). It is executed from the Jumpstart server.

EXTENDED DESCRIPTION

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

The following options are supported.

`-c` *client-host-name* Specifies the name of the JumpStart client to be installed.

`-h |-?` Displays usage descriptions.

Note – Use alone. Any option specified in addition to `-h` or `-?` is ignored.

`-i` *install-server* Specifies the name of the JumpStart install server. If no value is given, a list of available options is provided. If the system has only one network interface then `add-client` uses it by default.

`-m` *client-mach-class* Specifies the machine class of the JumpStart client. This value must be in the same format as the output of the `uname -n` command.

`-o` *solaris-os-instance* Specifies the revision of the Solaris Operating System to be installed on the client. If no value is given, a list of available options is provided. If only one instance is available, `add-client` uses it by default.

-s *sysidcfg-dir* Specifies the pathname to an alternate directory in which a system identification and configuration (sysidcfg) file is stored. By default, the value is set to the directory, JASS_HOME/Sysidcfg/Solaris_VERSION/. If this option is used, this path name should be specified relative to the JASS_HOME/Sysidcfg directory. For example, Hosts/alpha where JASS_HOME/Sysidcfg/Hosts/alpha exists and contains a sysidcfg file.

-v Displays the version information for this program.

EXAMPLES

EXAMPLE 1 Add a Client to a System Using Defaults

```
sc0:#:> /opt/SUNWjass/bin/add-client -c eng1 -m sun4u
Selecting default operating system, Solaris_ver.
Selecting default system interface, IP_address.
cleaning up preexisting install client "eng1"
removing eng1 from bootparams
updating /etc/bootparams
sc0:#:>
```

where:

Solaris_ver	Default version of the Solaris Operating System.
IP_ address	Internet Protocol address written as four sets of numbers separated by periods; for example, 172.16.0.59.
eng1	The hostname of the Jumpstart client.

EXAMPLE 2 Add a Client to a System Using Full Options

```
sc0:#:> /opt/SUNWjass/bin/add-client -c eng1 -i jumpserve1 -m
sun4u -o Solaris_9_2003-12 -s $JASS_HOME/Sysidcfg/Hosts/alpha/
cleaning up preexisting install client "eng1"
removing eng1 from bootparams
updating /etc/bootparams
sc0:#:>
```

where:

eng1	The hostname of the Jumpstart client
jumpserve1	The hostname of the Jumpstart install server.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion
1	An error occurred

ATTRIBUTES

See **attributes**`(5)` for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Unstable

SEE ALSO

jass-check-sum`(1M)`

jass-execute`(1M)`

make-jass-pkg`(1M)`

rm-client`(7)`

NAME

audit_public_funcs - change audit behavior of the Solaris Security Toolkit (JASS)

SYNOPSIS

`audit_public_funcs`

DESCRIPTION

Framework functions provide flexibility for you to change the behavior of the Solaris Security Toolkit (also known as JASS) software without modifying source code.

Note – Two types of audit functions are in the software: private and public. The functions defined in the `audit_private.funcs` file are private and not for public use. Never use the private scripts defined in this file. Only use the public scripts defined in the `audit_public.funcs` file.

`audit_public_funcs` define audit functions used in audit scripts, which are located in `JASS_AUDIT_DIR`. Functions defined in this file are public and can be freely used in both standard and custom audit scripts. In many cases, the functions defined in this file are stubs that call functions defined in the `audit_private.funcs` file. These stubs were implemented to allow users to code their scripts to these public interfaces without needing to care if the underlying code will be modified or enhanced in newer releases.

Use these functions as part of audit scripts to assess components of the system's stored and run-time configurations. The following functions are public interfaces to the Solaris Security Toolkit software's audit framework.

When customizing or creating audit scripts, use the following functions to perform standard operations:

- `check_fileContentsExist` and `check_fileContentsNotExist`
- `check_fileGroupMatch` and `check_fileGroupNoMatch`
- `check_fileModeMatch` and `check_fileModeNoMatch`
- `check_fileOwnerMatch` and `check_fileOwnerNoMatch`
- `check_fileTemplate`
- `check_fileTypeMatch` and `check_fileTypeNoMatch`
- `check_minimized`
- `check_packageExists` and `check_packageNotExists`
- `check_patchExists` and `check_patchNotExists`
- `check_processArgsMatch` and `check_processArgsNoMatch`
- `check_processExists` and `check_processNotExists`
- `check_serviceConfigExists` and `check_serviceConfigNotExists`
- `check_startScriptExists` and `check_startScriptNotExists`
- `check_stopScriptExists` and `check_stopScriptNotExists`
- `finish_audit`

- `start_audit`

For detailed information and instructions on the use of each of these functions please refer to the "Framework Functions" chapter of the *Solaris Security Toolkit 4.1 Reference Manual*.

EXAMPLES

EXAMPLE 1 Checking for the Existence of a File

```
check_fileExists /etc/inet/inetd.conf 1 LOG
```

EXAMPLE 2 Checking for the Existence of a Patch

```
check_packageExists SUNWsshdu 1 LOG
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUES
Availability	SUNWjass
Stability	Unstable

SEE ALSO

add-client(1M)

common_log_funcs(4)

common_misc_funcs(4)

driver_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(7)

security_drivers(7)

NAME

common_log_funcs - control all logging and reporting functions for the Solaris Security Toolkit (JASS)

SYNOPSIS

`common_log_funcs`

DESCRIPTION

Framework functions provide flexibility for you to change the behavior of the Solaris Security Toolkit (also known as JASS) software without modifying source code.

`common_log_funcs` control all logging and reporting functions and are located in the `Drivers` directory in a file called `common_log.funcs`. The logging and reporting functions are used in all of the Solaris Security Toolkit software's operational modes; therefore, they are considered common functions. For example, common functions such as `logWarning` and `logError` are in this file.

This following is a list of common log functions:

- `logBanner`
- `logDebug`
- `logError`
- `logFailure`
- `logFileContentsExist` and `logFileContentsNotExist`
- `logFileExists` and `logFileNotExists`
- `logFileGroupMatch` and `logFileGroupNoMatch`
- `logFileModeMatch` and `logFileModeNoMatch`
- `logFileNotFound`
- `logFileOwnerMatch` and `logFileOwnerNoMatch`
- `logFileTypeMatch` and `logFileTypeNoMatch`
- `logFinding`
- `logFormattedMessage`
- `logInvalidDisableMode`
- `logInvalidOSRevision`
- `logMessage`
- `logNotice`
- `logPackageExists` and `logPackageNotExists`
- `logPatchExists` and `logPatchNotExists`
- `logProcessArgsMatch` and `logProcessArgsNoMatch`
- `logProcessExists` and `logProcessNotExists`
- `logProcessNotFound`
- `logServiceConfigExists` and `logServiceConfigNotExists`

- logStartScriptExists and logStartScriptNotExists
- logStopScriptExists and logStopScriptNotExists
- logSuccess
- logWarning

For detailed information and instructions on the use of each of these functions please refer to the “Framework Functions” chapter of the *Solaris Security Toolkit 4.1 Reference Manual*.

EXAMPLES

EXAMPLE 1 Log Failure

```
Usage:
logFailure "Package SUNWatfsr is installed."
Output:
[FAIL] Package SUNWatfsr is installed.
```

EXAMPLE 2 Log File Existence

```
Usage:
logFileExists /etc/issue
Output:
[NOTE] File /etc/issue was found.
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUES
Availability	SUNWjass
Stability	Unstable

SEE ALSO

add-client(1M)

audit_public_funcs(1M)

common_misc_funcs(4)

driver_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(7)

security_drivers(7)

NAME

common_misc_funcs - miscellaneous framework functions for the Solaris Security Toolkit (JASS)

SYNOPSIS

`common_misc_funcs`

DESCRIPTION

Framework functions provide flexibility for you to change the behavior of the Solaris Security Toolkit (also known as JASS) software without modifying source code.

`common_misc_funcs` are used within several areas of the Solaris Security Toolkit software and are not specific to functionality provided by other framework functions (files ending with a .func suffix). These functions are in the Drivers directory in a file called `common_misc.funcs`. Common utility functions such as `isNumeric` and `printPretty` are in this file.

This following is a list of common miscellaneous functions:

- isNumeric
- invalidVulnVal
- checkLogStatus
- adjustScore
- printPretty
- printPrettyPath
- extractComments
- clean_path
- strip_path

For detailed information and instructions on the use of each of these functions please refer to the "Framework Functions" chapter of the *Solaris Security Toolkit 4.1 Reference Manual*.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUES
Availability	SUNWjass
Stability	Unstable

SEE ALSO

add-client(1M)

audit_public_funcs(1M)

common_log_funcs(4)

driver_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(7)

security_drivers(7)

NAME

driver_funcs - Solaris Security Toolkit (JASS) driver functions

SYNOPSIS

```
driver_funcs
```

DESCRIPTION

These functions are for Solaris Security Toolkit (also known as JASS) driver functionality. These functions are in the `driver.funcs` file, located in the Drivers directory. Functions such as `add_pkg` and `copy_a_file` are in this file.

When customizing or creating scripts, use the following functions to perform standard operations:

- `add_patch`
- `add_pkg`
- `add_to_manifest`
- `backup_file`
- `check_os_main_version`
- `check_os_revision`
- `copy_a_dir`
- `copy_a_file`
- `copy_a_symlink`
- `copy_files`
- `create_a_file`
- `create_file_timestamp`
- `disable_conf_file`
- `disable_file`
- `disable_rc_file`
- `is_patch_applied`

For detailed information and instructions on the use of each of these functions please refer to the “Framework Functions” chapter of the *Solaris Security Toolkit 4.1 Reference Manual*.

EXAMPLES

EXAMPLE 1 Adding a Single Patch

```
add_patch 123456-01
```

EXAMPLE 2 Adding a Patch List

```
add_patch -M ${JASS_PATCH_DIR}/OtherPatches patch_list.txt
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUES
Availability	SUNWjass

SEE ALSO

add-client(1M)

audit_public_funcs(1M)

common_log_funcs(4)

common_misc_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(7)

security_drivers(7)

NAME

jass-check-sum - identify file changes made since the last Solaris Security Toolkit (JASS) backup

SYNOPSIS

jass-check-sum

DESCRIPTION

This Solaris Security Toolkit (also known as JASS) script identifies those files that have been modified since their checksums were originally saved in the *jass* repository (`/var/opt/SUNWjass/run/*/jass-checksums.txt`)

Only the last (most recent) checksum of a file is compared to the current file. This aids in determining if a file has been changed after being configured by the Solaris Security Toolkit. If a given configuration has already been undone, this script will skip it.

EXTENDED DESCRIPTION

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

None.

EXAMPLES

EXAMPLE 1 Checking the Solaris Security Toolkit Files

```
sc0: #:> /opt/SUNWjass/bin/jass-check-sum

Checking for file signature conflicts associated with Toolkit run:
20040621172054

File Name              Saved CkSum              Current CkSum
-----------------------------------------------------------------
/etc/passwd            685593234:456            1703916610:489
/etc/shadow                 3216256103:185            3154547236:190

sc0:#:>
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion

1 Error detected

FILES

The following file is used by this command:

`/var/opt/SUNWjass/run/`*run_id*`/jass-checksums.txt` Files which are compared to the files being tested.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Evolving

SEE ALSO

add-client(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(7)

NAME

jass-execute - configure the Solaris Security Toolkit (JASS) application

SYNOPSIS

jass-execute [`-r` *root_directory* `-p` *os_version*] [`-q|-o` *output_file*] [`-m` *e-mail_address*] [`-V` *verbosity_level*] [`-d` *driver*]

jass-execute `-u` [`-b|-f|-k`] [`-q|-o` *output_file*] [`-m` *email_address*] [`-V` *verbosity_level*]

jass-execute `-a` *driver* [`-V` *verbosity_level*] [`-q|-o` *output_file*] [`-m` *email_address*]

jass-execute `-H`

jass-execute `-l`

jass-execute `-h|-?`

jass-execute `-v`

DESCRIPTION

`jass-execute` configures the Solaris Security Toolkit (also known as JASS) depending on the options used.

EXTENDED DESCRIPTION

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

The following options are supported.

`-a` *driver*
Determines if the system is in compliance with its security profile.

`-b`
Used with the -u option.

Backs up any files that have been manually changed since the last hardening run, then restores system to its original state.

`-d` *driver*
Specifies the driver to be run. Cannot be used with the -a, -h, -H or -u options.

`-f`
Used with the -u option.

Reverses changes made during a hardening run without asking you about exceptions, even if files were manually changed after a hardening run.

`-H`
Displays the history of Solaris Security Toolkit applications on the system.

`-h |-?`
Displays usage descriptions.

Note – Use alone. Any option specified in addition to `-h|-?` is ignored.

`-k`
Used with the -u option.

Keeps any manual changes you made to files after a hardening run.

`-l`
Displays the last application of the Solaris Security Toolkit installed on the system.

`-m` *email_address*
Specifies an email address for in-house support.

`-o` *output_file*
Specifies a filename for jass run output.

`-p` *os_version*
Must be used with the `-r` *root_directory*.

Specifies the OS version of Solaris. The format is the same as that of `uname -r`.

`-q`
Quiet mode. Messages are not displayed while running this command. Output is stored in `JASS_REPOSITORY/`

`-r` *root_directory*
Must be used with the `-p` *os_version*.

Specifies the root directory used during jass-execute runs. By default, the root file system is `/`. This root directory is defined by the Solaris Security Toolkit (JASS) environment variable, `JASS_ROOT_DIR`. The Solaris OS being secured is available through `/`. For example, if you wanted to secure a separate OS directory, temporarily mounted under `/mnt` then use the `-r` option to specify `/mnt`.

`-u`
Undoes a previous application of the Solaris Security Toolkit.

Cannot be used with the `-d`, `-a`, `-h`, `-l` or `-H` options.

`-V` *verbosity*

Specifies the level of verbosity for an audit run. There are five levels (0-4)

`0` Single line indicating pass or fail.

`1` For each script, a single line indicating pass or fail. one grand total score line below all the script lines.

`2` For each script, provides results of all checks.

`3` Multiple lines providing full output, including banner and header messages. This is the default.

`4` Multiple lines (all data provided from level 3) plus all entries that are generated by the `logDebug` logging function. This level is for debugging.

`-v`

Displays the version information for this program.

EXAMPLES

EXAMPLE 1 Configure a Solaris Security Toolkit Application

```
sc0:#:> /opt/SUNWjass/bin/jass-execute -r /mnt -p 5.9 -o
output.txt -m support@mycompany.com -d secure.driver

[NOTE] The following prompt can be disabled by setting JASS_NOVICE_USER
to 0.
[WARN] Depending on how the Solaris Security Toolkit is configured, it is
both possible and likely that by default all remote shell and file transfer
access to this system will be disabled upon reboot effectively locking out
any user without console access to the system.
Are you sure that you want to continue? (YES/NO) [NO] YES
[NOTE] Executing driver, secure.driver
[NOTE] Recording output to output.txt
sc0:#:>
```

EXAMPLE 2 Undo a Previous Jass Application

```
sc0:#:> /opt/SUNWjass/bin/jass-execute -u -b -q -m
support@mycompany.com -V 3
[WARN] Creating backup copies of some files may cause unintended affects.
[WARN] This is particularly true of /etc/hostname.[interface] files as
well as crontab files in /var/spool/cron/crontabs.
[NOTE] Executing driver, undo.driver

Please select a Solaris Security Toolkit run to restore through:
1.  June 28, 2004 at 19:11:49 (/var/opt/SUNWjass/run/20040628191149)
2.  June 21, 2004 at 17:20:54 (/var/opt/SUNWjass/run/20040621172054)
3.  June 17, 2004 at 10:45:23 (/var/opt/SUNWjass/run/20040617104523)
Choice ('q' to exit)? 1
[NOTE] Restoring to previous run from
/var/opt/SUNWjass/run/20040628191149
sc0:#:>
```

EXAMPLE 3 Audit the System Against a Pre-Defined Profile

```
sc0:#:> /opt/SUNWjass/bin/jass-execute -a secure.driver -V 2 -o
output.txt -m support@mycompany.com

jass-execute                  [NOTE] Executing driver, secure.driver
jass-execute                  [NOTE] Recording output to output.txt
sc0:#:>
```

EXAMPLE 4 Display the Last Installed Solaris Security Toolkit Application

```
sc0:#:> /opt/SUNWjass/bin/jass-execute -l

# ./jass-execute -l
This information is only applicable for applications of the
Solaris Security Toolkit starting with version 0.3.
The last application of the Solaris Security Toolkit was:
1.   June 28, 2004 at 19:11:49 (20040628191149) (UNDONE)
sc0:#:>
```

EXIT STATUS

The following exit values are returned:

0	Successful completion
1	An error has occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Evolving

SEE ALSO

add-client(1M)

jass-check-sum(1M)

make-jass-pkg(1M)

rm-client(7)

NAME

make-jass-pkg - create Solaris Security Toolkit (JASS) package stream file

SYNOPSIS

make-jass-pkg [`-b` *new-base-dir*] [`-e` *excl-list*] [`-m` *new-email-address*] [`-p` *package-name*] [`-q`] [`-t` *new-title*]

make-jass-pkg`-v`

make-jass-pkg`-?|-h`

DESCRIPTION

`make-jass-pkg` creates a Solaris package stream file from the Solaris Security Toolkit, also known as JASS, distribution. The resulting file is installed using `pkgadd` and removed using `pkgrm`. Information about the installed distribution can be obtained using `pkginfo`.

EXTENDED DESCRIPTION

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

The following options are supported.

`-b` *new_base_dir*	Specifies an alternate installation base directory.
`-e` *excl-list*	Excludes top level files or directories from the package. This is done y specifying a pipe (`|`) separated list such as '`a|b/c|d`'
`-h |-?`	Displays usage descriptions. **Note –** Use alone. Any option specified in addition to `-h` or `-?` is ignored.
`-m` *new_email_address*	Specifies an email address to use for in-house support.
`-p` *package-name*	Specify a custom package name. The default is `JASScustm`.
`-q`	Quiet mode. No messages are displayed when this command is run.
`-t` *new-title*	Specifies an alternative package title. The default title is "Solaris Security Toolkit".
`-v`	Displays the version information for this program.

EXAMPLES

EXAMPLE 1 Using Defaults

```
sc0: #:> /opt/SUNWjass/bin/make-jass-pkg

[NOTE] Creating the package’s prototype file.  This may take a few minutes.
[NOTE] Excluded file: ./jass-include-list.tmp
[NOTE] Creating the package’s info file.
[NOTE] Creating the package in a scratch directory.
## Building pkgmap from package prototype file.
## Processing pkginfo file.
WARNING: parameter <PSTAMP> set to "eng120040623143146"
WARNING: parameter <CLASSES> set to "none"
## Attempting to volumize 360 entries in pkgmap.
part  1 -- 2934 blocks, 357 entries
## Packaging one part.
/opt/SUNWjass/SUNWjass/pkgmap
/opt/SUNWjass/SUNWjass/pkginfo
.
.[list of files...]
.
/opt/SUNWjass/SUNWjass/reloc/rules.SAMPLE
/opt/SUNWjass/SUNWjass/install/tsolinfo
## Validating control scripts.
## Packaging complete.
[NOTE] Creating the package’s stream format (package file).
The following packages are available:
  1 JASScustm Solaris Security Toolkit 4.1.0
                  (Solaris) 4.1.0
Select package(s) you wish to process (or ’all’ to process
all packages). (default: all) [?,??,q]: Transferring <JASScustm> package
instance
[NOTE] The package has been created as JASScustm.pkg.
sc0: #:>
```

EXAMPLE 2 Specifying Options

```
sc0: #:> /opt/SUNWjass/bin/make-jass-pkg -b /opt/SUNWjass/otherdir -e
/opt/SUNWjass/test -m eng_support@mycompany.com -p MYJASS -t MyToolkit

[NOTE] Creating the package's prototype file.  This may take a few
minutes.
[NOTE] Creating the package's info file.
[NOTE] Creating the package in a scratch directory.
## Building pkgmap from package prototype file.
## Processing pkginfo file.
WARNING: parameter <PSTAMP> set to "eng120040623150621"
WARNING: parameter <CLASSES> set to "none"
## Attempting to volumize 363 entries in pkgmap.
part  1 -- 5612 blocks, 359 entries
## Packaging one part.
/opt/SUNWjass/SUNWjass/pkgmap
/opt/SUNWjass/SUNWjass/pkginfo
.
.
.[list of files]
/opt/SUNWjass/SUNWjass/reloc/rules.SAMPLE
/opt/SUNWjass/SUNWjass/install/tsolinfo
## Validating control scripts.
## Packaging complete.
[NOTE] Creating the package's stream format (package file).
The following packages are available:
  1 MYJASS Solaris Security Toolkit 4.1.0 / MyToolkit
                  (Solaris) 4.1.0
Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: Transferring <MYJASS> package
instance
[NOTE] The package has been created as MYJASS.pkg.
sc0: #:>
```

EXIT STATUS

The following exit values are returned:

0	Successful completion
1	An error has occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Evolving

SEE ALSO

add-client(1M)

jass-check-sum(1M)

jass-execute(1M)

rm-client(7)

NAME

rm-client - remove JumpStart client for the Solaris Security Toolkit (JASS).

SYNOPSIS

rm-client [-c] *client-host-name*

rm-client -? | -h

rm-client -v

DESCRIPTION

`rm-client` removes the JumpStart client and configuration information needed by the Solaris Security Toolkit (also known as JASS). It is executed from the Jumpstart server.

EXTENDED DESCRIPTION

Group Privileges Required

You must have superuser privileges to run this command.

OPTIONS

The following option is supported.

`-c` *client-host-name* Removes the installed JumpStart client as well as all configuration information with it, needed by JASS.

`-h | -?` Displays usage descriptions.

Note – Use alone. Any option specified in addition to `-h` or `-?` is ignored.

`-v` Displays the version information for this program.

EXAMPLES

EXAMPLE 1 Remove Client

```
sc0: #:> /opt/SUNWjass/bin/rm-client -c eng1
removing eng1 from bootparams
```

where:

eng1 The hostname of the client to be removed

EXIT STATUS

The following exit values are returned:

0	Successful completion
1	An error occurred.

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

Attribute Types	Attribute Values
Availability	SUNWjass
Interface Stability	Unstable

SEE ALSO

add-client(1M)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

NAME

security_drivers - Solaris Security Toolkit (JASS) drivers

SYNOPSIS

```
config.driver

hardening.driver

secure.driver

undo.driver

desktop-secure.driver

install-Sun_One-WS.driver

jumpstart-secure.driver

suncluster3x-secure.driver

sunfire_mf_msp-secure.driver

starfire_ssp-secure.driver

sunfire_15k_domain-secure.driver

sunfire_15k_sc-secure.driver
```

DESCRIPTION

`security_drivers` refers to the collection of drivers used by the Solaris Security Toolkit (also known as JASS). The following is a list of the standard drivers:

- `config.driver`-implements tasks associated with the driver set.
- `hardening.driver`- contains most of the security specific scripts.
- `secure.driver`- default driver used in the rules for client installation. Implements all the hardening functionality.
- `undo.driver`- provides undo functionality during an undo run.
- `desktop-secure.driver` - based on secure.driver, highlights what may be necessary to secure a desktop system.
- `install-Sun_ONE-WS.driver` - applicable only to JumpStart mode, allows the Solaris Security Toolkit to install the Sun™ ONE Web Server software.
- `jumpstart-secure.driver` - based on secure.driver, highlights what may be necessary to secure a JumpStart server.
- `suncluster3x-secure.driver`- provides a baseline configuration for hardening SunPlex™, formerly Sun Cluster 3.x, software releases.
- `sunfire_mf_msp-secure.driver`- provides hardening for the midframe service processor (MSP) when building secured Sun Fire midframe environments.
- `starfire_ssp-secure.driver`- provides hardening for the Sun Enterprise™ 10000 system service processors (SSP).

- `sunfire_15k_domain-secure.driver`- provides a baseline for developing hardened Sun Fire high-end system domains.
- `sunfire_15k_sc-secure.driver`- the only supported mechanism by which the Sun Fire high-end system controller can be secured.

For detailed information and instructions on the use of each of these drivers please refer to the “Drivers” chapter in the *Solaris Security Toolkit 4.1 Reference Manual*.

EXAMPLES

EXAMPLE 1 Contents of the `secure.driver`

```
DIR="‘/bin/dirname $0‘"
export DIR

. ${DIR}/driver.init

. ${DIR}/config.driver

. ${DIR}/hardening.driver
```

EXAMPLE 2 Contents of the `undo.driver`

```
DIR="‘/bin/dirname $0‘"
export DIR

. ${DIR}/driver.init

. ${DIR}/undo.run
```

ATTRIBUTES

See **attributes**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjass
Stability	Unstable

SEE ALSO

add-client(1M)

audit_public_funcs(1M)

common_log_funcs(4)

common_misc_funcs(4)

driver_funcs(4)

jass-check-sum(1M)

jass-execute(1M)

make-jass-pkg(1M)

rm-client(7)