

SOPHOS



sav interface

User manual

SAV Interface Developer Toolkit for
SAV Dynamic Interface

Version 1.4



Contents

1 SAVDI	4
1.1 Introduction	4
1.2 Features	4
1.3 Command Line (*nix)	5
1.4 Command Line (Windows)	6
1.5 Installing SAVDI	7
1.6 Architecture	8
1.7 DoS Protection	9
1.8 Internationalisation etc	10
1.9 Logging	10
1.10 Scanning permissions	11
1.11 Security	12
1.12 Exit modes	13
1.13 Signals (*nix)	14
2 Configuration Options	17
2.1 Configuration file errors	18
2.2 Options Summary	19
2.3 Daemon	19
2.4 Log	19
2.5 Channel	22
2.6 Commprotocol	22
2.7 Scanprotocol	24
2.8 Scanner	26
Appendix: Return Codes	28
Appendix: REject Codes	29
Appendix: SAV Dynamic Interface Performance	30
Appendix: re file for *nix	32

1 SAVDI

1.1 Introduction

The SAV Dynamic Interface (SAVDI) is a daemon on *nix systems or service on Windows systems providing a networked interface to SAVI and the virus engine. It thus enables programs written in any language on any system with networking capability to scan files and data for malware.

SAVDI clients can use either the SSSP or Sophie protocols. The SSSP and Sophie protocols are described in the SAV Interface Sophos Simple Scanning Protocol User Manual.

- ❗ Sophie was developed by Vanja Hrustic as a free open source project to provide daemon access to the Sophos virus engine.

1.2 Features

1. Implemented on Unix/Linux and recent versions of Windows. Windows 9x is not supported.
2. Implements SSSP and also provides Sophie functionality for compatibility.
3. Scans can be aborted if the server detects a problem, or if the client breaks the connection.
4. Limits the possibility for DoS whether by the use of problem files or by hogs.
5. Updates to virus data can be achieved smoothly.
6. Communication is via configurable TCP ports, Unix domain sockets and Windows named pipes.

See the SSSP and Sophie protocol descriptions for the functionality that they provide.

1.3 Command Line (*nix)

SAVDI can be run as an ordinary command line process or as a daemon using the `-d` switch. There is a slight difference in behaviour between the two run modes: if it is running as a daemon SAVDI will under some circumstances respawn itself (see below), but not otherwise.

If SAVDI is run in a non-daemon mode it can be interrupted and stopped with a CTRL-C from the keyboard.

```
savdid [-d] [-c config] [-l] [-p] [-s]
```

`-d` will cause the process to switch to daemon mode.

`-c` specifies a configuration file other than the default.

`-f` specifies a pid file.

`-l` will cause logging to be sent to the console provided `-d` is not used.

`-p` will print details of the configuration file format and exit.

`-s` will suppress the initial version and copyright messages.

By default `savdid` runs as an attached process and will only switch to daemon mode if `-d` is specified. `savdid` will not respawn itself if it is not running as a daemon.

The default locations for the configuration file are `/usr/local/savdi/savdid.conf` and `/etc/savdi/savdid.conf` in that order of priority.

1.4 Command Line (Windows)

SAVDI can be run as an ordinary command line process or as a Windows service.

If SAVDI is run in command-line mode it can be interrupted and stopped with a CTRL-C from the keyboard.

```
savdid [-c config] [-l] [-v] [-h]
```

-c specifies a configuration file other than the default.

-l will cause savdid to send logging to the console if running in command-line mode.

-v will display version and copyright information and then exit.

-h will display usage information and then exit.

By default savdid runs as an attached process and will only run as a service if it has been installed as such.

If the -c command-line argument is omitted, savdid will check the registry for the following string value:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Sophos\Savdid\  
ConfigurationPath
```

If this value is present, its value will be interpreted as the path of the configuration file. If the -c command-line argument is omitted and the above registry value is absent, savdid will search its own directory for savdid.conf.

To install savdid as a service, use the following commands:-

savdid -install	Installs savdid as a service
savdid -uninstall	Uninstalls the savdid service
savdid -reinstall	Equivalent to savdid -uninstall followed by savdid -install.
savdid -start	Starts the savdid service
savdid -stop	Stops the savdid service

1.5 Installing SAVDI

A SAVDI installation consists of:-

- the savdid executable
- a DLL for Windows Event Log messages, message.dll
- a sample savdid.conf configuration file
- a language file, savdidlang_en.txt.

For *nix, savdid may be placed anywhere but the default locations for the savdid.conf and savdidlang_en.txt files are /usr/local/savdi and /etc/savdi. The location for the configuration file may be overridden by a command line option.

For Windows the savdid.conf and savdidlang_en.txt files are placed, and expected to be in the normal installation folder. However the location of the configuration file may be overridden by a command line option.

Create the configuration file either from scratch or by copying and editing the sample configuration. To verify the configuration file initially run savdid as a command line executable with the log to console switch:-

```
./savidid -l [-c config-file]
```

Use the -c switch to specify the configuration file if it is not in the standard location.

This makes debugging the configuration file simpler as errors are reported to the console.

Check the operation of savdid by running one or more of the sample applications from another terminal session.

Once savdid is running as desired it can be run as a daemon or service.

1.6 Architecture

On Windows savdid runs as a single process either from the command line or as a service.

On *nix it will also run as a single process if run in non-daemon mode. However if run in daemon mode it will run as two processes, a parent and child. The parent spawns the child process, controlling and monitoring it. The parent is the one that will respond to signals, spawning new child processes as requested, and terminating the old ones. Also, if the child process should exit unexpectedly, the parent will start a new one, maintaining the service.

On all platforms, the process providing the scanning service has a main thread with a configurable number of session, or worker, threads.

The main thread fields the initial connections from the clients which are queued to the session threads. An idle session thread will pick up the connection and handle the rest of the communication with the client until disconnection.

Any number of channels may be defined, a channel consisting of: a communications port defining the transport mechanism to be used; the definition of the protocol to be used; the definition of the SAVI object to be used. Each channel can support any number of clients, if permitted by the transport mechanism.

To minimise a session's startup time SAVI objects are pre-allocated and initialised.

SAVDI provides for multiple:-

- Protocols (SSSP and Sophie have been implemented).
- Transport mechanisms (TCP/IP, unix domain sockets and Windows named pipes have been implemented).
- Logging mechanisms (to the console, to a daily log file to syslog on *nix systems).
- Languages.

1.7 Denial of Service Protection

There is no protection against the daemon being flooded with scan requests denying service to legitimate clients however there are some measures that can be taken to limit the effect of accidental and malicious DoS attacks.

Timers can be used to limit the effect of careless usage:-

1. There is a configurable time limit imposed on a scan.
2. There is a configurable time limit imposed on a request.
3. There are timeouts on sending and receiving data from the client.
4. An idle timer closes the connection if the client is silent for too long.

Unidentified protocol requests will cause the connection to be closed.

Remote connections can be restricted to a subnet.

If a client disconnects during a scan, the scan will be aborted.

Administrators can create channels with different ownership and protections, and with different properties such as timeouts, allowing for different classes of client. Multiple versions of the daemon, with different configuration files, can be run further enhancing the separation of clients.

Memory faults and similar exceptions are caught and will cause `savdid`, if it is running in daemon mode, to spawn a new version of itself to handle new connections, while the old daemon completes its current work as possible before it exits.

`savdid` has an internal pending connection queue. Once the queue is full any further connections will be rejected. This means the maximum number of connections to `savdid` is the no of threads + the maximum length of the queue.

When scanning directories there is no protection built in from a link which loops back in the directory tree. However on *nix systems soft links are ignored and provided hard links are only permitted to files and not directories this possibility is precluded.

1.8 Internationalisation

Log messages, including SAVI result codes, are available for translation, or amendment. The language as defined by the locale is used by default if it is available. If it is not available or if a translation of a message is not available the English version will be used instead.

❗ At the moment only the English version of the text is available.

1.9 Logging

savdid has a single logger for all log messages. The log is sent to any one of a set of configurable destinations, the console, a file, or on *nix systems, the syslog. A log entry consists of a message, represented by a number and a list of parameters. The message number is converted to a text string before the message is output

On *nix systems the parent process, in daemon mode, will always log significant events to the syslog.

Savdid logs the following events, in order of priority:-

1. Errors and Detected threats.
2. Startup and shutdown of savdid.
3. Start and end of connections and sessions.

Savdid can be configured to limit the events it logs according to the above priority. Level 0 events, which are always logged, are the errors and threats. The default level is level 2 and includes savdid startup and shutdown events.

In addition, individual channels can be requested to log requests from clients. This is independent of the selected logging level.

All log entries include a timestamp and events relating to client communications also include a unique serial number which identifies the session and request. This serial number is the same number specified in the SSSP ACC message.

If the log entry is recording a threat the threat name will be enclosed in single quotes and the location of the threat will be enclosed in double quotes. Should the location of the threat include a double

quote, the double quote will be preceded by a double quote.

When logging to a serial file the log files are named with the date and a new one is started at midnight.

See the Log sub-configuration in Configuration Options.

The message number in a log message is translated before output. The translations are contained in a configurable directory which can contain more than one translation file. The file to be used for translation is also configurable. If a message number cannot be translated the translation file 'en' (English) is used, and if that also fails, the number itself is output.

1.10 Scanning permissions

SSSP and Sophie allow scanning of named files and by passing the contents to be scanned. SSSP has different requests, SCANFILE and SCANDATA. Sophie limits scanning by file and scanning by data according to the transport mechanism. With Sophie scanning by data is required for IP connections and scanning by filename is required for Unix domain sockets and Windows named pipes. SSSP allows both on any transport mechanism. However, the configuration for SSSP includes options to disallow scanning by file and scanning by data by channel (allowscanfile and allowscandata).

SSSP and Sophie also permit the scanning of directories if allowed. Sophie will accept a directory specification instead of a file specification and scan the directory and sub-directories. SSSP has specific requests for scanning a directory and for scanning sub-directories, SCANDIR and SCANDIRR. For both protocols the ability to scan directories and to scan sub-directories are separately controlled by configuration options, allowscanfile for SSSP and allowscandir for Sophie.

Note that the SSSP scan directory requests permit the use of a file name instead of a directory name, whereas a scan file request does not permit the use of a directory name.

See the Scanprotocol sub-configuration in Configuration Options.

Irrespective of the protocol, only files and directories will be scanned,

other types of file system object, for example symbolic links will be ignored or rejected.

For the SSSP SCANFILE, SCANDIR and SCANDIRR and Sophie scan file by name, requests it is possible to limit the areas that may be scanned. The configuration options 'deny' and 'allow' permit the administrator to prevent scanning of directories (/dev on *nix systems may be denied, for example), or to limit clients to an allowed area. The logic is:-

If the requested path is denied, deny the request.

If the requested path is allowed, allow the request.

If there are any allowed paths, deny the request otherwise allow it.

While the first two tests are obvious the last one means that a single allow option will limit all requests to that sub-tree, allowing the administrator to set up a special scanning directory, say.

This is intended to provide for either a restrictive system where only specific areas are allowed, or a permissive system where specific areas are denied, so that in general only either the allow, or the deny options would be used. Mixing allow and deny options is not recommended.

See the Scanner sub-configuration in Configuration Options.

1.11 Security

At the process level:-

- savdid will accept only a configurable number of concurrent sessions.

At the channel level:-

- TCP/IP sockets can be limited to a specific sub-set of IP addresses from which connections will be accepted.
- To allow or deny the ability to scan by filename or scan by data if SSSP is used. The ability to scan directories and to recurse into sub-directories is also configurable and for all application protocols.

- Timeouts in seconds can be set for:-
 - Idle time between client requests.
 - Wait time for a receive to complete.
 - Wait time for a send to complete.
 - Maximum time allowed for a file scan to complete.
 - Maximum time allowed for a request to complete.
 - The maximum amount of data that may be sent for scanning.
- However:-
- There is no encryption so network socket users have no protection from network sniffers.

Security (*nix)

At the process level:-

- savdid, if run from a super-user account, can switch to running as a configurable user.
- savdid will refuse to scan a file for which the user of a unix domain socket client, does not have read permissions where possible.

At the channel level:-

- Unix domain sockets can be created with configurable ownership and permissions.

Security (Windows)

At the process level:-

- savdid impersonates the client of a named pipe connection and attempts all file accesses from within that security context.

1.12 Exit modes

savdid has three ways of exiting:

- Ungracefully. The process exits immediately.

- Request gracefully. It will complete all current requests before disconnecting from the client and exiting when all have finished. New connections are rejected.
- Session gracefully. It completes all current sessions and exits when all have completed. New connections are rejected.

The choice between these three modes of exiting is configurable using the `onrequest` daemon option.

1.13 Signals (*nix)

The daemon will trap and respond to the following signals:-

SIGINT	To exit immediately and without waiting for any completions, i.e. the daemon will exit ungracefully.
SIGTERM	To finish gracefully, waiting for current requests or sessions to complete according to the <code>onrequest</code> configuration option.
SIGHUP	To spawn a new daemon which will handle new connections while the old one finishes the current requests or sessions, depending on the <code>onrequest</code> configuration option, before exiting. The new process will reload the configuration file and virus data. If <code>savdid</code> is not running in daemon mode it will not spawn a new process but will exit according to the value of the <code>onrequest</code> configuration option.

In addition the daemon traps exceptions and signals originating in the virus engine and will treat it as if a SIGHUP signal had been received, except that the exit mode will depend on the `onexception` configuration option. The `onexception` option is similar to the `onrequest` option.

This table summarises these actions:

Signal	Respawn?	Exit mode
INT	No	Immediate
TERM	No	onrequest option
HUP	Yes (if in daemon mode)	onrequest option
Non-recoverable error Non-recoverable error	Yes (if in daemon mode)	onexception option

To enable the correct `savdid` process to be signalled, `savdid` writes its process ID, `pid`, to a file from which it can be read and used for signalling. The `pid` file may be specified on the command line or in the configuration file. If it is not specified at all, the default is `/var/run/savdid.pid`. This shell command line will correctly signal the daemon, if using the default `pid` file:-

```
$ kill -HUP `cat /var/run/savdid.pid`
```

❗ Note the quotes are back quotes not the more usual single quotes.

Moribund processes (*nix)

On occasion `savdid` daemon will either exit or restart itself as a result of a request or an exception. When restarting, the new `savdid` will take on new connections, while the old one completes its current work. So before it fully exits, the old `savdid` will enter a moribund state while it waits for current work to complete. Depending on circumstances, it is possible for there to be more than one moribund process though such circumstances should be very rare.

The parent process monitors the number of child processes and will forcibly terminate the older processes when the number of child processes exceeds five, i.e. one active and four moribund.

See the `onexception` and `onrequest` configuration options.

Virus and Engine Updates

On Windows, provided the host system has the Sophos automatic updates included, the savdi service will automatically be updated with the latest virus data and version of the virus engine. If there is no automatic updating then the service must be stopped and re-started to load the latest version of the virus data and engine.

On *nix systems, the daemon can be signalled with a HUP to cause it to reload the virus data, see description of signals above. To use a new version of the virus engine the daemon must be stopped using either the INT or TERM signals and re-started.

2 Configuration Options

The configuration file syntax is essentially a list of name/value pairs which are grouped into named sub-configurations that define the configuration of well-defined elements of the daemon. Sub-configurations can in turn contain sub-configurations.

Syntax is:-

<statement list> := <statement> {<statement list>}

<statement> := <option> | <subconfiguration>

<subconfiguration> := <name> “{”<eol><statement list> “}”<eol>

<option> := <option name><option sep><value><sep>
<value list><eol>

<value list> := [|<value><value list>]

<option sep> := “:”

<sep> := <spaces>

<eol> represents the end of the line.

<spaces> represents one or more space characters.

Comments start with a “#” and end at the end of line and are ignored.

As an example, this is a fragment from a savdid configuration file:-

```
user: savdid
group: savdid
threadcount: 3

onrequest: REQUEST
onexception: DONTWAIT

log {
    type: FILE
    logdir: /var/log/savdid/log
} loglevel: 0
channel {
```

```
commprotocol {
    type: IP
    address: 172.18.33.14
    port: 4010
    subnet: 172.18.33.0/24
    # etc  scanprotocol {
    type: SSSP
    allowscandata: YES
    allowscanfile: NO
    # etc
    }
    scanner {
        type: SAVI
        savigrp: GrpInternet 1
        deny: /dev
        # etc
    }
}
channel {
    #etc
}.

```

2.1 Configuration file errors

When savdid starts it will do some checks on the configuration and may display one or more errors. The errors have the following format:-

<lineno>: <option name>: <option value>: <error message>

The <option name> will include the sub-configurations, if any, where the option is located. So for example “channel[3]-scanner[1]-deny” specifies the deny option in the first scanner sub-configuration of the third channel.

2.2 Options Summary

In the following tables, the columns are:-

Option - the option name and the format of the option.

N - the cardinality of the option, ie how many there can or should be. A single number (usually '1') indicates that there must be that number of options present. A range, 0..1 for example, indicates the minimum and maximum number of occurrences. The example, 0..1 is for something that can occur at most once. 1..N means that it must occur at least once but there is otherwise no limit.

Default - the default value if there is one. '-' indicates that there is no default.

Comment - self-explanatory.

2.3 Daemon

Option	N	Default	Comment
pidfile: <filename>	0..1	/var/run/savdid.pid	For *nix systems, Where savdid will store the process ID of the controlling process. If a pidfile is specified on the command line, the command line will take precedence.
user: <user name>	0..1	-	For Unix systems. The user to switch to on start up. See note 1.
group: <group name>	0..1	-	For Unix systems. The group to switch to on start up. See note 1.

Option	N	Default	Comment
threadcount: N	0..1	3	The number of threads to use and hence the maximum number of concurrent scans.
maxqueuedsessions	0..1	0	The maximum number of connections/sessions to queue to be queued internally. 0 means there will be no queueing.
virusdatadir: <dir name>	0..1	Defaults to SAVI default	Where to find the virus data. See note 2.
virusdataname <name>	0..1	Defaults to SAVI default	The name of the main virus data file, without its extension. See note 2.
idedir: <dir name>	0..1	Defaults to SAVI default	Where to find the IDE files. See note 3.
onexception: "DONTWAIT" "REQUEST" SESSION"	0..1	"REQUEST"	What to do when exiting as a result of an exception. See note 3.
onrequest: "DONTWAIT" "REQUEST" SESSION"	0..1	"SESSION"	What to do when exiting as a result of a request to exit. See note 3.
log	0..1	-	The sub-configuration for logging.. See note 4.
channel	1..N	-	The sub-configuration for the communication channels.

Note 1: To change user or group, the daemon must have suitable privileges, and possibly running as super-user.

Note 2: The virusdatadir, virusdataname and idedir options can also be set using the savistr options in the channel scanner configuration. However since the virus data is common to all this is not advised. In such a situation the set of virus data that will be used is undefined.

Note 3: When exiting the daemon can exit immediately without

waiting, or it can wait for the current requests to complete, or it can wait for the current sessions to complete. The choice depends on the circumstances (exception or request) and in particular whether sessions are long running or not.

Note 4: If a log configuration is not specified the logging is to either the console or system/event log depending on whether the daemon is running interactively or not.

2.4 Log

Option	N	Default	Comment
type: "FILE" "CONSOLE" "SYSLOG"	0..1	See note 1.	
logdir: <directory name>	0..1	-	Where to write the log files if FILE is specified. See note 2.
loglevel: <level>	0..1	2	The level of event logging required. Level 0 is the minimum and includes errors and threats. Level 2 includes everything.

Note 1: The log type defaults depending on whether or not logdir is defined: if it is defined FILE is assumed, if not CONSOLE is assumed.

Note 2: Log files are named yymmdd.log and a new file is started with the first log message after midnight.

2.5 Channel

Configuration options per channel.

Option	N	Default	Comment
commprotocol	1	-	The sub-configuration defining the transport protocol to be used.
scanprotocol	1	-	The sub-configuration defining the application protocol to be used.
scanner	1	-	The sub-configuration defining the scanner.

2.6 Commprotocol

The options for the commprotocol configuration depend on the choice of transport , IP, Unix or Pipe.

Option	N	Default	Comment
type: "IP"	1	-	Specifies TCP/IP as the transport layer.
address: <ip address>	0..1	0.0.0.0	Specifies the IP address for the socket to bind to.
port: N	1	-	Specifies the port no. to use for an IP channel (1 to 65535).

Option	N	Default	Comment
subnet: <ip address>/ <subnet>	0..1	-	The client IP addresses from which the daemon will accept connections. <subnet> is the number of significant bits in the address and so is in the 0 to 32.
requesttimeout: N	0..1	-1	The timeout when waiting for a new request in seconds. -1 means wait forever.
sendtimeout: N	0..1	5	The timeout for sending in seconds.
recvtimeout: N	0..1	5	The timeout for receiving in seconds.

Option	N	Default	Comment
type: "UNIX"	1	-	Specifies a Unix domain socket as the transport layer.
socket: <filename>	1	-	The file name for the socket for a UNIX channel. Not used otherwise.
user: <user name>	0..1	Daemon user	The user name for ownership of the socket.
group: <group name>	0..1	Daemon group	The group name for ownership of the socket.
mode: <mode>	0..1	user	One of user, group, all specifying who may use the socket.
requesttimeout: N	0..1	-1	The timeout when waiting for a new request in seconds. -1 means wait forever.

Option	N	Default	Comment
sendtimeout: N	0..1	5	The timeout for sending in seconds.
recvtimeout: N	0..1	5	The timeout for receiving in seconds.

Note: To set user, group and protections on a socket, the daemon must have suitable privileges, and possibly running as super-user

Option	N	Default	Comment
type: "Pipe"	1	-	Specifies a Windows named pipe as the transport layer.
name: <name>	1	-	The name for the named pipe.
requesttimeout: N	0..1	-1	The timeout when waiting for a new request in seconds. -1 means wait forever.
sendtimeout: N	0..1	3	The timeout for sending in seconds.
recvtimeout: N	0..1	3	The timeout for receiving in seconds.

2.7 Scanprotocol

Option	N	Default	Comment
type: "SSSP" "SOPHIE"	1	-	Specifies the scan protocol to be used.
tmpfilestub: <partial file name>	0..1	-/tmp/savdid_ tmp	Species the directory and file name stub to be used for temporary files used when scanning data sent to the server - see maxmemoryscansize below.

Option	N	Default	Comment
maxscandata: N	0..1	0 (unlimited)	In bytes the maximum amount of data a client can send for scanning.
maxmemorysize: N	0..1	250000	The maximum size of data from the client before using a temporary file instead. See tmpfilestub above.
allowscanfile: "NO" "FILE" "DIR" "SUBDIR"	0..1	NO	For SSSP only. Specifies the level of scanning a client is permitted. NO means none at all, FILE means only explicit files may be scanned, DIR means a directory or file may be scanned, SUBDIR means that sub-directories will also be scanned. See notes (1) and (2).
allowscandir: "NO" "DIR" "SUBDIR"	0..1	NO	For SOPHIE only. Specifies the level of scanning a client is permitted. NO means that directories cannot be scanned, DIR means a directory or file may be scanned, SUBDIR means that sub-directories will also be scanned. See note (1).
allowscandata: "YES" "NO"	0..1	NO	For SSSP only. Allow the client to send data for scanning. See notes (1) and (2).
logrequests: "YES" "NO"	0..1	NO	Log all client requests.

Note 1: The allowscanfile and allowscandata options do not apply for Sophie as the protocol will only allow scanning by filename on Unix domain sockets and scanning by data on TCP/IP sockets.

Note 2: SSSP will enable the requests SCANDATA, SCANFILE, SCANDIR and SCANDIRR according to the settings of these options.

2.8 Scanner

Specifies the scanner and its configuration options.

Option	N	Default	Comment
type: "SAVI"	0..1	"SAVI"	Specifies the type of scanner. For now only SAVI is accepted.
inprocess: "YES" "NO"	0..1	"YES"	Specifies whether the scanner is in process or not, the alternative is out of process. For now only "YES" is accepted.
maxscantime: N	0..1	-1 (unlimited)	Maximum time, in seconds, to allow a scan to complete.
maxrequesttime: N	0..1	-1 (unlimited)	Maximum time in seconds to allow servicing a request. This should be greater than or equal to maxscantime.
deny: <directory name>	0..N	-	Specifies a directory which is not allowed to be scanned. All files and sub-directories will be barred.
allow: <directory name>	0..N	-	Specifies a directory which is allowed to be scanned. By implication all other directories are denied unless explicitly allowed.
saviint: <name> <value>	0..N	-	Specifies a SAVI/Engine integer (U16) option. See Note 1.

Option	N	Default	Comment
savigrp: <name> <value>	0..N	-	Specifies a SAVI/Engine group configuration option. See Note 1.
savistr: <name> <value>	0..N	-	Specifies a SAVI Engine string option. See Notes 1 and 2.
savists: <name> <value>	0..N	-	Specifies a SAVI/Engine status (U32) option. See Note 1.

Note 1: savdid does not validate or verify the SAVI options; they are passed through to the SAVI objects as given.

Note 2: It is inadvisable to use these options to set the virus data directory and associated options. These options should be set at the daemon level.

Appendix: Return Codes

In addition to the SAVI return codes documented in `savi_sen.pdf` as part of the SAVI Developer Toolkit, the following return codes can also be returned by the SAV Dynamic Interface in response to a scan request:-

- | | |
|------|---|
| 060D | Engine exception. Please report the file to Sophos Technical Support. |
| 060E | Request terminated due to timeout.
See the <code>maxrequesttime</code> configuration option. |
| 060F | Scan terminated due to timeout.
See the <code>maxscantime</code> configuration option. |

Appendix: REJect Codes

If SSSP rejects a request, the REJ message is accompanied by a numeric code indicating the cause:-

- 1 The request was not recognised. After sending this the server will disconnect.
- 2 The SSSP version number was incorrect.
- 3 There was an error in the OPTIONS list.
- 4 SCANDATA was trying to send too much data.
- 5 The request is not permitted.

Appendix: SAV Dynamic Interface Performance

The method used by the client to interact with the SAV Dynamic Interface will be constrained by circumstances. Nonetheless the following are offered as guidelines in selecting the best method:-

1. Connect and stay connected. There is a considerable overhead to establishing the connection, and breaking it.
2. Use a Unix domain socket, or windows named pipe rather than TCP/IP. There is an extra overhead using TCP/IP.
3. Avoid using a one-off script, i.e. executing a script for each scan. Not only is there the overhead of making a new connection but also the overhead of starting the interpreter, script compilation, and so on.
4. Use the SSSP OPTIONS request only if necessary, it is cheaper to have the options set in the configuration file. If a client has different needs to other clients, create another channel tailored for that client.
5. If possible scan a whole directory rather than making requests to scan each file.

To illustrate the differences in performance, the following table, ordered with the fastest first, shows throughput of files being scanned relative to the fastest method (indexed at 100):-

- a. A C program which connects, scans a file 100 times, and disconnects is run once .

Unix domain socket	100
TCP/IP (localhost:4010)	97

- b. A Perl script which connects, scans a file 100 times, and disconnects is run once.

Unix domain socket	83
TCP/IP (localhost:4010)	81

- c. A Perl script which connects, scans a file, and disconnects 100 times is run once.

Unix domain socket 56

TCP/IP (localhost:4010) 50

- d. A C program which connects, scans a file, and disconnects is run 100 times.

Unix domain socket 17

TCP/IP (localhost:4010) 6

- e. A Perl script which connects, scans a file, and disconnects is run 100 times.

Unix domain socket 4

TCP/IP (localhost:4010) 3

Appendix: rc file for *nix

This is a basic rc file for a Redhat distribution as an illustration of an rc file and how to control savdid.

```
#!/bin/bash
#
# description: Sophos SAV Dynamic Interface
#
# processname: savdid

SAVDID=/usr/local/savdi/savdid
PIDFILE=/var/run/savdid.pid

# source function library
. /etc/rc.d/init.d/functions

RETVAL=0

prog="savdid"

case "$1" in
    start)
        echo -n $"Starting $prog: "

        # Start savdid in daemon mode, no banner, and specifying
        # the pidfile
        $SAVDID -d -s -f $PIDFILE
        RETVAL=$?

        # Sleep a moment to let savdid get things worked out
        sleep 1

        # The presence of the pidfile indicates that it is still running
        [ -f $PIDFILE ] && RETVAL=0
        if [ $RETVAL -eq 0 ]; then
            echo_success
            touch /var/lock/subsys/savdid
        else
            echo_failure
        fi
    ;;
esac
```



```
fi
echo
;;
stop)
    echo -n $"Shutting down $prog: "
    # Tell savdid to stop dead
    [ -f $PIDFILE ] && kill -INT `cat $PIDFILE`
    while [ -f $PIDFILE ]; do sleep 1; done

    echo_success
    rm -f /var/lock/subsys/savdid
    echo
    RETVAL=0
    ;;
restart)
    echo -n $"Shutting down $prog: "
    # Tell savdid to exit gracefully
    [ -f $PIDFILE ] && kill -TERM `cat $PIDFILE`
    while [ -f $PIDFILE ]; do sleep 1; done

    echo_success
    echo
    $0 start
    RETVAL=$?
    ;;
reload)
    echo -n $"Reloading $prog"
    if [ ! -f $PIDFILE ]; then
        echo " $prog is not running"
        RETVAL=1
    else
        kill -HUP `cat $PIDFILE`
        RETVAL=0
    fi
    echo
    ;;
condrestart)
    if [ -f /var/lock/subsys/savdid ]; then
        $0 stop
```

```
        $0 start
    fi
    RETVAL=$?
    ;;
status)
    status savdid
    RETVAL=$?
    ;;
*)
    echo $"Usage: $0 {start|stop|restart|reload|condrestart|status}"
    exit 1
esac

exit $RETVAL
```

Copyright 2007 Sophos Group. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise unless you are either a valid licensee where the documentation can be reproduced in accordance with the licence terms or you otherwise have the prior permission in writing of the copyright owner.

Sophos and Sophos Anti-Virus are registered trademarks of Sophos plc and Sophos Group. All other product and company names mentioned are trademarks or registered trademarks of their respective owners.