# Chapter 11: Analyzing and Storing Logs:

[**Describing System Log Architecture**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Quiz: Describing System Log Architecture**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s02/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Reviewing Syslog Files**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s03/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Guided Exercise: Reviewing Syslog Files**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s04/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Reviewing System Journal Entries**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s05/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Guided Exercise: Reviewing System Journal Entries**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s06/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Preserving the System Journal**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s07/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Guided Exercise: Preserving the System Journal**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s08/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Maintaining Accurate Time**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s09/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Guided Exercise: Maintaining Accurate Time**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s10/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Lab: Analyzing and Storing Logs**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s11/9a2ef70f-4e72-42df-a498-b694b274af27)

[**Summary**](https://rha.ole.redhat.com/rha/app/courses/rh124-8.2/pages/ch11s12/9a2ef70f-4e72-42df-a498-b694b274af27)

# Abstract

|  |  |
| --- | --- |
| **Goal** | * Locate and accurately interpret logs of system events for troubleshooting purposes. |
| **Objectives** | * Describe the basic logging architecture used by Red Hat Enterprise Linux to record events. * Interpret events in relevant syslog files to troubleshoot problems or review system status. * Find and interpret entries in the system journal to troubleshoot problems or review system status. * Configure the system journal to preserve the record of events when a server is rebooted. * Maintain accurate time synchronization using NTP and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs. |
| **Sections** | * Describing System Log Architecture (and Quiz) * Reviewing Syslog Files (and Guided Exercise) * Reviewing System Journal Entries (and Guided Exercise) * Preserving the System Journal (and Guided Exercise) * Maintaining Accurate Time (and Guided Exercise) |
| **Lab** | * Analyzing and Storing Logs |

# Describing System Log Architecture

**Objectives**

After completing this section, you should be able to describe the basic logging architecture used by Red Hat Enterprise Linux to record events.

**System Logging**

Processes and the operating system kernel record a log of events that happen. These logs are used to audit the system and troubleshoot problems.

Many systems record logs of events in text files which are kept in the /var/log directory. These logs can be inspected using normal text utilities such as **less** and **tail**.

A standard logging system based on the *Syslog* protocol is built into Red Hat Enterprise Linux. Many programs use this system to record events and organize them into log files. The systemd-journald and rsyslog services handle the syslog messages in Red Hat Enterprise Linux 8.

The systemd-journald service is at the heart of the operating system event logging architecture. It collects event messages from many sources including the kernel, output from the early stages of the boot process, standard output and standard error from daemons as they start up and run, and syslog events. It then restructures them into a standard format, and writes them into a structured, indexed system journal. By default, this journal is stored on a file system that does not persist across reboots.

However, the rsyslog service reads syslog messages received by systemd-journald from the journal as they arrive. It then processes the syslog events, recording them to its log files or forwarding them to other services according to its own configuration.

The rsyslog service sorts and writes syslog messages to the log files that do persist across reboots in /var/log. The rsyslog service sorts the log messages to specific log files based on the type of program that sent each message, or *facility*, and the priority of each syslog message.

In addition to syslog message files, the /var/log directory contains log files from other services on the system. The following table lists some useful files in the /var/log directory.

**Table 11.1. Selected System Log Files**

| **Log file** | **Type of Messages Stored** |
| --- | --- |
| /var/log/messages | Most syslog messages are logged here. Exceptions include messages related to authentication and email processing, scheduled job execution, and those which are purely debugging-related. |
| /var/log/secure | Syslog messages related to security and authentication events. |
| /var/log/maillog | Syslog messages related to the mail server. |
| /var/log/cron | Syslog messages related to scheduled job execution. |
| /var/log/boot.log | Non-syslog console messages related to system startup. |

Note

Some applications do not use syslog to manage their log messages, although typically, they do place their log files in a subdirectory of /var/log. For example, the Apache Web Server saves log messages to files in a subddirectory of the /var/log directory.

References

systemd-journald.service(8), rsyslogd(8), and rsyslog.conf(5) man pages

For more information refer to the *Troubleshooting problems using log files* section in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration>

# Quiz: Describing System Log Architecture

Choose the correct answer to the following questions:

1. Which of these log files stores most syslog messages, with the exception of those that are related to authentication, mail, scheduled jobs, and debugging?

C /var/log/messages

2. Which log file stores syslog messages related to security and authentication operations in the system?

D /var/log/secure

3. Which service sorts and organizes syslog messages into files in /var/log?

A rsyslog

4. Which directory accommodates the human-readable syslog files?

D /var/log

5. Which file stores syslog messages related to the mail server?

B /var/log/maillog

6. Which file stores syslog messages related to the scheduled jobs?

A /var/log/cron

7. What file stores console messages related to system startup?

C /var/log/boot.log

# Reviewing Syslog Files

**Objectives**

After completing this section, you should be able to interpret events in relevant syslog files to troubleshoot problems or review system status.

**Logging Events to the System**

Many programs use the syslog protocol to log events to the system. Each log message is categorized by a facility (the type of message) and a priority (the severity of the message). Available facilities are documented in the rsyslog.conf(5) man page.

The following table lists the standard eight syslog priorities from highest to lowest.

**Table 11.2. Overview of Syslog Priorities**

| **Code** | **Priority** | **Severity** |
| --- | --- | --- |
| 0 | emerg | System is unusable |
| 1 | alert | Action must be taken immediately |
| 2 | crit | Critical condition |
| 3 | err | Non-critical error condition |
| 4 | warning | Warning condition |
| 5 | notice | Normal but significant event |
| 6 | info | Informational event |
| 7 | debug | Debugging-level message |

The rsyslog service uses the facility and priority of log messages to determine how to handle them. This is configured by rules in the /etc/rsyslog.conf file and any file in the /etc/rsyslog.d directory that has a file name extension of .conf. Software packages can easily add rules by installing an appropriate file in the /etc/rsyslog.d directory.

Each rule that controls how to sort syslog messages is a line in one of the configuration files. The left side of each line indicates the facility and severity of the syslog messages the rule matches. The right side of each line indicates what file to save the log message in (or where else to deliver the message). An asterisk (\*) is a wildcard that matches all values.

For example, the following line would record messages sent to the authpriv facility at any priority to the file /var/log/secure:

authpriv.\* /var/log/secure

Log messages sometimes match more than one rule in rsyslog.conf. In such cases, one message is stored in more than one log file. To limit messages stored, the key word none in the priority field indicates that no messages for the indicated facility should be stored in the given file.

Instead of logging syslog messages to a file, they can also be printed to the terminals of all logged-in users. The rsyslog.conf file has a setting to print all the syslog messages with the emerg priority to the terminals of all logged-in users.

Sample Rules of Rsyslog

#### RULES ####

# Log all kernel messages to the console.

# Logging much else clutters up the screen.

#kern.\* /dev/console

# Log anything (except mail) of level info or higher.

# Don't log private authentication messages!

\*.info;mail.none;authpriv.none;cron.none /var/log/messages

# The authpriv file has restricted access.

authpriv.\* /var/log/secure

# Log all the mail messages in one place.

mail.\* -/var/log/maillog

# Log cron stuff

cron.\* /var/log/cron

# Everybody gets emergency messages

\*.emerg :omusrmsg:\*

# Save news errors of level crit and higher in a special file.

uucp,news.crit /var/log/spooler

# Save boot messages also to boot.log

local7.\* /var/log/boot.log

Note

The syslog subsystem has many more features beyond the scope of this course. For those who wish to explore further, consult the rsyslog.conf(5) man page and the extensive HTML documentation in /usr/share/doc/rsyslog/html/index.html contained in the rsyslog-doc package, available from the AppStream repository in Red Hat Enterprise Linux 8.

**Log File Rotation**

The **logrotate** tool rotates log files to keep them from taking up too much space in the file system containing the /var/log directory. When a log file is rotated, it is renamed with an extension indicating the date it was rotated. For example, the old /var/log/messages file may become /var/log/messages-20190130 if it is rotated on 2019-01-30. Once the old log file is rotated, a new log file is created and the service that writes to it is notified.

After a certain number of rotations, typically after four weeks, the oldest log file is discarded to free disk space. A scheduled job runs the **logrotate** program daily to see if any logs need to be rotated. Most log files are rotated weekly, but **logrotate** rotates some faster, or slower, or when they reach a certain size.

Configuration of **logrotate** is not covered in this course. For more information, see the **logrotate**(8) man page.

**Analyzing a Syslog Entry**

Log messages start with the oldest message on top and the newest message at the end of the log file. The rsyslog service uses a standard format while recording entries in log files. The following example explains the anatomy of a log message in the /var/log/secure log file.

Feb 11 20:11:48 localhost sshd[1433]: Failed password for student from 172.25.0.10 port 59344 ssh2

|  |  |
| --- | --- |
|  | The time stamp when the log entry was recorded |
|  | The host from which the log message was sent |
|  | The program or process name and PID number that sent the log message |
|  | The actual message sent |

**Monitoring Logs**

Monitoring one or more log files for events is helpful to reproduce problems and issues. The **tail -f */path/to/file*** command outputs the last 10 lines of the file specified and continues to output new lines in the file as they get written.

For example, to monitor for failed login attempts, run the **tail** command in one terminal and then in another terminal, run the **ssh** command as the root user while a user tries to log in to the system.

In the first terminal, run the following **tail** command:

**[root@host ~]# tail -f /var/log/secure**

In the second terminal, run the following **ssh** command:

**[root@host ~]# ssh root@localhost**

**root@localhost's password: *redhat***

*...output omitted...*

**[root@host ~]#**

Return to the first terminal and view the logs.

*...output omitted...*

Feb 10 09:01:13 host sshd[2712]: Accepted password for root from 172.25.254.254 port 56801 ssh2

Feb 10 09:01:13 host sshd[2712]: pam\_unix(sshd:session): session opened for user root by (uid=0)

Sending Syslog Messages Manually

The **logger** command can send messages to the **rsyslog** service. By default, it sends the message to the user facility with the notice priority (user.notice) unless specified otherwise with the -p option. It is useful to test any change to the rsyslog service configuration.

To send a message to the **rsyslog** service that gets recorded in the /var/log/boot.log log file, execute the following **logger** command:

**[root@host ~]# logger -p local7.notice "Log entry created on host"**

References

logger(1), tail(1), rsyslog.conf(5), and logrotate(8) man pages

rsyslog Manual

/usr/share/doc/rsyslog/html/index.html provided by the rsyslog-doc package

For more information refer to the *Troubleshooting problems using log files* section in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration>

# Guided Exercise: Reviewing Syslog Files

In this exercise, you will reconfigure rsyslog to write specific log messages to a new file.

**Outcomes**

You should be able to configure the rsyslog service to write all log messages with the debug priority to the /var/log/messages-debug log file.

Log in to workstation as student using student as the password.

On workstation, run **lab log-configure start** to start the exercise. This script ensures that the environment is setup correctly.

**[student@workstation ~]$ lab log-configure start**

From workstation, open an SSH session to servera as student.

**[student@workstation ~]$ ssh student@servera**

*...output omitted...*

**[student@servera ~]$**

Configure rsyslog on servera to log all messages with the debug priority, or higher, for any service into the new /var/log/messages-debug log file by adding the rsyslog configuration file /etc/rsyslog.d/debug.conf.

Use the **sudo -i** command to switch to the root user. Specify student as the password for the student user if asked while running the **sudo -i** command.

**[student@servera ~]$ sudo -i**

**[sudo] password for student: student**

**[root@servera ~]#**

Create the /etc/rsyslog.d/debug.conf file with the necessary entries to redirect all log messages having the debug priority to /var/log/messages-debug. You may use the **vim /etc/rsyslog.d/debug.conf** command to create the file with the following content.

\*.debug /var/log/messages-debug

This configuration line catches syslog messages with any facility and a debug or above priority level. The rsyslog service will write the matching messages to the /var/log/messages-debug file. The wildcard (**\***) in the facility or priority fields of the configuration line indicates any facility or priority.

Restart the rsyslog service.

**[root@servera ~]# systemctl restart rsyslog**

Verify that all the log messages with the debug priority appears in the /var/log/messages-debug file.

Use the **logger** command with the -p option to generate a log message with the user facility and the debug priority.

**[root@servera ~]# logger -p user.debug "Debug Message Test"**

Use the **tail** command to view the last ten log messages from the /var/log/messages-debug file and confirm that you see the Debug Message Test message among the other log messages.

**[root@servera ~]# tail /var/log/messages-debug**

Feb 13 18:22:38 servera systemd[1]: Stopping System Logging Service...

Feb 13 18:22:38 servera rsyslogd[25176]: [origin software="rsyslogd" swVersion="8.37.0-9.el8" x-pid="25176" x-info="http://www.rsyslog.com"] exiting on signal 15.

Feb 13 18:22:38 servera systemd[1]: Stopped System Logging Service.

Feb 13 18:22:38 servera systemd[1]: Starting System Logging Service...

Feb 13 18:22:38 servera rsyslogd[25410]: environment variable TZ is not set, auto correcting this to TZ=/etc/localtime [v8.37.0-9.el8 try http://www.rsyslog.com/e/2442 ]

Feb 13 18:22:38 servera systemd[1]: Started System Logging Service.

Feb 13 18:22:38 servera rsyslogd[25410]: [origin software="rsyslogd" swVersion="8.37.0-9.el8" x-pid="25410" x-info="http://www.rsyslog.com"] start

**Feb 13 18:27:58 servera student[25416]: Debug Message Test**

Exit both the root and student users' shells on servera to return to the student user's shell on workstation.

**[root@servera ~]# exit**

logout

**[student@servera ~]$ exit**

logout

Connection to servera closed.

**[student@workstation ~]$**

**Finish**

On workstation, run **lab log-configure finish** to complete this exercise. This script ensures that the environment is restored back to the clean state.

**[student@workstation ~]$ lab log-configure finish**

This concludes the guided exercise.

# Reviewing System Journal Entries

**Objectives**

After completing this section, you should be able to find and interpret entries in the system journal to troubleshoot problems or review system status.

**Finding Events**

The systemd-journald service stores logging data in a structured, indexed binary file called the journal. This data includes extra information about the log event. For example, for syslog events this includes the facility and the priority of the original message.

**Important**

In Red Hat Enterprise Linux 8, the /run/log directory stores the system journal by default. The contents of the /run/log directory get cleared after a reboot. You can change this setting, and how to do so is discussed later in this chapter.

To retrieve log messages from the journal, use the **journalctl** command. You can use this command to view all messages in the journal, or to search for specific events based on a wide range of options and criteria. If you run the command as root, you have full access to the journal. Regular users can also use this command, but might be restricted from seeing certain messages.

**[root@host ~]# journalctl**

*...output omitted...*

Feb 21 17:46:25 host.lab.example.com systemd[24263]: Stopped target Sockets.

Feb 21 17:46:25 host.lab.example.com systemd[24263]: Closed D-Bus User Message Bus Socket.

Feb 21 17:46:25 host.lab.example.com systemd[24263]: Closed Multimedia System.

Feb 21 17:46:25 host.lab.example.com systemd[24263]: Reached target Shutdown.

Feb 21 17:46:25 host.lab.example.com systemd[24263]: Starting Exit the Session...

Feb 21 17:46:25 host.lab.example.com systemd[24268]: pam\_unix(systemd-user:session): session c>

Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopped User Manager for UID 1001.

Feb 21 17:46:25 host.lab.example.com systemd[1]: user-runtime-dir@1001.service: Unit not neede>

Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopping /run/user/1001 mount wrapper...

Feb 21 17:46:25 host.lab.example.com systemd[1]: Removed slice User Slice of UID 1001.

Feb 21 17:46:25 host.lab.example.com systemd[1]: Stopped /run/user/1001 mount wrapper.

Feb 21 17:46:36 host.lab.example.com sshd[24434]: Accepted publickey for root from 172.25.250.>

Feb 21 17:46:37 host.lab.example.com systemd[1]: Started Session 20 of user root.

Feb 21 17:46:37 host.lab.example.com systemd-logind[708]: New session 20 of user root.

Feb 21 17:46:37 host.lab.example.com sshd[24434]: pam\_unix(sshd:session): session opened for u>

Feb 21 18:01:01 host.lab.example.com CROND[24468]: (root) CMD (run-parts /etc/cron.hourly)

Feb 21 18:01:01 host.lab.example.com run-parts[24471]: (/etc/cron.hourly) starting 0anacron

Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished 0anacron

**lines 1464-1487/1487 (END) q**

The **journalctl** command highlights important log messages: messages at notice or warning priority are in bold text while messages at the error priority or higher are in red text.

The key to successfully using the journal for troubleshooting and auditing is to limit journal searches to show only relevant output.

By default, **journalctl -n** shows the last 10 log entries. You can adjust this with an optional argument that specifies how many log entries to display. For the last five log entries, run the following **journalctl** command:

**[root@host ~]# journalctl -n 5**

Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:01:01 +07. --

*...output omitted...*

Feb 21 17:46:37 host.lab.example.com systemd-logind[708]: New session 20 of user root.

Feb 21 17:46:37 host.lab.example.com sshd[24434]: pam\_unix(sshd:session): session opened for u>

Feb 21 18:01:01 host.lab.example.com CROND[24468]: (root) CMD (run-parts /etc/cron.hourly)

Feb 21 18:01:01 host.lab.example.com run-parts[24471]: (/etc/cron.hourly) starting 0anacron

Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished 0anacron

**lines 1-6/6 (END) q**

Similar to the **tail -f** command, the **journalctl -f** command outputs the last 10 lines of the system journal and continues to output new journal entries as they get written to the journal. To exit the **journalctl -f** process, use the **Ctrl**+**C** key combination.

**[root@host ~]# journalctl -f**

Logs begin at Wed 2019-02-20 16:01:17 +07. --

*...output omitted...*

Feb 21 18:01:01 host.lab.example.com run-parts[24477]: (/etc/cron.hourly) finished 0anacron

Feb 21 18:22:42 host.lab.example.com sshd[24437]: Received disconnect from 172.25.250.250 port 48710:11: disconnected by user

Feb 21 18:22:42 host.lab.example.com sshd[24437]: Disconnected from user root 172.25.250.250 port 48710

Feb 21 18:22:42 host.lab.example.com sshd[24434]: pam\_unix(sshd:session): session closed for user root

Feb 21 18:22:42 host.lab.example.com systemd-logind[708]: Session 20 logged out. Waiting for processes to exit.

Feb 21 18:22:42 host.lab.example.com systemd-logind[708]: Removed session 20.

Feb 21 18:22:43 host.lab.example.com sshd[24499]: Accepted publickey for root from 172.25.250.250 port 48714 ssh2: RSA SHA256:1UGybTe52L2jzEJa1HLVKn9QUCKrTv3ZzxnMJol1Fro

Feb 21 18:22:44 host.lab.example.com systemd-logind[708]: New session 21 of user root.

Feb 21 18:22:44 host.lab.example.com systemd[1]: Started Session 21 of user root.

Feb 21 18:22:44 host.lab.example.com sshd[24499]: pam\_unix(sshd:session): session opened for user root by (uid=0)

**^C**

**[root@host ~]#**

To help troubleshoot problems, you might want to filter the output of the journal based on the priority of the journal entries. The **journalctl -p** takes either the name or the number of a priority level and shows the journal entries for entries at that priority and above. The **journalctl** command understands the debug, info, notice, warning, err, crit, alert, and emerg priority levels.

Run the following **journalctl** command to list journal entries at the err priority or higher:

**[root@host ~]# journalctl -p err**

Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:01:01 +07. --

*..output omitted...*

Feb 20 16:01:17 host.lab.example.com kernel: Detected CPU family 6 model 13 stepping 3

Feb 20 16:01:17 host.lab.example.com kernel: Warning: Intel Processor - this hardware has not undergone testing by Red Hat and might not be certif>

Feb 20 16:01:20 host.lab.example.com smartd[669]: DEVICESCAN failed: glob(3) aborted matching pattern /dev/discs/disc\*

Feb 20 16:01:20 host.lab.example.com smartd[669]: In the system's table of devices NO devices found to scan

**lines 1-5/5 (END) q**

When looking for specific events, you can limit the output to a specific time frame. The **journalctl** command has two options to limit the output to a specific time range, the --since and --until options. Both options take a time argument in the format *"YYYY-MM-DD hh:mm:ss"* (the double-quotes are required to preserve the space in the option). If the date is omitted, the command assumes the current day, and if the time is omitted, the command assumes the whole day starting at 00:00:00. Both options take yesterday, today, and tomorrow as valid arguments in addition to the date and time field.

Run the following **journalctl** command to list all journal entries from today's records.

**[root@host ~]# journalctl --since today**

Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:31:14 +07. --

*...output omitted...*

Feb 21 18:22:44 host.lab.example.com systemd-logind[708]: New session 21 of user root.

Feb 21 18:22:44 host.lab.example.com systemd[1]: Started Session 21 of user root.

Feb 21 18:22:44 host.lab.example.com sshd[24499]: pam\_unix(sshd:session): session opened for user root by (uid=0)

Feb 21 18:31:13 host.lab.example.com systemd[1]: Starting dnf makecache...

Feb 21 18:31:14 host.lab.example.com dnf[24533]: Red Hat Enterprise Linux 8.0 AppStream (dvd) 637 kB/s | 2.8 kB 00:00

Feb 21 18:31:14 host.lab.example.com dnf[24533]: Red Hat Enterprise Linux 8.0 BaseOS (dvd) 795 kB/s | 2.7 kB 00:00

Feb 21 18:31:14 host.lab.example.com dnf[24533]: Metadata cache created.

Feb 21 18:31:14 host.lab.example.com systemd[1]: Started dnf makecache.

**lines 533-569/569 (END) q**

Run the following **journalctl** command to list all journal entries ranging from 2019-02-10 20:30:00 to 2019-02-13 12:00:00.

**[root@host ~]# journalctl --since "2019-02-10 20:30:00" \**

**--until "2019-02-13 12:00:00"**

*...output omitted...*

You can also specify all entries since a time relative to the present. For example, to specify all entries in the last hour, you can use the following command:

**[root@host ~]# journalctl --since "-1 hour"**

*...output omitted...*

**Note**

You can use other, more sophisticated time specifications with the --since and --until options. For some examples, see the systemd.time(7) man page.

In addition to the visible content of the journal, there are fields attached to the log entries that can only be seen when verbose output is turned on. Any displayed extra field can be used to filter the output of a journal query. This is useful to reduce the output of complex searches for certain events in the journal.

**[root@host ~]# journalctl -o verbose**

Logs begin at Wed 2019-02-20 16:01:17 +07, end at Thu 2019-02-21 18:31:14 +07. --

*...output omitted...*

Thu 2019-02-21 18:31:14.509128 +07...

PRIORITY=6

\_BOOT\_ID=4409bbf54680496d94e090de9e4a9e23

\_MACHINE\_ID=73ab164e278e48be9bf80e80714a8cd5

SYSLOG\_FACILITY=3

SYSLOG\_IDENTIFIER=systemd

\_UID=0

\_GID=0

CODE\_FILE=../src/core/job.c

CODE\_LINE=826

CODE\_FUNC=job\_log\_status\_message

JOB\_TYPE=start

JOB\_RESULT=done

MESSAGE\_ID=39f53479d3a045ac8e11786248231fbf

\_TRANSPORT=journal

\_PID=1

\_COMM=systemd

\_EXE=/usr/lib/systemd/systemd

\_CMDLINE=/usr/lib/systemd/systemd --switched-root --system --deserialize 18

\_CAP\_EFFECTIVE=3fffffffff

\_SELINUX\_CONTEXT=system\_u:system\_r:init\_t:s0

\_SYSTEMD\_CGROUP=/init.scope

\_SYSTEMD\_UNIT=init.scope

\_SYSTEMD\_SLICE=-.slice

UNIT=dnf-makecache.service

MESSAGE=Started dnf makecache.

\_HOSTNAME=host.lab.example.com

INVOCATION\_ID=d6f90184663f4309835a3e8ab647cb0e

\_SOURCE\_REALTIME\_TIMESTAMP=1550748674509128

**lines 32239-32275/32275 (END) q**

The following list gives the common fields of the system journal that can be used to search for lines relevant to a particular process or event.

\_COMM is the name of the command

\_EXE is the path to the executable for the process

\_PID is the PID of the process

\_UID is the UID of the user running the process

\_SYSTEMD\_UNIT is the systemd unit that started the process

More than one of the system journal fields can be combined to form a granular search query with the **journalctl** command. For example, the following **journalctl** command shows all journal entries related to the sshd.service systemd unit from a process with PID 1182.

**[root@host ~]# journalctl \_SYSTEMD\_UNIT=sshd.service \_PID=1182**

Apr 03 19:34:27 host.lab.example.com sshd[1182]: Accepted password for root from ::1 port 52778 ssh2

Apr 03 19:34:28 host.lab.example.com sshd[1182]: pam\_unix(sshd:session): session opened for user root by (uid=0)

*...output omitted...*

Note

For a list of commonly used journal fields, consult the systemd.journal-fields(7) man page.

References

journalctl(1), systemd.journal-fields(7), and systemd.time(7) man pages

For more information refer to the *Troubleshooting problems using log files* section in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration>

# Guided Exercise: Reviewing System Journal Entries

In this exercise, you will search the system journal for entries recording events that match specific criteria.

**Outcomes**

You should be able to search the system journal for entries recording events based on different criteria.

Log in to workstation as student using student as the password.

On workstation, run **lab log-query start** to start the exercise. This script ensures that the environment is setup correctly.

**[student@workstation ~]$ lab log-query start**

1. From workstation, open an SSH session to servera as student.

**[student@workstation ~]$ ssh student@servera**

*...output omitted...*

**[student@servera ~]$**

1. Use the \_PID=1 match with the **journalctl** command to display only log events originating from the systemd process running with the process identifier of 1 on servera. To quit **journalctl**, press **q**.

**[student@servera ~]$ journalctl \_PID=1**

*...output omitted...*

Feb 13 13:21:08 localhost systemd[1]: Found device /dev/disk/by-uuid/cdf61ded-534c-4bd6-b458-cab18b1a72ea.

Feb 13 13:21:08 localhost systemd[1]: Started dracut initqueue hook.

Feb 13 13:21:08 localhost systemd[1]: Found device /dev/disk/by-uuid/44330f15-2f9d-4745-ae2e-20844f22762d.

Feb 13 13:21:08 localhost systemd[1]: Reached target Initrd Root Device.

**lines 1-5/5 (END) q**

**[student@servera ~]$**

1. Note

The **journalctl** command may produce a different output on your system.

1. Use the \_UID=81 match with the **journalctl** command to display all log events originating from a system service started with the user identifier of 81 on servera. To quit **journalctl** press **q**.

**[student@servera ~]$ journalctl \_UID=81**

*...output omitted...*

Feb 22 01:29:09 servera.lab.example.com dbus-daemon[672]: [system] Activating via systemd: service name='org.freedesktop.nm\_dispatcher'>

Feb 22 01:29:09 servera.lab.example.com dbus-daemon[672]: [system] Successfully activated service 'org.freedesktop.nm\_dispatcher'

**lines 1-5/5 (END) q**

**[student@servera ~]$**

1. Use the -p warning option with the **journalctl** command to display log events with priority warning and above on servera. To quit **journalctl** press **q**.

**[student@servera ~]$ journalctl -p warning**

*...output omitted...*

Feb 13 13:21:07 localhost kernel: Detected CPU family 6 model 13 stepping 3

Feb 13 13:21:07 localhost kernel: Warning: Intel Processor - this hardware has not undergone testing by Red Hat and might not >

Feb 13 13:21:07 localhost kernel: acpi PNP0A03:00: fail to add MMCONFIG information, can't access extended PCI configuration s>

Feb 13 13:21:07 localhost rpc.statd[288]: Running as root. chown /var/lib/nfs/statd to choose different user

Feb 13 13:21:07 localhost rpc.idmapd[293]: Setting log level to 0

*...output omitted...*

Feb 13 13:21:13 servera.lab.example.com rsyslogd[1172]: environment variable TZ is not set, auto correcting this to TZ=/etc/lo>

Feb 13 14:51:42 servera.lab.example.com systemd[1]: cgroup compatibility translation between legacy and unified hierarchy sett>

Feb 13 17:15:37 servera.lab.example.com rsyslogd[25176]: environment variable TZ is not set, auto correcting this to TZ=/etc/l>

Feb 13 18:22:38 servera.lab.example.com rsyslogd[25410]: environment variable TZ is not set, auto correcting this to TZ=/etc/l>

Feb 13 18:47:55 servera.lab.example.com rsyslogd[25731]: environment variable TZ is not set, auto correcting this to TZ=/etc/l>

**lines 1-17/17 (END) q**

**[student@servera ~]$**

1. Display all log events recorded in the past 10 minutes from the current time on servera.
   1. Use the --since option with the **journalctl** command to display all log events recorded in the past 10 minutes on servera. To quit **journalctl** press **q**.

**[student@servera ~]$ journalctl --since "-10min"**

*...output omitted...*

Feb 13 22:31:01 servera.lab.example.com CROND[25890]: (root) CMD (run-parts /etc/cron.hourly)

Feb 13 22:31:01 servera.lab.example.com run-parts[25893]: (/etc/cron.hourly) starting 0anacron

Feb 13 22:31:01 servera.lab.example.com run-parts[25899]: (/etc/cron.hourly) finished 0anacron

Feb 13 22:31:41 servera.lab.example.com sshd[25901]: Bad protocol version identification 'brain' from 172.25.250.254 port 37450

Feb 13 22:31:42 servera.lab.example.com sshd[25902]: Accepted publickey for root from 172.25.250.254 port 37452 ssh2: RSA SHA2>

Feb 13 22:31:42 servera.lab.example.com systemd[1]: Started /run/user/0 mount wrapper.

Feb 13 22:31:42 servera.lab.example.com systemd[1]: Created slice User Slice of UID 0.

Feb 13 22:31:42 servera.lab.example.com systemd[1]: Starting User Manager for UID 0...

Feb 13 22:31:42 servera.lab.example.com systemd[1]: Started Session 118 of user root.

Feb 13 22:31:42 servera.lab.example.com systemd-logind[712]: New session 118 of user root.

Feb 13 22:31:42 servera.lab.example.com systemd[25906]: pam\_unix(systemd-user:session): session opened for user root by (uid=0)

*...output omitted...*

**lines 1-32/84 39% q**

**[student@servera ~]$**

1. Use the --since option and the \_SYSTEMD\_UNIT="sshd.service" match with the **journalctl** command to display all the log events originating from the sshd service recorded since 09:00:00 this morning on servera. To quit **journalctl** press **q**.

Note

You may or may not be located in the same timezone as your classroom. Check the time on servera and adjust the --since value accordingly if required.

**[student@servera ~]$ journalctl --since 9:00:00 \_SYSTEMD\_UNIT="sshd.service"**

*...output omitted...*

Feb 13 13:21:12 servera.lab.example.com sshd[727]: Server listening on 0.0.0.0 port 22.

Feb 13 13:21:12 servera.lab.example.com sshd[727]: Server listening on :: port 22.

Feb 13 13:22:07 servera.lab.example.com sshd[1238]: Accepted publickey for student from 172.25.250.250 port 50590 ssh2: RSA SH>

Feb 13 13:22:07 servera.lab.example.com sshd[1238]: pam\_unix(sshd:session): session opened for user student by (uid=0)

Feb 13 13:22:08 servera.lab.example.com sshd[1238]: pam\_unix(sshd:session): session closed for user student

Feb 13 13:25:47 servera.lab.example.com sshd[1289]: Accepted publickey for root from 172.25.250.254 port 37194 ssh2: RSA SHA25>

Feb 13 13:25:47 servera.lab.example.com sshd[1289]: pam\_unix(sshd:session): session opened for user root by (uid=0)

Feb 13 13:25:47 servera.lab.example.com sshd[1289]: pam\_unix(sshd:session): session closed for user root

Feb 13 13:25:48 servera.lab.example.com sshd[1316]: Accepted publickey for root from 172.25.250.254 port 37196 ssh2: RSA SHA25>

Feb 13 13:25:48 servera.lab.example.com sshd[1316]: pam\_unix(sshd:session): session opened for user root by (uid=0)

Feb 13 13:25:48 servera.lab.example.com sshd[1316]: pam\_unix(sshd:session): session closed for user root

Feb 13 13:26:07 servera.lab.example.com sshd[1355]: Accepted publickey for student from 172.25.250.254 port 37198 ssh2: RSA SH>

Feb 13 13:26:07 servera.lab.example.com sshd[1355]: pam\_unix(sshd:session): session opened for user student by (uid=0)

Feb 13 13:52:28 servera.lab.example.com sshd[1473]: Accepted publickey for root from 172.25.250.254 port 37218 ssh2: RSA SHA25>

Feb 13 13:52:28 servera.lab.example.com sshd[1473]: pam\_unix(sshd:session): session opened for user root by (uid=0)

*...output omitted...*

**lines 1-32 q**

**[student@servera ~]$**

1. Log out of servera.

**[student@servera ~]$ exit**

logout

Connection to servera closed.

**[student@workstation ~]$**

**Finish**

On workstation, run **lab log-query finish** to complete this exercise. This script ensures that the environment is restored back to the clean state.

**[student@workstation ~]$ lab log-query finish**

This concludes the guided exercise.

# Preserving the System Journal

**Objectives**

After completing this section, you should be able to configure the system journal to preserve the record of events when a server is rebooted.

**Storing the System Journal Permanently**

By default, the system journals are kept in the /run/log/journal directory, which means the journals are cleared when the system reboots. You can change the configuration settings of the systemd-journald service in the /etc/systemd/journald.conf file to make the journals persist across reboot.

The Storage parameter in the /etc/systemd/journald.conf file defines whether to store system journals in a volatile manner or persistently across reboot. Set this parameter to persistent, volatile, auto, or none as follows:

* persistent: stores journals in the /var/log/journal directory which persists across reboots.
* If the /var/log/journal directory does not exist, the systemd-journald service creates it.
* volatile: stores journals in the volatile /run/log/journal directory.
* As the /run file system is temporary and exists only in the runtime memory, data stored in it, including system journals, do not persist across a reboot.
* auto: if the /var/log/journal directory exists, then systemd-journald uses persistent storage, otherwise it uses volatile storage.
* This is the default action if the Storage parameter is not set.
* none: do not use any storage. All logs are dropped but log forwarding will still work as expected.

The advantage of persistent system journals is that the historic data is available immediately at boot. However, even with a persistent journal, not all data is kept forever. The journal has a built-in log rotation mechanism that triggers monthly. In addition, by default, the journals are not allowed to get larger than 10% of the file system it is on, or leave less than 15% of the file system free. These values can be tuned for both the runtime and persistent journals in /etc/systemd/journald.conf. The current limits on the size of the journal are logged when the systemd-journald process starts. The following command output shows the journal entries that reflect the current size limits:

**[user@host ~]$ journalctl | grep -E 'Runtime|System journal'**

**Feb 25 13:01:46 localhost systemd-journald[147]: Runtime journal (/run/log/journal/ae06db7da89142138408d77efea9229c) is 8.0M, max 91.4M, 83.4M free.**

**Feb 25 13:01:48 remotehost.lab.example.com systemd-journald[548]: Runtime journal (/run/log/journal/73ab164e278e48be9bf80e80714a8cd5) is 8.0M, max 91.4M, 83.4M free.**

**Feb 25 13:01:48 remotehost.lab.example.com systemd-journald[548]: System journal (/var/log/journal/73ab164e278e48be9bf80e80714a8cd5) is 8.0M, max 3.7G, 3.7G free.**

Feb 25 13:01:48 remotehost.lab.example.com systemd[1]: Starting Tell Plymouth To Write Out Runtime Data...

Feb 25 13:01:48 remotehost.lab.example.com systemd[1]: Started Tell Plymouth To Write Out Runtime Data.

**Note**

In the **grep** above, the pipe (|) symbol acts as an *or* operator. That is, **grep** matches any line containing either the Runtime string or the System journal string from the **journalctl** output. This fetches the current size limits on the volatile (Runtime) journal store as well the persistent (System) journal store.

**Configuring Persistent System Journals**

To configure the systemd-journald service to preserve system journals persistently across reboot, set Storage to persistent in the /etc/systemd/journald.conf file. Run the text editor of your choice as the superuser to edit the /etc/systemd/journald.conf file.

[Journal]

Storage=persistent

*...output omitted...*

After editing the configuration file, restart the systemd-journald service to bring the configuration changes into effect.

**[root@host ~]# systemctl restart systemd-journald**

If the systemd-journald service successfully restarts, you can see that the /var/log/journal directory is created and contains one or more subdirectories. These subdirectories have hexadecimal characters in their long names and contain \*.journal files. The \*.journal files are the binary files that store the structured and indexed journal entries.

**[root@host ~]# ls /var/log/journal**

73ab164e278e48be9bf80e80714a8cd5

**[root@host ~]# ls /var/log/journal/*73ab164e278e48be9bf80e80714a8cd5***

system.journal user-1000.journal

While the system journals persist across reboot, you get an extensive number of entries in the output of the **journalctl** command that includes entries from the current system boot as well as the previous ones. To limit the output to a specific system boot, use the -b option with the **journalctl** command. The following **journalctl** command retrieves the entries limited to the first system boot:

**[root@host ~]# journalctl -b *1***

*...output omitted...*

The following **journalctl** command retrieves the entries limited to the second system boot. The following argument is meaningful only if the system has been rebooted at least twice:

**[root@host ~]# journalctl -b 2**

The following **journalctl** command retrieves the entries limited to the current system boot:

**[root@host ~]# journalctl -b**

**Note**

When debugging a system crash with a persistent journal, it is usually required to limit the journal query to the reboot before the crash happened. The **-b** option can be accompanied by a negative number indicating how many prior system boots the output should include. For example, **journalctl -b -1** limits the output to only the previous boot.

**References**

systemd-journald.conf(5), systemd-journald(8) man pages

For more information refer to the *Troubleshooting problems using log files* section in the *Red Hat Enterprise Linux 8 Configuring basic system settings Guide* at <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration>

# Guided Exercise: Preserving the System Journal

In this exercise, you will configure the system journal to preserve its data after a reboot.

**Outcomes**

You should be able to configure the system journal to preserve its data after a reboot.

Log in to workstation as student using student as the password.

On workstation, run **lab log-preserve start** to start the exercise. This script ensures that the environment is set up correctly.

**[student@workstation ~]$ lab log-preserve start**

1. From workstation, open an SSH session to servera as student.

**[student@workstation ~]$ ssh student@servera**

*...output omitted...*

**[student@servera ~]$**

1. As the superuser, confirm that the /var/log/journal directory does not exist. Use the **ls** command to list the /var/log/journal directory contents. Use **sudo** to elevate the student user privileges. Use student as the password if asked.

**[student@servera ~]$ sudo ls /var/log/journal**

**[sudo] password for student: student**

ls: cannot access '/var/log/journal': No such file or directory

As the /var/log/journal directory does not exist, the systemd-journald service is not preserving its log data.

1. Configure the systemd-journald service on servera to preserve journals across a reboot.
   1. Uncomment the Storage=auto line in the /etc/systemd/journald.conf file and set Storage to persistent. You may use the **sudo vim /etc/systemd/journald.conf** command to edit the configuration file. Type **/ Storage=auto** from **vim** command mode to search for the Storage=auto line.

*...output omitted...*

[Journal]

Storage=persistent

*...output omitted...*

* 1. Use the **systemctl** command to restart the systemd-journald service to bring the configuration changes into effect.

**[student@servera ~]$ sudo systemctl restart systemd-journald.service**

1. Confirm that the systemd-journald service on servera preserves its journals such that the journals persist across reboots.
   1. Use the **systemctl reboot** command to restart servera.

**[student@servera ~]$ sudo systemctl reboot**

Connection to servera closed by remote host.

Connection to servera closed.

**[student@workstation ~]$**

Notice that the SSH connection was terminated as soon as you restarted the servera system.

* 1. Open an SSH session to servera again.

**[student@workstation ~]$ ssh student@servera**

*...output omitted...*

**[student@servera ~]$**

* 1. Use the **ls** command to confirm that the /var/log/journal directory exists. The /var/log/journal directory contains a subdirectory with a long hexadecimal name. The journal files are found in that directory. The subdirectory name on your system will be different.

**[student@servera ~]$ sudo ls /var/log/journal**

**[sudo] password for student: student**

73ab164e278e48be9bf80e80714a8cd5

**[student@servera ~]$ sudo ls \**

**/var/log/journal/*73ab164e278e48be9bf80e80714a8cd5***

system.journal user-1000.journal

* 1. Log out of servera.

**[student@servera ~]$ exit**

logout

Connection to servera closed.

**Finish**

On workstation, run **lab log-preserve finish** to complete this exercise. This script ensures that the environment is restored back to the clean state.

**[student@workstation ~]$ lab log-preserve finish**

This concludes the guided exercise.

# Maintaining Accurate Time

Objectives

After completing this section, you should be able to maintain accurate time synchronization using NTP and configure the time zone to ensure correct time stamps for events recorded by the system journal and logs.

Setting Local Clocks and Time Zones

Correct synchronized system time is critical for log file analysis across multiple systems. The *Network Time Protocol* (*NTP*) is a standard way for machines to provide and obtain correct time information on the Internet. A machine may get accurate time information from public NTP services on the Internet, such as the NTP Pool Project. A high-quality hardware clock to serve accurate time to local clients is another option.

The **timedatectl** command shows an overview of the current time-related system settings, including current time, time zone, and NTP synchronization settings of the system.

**[user@host ~]$ timedatectl**

Local time: Fri 2019-04-05 16:10:29 CDT

Universal time: Fri 2019-04-05 21:10:29 UTC

RTC time: Fri 2019-04-05 21:10:29

Time zone: America/Chicago (CDT, -0500)

System clock synchronized: yes

NTP service: active

RTC in local TZ: no

A database of time zones is available and can be listed with the **timedatectl list-timezones** command.

**[user@host ~]$ timedatectl list-timezones**

Africa/Abidjan

Africa/Accra

Africa/Addis\_Ababa

Africa/Algiers

Africa/Asmara

Africa/Bamako

...

Time zone names are based on the public time zone database that IANA maintains. Time zones are named based on continent or ocean, then typically but not always the largest city within the time zone region. For example, most of the US Mountain time zone is America/Denver.

Selecting the correct name can be non-intuitive in cases where localities inside the time zone have different daylight saving time rules. For example, in the USA, much of the state of Arizona (US Mountain time) does not have a daylight saving time adjustment at all and is in the time zone America/Phoenix.

The command **tzselect** is useful for identifying correct zoneinfo time zone names. It interactively prompts the user with questions about the system's location, and outputs the name of the correct time zone. It does not make any change to the time zone setting of the system.

The superuser can change the system setting to update the current time zone using the **timedatectl set-timezone** command. The following **timedatectl** command updates the current time zone to America/Phoenix.

**[root@host ~]# timedatectl set-timezone America/Phoenix**

**[root@host ~]# timedatectl**

Local time: Fri 2019-04-05 14:12:39 MST

Universal time: Fri 2019-04-05 21:12:39 UTC

RTC time: Fri 2019-04-05 21:12:39

Time zone: America/Phoenix (MST, -0700)

System clock synchronized: yes

NTP service: active

RTC in local TZ: no

**Note**

Should you need to use the Coordinated Universal Time (UTC) on a particular server, set its time zone to UTC. The **tzselect** command does not include the name of the UTC time zone. Use the **timedatectl set-timezone UTC** command to set the system's current time zone to UTC.

Use the **timedatectl set-time** command to change the system's current time. The time is specified in the *"YYYY-MM-DD hh:mm:ss"* format, where either date or time can be omitted. The following **timedatectl** command changes the time to 09:00:00.

**[root@host ~]# timedatectl set-time 9:00:00**

**[root@host ~]# timedatectl**

Local time: Fri 2019-04-05 09:00:27 MST

Universal time: Fri 2019-04-05 16:00:27 UTC

RTC time: Fri 2019-04-05 16:00:27

Time zone: America/Phoenix (MST, -0700)

System clock synchronized: yes

NTP service: active

RTC in local TZ: no

The **timedatectl set-ntp** command enables or disables NTP synchronization for automatic time adjustment. The option requires either a **true** or **false** argument to turn it on or off. The following **timedatectl** command turns on NTP synchronization.

**[root@host ~]# timedatectl set-ntp true**

**Note**

In Red Hat Enterprise Linux 8, the **timedatectl set-ntp** command will adjust whether or not chronyd NTP service is operating. Other Linux distributions might use this setting to adjust a different NTP or SNTP service.

Enabling or disabling NTP using other utilities in Red Hat Enterprise Linux, such as in the graphical GNOME Settings application, also updates this setting.

**Configuring and Monitoring Chronyd**

The chronyd service keeps the usually-inaccurate local hardware clock (RTC) on track by synchronizing it to the configured NTP servers. If no network connectivity is available, chronyd calculates the RTC clock drift, which is recorded in the driftfile specified in the /etc/chrony.conf configuration file.

By default, the chronyd service uses servers from the NTP Pool Project for the time synchronization and does not need additional configuration. It may be useful to change the NTP servers when the machine in question is on an isolated network.

The *stratum* of the NTP time source determines its quality. The stratum determines the number of hops the machine is away from a high-performance reference clock. The reference clock is a stratum 0 time source. An NTP server directly attached to it is a stratum 1, while a machine synchronizing time from the NTP server is a stratum 2 time source.

The *server* and *peer* are the two categories of time sources that you can declare in the /etc/chrony.conf configuration file. The server is one stratum above the local NTP server, and the peer is at the same stratum level. More than one server and more than one peer can be specified, one per line.

The first argument of the server line is the IP address or DNS name of the NTP server. Following the server IP address or name, a series of options for the server can be listed. It is recommended to use the iburst option, because after the service starts, four measurements are taken in a short time period for a more accurate initial clock synchronization.

The following server classroom.example.com iburst line in the /etc/chrony.conf file causes the chronyd service to use the classroom.example.com NTP time source.

# Use public servers from the pool.ntp.org project.

*...output omitted...*

server classroom.example.com iburst

*...output omitted...*

After pointing **chronyd** to the local time source, classroom.example.com, you should restart the service.

**[root@host ~]# systemctl restart chronyd**

The **chronyc** command acts as a client to the chronyd service. After setting up NTP synchronization, you should verify that the local system is seamlessly using the NTP server to synchronize the system clock using the **chronyc sources** command. For more verbose output with additional explanations about the output, use the **chronyc sources -v** command.

**[root@host ~]# chronyc sources -v**

210 Number of sources = 1

.-- Source mode '^' = server, '=' = peer, '#' = local clock.

/ .- Source state '\*' = current synced, '+' = combined , '-' = not combined,

| / '?' = unreachable, 'x' = time may be in error, '~' = time too variable.

|| .- xxxx [ yyyy ] +/- zzzz

|| / xxxx = adjusted offset,

|| Log2(Polling interval) -. | yyyy = measured offset,

|| \ | zzzz = estimated error.

|| | |

MS Name/IP address Stratum Poll Reach LastRx Last sample

===============================================================================

^\* classroom.example.com 8 6 17 23 -497ns[-7000ns] +/- 956us

The \* character in the S (Source state) field indicates that the classroom.example.com server has been used as a time source and is the NTP server the machine is currently synchronized to.

References

timedatectl(1), tzselect(8), chronyd(8), chrony.conf(5), and chronyc(1) man pages

[NTP Pool Project](http://www.pool.ntp.org/)

[Time Zone Database](http://www.iana.org/time-zones)

# Guided Exercise: Maintaining Accurate Time

In this exercise, you will adjust the time zone on a server and ensure that its system clock is synchronized with an NTP time source.

**Outcomes**

You should be able to:

Change the time zone on a server.

Configure the server to synchronize its time with an NTP time source.

Log in to workstation as student using student as the password.

On workstation, run **lab log-maintain start** to start the exercise. This script ensures that the time synchronization is disabled on the servera system to provide you with the opportunity to manually update the settings on the system and enable the time synchronization.

**[student@workstation ~]$ lab log-maintain start**

From workstation, open an SSH session to servera as student.

**[student@workstation ~]$ ssh student@servera**

*...output omitted...*

**[student@servera ~]$**

For the sake of the activity, pretend that the servera system is relocated to Haiti and so you need to update the time zone appropriately. Use **sudo** to elevate the privileges of the student user while running the **timedatectl** command to update the time zone. Use student as the password if asked.

Use the **tzselect** command to determine the appropriate time zone for Haiti.

**[student@servera ~]$ tzselect**

Please identify a location so that time zone rules can be set correctly.

Please select a continent, ocean, "coord", or "TZ".

1) Africa

2) Americas

3) Antarctica

4) Asia

5) Atlantic Ocean

6) Australia

7) Europe

8) Indian Ocean

9) Pacific Ocean

10) coord - I want to use geographical coordinates.

11) TZ - I want to specify the time zone using the Posix TZ format.

**#? 2**

Please select a country whose clocks agree with yours.

1) Anguilla 19) Dominican Republic 37) Peru

2) Antigua & Barbuda 20) Ecuador 38) Puerto Rico

3) Argentina 21) El Salvador 39) St Barthelemy

4) Aruba 22) French Guiana 40) St Kitts & Nevis

5) Bahamas 23) Greenland 41) St Lucia

6) Barbados 24) Grenada 42) St Maarten (Dutch)

7) Belize 25) Guadeloupe 43) St Martin (French)

8) Bolivia 26) Guatemala 44) St Pierre & Miquelon

9) Brazil 27) Guyana 45) St Vincent

10) Canada 28) Haiti 46) Suriname

11) Caribbean NL 29) Honduras 47) Trinidad & Tobago

12) Cayman Islands 30) Jamaica 48) Turks & Caicos Is

13) Chile 31) Martinique 49) United States

14) Colombia 32) Mexico 50) Uruguay

15) Costa Rica 33) Montserrat 51) Venezuela

16) Cuba 34) Nicaragua 52) Virgin Islands (UK)

17) Curaçao 35) Panama 53) Virgin Islands (US)

18) Dominica 36) Paraguay

**#? 28**

The following information has been given:

Haiti

Therefore TZ='America/Port-au-Prince' will be used.

Selected time is now: Tue Feb 19 00:51:05 EST 2019.

Universal Time is now: Tue Feb 19 05:51:05 UTC 2019.

Is the above information OK?

1) Yes

2) No

**#? 1**

You can make this change permanent for yourself by appending the line

TZ='America/Port-au-Prince'; export TZ

to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you

can use the /usr/bin/tzselect command in shell scripts:

America/Port-au-Prince

Notice that the preceding **tzselect** command displayed the appropriate time zone for Haiti.

Use the **timedatectl** command to update the time zone on servera to America/Port-au-Prince.

**[student@servera ~]$ sudo timedatectl set-timezone \**

**America/Port-au-Prince**

**[sudo] password for student: student**

Use the **timedatectl** command to verify that the time zone has been updated to America/Port-au-Prince.

**[student@servera ~]$ timedatectl**

Local time: Tue 2019-02-19 01:16:29 EST

Universal time: Tue 2019-02-19 06:16:29 UTC

RTC time: Tue 2019-02-19 06:16:29

**Time zone: America/Port-au-Prince (EST, -0500)**

System clock synchronized: no

NTP service: inactive

RTC in local TZ: no

Configure the chronyd service on servera to synchronize the system time with the NTP time source classroom.example.com.

Edit the /etc/chrony.conf file to specify the classroom.example.com server as the NTP time source. You may use the **sudo vim /etc/chrony.conf** command to edit the configuration file. The following output shows the configuration line you must add to the configuration file:

*...output omitted...*

server classroom.example.com iburst

*...output omitted...*

The preceding line in the /etc/chrony.conf configuration file includes the iburst option to speed up initial time synchronization.

Use the **timedatectl** command to turn on the time synchronization on servera.

**[student@servera ~]$ sudo timedatectl set-ntp yes**

The preceding **timedatectl** command activates the NTP server with the changed settings in the /etc/chrony.conf configuration file. The preceding **timedatectl** command may activate either the chronyd or the ntpd service, based on what is currently installed on the system.

Verify that the time settings on servera are currently configured to synchronize with the classroom.example.com time source in the classroom environment.

Use the **timedatectl** command to verify that the servera currently has the time synchronization enabled.

**[student@servera ~]$ timedatectl**

Local time: Tue 2019-02-19 01:52:17 EST

Universal time: Tue 2019-02-19 06:52:17 UTC

RTC time: Tue 2019-02-19 06:52:17

Time zone: America/Port-au-Prince (EST, -0500)

**System clock synchronized: yes**

NTP service: active

RTC in local TZ: no

Note

If the preceding output shows that the clock is not synchronized, wait for two seconds and re-run the **timedatectl** command. It takes a few seconds to successfully synchronize the time settings with the time source.

Use the **chronyc** command to verify that the servera system is currently synchronizing its time settings with the classroom.example.com time source.

**[student@servera ~]$ chronyc sources -v**

210 Number of sources = 1

.-- Source mode '^' = server, '=' = peer, '#' = local clock.

/ .- Source state '\*' = current synced, '+' = combined , '-' = not combined,

| / '?' = unreachable, 'x' = time may be in error, '~' = time too variable.

|| .- xxxx [ yyyy ] +/- zzzz

|| Reachability register (octal) -. | xxxx = adjusted offset,

|| Log2(Polling interval) --. | | yyyy = measured offset,

|| \ | | zzzz = estimated error.

|| | | \

MS Name/IP address Stratum Poll Reach LastRx Last sample

===============================================================================

**^\* classroom.example.com 2 6 377 62 +105us[ +143us] +/- 14ms**

Notice that the preceding output shows an asterisk (\*) in the source state (S) field for the classroom.example.com NTP time source. The asterisk indicates that the local system time is currently in successful synchronization with the NTP time source.

Log out of servera.

**[student@servera ~]$ exit**

logout

Connection to servera closed.

**[student@workstation ~]$**

**Finish**

On workstation, run **lab log-maintain finish** to complete this exercise. This script ensures that the original time zone is restored along with all the original time settings on servera.

**[student@workstation ~]$ lab log-maintain finish**

This concludes the guided exercise.

# Summary

In this chapter, you learned:

* The systemd-journald and rsyslog services capture and write log messages to the appropriate files.
* The /var/log directory contains log files.
* Periodic rotation of log files prevent them from filling up the file system space.
* The systemd journals are temporary and do not persist across reboot.
* The chronyd service helps to synchronize time settings with a time source.
* The time zone of the server can be updated based on its location.