4.3.6 Unit Files Facts

Systemd manages many functions of a Linux server including services and system states. A unit refers to any resource that systemd knows how to operate on and manage. Systemd uses *unit files* to manage these functions. Unit files are configuration files that define the actions of resources and are managed by daemons and utilities.

Examples of resources controlled by unit files are:

- Sockets
- Devices
- Mountpoints
- Automount points
- Filesystem
- Targets

This lesson covers the following topics:

- Unit File Structure
- Directory Location
- Target Units

Unit File Structure

Unit files consist of sections that specify behavior and metadata. Sections contain directives for the unit. Some section directives are environment parameters that control how a unit file functions.

Each section within the unit file is formatted as:

- [Section]
- Directive1=value
- Directive2=value

Example: iscsi.service unit file

[Unit] Description=Login and scanning of iSCSI devices
Documentation=man:iscsid (8) man:iscsiadm(8)
DefaultDependencies=no
Conflicts=shutdown.target
After=systemd-remount-fs.service network.target iscsid.service iscsiuio.service
Before=remote-fs-pre.target
Wants=remote-fs-pre.target iscsi-shutdown.service
ConditionDirectoryNotEmpty=|/var/lib/iscsi/nodes ConditionDirectoryNotEmpty=|/sys/cIass/iscsi session

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=-/usr/libexec/iscsi-mark-root-nodes
ExecStart=-/sbin/iscsiadm -m node --loginall=automatic
ExecReload=-/sbin/iscsiadm -m node --loginall=automatic

[Install] WantedBy=sysinit.target

The following table identifies some commonly used sections.

<table>
<tr><th>Section</th><th>Description</th></tr>
<tr><td>Unit Section</td><td>The Unit directives provide an overview of the unit.

Unit section directives include:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Documentation=</td><td>Lists referencing documentation for this unit or its configuration.</td></tr>
<tr><td>DefaultDependencies=</td><td>Lists 'yes' or 'no'. These are similar to implicit dependencies, but can be turned on and off by setting this option to yes or no.</td></tr>
<tr><td>Conflicts=</td><td>Lists negative requirement dependencies. If a unit has a Conflicts= setting on another unit, starting the former will stop the latter and vice versa. Note that this setting is independent of to the After= and Before= ordering dependencies.</td></tr>
</table></td></tr>
</table>

Requires=	Lists the units that must be activated for a unit to function. By default, the other units listed by a directive are activated at the same time as the unit.
After=	Lists the units to start before this unit is started.
Before=	Lists the units to start after this unit is started.
Wants=	Lists the units recommended to be in effect or started for the unit to function.

Unit-Specific Sections

Systemd categories units according to the type of resource they describe. The easiest way to determine the type of a unit is by the suffix type appended to the end of the resource name. For example, a unit file name *my.service* would be a service unit file that would describe how to manage a service.

Therefore, unit files will typically contain a unit-specific section. The following table shows a few example of unit files that have directives specific to their type:

Type	Description
Service	Describes how to manage a service or application on the server.
Socket	Describes a network or IPC socket, or a FIFO buffer that systemd uses for socket-based activation.
Mount	Defines a mountpoint on the system to be managed by systemd.
Automount	Configures a mountpoint that will be automatically mounted.

When a unit does not include a directive type, it does have unit-specific sections.

Install Sections

The install section is typically the last section in a unit file. Typically, a unit is installed, or enabled, by another unit that is started at boot. Directives in the Install section specify what happens when a unit is enabled. Directives found in the Install section include:

Option	Descriptions
WantedBy=	Specifies the unit requesting this unit to be enabled.
RequiredBy=	Specifies that another unit requires this unit to be enabled.
Also=	Specifies units to be enabled or disabled as a set.

Unit files can include one or more environment directives to set environment variables for executed processes. This is done using the **Environment=** option, often found in the unit-specific section named services, such as: **Environment="VAR1=word1 word2" VAR2=word3 "VAR3=$word 5 6"**
In this example you would have three variables "VAR1", "VAR2", "VAR3" with the values "word1 word2", "word3", "$word 5 6".

The following example shows an environment variable using in the unit-specific section named service:

```
[Service]
Type=forking
ExecStart=/usr/bin/emacs --daemon
ExecStop=/usr/bin/emacsclient --eval "(kill-emacs)"
Environment=SSH_AUTH_SOCK=%t/keyring/ssh
Restart=always
```

Environment variables can also be read from a file using **EnvironmentFile=** as shown in the following example:

```
[Service]
Type=notify
EnvironmentFile=-/etc/sysconfig/libvirtd
ExecStart=
```

Settings from these files override settings made with **Environment=**. If the same variable is set twice from these files, the files will be read in the order they are specified, and the later setting will override the earlier setting.

Directory Locations

Although each distribution may vary, unit files generally adhere to the following directory locations and rules:

- Unit files are generally kept in the /lib/systemd/system directory.

  Do not edit files in this directory. You can change a unit's function as indicated in the following bullets.

- To change a unit function, place a unit file with the preferred behavior in /etc/systemd/system. Units in this directory take precedence.
- You can override specific directives in a unit file by creating another unit file that contains snippets of the function changes in a subdirectory of the unit file.
  - To do this, you create a directory named after the unit file with a .d appended to the directory name, for example **hd0.device.d**.
  - Within this directory, you create a unit file ending with .conf. You enter directives in this file to override or extend the attributes of the unit file.
  - You can reset or override a directive by assigning an empty string to a directive in the unit file ending in .conf.
- Run-time unit definitions can the stored in /run/systemd/system. The priority of these unit files is lower than units in /etc/systemd/system, but higher than units in /lib/systemd/system. Dynamically created unit files are stored in this directory.
- High-level hostnames generated by **hostnamectl** are stored in /etc/machine-info. Static hostnames generated by **hostnamectl** are stored in /etc/hostname.

Target Units

A target unit is used to provide a synchronization point for other units when the system is booting up or changing states. A target unit can be used to group unit dependencies and to establish names for dependencies between units. It also can be used to bring the system to a new state. Other units specify their relation to targets to become tied to the target's operations.
