Rocky Enterprise Linux 9.2 Manual Pages on command 'systemd.socket.5'

C:\>man systemd.socket.5

SYSTEMD.SOCKET(5) systemd.socket SYSTEMD.SOCKET(5)

NAME

systemd.socket - Socket unit configuration

SYNOPSIS

socket.socket

DESCRIPTION

A unit configuration file whose name ends in ".socket" encodes information about an IPC or network socket or a file system FIFO controlled and supervised by systemd, for socket-based activation.

This man page lists the configuration options specific to this unit type. See systemd.unit(5) for the common options of all unit configuration files. The common configuration items are configured in the generic "[Unit]" and "[Install]" sections. The socket specific configuration options are configured in the "[Socket]" section.

Additional options are listed in systemd.exec(5), which define the execution environment the ExecStartPre=, ExecStartPost=, ExecStopPre= and ExecStopPost= commands are executed in, and in systemd.kill(5), which define the way the processes are terminated, and in systemd.resource-control(5), which configure resource control settings for the processes of the socket.

For each socket unit, a matching service unit must exist, describing the service to start on incoming traffic on the socket (see systemd.service(5) for more information about .service units). The name of the .service unit is by default the

same as the name of the .socket unit, but can be altered with the Service= option described below. Depending on the setting of the Accept= option described below, this .service unit must either be named like the .socket unit, but with the suffix replaced, unless overridden with Service=; or it must be a template unit named the same way. Example: a socket file foo.socket needs a matching service foo.service if Accept=no is set. If Accept=yes is set, a service template foo@.service must exist from which services are instantiated for each incoming connection.

No implicit WantedBy= or RequiredBy= dependency from the socket to the service is added. This means that the service may be started without the socket, in which case it must be able to open sockets by itself. To prevent this, an explicit Requires= dependency may be added.

Socket units may be used to implement on-demand starting of services, as well as parallelized starting of services. See the blog stories linked at the end for an introduction.

Note that the daemon software configured for socket activation with socket units needs to be able to accept sockets from systemd, either via systemd's native socket passing interface (see sd_listen_fds(3) for details) or via the traditional inetd(8)-style socket passing (i.e. sockets passed in via standard input and output, using StandardInput=socket in the service file).

All network sockets allocated through .socket units are allocated in the host's network namespace (see network_namespaces(7)). This does not mean however that the service activated by a configured socket unit has to be part of the host's network namespace as well. It is supported and even good practice to run services in their own network namespace (for example through PrivateNetwork=, see systemd.exec(5)), receiving only the sockets configured through socket-activation from the host's namespace. In such a set-up communication within the host's network namespace is only permitted through the activation sockets passed in while all sockets allocated from the service code itself will be associated with the service's own namespace, and thus possibly subject to a a much more restrictive configuration.

AUTOMATIC DEPENDENCIES

Implicit Dependencies

The following dependencies are implicitly added:

? Socket units automatically gain a Before= dependency on the service units they

activate.

? Socket units referring to file system paths (such as AF_UNIX sockets or FIFOs) implicitly gain Requires= and After= dependencies on all mount units necessary to access those paths.

? Socket units using the BindToDevice= setting automatically gain a BindsTo= and After= dependency on the device unit encapsulating the specified network interface.

Additional implicit dependencies may be added as result of execution and resource control parameters as documented in systemd.exec(5) and systemd.resource-control(5).

Default Dependencies

The following dependencies are added unless DefaultDependencies=no is set:

? Socket units automatically gain a Before= dependency on sockets.target.

? Socket units automatically gain a pair of After= and Requires= dependency on sysinit.target, and a pair of Before= and Conflicts= dependencies on shutdown.target. These dependencies ensure that the socket unit is started before normal services at boot, and is stopped on shutdown. Only sockets involved with early boot or late system shutdown should disable DefaultDependencies= option.

OPTIONS

Socket files must include a [Socket] section, which carries information about the socket or FIFO it supervises. A number of options that may be used in this section are shared with other unit types. These options are documented in systemd.exec(5) and systemd.kill(5). The options specific to the [Socket] section of socket units are the following:

ListenStream=, ListenDatagram=, ListenSequentialPacket=

Specifies an address to listen on for a stream (SOCK_STREAM), datagram (SOCK_DGRAM), or sequential packet (SOCK_SEQPACKET) socket, respectively. The address can be written in various formats:

If the address starts with a slash ("/"), it is read as file system socket in the AF_UNIX socket family.

If the address starts with an at symbol ("@"), it is read as abstract namespace socket in the AF_UNIX family. The "@" is replaced with a NUL character before

binding. For details, see unix(7).

If the address string is a single number, it is read as port number to listen on via IPv6. Depending on the value of BindIPv6Only= (see below) this might result in the service being available via both IPv6 and IPv4 (default) or just via IPv6.

If the address string is a string in the format v.w.x.y:z, it is read as IPv4 specifier for listening on an address v.w.x.y on a port z.

If the address string is a string in the format [x]:y, it is read as IPv6 address x on a port y. Note that this might make the service available via IPv4, too, depending on the BindIPv6Only= setting (see below).

If the address string is a string in the format "vsock:x:y", it is read as CID "x" on a port "y" address in the AF_VSOCK family. The CID is a unique 32-bit integer identifier in AF_VSOCK analogous to an IP address. Specifying the CID is optional, and may be set to the empty string.

Note that SOCK_SEQPACKET (i.e. ListenSequentialPacket=) is only available for AF_UNIX sockets. SOCK_STREAM (i.e. ListenStream=) when used for IP sockets refers to TCP sockets, SOCK_DGRAM (i.e. ListenDatagram=) to UDP.

These options may be specified more than once, in which case incoming traffic on any of the sockets will trigger service activation, and all listed sockets will be passed to the service, regardless of whether there is incoming traffic on them or not. If the empty string is assigned to any of these options, the list of addresses to listen on is reset, all prior uses of any of these options will have no effect.

It is also possible to have more than one socket unit for the same service when using Service=, and the service will receive all the sockets configured in all the socket units. Sockets configured in one unit are passed in the order of configuration, but no ordering between socket units is specified.

If an IP address is used here, it is often desirable to listen on it before the interface it is configured on is up and running, and even regardless of whether it will be up and running at any point. To deal with this, it is recommended to set the FreeBind= option described below.

ListenFIFO=

Specifies a file system FIFO to listen on. This expects an absolute file system

path as argument. Behavior otherwise is very similar to the ListenDatagram= directive above.

ListenSpecial=

Specifies a special file in the file system to listen on. This expects an absolute file system path as argument. Behavior otherwise is very similar to the ListenFIFO= directive above. Use this to open character device nodes as well as special files in /proc and /sys.

ListenNetlink=

Specifies a Netlink family to create a socket for to listen on. This expects a short string referring to the AF_NETLINK family name (such as audit or kobject-uevent) as argument, optionally suffixed by a whitespace followed by a multicast group integer. Behavior otherwise is very similar to the ListenDatagram= directive above.

ListenMessageQueue=

Specifies a POSIX message queue name to listen on. This expects a valid message queue name (i.e. beginning with /). Behavior otherwise is very similar to the ListenFIFO= directive above. On Linux message queue descriptors are actually file descriptors and can be inherited between processes.

ListenUSBFunction=

Specifies a USB FunctionFS[1] endpoints location to listen on, for implementation of USB gadget functions. This expects an absolute file system path of functionfs mount point as the argument. Behavior otherwise is very similar to the ListenFIFO= directive above. Use this to open the FunctionFS endpoint ep0. When using this option, the activated service has to have the USBFunctionDescriptors= and USBFunctionStrings= options set.

SocketProtocol=

Takes one of udplite or sctp. Specifies a socket protocol (IPPROTO_UDPLITE) UDP-Lite (IPPROTO_SCTP) SCTP socket respectively.

BindIPv6Only=

Takes one of default, both or ipv6-only. Controls the IPV6_V6ONLY socket option (see ipv6(7) for details). If both, IPv6 sockets bound will be accessible via both IPv4 and IPv6. If ipv6-only, they will be accessible via IPv6 only. If default (which is the default, surprise!), the system wide default setting is

used, as controlled by /proc/sys/net/ipv6/bindv6only, which in turn defaults to the equivalent of both.

Backlog=

Takes an unsigned integer argument. Specifies the number of connections to queue that have not been accepted yet. This setting matters only for stream and sequential packet sockets. See listen(2) for details. Defaults to SOMAXCONN (128).

BindToDevice=

Specifies a network interface name to bind this socket to. If set, traffic will only be accepted from the specified network interfaces. This controls the SO_BINDTODEVICE socket option (see socket(7) for details). If this option is used, an implicit dependency from this socket unit on the network interface device unit (systemd.device(5) is created. Note that setting this parameter might result in additional dependencies to be added to the unit (see above).

SocketUser=, SocketGroup=

Takes a UNIX user/group name. When specified, all AF_UNIX sockets and FIFO nodes in the file system are owned by the specified user and group. If unset (the default), the nodes are owned by the root user/group (if run in system context) or the invoking user/group (if run in user context). If only a user is specified but no group, then the group is derived from the user's default group.

SocketMode=

If listening on a file system socket or FIFO, this option specifies the file system access mode used when creating the file node. Takes an access mode in octal notation. Defaults to 0666.

DirectoryMode=

If listening on a file system socket or FIFO, the parent directories are automatically created if needed. This option specifies the file system access mode used when creating these directories. Takes an access mode in octal notation. Defaults to 0755.

Accept=

Takes a boolean argument. If true, a service instance is spawned for each incoming connection and only the connection socket is passed to it. If false,

all listening sockets themselves are passed to the started service unit, and only one service unit is spawned for all connections (also see above). This value is ignored for datagram sockets and FIFOs where a single service unit unconditionally handles all incoming traffic. Defaults to false. For performance reasons, it is recommended to write new daemons only in a way that is suitable for Accept=no. A daemon listening on an AF_UNIX socket may, but does not need to, call close(2) on the received socket before exiting. However, it must not unlink the socket from a file system. It should not invoke shutdown(2) on sockets it got with Accept=no, but it may do so for sockets it got with Accept=yes set. Setting Accept=yes is mostly useful to allow daemons designed for usage with inetd(8) to work unmodified with systemd socket activation.

For IPv4 and IPv6 connections, the REMOTE_ADDR environment variable will contain the remote IP address, and REMOTE_PORT will contain the remote port. This is the same as the format used by CGI. For SOCK_RAW, the port is the IP protocol.

Writable=

Takes a boolean argument. May only be used in conjunction with ListenSpecial=. If true, the specified special file is opened in read-write mode, if false, in read-only mode. Defaults to false.

MaxConnections=

The maximum number of connections to simultaneously run services instances for, when Accept=yes is set. If more concurrent connections are coming in, they will be refused until at least one existing connection is terminated. This setting has no effect on sockets configured with Accept=no or datagram sockets. Defaults to 64.

MaxConnectionsPerSource=

The maximum number of connections for a service per source IP address. This is very similar to the MaxConnections= directive above. Disabled by default.

KeepAlive=

Takes a boolean argument. If true, the TCP/IP stack will send a keep alive message after 2h (depending on the configuration of /proc/sys/net/ipv4/tcp_keepalive_time) for all TCP streams accepted on this

socket. This controls the SO_KEEPALIVE socket option (see socket(7) and the TCP Keepalive HOWTO[2] for details.) Defaults to false.

KeepAliveTimeSec=

Takes time (in seconds) as argument. The connection needs to remain idle before TCP starts sending keepalive probes. This controls the TCP_KEEPIDLE socket option (see socket(7) and the TCP Keepalive HOWTO[2] for details.) Defaults value is 7200 seconds (2 hours).

KeepAliveIntervalSec=

Takes time (in seconds) as argument between individual keepalive probes, if the socket option SO_KEEPALIVE has been set on this socket. This controls the TCP_KEEPINTVL socket option (see socket(7) and the TCP Keepalive HOWTO[2] for details.) Defaults value is 75 seconds.

KeepAliveProbes=

Takes an integer as argument. It is the number of unacknowledged probes to send before considering the connection dead and notifying the application layer. This controls the TCP_KEEPCNT socket option (see socket(7) and the TCP Keepalive HOWTO[2] for details.) Defaults value is 9.

NoDelay=

Takes a boolean argument. TCP Nagle's algorithm works by combining a number of small outgoing messages, and sending them all at once. This controls the TCP_NODELAY socket option (see tcp(7) Defaults to false.

Priority=

Takes an integer argument controlling the priority for all traffic sent from this socket. This controls the SO_PRIORITY socket option (see socket(7) for details.).

DeferAcceptSec=

Takes time (in seconds) as argument. If set, the listening process will be awakened only when data arrives on the socket, and not immediately when connection is established. When this option is set, the TCP_DEFER_ACCEPT socket option will be used (see tcp(7)), and the kernel will ignore initial ACK packets without any data. The argument specifies the approximate amount of time the kernel should wait for incoming data before falling back to the normal behavior of honoring empty ACK packets. This option is beneficial for protocols

where the client sends the data first (e.g. HTTP, in contrast to SMTP), because the server process will not be woken up unnecessarily before it can take any action.

If the client also uses the TCP_DEFER_ACCEPT option, the latency of the initial connection may be reduced, because the kernel will send data in the final packet establishing the connection (the third packet in the "three-way handshake").

Disabled by default.

ReceiveBuffer=, SendBuffer=

Takes an integer argument controlling the receive or send buffer sizes of this socket, respectively. This controls the SO_RCVBUF and SO_SNDBUF socket options (see socket(7) for details.). The usual suffixes K, M, G are supported and are understood to the base of 1024.

IPTOS=

Takes an integer argument controlling the IP Type-Of-Service field for packets generated from this socket. This controls the IP_TOS socket option (see ip(7) for details.). Either a numeric string or one of low-delay, throughput, reliability or low-cost may be specified.

IPTTL=

Takes an integer argument controlling the IPv4 Time-To-Live/IPv6 Hop-Count field for packets generated from this socket. This sets the IP_TTL/IPV6_UNICAST_HOPS socket options (see ip(7) and ipv6(7) for details.)

Mark=

Takes an integer value. Controls the firewall mark of packets generated by this socket. This can be used in the firewall logic to filter packets from this socket. This sets the SO_MARK socket option. See iptables(8) for details.

ReusePort=

Takes a boolean value. If true, allows multiple bind(2)s to this TCP or UDP port. This controls the SO_REUSEPORT socket option. See socket(7) for details.

SmackLabel=, SmackLabelIPIn=, SmackLabelIPOut=

Takes a string value. Controls the extended attributes "security.SMACK64", "security.SMACK64IPIN" and "security.SMACK64IPOUT", respectively, i.e. the security label of the FIFO, or the security label for the incoming or outgoing

connections of the socket, respectively. See Smack.txt[3] for details.

SELinuxContextFromNet=

Takes a boolean argument. When true, systemd will attempt to figure out the SELinux label used for the instantiated service from the information handed by the peer over the network. Note that only the security level is used from the information provided by the peer. Other parts of the resulting SELinux context originate from either the target binary that is effectively triggered by socket unit or from the value of the SELinuxContext= option. This configuration option only affects sockets with Accept= mode set to "true". Also note that this option is useful only when MLS/MCS SELinux policy is deployed. Defaults to "false".

PipeSize=

Takes a size in bytes. Controls the pipe buffer size of FIFOs configured in this socket unit. See fcntl(2) for details. The usual suffixes K, M, G are supported and are understood to the base of 1024.

MessageQueueMaxMessages=, MessageQueueMessageSize=

These two settings take integer values and control the mq_maxmsg field or the mq_msgsize field, respectively, when creating the message queue. Note that either none or both of these variables need to be set. See mq_setattr(3) for details.

FreeBind=

Takes a boolean value. Controls whether the socket can be bound to non-local IP addresses. This is useful to configure sockets listening on specific IP addresses before those IP addresses are successfully configured on a network interface. This sets the IP_FREEBIND socket option. For robustness reasons it is recommended to use this option whenever you bind a socket to a specific IP address. Defaults to false.

Transparent=

Takes a boolean value. Controls the IP_TRANSPARENT socket option. Defaults to false.

Broadcast=

Takes a boolean value. This controls the SO_BROADCAST socket option, which allows broadcast datagrams to be sent from this socket. Defaults to false.

PassCredentials=

Takes a boolean value. This controls the SO_PASSCRED socket option, which allows AF_UNIX sockets to receive the credentials of the sending process in an ancillary message. Defaults to false.

PassSecurity=

Takes a boolean value. This controls the SO_PASSSEC socket option, which allows AF_UNIX sockets to receive the security context of the sending process in an ancillary message. Defaults to false.

TCPCongestion=

Takes a string value. Controls the TCP congestion algorithm used by this socket. Should be one of "westwood", "veno", "cubic", "lp" or any other available algorithm supported by the IP stack. This setting applies only to stream sockets.

ExecStartPre=, ExecStartPost=

Takes one or more command lines, which are executed before or after the listening sockets/FIFOs are created and bound, respectively. The first token of the command line must be an absolute filename, then followed by arguments for the process. Multiple command lines may be specified following the same scheme as used for ExecStartPre= of service unit files.

ExecStopPre=, ExecStopPost=

Additional commands that are executed before or after the listening sockets/FIFOs are closed and removed, respectively. Multiple command lines may be specified following the same scheme as used for ExecStartPre= of service unit files.

TimeoutSec=

Configures the time to wait for the commands specified in ExecStartPre=, ExecStartPost=, ExecStopPre= and ExecStopPost= to finish. If a command does not exit within the configured time, the socket will be considered failed and be shut down again. All commands still running will be terminated forcibly via SIGTERM, and after another delay of this time with SIGKILL. (See KillMode= in systemd.kill(5).) Takes a unit-less value in seconds, or a time span value such as "5min 20s". Pass "0" to disable the timeout logic. Defaults to DefaultTimeoutStartSec= from the manager configuration file (see systemd-

system.conf(5)).

Service=

Specifies the service unit name to activate on incoming traffic. This setting is only allowed for sockets with Accept=no. It defaults to the service that bears the same name as the socket (with the suffix replaced). In most cases, it should not be necessary to use this option. Note that setting this parameter might result in additional dependencies to be added to the unit (see above).

RemoveOnStop=

Takes a boolean argument. If enabled, any file nodes created by this socket unit are removed when it is stopped. This applies to AF_UNIX sockets in the file system, POSIX message queues, FIFOs, as well as any symlinks to them configured with Symlinks=. Normally, it should not be necessary to use this option, and is not recommended as services might continue to run after the socket unit has been terminated and it should still be possible to communicate with them via their file system node. Defaults to off.

Symlinks=

Takes a list of file system paths. The specified paths will be created as symlinks to the AF_UNIX socket path or FIFO path of this socket unit. If this setting is used, only one AF_UNIX socket in the file system or one FIFO may be configured for the socket unit. Use this option to manage one or more symlinked alias names for a socket, binding their lifecycle together. Note that if creation of a symlink fails this is not considered fatal for the socket unit, and the socket unit may still start. If an empty string is assigned, the list of paths is reset. Defaults to an empty list.

FileDescriptorName=

Assigns a name to all file descriptors this socket unit encapsulates. This is useful to help activated services identify specific file descriptors, if multiple fds are passed. Services may use the sd_listen_fds_with_names(3) call to acquire the names configured for the received file descriptors. Names may contain any ASCII character, but must exclude control characters and ":", and must be at most 255 characters in length. If this setting is not used, the file descriptor name defaults to the name of the socket unit, including its .socket suffix.

TriggerLimitIntervalSec=, TriggerLimitBurst=

Configures a limit on how often this socket unit my be activated within a specific time interval. The TriggerLimitIntervalSec= may be used to configure the length of the time interval in the usual time units "us", "ms", "s", "min", "h", ... and defaults to 2s (See systemd.time(7) for details on the various time units understood). The TriggerLimitBurst= setting takes a positive integer value and specifies the number of permitted activations per time interval, and defaults to 200 for Accept=yes sockets (thus by default permitting 200 activations per 2s), and 20 otherwise (20 activations per 2s). Set either to 0 to disable any form of trigger rate limiting. If the limit is hit, the socket unit is placed into a failure mode, and will not be connectible anymore until restarted. Note that this limit is enforced before the service activation is enqueued.

Check systemd.exec(5) and systemd.kill(5) for more settings.

SEE ALSO

systemd(1), systemctl(1), systemd-system.conf(5), systemd.unit(5), systemd.exec(5), systemd.kill(5), systemd.resource-control(5), systemd.service(5), systemd.directives(7), sd_listen_fds(3), sd_listen_fds_with_names(3)

For more extensive descriptions see the "systemd for Developers" series: Socket Activation[4], Socket Activation, part II[5], Converting inetd Services[6], Socket Activated Internet Services and OS Containers[7].

NOTES

1. USB FunctionFS

   https://www.kernel.org/doc/Documentation/usb/functionfs.txt

2. TCP Keepalive HOWTO

   http://www.tldp.org/HOWTO/html_single/TCP-Keepalive-HOWTO/

3. Smack.txt

   https://www.kernel.org/doc/Documentation/security/Smack.txt

4. Socket Activation

   http://0pointer.de/blog/projects/socket-activation.html

5. Socket Activation, part II

   http://0pointer.de/blog/projects/socket-activation2.html

6. Converting inetd Services

http://0pointer.de/blog/projects/inetd.html

7. Socket Activated Internet Services and OS Containers

http://0pointer.de/blog/projects/socket-activated-containers.html