Solstice Bandwidth Reservation Protocol 1.0 Administration Guide

THE NETWORK IS THE COMPUTER™

SunSoft, Inc.
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043 USA
415 960-1300 fax 415 969-9131

Part No.: 805-0378-10
Revision A, March 1997

Contents

Figures

Tables

Preface

The *Solstice Bandwidth Reservation Protocol Administration Guide* provides information on how to configure and maintain the Bandwidth Reservation Protocol product.

There is a brief explanation of the RSVP protocol, an analysis of the configuration file and details of the shared path installation procedure.

It explains how to use the RSVP management tools and provides recommended levels of logging.

Who Should Use This Book

This book is written for system administrators who wish to configure and maintain the Solstice Bandwidth Reservation Protocol product. For developers wishing to use the RSVP API to enable RSVP applications, refer to the *Solstice Bandwidth Reservation Protocol Programming Guide*.

How This Book Is Organized

Chapter 1, "Introducing Solstice Bandwidth Reservation Protocol" briefly explains the RSVP protocol and how messages are sent and received.

Chapter 2, "Configuring Solstice Bandwidth Reservation Protocol," describes the parameters contained within the RSVP configuration file, where the file can be located, and the shared path installation procedure.

Chapter 3, "Solstice Bandwidth Reservation Protocol Management Tools," explains how to use the RSVP management tools, including setting the log levels.

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals using this program.

For a list of documents and how to order them, see the catalog section of the SunExpress™ Internet site at `http://www.sun.com/sunexpress`.

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

Abbreviations

Table P-3 Abbreviations

Abbreviation	Meaning
API	Application Program Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CCITT	International Telegraph and Telephone Consultative Committee
IA5	International Alphabet No. 5
ISO	International Organization for Standardization
OM	Object Management
OSI	Open Systems Interconnection
UCT	Universal Coordinated Time

Introducing Solstice Bandwidth Reservation Protocol 1

Solstice™ Bandwidth Reservation Protocol is an implementation of the RSVP protocol for reserving sufficient network bandwidth and resources for IP applications in the Internet and intranet world.

It specifies standard ways for requesting a particular Quality of Service (QoS) from network devices such as gateways, routers and switches along the data path. Operations are transparent through routers that do not support the RSVP protocol.

Solstice Bandwidth Reservation Protocol is significant for the development of distributed multi-media applications. It helps ensure that there is sufficient bandwidth to support a combination of voice, data and videoconferencing applications with quality levels that meet the needs for each traffic type.

RSVP (Resource Reservation Protocol) has been developed by the Internet Engineering Taskforce (IETF).

Solstice Bandwidth Reservation Protocol consists of two main components:

1. A user level daemon that processes requests from the application and translates them into the RSVP protocol. See "The Solstice Bandwidth Reservation Protocol Daemon" on page 4 for more details.

2. A user level library that implements the Application Programming Interface (API). An application must be linked to this library to use Solstice Bandwidth Reservation Protocol. Refer to the *Solstice Bandwidth Reservation Protocol Programming Guide* for more details.

Sending and Receiving RSVP Messages

The RSVP protocol is receiver oriented which means it is the responsibility of the receiver application to initiate and maintain the resource reservation. It does this by sending Quality of Service reservation requests through all the routers along the data path to the sender application.

The reservation request (`RESV` message) from the receiver contains:

- A flowspec. This contains the desired Quality of Service.
- A filterspec. This determines which sender's packets receive the corresponding Quality of Service.

The router will accept or deny the request for bandwidth reservation depending on its available resources. If the router accepts the request, a confirmation message will be sent back to the receiver (if specified).

The Quality of Service reservation request will then be passed upstream to the next router in the data path. If any of the routers along the data path deny the reservation request, an error message will be generated and the receiver application will be notified through the API.

Once the sender application receives and accepts the request message, it knows that any data sent back to the receiver application along the reserved path will be covered by the receivers' reservation request.

The `rsvpd` daemon regularly transmits RSVP packets with the data stream. These packets are `PATH` messages which contain general specifications about the data being transmitted.

The RSVP packets (`PATH` messages) from the sender application contain:

- A sender template. This describes the format of data packets to be transmitted by the sender.
- A sender Tspec. This defines the traffic characteristics of the data stream that the sender will generate.
- An Adspec. Adspecs are used to propagate information about the path to receivers.
- The previous hop address.

This information enables the receiver applications to generate appropriate bandwidth reservation requests.

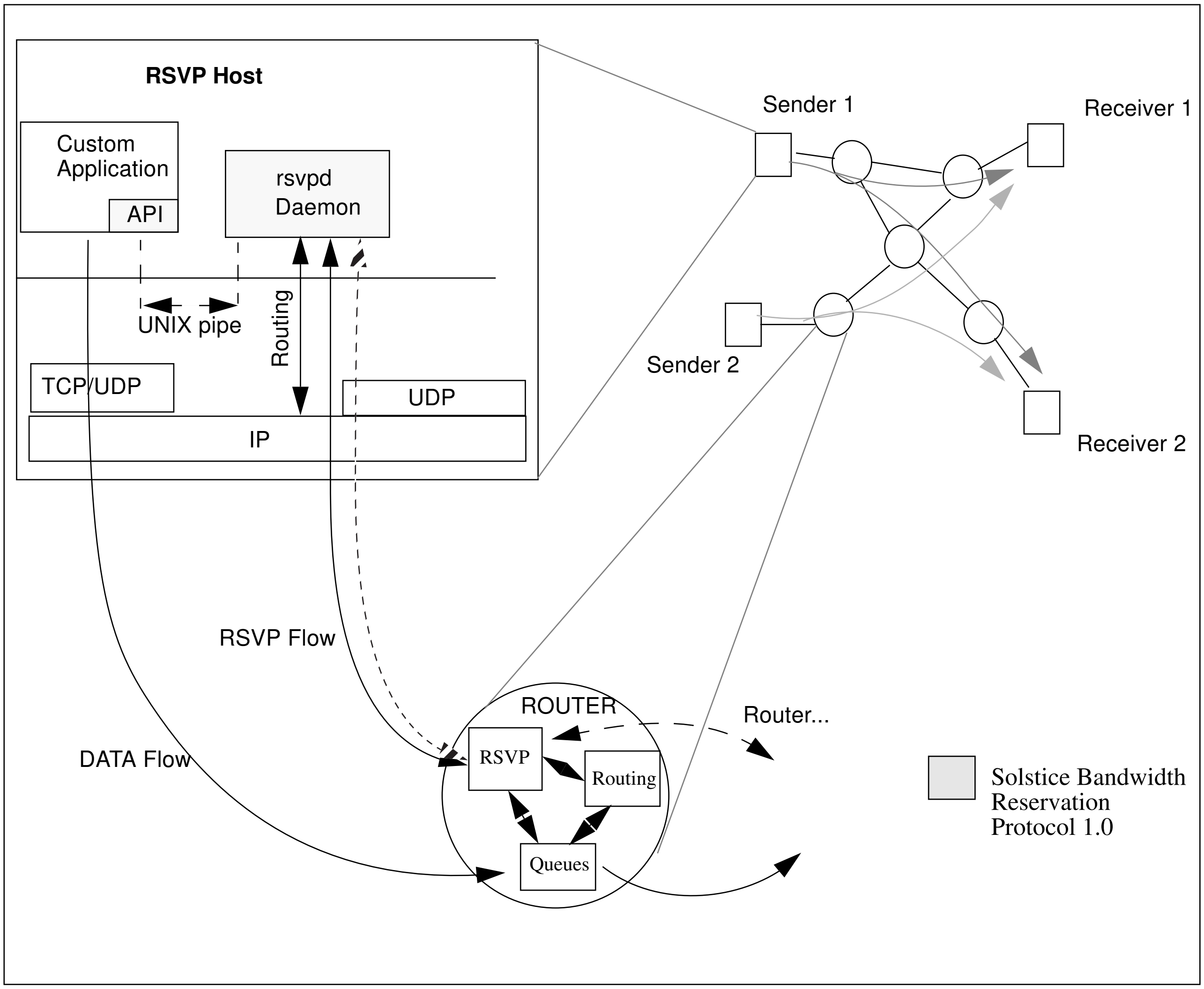

Figure 1-1 Solstice Bandwidth Reservation Protocol operating environment

The Solstice Bandwidth Reservation Protocol Daemon

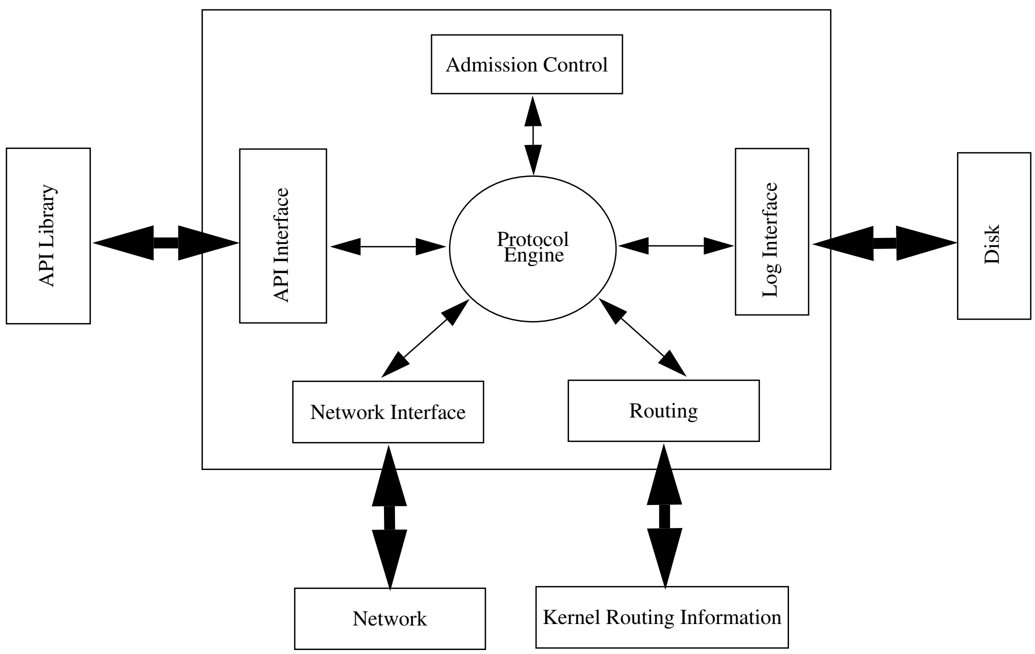

Figure 1-2 The Solstice Bandwidth Reservation Protocol Daemon

The `rsvpd` daemon consists of the following modules:

Admission Control Module

The Admission Control Module is responsible for local resource management, and prevents excessive demand on local resources.

API Interface

This interface listens for incoming connections from the RSVP clients over a UNIX domain socket and sends instructions to the API. Each time the application uses the `rapi_session()` function to open an RSVP session, the

RSVP library connects to the `rsvpd` daemon over the `/tmp/RSVP.daemon` socket. The API is described in detail in the *Solstice Bandwidth Reservation Protocol Programming Guide*.

Network Interface

This interface uses the Internet Protocol (IP) which requires a protocol number to dispatch transport level traffic to the right protocol. The IP number for RSVP is `IPPROTO_RSVP`=46.

RSVP uses Solaris raw sockets to communicate to the IP kernel module. If a receiver does not support raw IP, the daemon uses a UDP encapsulated mode.

Routing Interface

This interface supplies routing information to the daemon, instructing it which interface to use. The daemon uses a Kernel routing socket if available, otherwise the IP MIB2 data is searched.

Log Interface

This interface manages the log files and formats system messages. It ensures that the logs do not use more than a specified amount of disk space. The location, maximum size and content of the log files are fully configurable.

Configuring Solstice Bandwidth Reservation Protocol 2

This chapter explains the parameters contained within the RSVP configuration file, where the file can be located, and details of shared path installation.

Modifying the Default Configuration File

Configuration data for Solstice Bandwidth Reservation Protocol are contained in the text file `/etc/opt/SUNWconn/rsvp/rsvpd.conf`. Each parameter must be assigned a value and default values are supplied. If you want to change the default values, edit the file using a text editor. The parameters are described in the following subsections.

Configuration Parameters

interface

Syntax

```
interface interface_name bandwidth b
```

Description

An interface specification is provided for each network interface under the control of RSVP. A multihomed system could have more than one interface specification.

Arguments

interface_name

One specification is required for each RSVP controlled network interface. Use `default` to select the default interface on your machine. For example, `le0` is the standard ethernet interface on SPARCstation desktop workstations. Use `all` to configure all available interfaces with the same bandwidth. Interfaces may also be specified individually on separate lines. If an interface is omitted then Solstice Bandwidth Reservation Protocol will not use this interface.

b

The maximum bandwidth that the admission control will allow on this interface. This value is in bits per second.

log

Syntax

```
log directory size s max_rank r logging_level l logging_mask m
```

Description

This parameter configures the logging facility. For recommended levels of logging see "Recommended Logging Levels" on page 18.

Arguments

directory

The directory to which log files will be written. The default is `/var/opt/SUNWconn/rsvp/log`.

s

The maximum size of each individual log file in bytes. The default value is 8192.

r

The maximum number of log files that the log mechanism will create. Each file name will be `rsvpd.log.`*rank number*`.` The default value is 8.

l

Determines the amount of information logged by the `rsvpd` daemon. Permitted values are:

- 2 - critical conditions
- 3 - error conditions
- 4 - warning conditions
- 5 - normal but significant conditions
- 6 - informational
- 7 - debug-level messages
- 8 - includes hex dumps of interface events

Each level automatically includes the previous value. The default is 3.

m

This is a hexadecimal value of the form *0xdd*. It changes the type of information that the `rsvpd` daemon will log. Permitted values are any added combination of the following:

- 0X01 - Log packets sent and received
- 0X02 - Dump data structures
- 0X04 - Log API, kernel reservation events
- 0X08 - Log routes
- 0X10 - Multicast RSVP state (to `dmap`)
- 0X20 - Timer events
- 0XFF - All the above

For example, to dump data structures and log API and kernel reservation events, the value would be 0X06. The default value is 0XFF.

`version`

Syntax

```
version v
```

Description

The version number of the configuration file.

Argument

v

This parameter enables version protection and should not be changed. This release is 1.0.

syslog

Syntax

```
syslog f
```

Description

This parameter is used to log critical information into syslog. This is defined in the `syslog(3)` man page.

Arguments

f

Permitted values are 0-7 corresponding to `LOG_LOCAL` *value*. The default value is 0.

router

Syntax

```
router r
```

Description

This is the IP address or hostname of the next RSVP-capable router supporting UDP encapsulated RSVP.

Arguments

r

This is the IP address or hostname of the next RSVP-capable router. The default value is defined in the routing table.

inact_to

Syntax

```
inact_to t
```

Description

This parameter controls the autostop feature. It specifies how long the rsvpd daemon waits for RSVP clients before terminating.

Arguments

t

The number of seconds the rsvpd daemon waits for RSVP clients before terminating. If this value is set to 0 then the daemon will not terminate. The default is 300 (5 minutes).

Example rsvpd.conf *Configuration File*

```
interface le0 bandwidth 5000000
version 1.0
syslog 0
log /var/opt/SUNWconn/rsvp/log size 1024 max_rank 8  logging_level 3
logging_mask 0xff
router 192.0.2.10
inact_to 300
```

Locating the Daemon Configuration File

You can use the environment variable RSVPDCONF to tell the daemon where to find its configuration file rsvpd.conf. If RSVPDCONF is set, it is used as the location of the configuration file and rsvpd.conf is started from this directory. If the environment variable RSVPDCONF is not set, the RSVP library will set RSVPDCONF to *rsvpd_location*/../etc before starting the daemon. The daemon then uses RSVPDCONF to find the location of rsvpd.conf.

You only need to set RSVPCONF if rsvpd.conf is in a non-standard location.

Note – If the daemon is started manually and not by the API then the environment variable RSVPDCONF may not exist when the daemon is executed. In this case, the daemon uses the standard default location /etc/opt/SUNWconn/rsvp.

Locating the Daemon

You can use the environment variable RSVPDPATH to tell the API where to find the daemon binary, in the event that the API needs to start the daemon. If RSVPDPATH is set then this is used as the location of the daemon and the daemon is started from this directory. If RSVPDPATH is not set, then the standard daemon locations /usr/lib and /opt/SUNWconn/rsvp/lib will be added to the PATH environment variable so that the daemon may be started. The API then uses the PATH variable locate the daemon.

You only need to set RSVPDPATH if the daemon binary is in a non-standard location.

Starting and Stopping the Daemon Manually

The daemon is usually started and stopped automatically through use of the library. There are two ways to do this manually.

1. To start rsvpd, become root and type:

```
prompt%/etc/rc2.d/S96rsvpd start
```

   To stop rsvpd, become root and type:

```
prompt%/etc/rc2.d/S96rsvpd stop
```

2. To execute rsvpd in debugging mode, use the argument -D. It will run in the foreground and print debugging information to the console (stdout). To display help information, use the argument -h.

   State and event information will appear on the console (stdout) if rsvpd is executed in debug mode, for instance, with the -D flag.

Note – Note Before you start the `rsvpd` daemon, make sure that you have set all the required configuration parameters.

See "Decoding and Displaying the rsvpd Log Files" on page 21 for further information on viewing RSVP logs.

Shared Path Installation

If Solstice Bandwidth Reservation Protocol is to be installed on a large number of hosts, it is recommended that you implement a server based NFS installation. This means installing Solstice Bandwidth Reservation Protocol on one server only, with all clients using the package from a mounted point.

There are several points to be aware of if using an NFS mounted library. There is no local startup script, local host specific configuration or well-known daemon location.

In addition, the `rsvpd` binary may lose its `SETUID` flag, depending on the NFS configuration. Without the `SETUID` flag, the API's autostart feature cannot start the daemon as root and the daemon will start in a degraded mode. This would mean UDP encapsulation, default routing on the default interface, and no interoperability with IP-only RSVP implementations. There are two solutions to avoid this:

1. Allow the `SETUID` bit on `rsvpd.`
2. Disable the daemon's autostop feature by:

- Setting `inact_to` equal to 0 in the `rsvpd.conf` file.
- Copying the `rc2.d` scripts provided with the package to each client's `/etc/rc2.d` directory. This will start the daemon as root during the boot phase. Then, you can either:

   a. Manually link the mounted `rsvpd` to `/usr/lib/rsvpd` and `rsvpd.conf` to `/etc/opt/SUNWconn.`

   or

   b. Change the daemon location paths in the `/etc/rc2.d/S96rsvp` and `/etc/rc2.d/K96rsvp` daemon startup files. Set `inact_to` to 0 in the `rsvpd.conf` file and copy these files to every client.

To set the environment variables RSVPDPATH and RSVPDCONF, see the above sections “Locating the Daemon” and “Locating the Daemon Configuration File” on page 12.

If you are not using a standard setup, the configuration file will need to be customized for each client using the NFS mount point. You must alter the log file location (log) in rsvpd.conf to point to a directory that is present on every client system, for example /var/tmp.

You should also check that the bandwidth and interface values in the rsvpd.conf file are appropriate to each users system.

Solstice Bandwidth Reservation Protocol Management Tools 3

This chapter describes various tools to help monitor and maintain Solstice Bandwidth Reservation Protocol.

RSVP Logging Facility

Solstice Bandwidth Reservation Protocol creates a set of log files which contain information to assist in the debugging of RSVP problems. By default, these are stored in the directory `/var/opt/SUNWconn/rsvp/log`.

Logging information is available using `rsvplay` and `syslog` in addition to the console window. Refer to "Decoding and Displaying the rsvpd Log Files" on page 21 for more information.

Trace level setting and trace activation are done through an administrative message sent on the API socket.

The daemon traces events in non-human readable format. The `rsvplay` management tool converts this format into a readable form according to the preferences set in the message catalog, `rsvp.cat` (see "Localizing Message Output" on page 37).

Recommended Logging Levels

Level of log information
Determines the amount of information logged by the `rsvpd` daemon. The parameter *l* may be changed in the `rsvpd.conf` file (see “log” on page 9) or by using the management tool `rsvpadm` (see “Setting the Logging Levels and Trace Output” on page 19).

Type of log information
Determines the type of information that the `rsvpd` daemon will log. The parameter *m* may be changed in the `rsvpd.conf` file or by using the management tool `rsvpadm`. This is a hexadecimal value, expressed in the form *0Xdd*.

The following values are useful settings for the level and type of log information.

Table 3-1 Useful Settings for the Log Level and Mask

l, m	Description
3, 0X	Captures critical logs and error logs only, for example, if the daemon cannot find or create files that it needs or if any system calls fail.
6, 0X	Captures informational, warning and notice logs. Use this level for more detailed information. For example, it will display the current working directory of the daemon and whether UDP encapsulation is being used. This will assist in solving configuration problems.
7, 7X	Captures debug logs and most RSVP events, including the transmission and reception of API calls and RSVP frames. Any Routing problems will also be identified. Use this level for debugging daemon problems and tracing interoperability problems with other RSVP implementations or routers.
8, 0XFF	Captures the hexadecimal dumps of structures and internal events, for example the expiration of timers. Use this level for detailed debugging information.

Each increment automatically includes the lesser values.

Note – If the above levels are changed using `rsvpadm`, the logging output may be redirected to the console.

Errors

Error and logging information is available using `rsvplay` and `syslog` in addition to the console window. Refer to “Decoding and Displaying the rsvpd Log Files” on page 21 for more information.

Setting the Logging Levels and Trace Output

Use `rsvpadm` to set the debug and mask logging levels and control the trace output.

Synopsis

```
/opt/SUNWconn/rsvp/bin/rsvpadm -l level -m mask
```

Description

`rsvpadm` changes the debug level and mask values from the command line without restarting the `rsvpd` daemon. Further options are available from the command interpreter. For example, the debug output from the `rsvpd` daemon may be redirected to `stdout`.

Arguments

`-l` <*level*>

This changes the amount of information logged by the `rsvpd` daemon. Permitted values are:

2 critical conditions

3 error conditions

4 warning conditions

5 normal but significant conditions

6 informational

7 debug-level messages

8 include hex dumps of interface events

Each level automatically includes the previous values. The default is 3.

`-m` <*mask*>

A hexadecimal value of the form 0xdd. This changes the type of information that the rsvpd daemon will log. Permitted values are any added combination of:

0X01 Log packets sent and received

0X02 Dump data structures

0X04 Log API, kernel reservation events

0X08 Log routes

0X10 Multicast RSVP state (to dmap)

0X20 Timer events

0XFF All the above

For example, to dump data structures and log API and kernel reservation events, the value would be 0X06. The default value is 0XFF.

`-h`

Displays help information.

Command Interpreter Options

To invoke the command interpreter, start `rsvpadm` with no parameters.

Table 3-2 Command Interpreter Options

Command	**Function**
`help or ?`	Displays the available commands
`quit`	Exits the command interpreter
`level`	Changes the value for the logging level
`mask`	Changes the value for the logging mask
`trace`	Captures the log output and direct it to `stdout`
`notrace`	Stops the log output to `stdout`

See Also

`rsvpd.conf` (page 7), `rsvpd` (page 13)

Decoding and Displaying the `rsvpd` Log Files

Use `rsvplay` to decode and display the `rsvpd` log files.

Synopsis

```
/opt/SUNWconn/rsvp/bin/rsvplay -h -d log file location
```

Description

The log files generated by `rsvpd` are in a binary form and require `rsvplay` to decode them. The output from `rsvplay` can be localized by installing the appropriate catalog, `rsvp.cat`.

`rsvpd.log` contains the number of the current log file and the log files themselves. They are numbered from `rsvpd.log.0`, through to `rsvpd.log.`*N* where *N* is the maximum log file rank. The file size, the maximum rank and the log file location are all defined in `rsvpd.conf`.

`rsvplay` uses `rsvpd.log` to obtain log information and will terminate if it cannot be found. The `rsvplay` program can be configured to start logfile decoding at a specific file by editing the `rsvpd.log` file. For example:

```
Current log file is /home/jsmith/test/rsvp/rsvpd.log.i
```

where *i* is the log file number to display.

`rsvplay` decodes and displays each log file incrementally, starting with the file defined in `rsvpd.log`. Once the largest numbered log file has been displayed, it will wrap back to `rsvpd.log.0` and continue the logging cycle. The log file entries run in chronological order, displaying the latest file last. If a log file is missing from the sequence, a warning message is displayed and `rsvplay` moves to the next log file.

Note – The rsvpd daemon may also log critical messages using syslog(3), for example if the standard log file mechanism has failed. The daemon uses the message prefix 'rsvp' and logs at level LOG_CRIT. The default facility it logs to is LOG_LOCAL0, which may be configured in rsvpd.conf. Therefore, the syslog entries should be checked for RSVP messages when troubleshooting or debugging.

Arguments

-h

Displays the help screen

-d

Specifies the log file location. rsvplay will look in this directory for the log files. If this option is not specified, rsvplay will search the current directory, and then the default Solstice Bandwidth Reservation Protocol package location for log files. This is /var/opt/SUNWconn/rsvp/log.

See Also

rsvpstat (page 36), rsvpadm (page 19), rsvp.cat (page 37), rsvpd.conf (page 7), rsvpd (page 13).

Example Log Traces

This section examines example log files for an RSVP session between two machines on the same Ethernet segment with no routers between them. The sender machine is jamima, the receiver machine is riemann (192.0.2.10).

This information is particularly useful for developers and is viewed using rsvplay.

The log state for each of the following events is illustrated and decoded:

1. The rsvpd daemon on jamima is started.
2. The RSVP application on jamima calls the rapi_session() function.
3. The RSVP application on jamima calls the rapi_sender() function.
4. The rsvpd daemon on jamima generates a PATH refresh message.

5. The RSVP application on riemann calls the `rapi_session()` function (this trace is not visible as it is local to riemann).
6. The RSVP application on riemann calls the `rapi_reserve()` function.
7. The RSVP application on riemann calls the `rapi_release()` function.
8. The RSVP application on jamima calls the `rapi_release()` function.

A log trace is shown for each event and its significant elements are explained.

The traces are generated by the `rsvpd` daemon on the sender machine jamima. The `rsvpd.conf` file on jamima is configured as follows

```
interface default bandwidth 5000000
version 1.0
syslog 0
log /var/opt/SUNWconn/rsvp/log size 8192 max_rank 8 logging_level 7
 logging_mask 0xff
inact_to 300
```

The log level of 7 and log mask of 0XFF request a high level of logging so the traces are relatively detailed (the default values for these parameters is 3 and 0 respectively).

The* `rsvpd` *daemon on jamima is started.

```
08/01/1997 19:08:17.859 Solstice Bandwidth Reservation Protocol 1.0
Compiled: Jan 03 1997 at 15:32  Packet revision 1
Log Level=7 Debug Mask=255
08/01/1997 18:39:09.617 Config File version=1.0 Log file
location=/var/opt/SUNWconn/rsvp/log/
08/01/1997 18:39:09.620 Log file max size=8192 Log file max rank=8
08/01/1997 18:39:09.706 Current Working Directory is /var/tmp
Configured vif 0 with 192.0.2.5
BSD routing socket unavailable:
Protocol not supported
08/01/1997 18:39:09.744 Using MIB2 routing tables
Added timer event - Autostop Timer in 300 seconds
Autostop Timer in 300 seconds
08/01/1997 18:39:09.777 Physical interfaces are:
08/01/1997 18:39:09.780 0 le0      192.0.2.5  TCup
08/01/1997 18:39:09.782 1 API      0.0.0.0          NoIS
```

At this log level various configuration and status information are displayed.

The RSVP application on jamima calls the* `rapi_session()` *function.

```
08/01/1997 18:39:33.442 18:39:33.442|  API Reg
192.0.2.10:5000[17] <API pid=10860 Asid=1
```

The `rapi_session()` call specifies the following information:

- `192.0.2.10 (riemann)`. This is the destination address.
- `5000`. This is the port number.
- `17`. This is the protocol, UDP.
- `pid`. This is the Process Identifier of the RSVP application.
- `Asid`. This is the Session ID of the RSVP session opened.

The RSVP application on jamima calls the* `rapi_sender()` *function.

```
08/01/1997 18:40:09.003 18:40:08.999|  API Reg  R
192.0.2.10:5000[17] <API pid=10860 Asid=1

   Register sender: jamima/*  [Gen T=[400(410) 800B/s 0 30K]]
```

```
08/01/1997 18:40:09.018 18:40:09.018|  Rcv API  PATH
192.0.2.10:5000[17] <API ttl=/63
  PATH: Dest=riemann:5000[17]       R: 30000   Phop: 0.0.0.0
        jamima/*  [Gen T=[400(410) 800B/s 0 30K]]  riemann
                Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})

Added timer event - Refresh path 192.0.2.10:5000 in 30 seconds
Refresh path 192.0.2.10:5000 in 30 seconds
Autostop Timer in 240 seconds
Trying to route .192.0.2.10.
route out interface le0

18:40:09.073 >>>>>>>>>>  Internal STATE: <<<<<<< 59205 <<<<<<
 Dest: riemann:5000 [17]
   Refresh intervals: Path R= 30000  Resv R=0
   Sender: jamima/*  [Gen T=[400(410) 800B/s 0 30K]]  TTD 216705
      Adspec: Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})

     Phop <(API) LIH=0>  flags *E  ip_ttl 63  In_if 1  Outlist 0x1
   ------------- End of Dest state dump ---------------

08/01/1997 18:40:09.084 18:40:09.084|  Snd Raw  PATH
192.0.2.10:5000[17] 0> 192.0.2.10/62
  PATH: Dest=riemann:5000[17]       R: 30000   Phop: jamima
        jamima/*  [Gen T=[400(410) 800B/s 0 30K]]
                Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})

Added timer event - Refresh API sid= 476 in 30 seconds
Refresh path 192.0.2.10:5000 in 30 seconds
Refresh API sid= 476 in 30 seconds
Autostop Timer in 240 seconds
```

The RSVP application has specified that jamima is the sender. Additional information includes:

- `[Gen T=[400(410) 800B/s 0 30K]]`. This is the Tspec for the flow which includes a token bucket rate of 400 bytes/s, a token bucket depth of 410 Bytes, a peak data rate of 800 bytes, a minimum policed unit of 0 and a maximum packet size of 30,000 bytes.
- `192.0.2.10:5000[17]`.This is the same destination as for calling the `rapi_session` function, `riemann:5000[UDP]`.

The RSVP API has received the request to register an RSVP sender. This prompts the API to request `PATH` messages from the `rsvpd` daemon.

- `Phop` (Previous Hop value) is 0.0.0.0, which is the API.

Following this is some routing information. This indicates which interface the daemon will send `PATH` messages on when there are multiple interfaces.

Similar information is contained within the internal session segment.

- `Refresh intervals: Path R= 30000  Resv R=0`.This is the refresh rate for `PATH` and `RESV` packets.
- `TTD`. This is the IP 'Time To Die' value for this session.
- `Phop LIH`. This is the Previous Hop Logical Interface Handle number.
- Various flags which may be any of:

  '*U' (Use UDP for this `PATH` packet)

  '*L' (Match only local `RESV` requests)

  '*E' (Entry policing for Sender is on)

  '*N' (Non-RSVP hop experienced)
- `ip_ttl`. This is the IP 'Time To Live' field of the `PATH` packets.
- `In_if` and `Outlist`. This is incoming and outgoing interface information (expressed as a bit mask).

Finally, there is information describing refresh rates and timers.

The* `rsvpd` *daemon on jamima generates a* `PATH` *refresh message.

```
Activate timer event - Refresh path 192.0.2.10:5000 in 0 seconds
Refresh API sid= 476 in 0 seconds
Autostop Timer in 210 seconds
08/01/1997 18:40:39.125 18:40:39.125|T Snd Raw  PATH
192.0.2.10:5000[17] 0> 192.0.2.10/62
  PATH: Dest=riemann:5000[17]       R: 30000   Phop: jamima
        jamima/*  [Gen T=[400(410) 800B/s 0 30K]]
                Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})


18:40:39.135 >>>>>>>>>>  Internal STATE: <<<<<<< 89213 <<<<<<
 Dest: riemann:5000 [17]
   Refresh intervals: Path R= 30000  Resv R=0
   Sender: jamima/*  [Gen T=[400(410) 800B/s 0 30K]]  TTD 216705
      Adspec: Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})

     Phop <(API) LIH=0>  flags *E  ip_ttl 63  In_if 1  Outlist 0x1
   ------------- End of Dest state dump ---------------

Added timer event - Refresh path 192.0.2.10:5000 in 15 seconds
Refresh API sid= 476 in 0 seconds
Refresh path 192.0.2.10:5000 in 15 seconds
Autostop Timer in 210 seconds
Activate timer event - Refresh API sid= 476 in 0 seconds
Refresh path 192.0.2.10:5000 in 15 seconds
Autostop Timer in 210 seconds
08/01/1997 18:40:39.154 18:40:39.153|T Rcv API  PATH
192.0.2.10:5000[17] <API ttl=/63
  PATH: Dest=riemann:5000[17]       R: 30000   Phop: 0.0.0.0
        jamima/*  [Gen T=[400(410) 800B/s 0 30K]]
                Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})

Trying to route .192.0.2.10.
route out interface le0
Added timer event - Refresh API sid= 476 in 30 seconds
Refresh path 192.0.2.10:5000 in 15 seconds
Refresh API sid= 476 in 30 seconds
Autostop Timer in 210 seconds
Added timer event - Refresh path 192.0.2.10:5000 in 15 seconds
Refresh path 192.0.2.10:5000 in 15 seconds
Refresh API sid= 476 in 29 seconds
Autostop Timer in 180 seconds
```

The RSVP application on riemann calls the `rapi_reserve()` ***function.***

This results in a `RESV` packet being sent to jamima, detailed in the events below.

```
08/01/1997 18:41:20.015 18:41:20.015|  Rcv Raw  RESV
192.0.2.10:5000[17] < riemann/255
  RESV: Dest=riemann:5000[17]       R: 30000   Nhop: riemann
    FF   jamima/*    [CL T=[400(410) InfB/s 0 30K] ]

08/01/1997 18:41:20.027 18:41:20.026|  AddFlow  le0
192.0.2.10:5000[17]  F=0 => handle= 1413144
08/01/1997 18:41:20.030 18:41:20.030|  AddFilt  le0
192.0.2.10:5000[17]  F=0 => handle= 1413144
    Filter= jamima/*  Fhandle=0
08/01/1997 18:41:20.035 18:41:20.034|  API Upc  Resv Evt
192.0.2.10:5000[17] > API pid=10860 Asid=1

Added timer event - Refresh reservation 192.0.2.10:5000 in 30
seconds
Refresh path 192.0.2.10:5000 in 4 seconds
Refresh API sid= 476 in 19 seconds
Refresh reservation 192.0.2.10:5000 in 30 seconds
Autostop Timer in 170 seconds

18:41:20.053 >>>>>>>>>>  Internal STATE: <<<<<<< 129906 <<<<<<
 Dest: riemann:5000 [17]
   Refresh intervals: Path R= 30000  Resv R=30000
   Sender: jamima/*  [Gen T=[400(410) 800B/s 0 30K]]  TTD 276717
      Adspec: Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})

     Phop <(API) LIH=0>  flags *E  ip_ttl 63  In_if 1  Outlist 0x1
   FF Resv:   Iface 0 Nhop <riemann LIH=0>  TTD 287406
     Filter jamima/*    Flowspec [CL T=[400(410) InfB/s 0 30K] ]
   Kernel reservation: Iface 0 (192.0.2.5) Rhandle 159018
     Filter jamima/*    Flowspec [CL T=[400(410) InfB/s 0 30K] ]
   ------------- End of Dest state dump ---------------

Added timer event - Refresh path 192.0.2.10:5000 in 15 seconds
Refresh reservation 192.0.2.10:5000 in 10 seconds
Refresh API sid= 476 in 14 seconds
Refresh path 192.0.2.10:5000 in 15 seconds
Autostop Timer in 135 seconds
```

Log trace information is presented as follows:

- `RESV: Dest=riemann:5000[17]`. The reservation is for the flow riemann.
- `R:30000.`The refresh rate for the reserve message in milliseconds.
- `Nhop: riemann.`The Next Hop is riemann.
- `FF`. This is a Fixed Filter reservation style. This could also be 'WF' (Wildcard Filter) or 'SE' (Shared Explicit).
- `CL`. This is a Controlled Load Quality of Service reservation type. This could also be 'G'(Guaranteed Service).
- `Rspec (T=)`. The Quality of Service required (for the Token Bucket used in the previous `PATH` message).
- `AddFlow` and `AddFilt.`These are the flow and filter structures added to the session state in jamima. `le0` is the interface name on which the reservation is being made.
- `API Upc Resc Evt.`This is an upcall to the API and application to notify them of the event.

Finally, a revised internal state for this session is provided.

The RSVP application on riemann calls the `rapi_release()` *function.*

This results in the `RESV_TEAR` packet being sent to jamima, detailed in the events below.

```
08/01/1997 18:41:55.840 18:41:55.840|  Rcv Raw  RESV-TEAR
192.0.2.10:5000[17] < riemann/255
  RTEAR: Dest=riemann:5000[17]    Nhop: riemann
    FF   jamima/*    [CL T=[400(410) InfB/s 0 30K] ]


08/01/1997 18:41:55.848 18:41:55.848|  DelFlow  le0
192.0.2.10:5000[17]  F=1 => handle= 1413144
        flowspec= [CL T=[400(410) InfB/s 0 30K] ]
08/01/1997 18:41:55.853 18:41:55.853|  API Upc  Resv Evt
192.0.2.10:5000[17] > API pid=10860 Asid=1
Deleted timer event - Refresh reservation 192.0.2.10:5000 in 9
seconds

18:41:55.859 >>>>>>>>>>  Internal STATE: <<<<<<< 165480 <<<<<<
 Dest: riemann:5000 [17]
   Refresh intervals: Path R= 30000  Resv R=0
```

```
   Sender: jamima/*  [Gen T=[400(410) 800B/s 0 30K]]  TTD 306721
      Adspec: Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})

     Phop <(API) LIH=0>  flags *E  ip_ttl 63  In_if 1  Outlist 0x1
   ------------- End of Dest state dump ---------------
```

Log trace information is presented as follows:

- `Rcv Raw RESV-TEAR` is the receipt of a `RESV_TEAR` packet from riemann.
- `DelFlow le0 192.0.2.10:5000[17]` is the subsequent removal of this reservation from the session on jamima.
- `API Upc Resv Evt` is the call made first to the API, and then to the RSVP application to notify it of the receipt of the `RESV_TEAR` message.

Finally, a revised internal state for this session is provided.

The RSVP application on jamima calls the `rapi_release()` *function*

This results in the `PATH_TEAR` packet being sent to the flow destination, detailed in the events below.

```
08/01/1997 18:41:59.846 18:41:59.846|  API Cls
192.0.2.10:5000[17] 08/01/1997 18:41:59.855 Solstice RSVP 1.0
<API pid=10860 Asid=1
08/01/1997 18:41:59.878 18:41:59.877|  Rcv API  PATH-TEAR
192.0.2.10:5000[17] <API ttl=/63
  PTEAR: Dest=riemann:5000[17]       R: 30000   Phop: 0.0.0.0
        jamima/*  [Gen T=[400(410) 800B/s 0 30K]]
                Adspec( {0 hops InfB/s 0us 65535B}{G}{CL})

08/01/1997 18:41:59.911 18:41:59.911|  Snd Raw  PATH-TEAR
192.0.2.10:5000[17] 0> 192.0.2.10/62
  PTEAR: Dest=riemann:5000[17]    Phop: jamima
        jamima/*  [Gen T=[400(410) 800B/s 0 30K]]

Deleted timer event - Refresh path 192.0.2.10:5000 in 9 seconds
Deleted timer event - Refresh API sid= 476 in 9 seconds
```

Log trace information is presented as follows:

- `API Cls` is the `rapi_release()` function being received by the API.

- PTEAR: Dest=riemann:5000[17].A PATH_TEAR packet is sent to riemann. This indicates that the sending of PATH messages is to cease and the session on jamima is to be removed.

As illustrated, some timers are also deleted from the daemons timer lists.

Packet Error Code and Values

The following error codes may appear in ERROR_SPEC objects and be passed to end systems. They appear in the various RSVP packets which are exchanged between sender and receiver applications. Except where noted, these error codes appear only in ResvErr messages.

Error Code = 00: Confirmation

This code is reserved for use in the ERROR_SPEC object of a reservation confirmation message. The Error Value will also be zero.

Error Code = 01: Admission Control failure

The reservation request was rejected by Admission Control due to unavailable resources.

The 16 bits of the Error Value field are `ssur cccc cccc cccc`

These bits represent:

ss = 00

Low order 12 bits contain a globally-defined sub-code (values listed below).

ss = 10

Low order 12 bits contain an organization-specific sub-code. RSVP is not expected to be able to interpret this except as a numeric value.

ss = 11

Low order 12 bits contain a service-specific sub-code. RSVP is not expected to be able to interpret this except as a numeric value.

Since the traffic control mechanism might substitute a different service, this encoding may include some representation of the service in use.

u = 0

RSVP rejects the message without updating local state.

u = 1

RSVP may use message to update local state and forward the message. This means that the message is informational.

r

Reserved bit. This should be zero.

When `ssur`=0000, the following globally-defined sub-codes may appear in the `cccc cccc cccc` format:

Sub-code = 1

The delay bound cannot be met.

Sub-code = 2

Requested bandwidth is unavailable.

Sub-code = 3

MTU in flowspec larger than interface MTU.

Error Code = 02: Policy Control failure

Reservation or path message that has been rejected for administrative reasons. For example, required credentials not submitted, insufficient quota or balance, or administrative preemption. This Error Code may appear in a `PathErr` or `ResvErr` message.

Contents of the Error Value field are not currently used.

Error Code = 03: No path information for this Resv message.

No path state for this session. This `RESV` message cannot be forwarded.

Error Code = 04: No sender information for this Resv message.

There is PATH state for this session, but it does not include the sender matching some flow descriptor contained in the `RESV` message. This `RESV` message cannot be forwarded.

Error Code = 05: Conflicting reservation style

Reservation style conflicts with style(s) of existing reservation state. The Error Value field contains the low-order 16 bits of the Option Vector of the existing style with which the conflict occurred. This `RESV` message cannot be forwarded.

Error Code = 06: Unknown reservation style

Reservation style is unknown. This `RESV` message cannot be forwarded.

Error Code = 07: Conflicting dest port

Sessions for same destination address and protocol have appeared with both zero and non-zero dest port fields. This Error Code may appear in a `PathErr` or `ResvErr` message.

Error Code = 08: Ambiguous path

Sender port appears both zero and non-zero in same session in a `PATH` message. This Error Code may appear only in a `PathErr` message.

Error Code = 09: Ambiguous Filter Spec

Message contains a filter spec that matches more than one sender, but an explicit style that requires an exact match.

Error Code = 10, 11: (not currently used)

Error Code = 12: Service pre-empted

The service request defined by the `STYLE` object and the flow descriptor has been administratively pre-empted.

The 16 bits of the Error Value field are `ssur cccc cccc cccc`

The high-order bits `ssur` are as defined under Error Code 01.

The globally-defined sub-codes that may appear in the low-order 12 bits when `ssur` = 0000 are not currently used.

Error Code = 13: Unknown object class

Error Value contains 16-bit value composed of (Class-Num, C-Type) of unknown object. This error should be sent only if RSVP is going to reject the message, as determined by the high-order bits of the Class-Num. This Error Code may appear in a `PathErr` or `ResvErr` message.

Error Code = 14: Unknown object C-Type

Error Value contains 16-bit value composed of (Class-Num, C-Type) of object.

Error Code = 15-19: (not currently used)

Error Code = 20: Reserved for API

Error Value field contains an API error code, for an API error that was detected asynchronously and must be reported with a call.

Error Code = 21: Traffic Control Error

Reservation request was rejected by Traffic Control due to the format or contents of the request. This `Resv` message cannot be forwarded, and continued attempts would be futile.

For this Error Code, the 16 bits of the Error Value field are:

ss00 cccc cccc cccc

The bits `ss` are as defined under Error Code 01.

When `ss`=00, the following globally-defined sub-codes may appear in the `cccc cccc cccc` format:

Sub-code = 01

Service conflict. Attempting to merge two incompatible service requests.

Sub-code = 02

Service unsupported. Traffic control can provide neither the requested service nor an acceptable replacement.

Sub-code = 03

Bad Flowspec value. Malformed or unreasonable request.

Sub-code = 04

Bad Tspec value. Malformed or unreasonable request.

Error Code = 22: Traffic Control System error

A system error was detected and reported by the traffic control modules. The Error Value will contain a system-specific value giving more information about the error. RSVP is not expected to be able to interpret this value.

Error Code = 23: RSVP System error

The Error Value field will provide implementation-dependent information on the error. RSVP is not expected to be able to interpret this value.

Displaying the Resource Reservation Status

Use `rsvpstat` to display the resource reservation status.

Synopsis

```
/opt/SUNWconn/rsvp/bin/rsvpstat -f -p -n -h
```

Description

By default, `rsvpstat` displays the reservation status for each interface and each session for which there is a reservation. Each line contains the interface name, the reservation style, the session address and port, the sender(s), and the neighboring hop address. If there is `PATH` state but no reservation, `rsvpstat` displays '`no resv`' under the session heading.

Additional information can be displayed by using the arguments below.

If there is no `rsvpd` daemon on this node, `rsvpstat` will time out after 5 seconds.

Arguments

`-f`

Displays the flowspec for each reservation on the following line.

`-n`

Displays IP addresses in numeric form, not their hostnames.

`-p`

Displays `PATH` state for each session, following the corresponding reservation state. There will be one line for each sender, containing the session address and port, the sender template, and the neighboring hop address. If the `f` flag is also present, there will be a second line for each sender containing the sender Tspec.

`-h`

Displays help information.

See Also

`rsvpstat` (page 36), `rsvpadm` (page 19), `rsvp.cat` (page 37), `rsvpd.conf` (page 7), `rsvpd` (page 13).

Localizing Message Output

To display the message output from the management tools programs in a language other than English, use `rsvp.cat`. This is a message catalog used to localize messages from `rsvplay`, `rsvpstat`, `rsvpadm` and the API library.

To localize messages from the management tool programs, the environment variables `NLSPATH` and `LC_MESSAGES` must be set as follows:

1. Add

`/etc/opt/SUNWconn/rsvp/lib/locale/%L/LC_MESSAGES/%N.cat`

to `NLSPATH`.

2. Set `LC_MESSAGES` to the required locale.

These variables determine which message catalog is used. For example, if the above `NLSPATH` is defined then the message catalog `/etc/opt/SUNWconn/rsvp/lib/locale/C/LC_MESSAGES/rsvp.cat` is used in the default `C` locale.

If `NLSPATH` is not set or if `rsvp.cat` cannot be found then the default `C` locale message settings will be used.

Currently, only the default locale of English is supported.

If the localization of Solaris messages is required, the appropriate Solaris localization packages must also be installed.

Index