rsyslog: Linux log routing

Inputs, facilities, priorities, actions, rulesets and remote transport

Role in the logging pipeline

rsyslog is a syslog processing daemon. It receives events from sources such as Unix sockets, kernel messages or network inputs, applies filters and rules, and writes or forwards messages to destinations. systemd-journald is a separate component that stores structured journal entries. On many Linux systems both coexist, but the hand-off between them depends on distribution configuration.

Traditional syslog messages carry a facility and severity. Facilities indicate broad sources (for example authpriv, daemon or local0); severities range from emerg through debug. The selector *.info matches info and more severe priorities, not just messages labelled exactly info.

```
systemctl status rsyslog
logger -p local0.notice "rsyslog route test"
journalctl -t logger --since "5 minutes ago"
```

Configuration structure and ordering

Modern rsyslog supports RainerScript syntax with explicit input(), ruleset() and action() objects. Older selector/action syntax is still widely encountered. Main configuration commonly resides in /etc/rsyslog.conf with included fragments in /etc/rsyslog.d/. Rules are evaluated in order within a ruleset; a stop statement prevents subsequent rules from processing that message.

Avoid writing a broad catch-all rule before a selective stop rule unless duplication is intended. Validate configuration syntax before restarting or reloading the service.

```
sudo rsyslogd -N1
sudo grep -R "include\|action\|stop" /etc/rsyslog.conf /etc/rsyslog.d/
```

Filtering and file actions

A property-based filter can select messages by programname, hostname, syslogtag or content. A file action writes matching messages to a path. The example routes messages whose programname is myapp to a dedicated log and then stops further handling in the current ruleset. Programname values depend on the sender and input; verify the actual properties before relying on them.

Directory permissions and log rotation must be configured separately. Writing a dedicated file does not automatically create a retention policy.

```
if ($programname == "myapp") then {
    action(type="omfile" file="/var/log/myapp.log")
    stop
}
```

Forwarding to another host

Remote syslog can use UDP, TCP or TLS-protected transports. UDP is simple but offers no delivery guarantee; plain TCP improves transport reliability but does not provide encryption or peer authentication. For sensitive logs crossing untrusted networks, use TLS with certificate validation and a configured trust chain.

Queues buffer events when a destination is temporarily unavailable. Disk-assisted queues can improve resilience, but they require storage limits, monitoring and a deliberate failure policy. Do not assume a remote action is reliable merely because it uses TCP.

```
action(
  type="omfwd"
  target="logs.example.net"
  port="514"
  protocol="tcp"
  action.resumeRetryCount="-1"
  queue.type="linkedList"
  queue.size="10000"
)
```

Testing and diagnosis

Use logger to generate controlled messages, rsyslogd -N1 to validate syntax, and journalctl to inspect daemon failures. If a message is missing, trace the complete path: source, journald/syslog socket, input module, ruleset, filter, action and filesystem permissions.

Do not assume the presence of /var/log/syslog on every distribution; some systems retain events only in journald until rsyslog is installed or configured. Rotation of rsyslog output files must coordinate with the daemon's reopen behaviour.

```
sudo rsyslogd -N1
logger -t myapp -p local0.info "routing check"
sudo journalctl -u rsyslog --since today
sudo tail -n 30 /var/log/myapp.log
```

Reference scope: commands and defaults vary by distribution and software version. Consult the locally installed manual pages for exact syntax and behaviour.