Networking with Wicked in SUSE® Linux Enterprise 12

Something Wicked This Way Comes

Guide

www.suse.com

Solution Guide

Server

SUSE
We adapt. You succeed.

Wicked QuickStart Guide

Abstract: Introduced with SUSE® Linux Enterprise 12, Wicked is the new network management tool for Linux, largely replacing the sysconfig package to manage the ever-more-complicated network configurations. Wicked provides network configuration as a service, enabling you to change your configuration dynamically.

This paper covers the basics of Wicked with an emphasis on providing correlations between how things were done previously and how they need to be done now.

Introduction

When S.u.S.E.[1] first introduced its Linux distribution, networking requirements were relatively simple and static. Over time networking evolved to become far more complex and dynamic. For example, automatic address configuration protocols such as DHCP or IPv6 auto-configuration entered the picture along with a plethora of new classes of network devices.

While this evolution was happening, the concepts behind managing a Linux system's network configuration didn't change much. The basic idea of storing a configuration in some files and applying it at system boot up using a collection of scripts and system programs was pretty much the same. To help cope with dynamic environments, various helper daemons were introduced, and a lot of time and effort went into ensuring that all the components played properly with each other.

Recently, new technologies have accelerated the trend toward complexity. Virtualization requires on-demand provisioning of resources, including networks. Converged networks that mix data and storage traffic on a shared link require a tighter integration between stacks that were previously mostly independent.

Today, more than 20 years after the first SUSE distribution, network configurations are very difficult to manage properly, let alone easily *(see Figure 1)*.

Modern Network Landscape

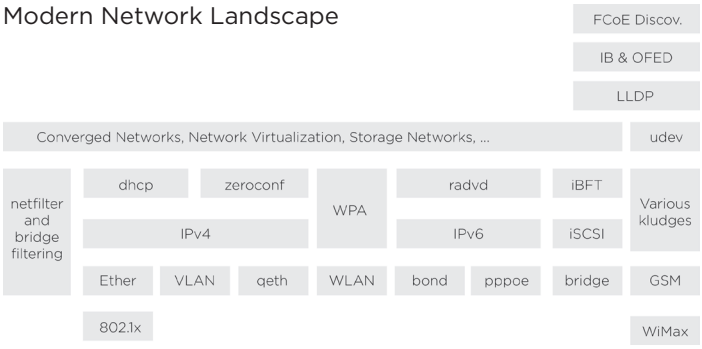

Figure 1

1 *That was indeed how we referred to our company back then. Now, of course, it's SUSE.*

This makes it fairly obvious that in today's data centers, the traditional approach of using the `ifup` scripts of yore has reached its end. During the past few years, a number of attempts have been made to implement more flexible and sophisticated network management tools with some level of success. It seemed clear, however, that something more was needed, which lead to the creation of Wicked[2].

Design Considerations

There were a number of ideas/constraints that went into the design of Wicked. We'll discuss some of the more important ones here:

Compatibility

Compatibility with the prior sysconfig package was considered extremely important. The intent was that, as much as possible, Wicked would be a "drop-in" replacement for the old `ifup` script, etc. As a result, Wicked is able to use the familiar `/etc/sysconfig/network/ifcfg-*` configuration files. If their contents contain only the functionality covered by the variables documented in the `ifcfg*` man pages, there should be no modifications necessary to work with Wicked. Along the same lines, commands such as `ifup`, `ifdown`, `ifprobe` and `netconfig` are still provided and work as expected. *See the section on "Important Commands and Tools" for more information.*

Capability

Given the need to cope with increasingly complex and dynamic configurations that drove its creation in the first place, Wicked is expected to work with a wide variety of network objects, such as Ethernet, Infiniband, Channel to Channel (CTC), Inter-User Communication Vehicle (IUCV), HiperSockets, Open System Adapters (OSA), IEEE VLANs, bridges, macvlan, macvtap, wireless (wifi) and more.

This is partly achieved by implementing Wicked in such a way as to provide "network configuration as a service." Wicked will react flexibly to network changes whether initiated by the system administrator, hypervisor, external network events, etc.

With the range of hardware platforms that SUSE Linux Enterprise runs on, Wicked had to be architecture-independent and extensible.

Usability

The target audience for Wicked was both data center staff and end users. Therefore, it needed to be straightforward to use and understand.

Finally, the intent is *not* to replace NetworkManager completely. While NetworkManager is mainly targeted at desktop/laptop users, Wicked is aimed more at servers.

Important Commands and Tools

Working with Individual Network Interfaces

COMMAND BACKWARD COMPATIBILITY

As mentioned previously, the traditional methods of working with network interfaces have been preserved. This is both for smoothing the transition from the old method and for compatibility with existing scripts that may have been written by system administrators. For example:

```
ifup eth0
ifdown wlan0
ifstatus br0
ifcheck eth1
```

These commands wind up invoking the `/usr/sbin/wicked` command "under the covers" so the functionality they provide is no different from using the `"wicked sub-command"` form described next.

2 *The name "Wicked" comes from early in development when experiments with a Representational State Transfer interface (REST) led the developers to decide against using that in the design. Humor being important when things are not going well resulted in "No REST for... the wicked."*

WICKED

The `/usr/sbin/wicked` command is the primary method for working with the various pieces of the wicked service. Both `/usr/sbin/wicked` and `/usr/sbin/ifup` are frequently referred to as the "wicked client." `/usr/sbin/wicked` has a number of sub-commands that be invoked to manage individual interfaces, such as `ifup`, `ifdown`, `show-config`, etc. For example:

```
wicked ifup eth0
wicked ifdown wlan0
wicked ifreload br0
wicked ifstatus all
```

See Appendix B for a complete list of sub-commands.

At this time, YaST® is the only tool provided that creates/modifies/deletes network interface files in `/etc/sysconfig/network`.

Working with the Network Service

COMMAND BACKWARD COMPATIBILITY

To start, stop or restart the network service, the `/sbin/rcnetwork` symbolic link is still provided, as part of the sysconfig RPM. This means that you can still issue the `rcnetwork` command with the `start`, `stop`, `restart`, etc., options. Instead of being a symbolic link to the init script at `/etc/init.d/network`, however, it now points to `/usr/sbin/service`, which in turn will invoke the appropriate `systemctl` command for systemd to execute. *See the following section on systemd for more details.*

SYSTEMD

Wicked was implemented as a group of DBus services that are integrated with systemd. So the usual `systemctl` commands will apply to Wicked.

`systemctl start network.service` (→ wicked.service)
Configures the network interfaces (and triggers wicked daemons to start).

`systemctl stop network.service` (→ wicked.service)
Unconfigures the network interfaces (but leaves the Wicked daemons running).

`systemctl restart network.service` (→ wicked.service)
Restarts the network interface configuration.

`systemctl restart wickedd.service`
Restarts Wicked daemons without reconfiguring the network interfaces.

`systemctl enable wicked.service`
This will automatically enable the wickedd.service also. Additionally, it will create a `network.service` "alias." This is so that starting, stopping, etc., the network service doesn't require knowing whether Wicked or NetworkManager will be handling the request.

`systemctl disable wicked.service`
This will automatically disable wickedd.service also. Note that disabling a service does *not* stop that service.

`systemctl show -p Id network.service`
Shows the currently enabled network service (Wicked or Network Manager).

`systemctl start wickedd.service`
Starts all Wicked daemons.

And finally, the usual "`rc*`" symbolic links for services are provided by the wicked-service package: `rcwicked`, `rcwickedd`, `rcwickedd-auto4`, `rcwickedd-dhcp4`, `rcwickedd-dhcp6`, `rcwickedd-nanny`.

Configuration Files

The format of the `/etc/sysconfig/network/ifcfg-*` files that most people are familiar with hasn't been changed. As was discussed in the "Design Considerations" section on compatibility, nearly everything should continue to work as before.

Internally, Wicked uses a structured, and much richer, representation of all configuration data. This is currently in XML, and we plan to expose all of this in a future release.

There is a variety of new configuration files shipped with wicked. Most of them should never need to be modified by the system administrator unless requested by SUSE technical support when performing debugging.

`/etc/dbus-1/system.d/`—contains the various `org.opensuse.Network.*` files provided by Wicked for its use of DBus.

`/etc/sysconfig/network/`—traditionally contains the various network interface configuration files, "hook" scripts, etc. This is still true with Wicked.

`/etc/wicked/common.xml`—contains common definitions that should be used by all applications. It is sourced/included by the other configuration files in this directory. While it could be used to enable debugging across all Wicked components, for instance, the recommendation is to put things like that in `/etc/wicked/local.xml`, which is included by `common.xml` if it exists.

`/etc/wicked/server.xml`—read by the `wickedd` server process at startup.

`/etc/wicked/client.xml`—read by the `wicked` command.

`/etc/wicked/nanny.xml`—read by the `wickedd-nanny` server process at startup.

For `wickedd`, `wicked`, and `wickedd-nanny`, if their respective XML file does not exist, the program will try to read `common.xml` directly.

Block Diagram of the Components

Wicked Components

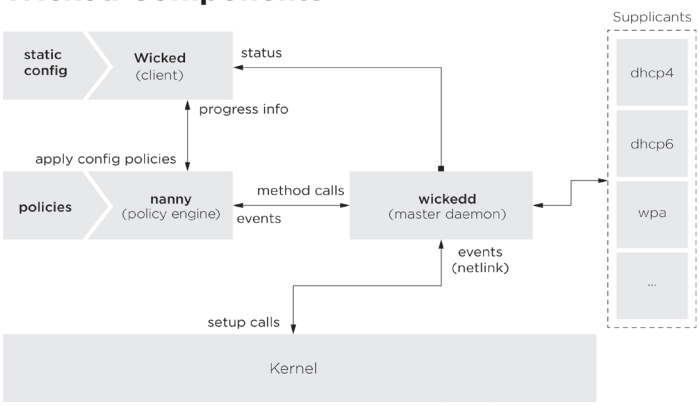

Figure 2

3 **http://en.wikipedia.org/wiki/Finite-state_machine**

Figure 2 shows a high-level view of Wicked's architecture. As in prior releases, static configuration information is kept under `/etc/sysconfig/network/`. When invoked, the Wicked client reads these configuration files and sends requests to nanny.

The nanny daemon is a policy engine that is responsible for asynchronous or unsolicited events such as hot-plugging devices and interacts with wickedd to have those requests executed. Wickedd makes calls to the kernel to actually implement the request. Status information is sent back to nanny. In turn, nanny will send progress updates back to the client as events occur. The wickedd daemon also listens for netlink events from the kernel and can respond to them dynamically. Information about these events is also passed along to nanny. In order to manage all the complexities inherent in this, wickedd was implemented as a finite-state machine (FSM).[3]

Finally, there are several "helper" services (or supplicants) for managing protocols such as DHCP (Dynamic Host Configuration Protocol), or WPA (Wi-Fi Protected Access).

Note: *The nanny framework is not enabled by default in SUSE Linux Enterprise 12. It will be enabled by default with SUSE Linux Enterprise 12 Service Pack 1. To enable it before then, see the following section.*

When nanny is *not* enabled, `/sbin/ifup` is a "one shot" command that talks directly to wickedd. In this state, Wicked will not react to hot-plugging of interfaces or carrier/link becoming available.

Enabling Nanny

Since the nanny framework is not enabled by default in SUSE Linux Enterprise 12, it must be enabled manually by the system administrator. Before doing so, it is recommended to have at least wicked-0.6.15 installed.

Nanny can be enabled either by specifying "`nanny=1`" in the installer (linuxrc) as a boot parameter or after installation by creating or modifying `/etc/wicked/local.xml` to contain the following:

```
<config>
<use-nanny>true</use-nanny>
<config>
```

Save the change and then restart the network:

```
systemctl restart wickedd.service
wicked ifup all
```

Note that `/etc/wicked/common.xml` contains:

```
<use-nanny>false</use-nanny>
```

Adding the `<use-nanny>true</use-nanny>` statement to `local.xml` will override that.

Troubleshooting

When problems arise, there are a number of ways to generate diagnostic information:

Command Line Options

There are three important debug-related command line options for the `wicked` command: `--debug`, `--log-level`, and `--log-target`. The `--debug` option specifies one or more Wicked "facilities" in a comma-separated list to be traced and reported on. The list of all available facilities can be determined by executing the

```
wicked --debug help
```

command. Three of those facilities, `mini`, `most`, and `all` will result in multiple facilities being traced. When using one of these names, you can also turn off individual facilities by prepending them with a minus sign, "-". For example,

```
wicked --debug all,-events,-socket,-objectmodel
```

will trace all facilities except events, socket, and objectmodel.

The `--log-level` option determines how verbose Wicked will be when writing out events to be logged. In order of increasing verbosity you can specify one of the following: `error, warning, notice, info, debug`. If `wicked --debug` has been executed or the `WICKED_DEBUG` environment variable has been set (see below), Wicked will automatically set the log level to "debug" for you.

The `--log-target` option can be used to direct the debugging output to either stderr or syslog. For example:

```
wicked –debug all –log-target syslog ifstatus all
```

See `man 8 wicked` *for details on what specific parameters are available for both targets.*

Environment Variables

All Wicked binaries will accept/respect the `WICKED_DEBUG` and `WICKED_LOG_LEVEL` environment variables, if specified. If `WICKED_DEBUG` is not set, a check is also made for `DEBUG=yes`. If it is set to "yes," that is equivalent to having `WICKED_DEBUG=most` specified. System-wide settings for these variables can be found in `/etc/sysconfig/network/config`.

Just as with the `--debug` command line option, `WICKED_DEBUG` can specify a single facility or a comma-separated list of facilities to be reported on or excluded.

Note that these environment variables are applied very early: before command line parsing is performed. That means that the `--debug` and `--log-level` command line options will override them.

Wicked Configuration File Options

As mentioned previously, `/etc/wicked/local.xml` can be used to turn on debugging systemwide. This is done via inserting the following XML stanza:

```
<config>
  <debug>all</debug>
</config>
```

As with the `--debug` command line option and the `WICKED_DEBUG` environment variable, the debug element in `/etc/wicked/local.xml` can specify a single facility or a comma-separated list of facilities.

The debug values set in `/etc/wicked/local.xml` will be used only if no command line debug options or environment variables are specified.

Collecting Logs

You may be asked to provide system logs by technical support. The easiest way to do that is with the `journalctl` command included with the systemd package:

```
journalctl -b -o short-precise > journal.txt
```

The `-o short-precise` option is preferred because timestamps are written to the microsecond level, which can be necessary to determine just what events happened in what order.

Appendix A

Terminology

Interface(s)—Network device(s)

Nanny—Policy engine that is responsible for asynchronous or unsolicited events such as hotplugging devices

FSM—Finite State Machine

Wicked client—The `wicked` command or any script calling `ifup`, `ifdown`, etc.

Appendix B

Wicked Sub-Commands

`ifup`—bring up one or more interfaces

`ifdown`—bring down one or more interfaces

`ifreload`—checks whether a configuration has changed and applies accordingly

`ifstatus/show`—displays detailed interface information

`ifcheck`—inspects particular interface details or state

`show-config`—reads, converts and displays all available configuration files

`show-xml`—displays the internal XML for an interface

`convert`—convert configuration files to internal XML

`getnames`—obtain different names for an interface

`xpath`—retrieve data from an XML blob

`nanny`—send configuration commands to wickedd-nanny

`arp`—check to see if an IP address is already in use on a local subnet

For details and additional parameters see `man 8 wicked`.

Appendix C

Samples of Output from Wicked Commands

```
# wicked ifstatus all
lo              up
      link:     #1, state up
      type:     loopback
      config:   compat:/etc/sysconfig/network/ifcfg-lo
      leases:   ipv4 static granted
      addr:     ipv4 127.0.0.1/8 [static]

eth0            up
      link:     #2, state up, mtu 1500
      type:     ethernet, hwaddr 52:54:00:5a:ec:a4
      config:   compat:/etc/sysconfig/network/ifcfg-eth0
      leases:   ipv4 dhcp granted
      addr:     ipv4 192.168.0.141/24 [dhcp]
      route:    ipv4 default via 192.168.0.30

# wicked ifdown eth0
eth0            device-ready

# wicked ifstatus all
eth0            device-ready
lo              up
      link:     #1, state up
      type:     loopback
      config:   compat:/etc/sysconfig/network/ifcfg-lo
      leases:   ipv4 static granted
      addr:     ipv4 127.0.0.1/8 [static]

eth0            device-unconfigured
      link:     #2, state down, mtu 1500
      type:     ethernet, hwaddr 52:54:00:5a:ec:a4

# wicked ifup eth0
eth0            up
```

Appendix D

Future Enhancements (Roadmap)

To provide a baseline, the following network tasks and objects were supported with the initial release of SUSE Linux Enterprise Server 12:

- *Setup of existing interfaces*

Ethernet, Infiniband, Channel to Channel (CTC), Inter-User Communication Vehicle (IUCV), Hipersockets, IBM Open System Adapters (OSA)

- *Creation and setup of new interfaces*

IEEE VLANs, bridge, dummy, macvlan, macvtap, Infiniband-child, Infiniband/Ethernet-bond, sit, gre, ipip

- *Creation and setup, but no driver support. These have to be started by another service after network setup is complete. For example, openvpn.*
  - Tun, tap.
- *Setup of wireless (WiFi). This is currently limited to one (1) WPA-PSK/EAP network as is the case within YaST.*
- *Address configuration*
  - Static IP addresses
  - Dynamic Host Configuration Protocol (dhcp) for both IPv4 and IPv6
  - IPv6 auto configuration
  - IPv4 zeroconf

Point-to-Point Protocol over Ethernet (pppoe) is not yet available, but is intended to be delivered as a maintenance update to SUSE Linux Enterprise Server 12.

With SUSE Linux Enterprise Server 12 Service Pack 1, the following new network objects are intended to be supported:

- *PPP (point-to-point) devices*
  - Serial modems
  - Universal Mobile Telecommunications System modems (UMTS)
  - Long-Term Evolution (LTE, frequently also referred to as 4G networking)
- *Teaming. A user space bonding variant using a teamd driver daemon*
- *Multiple routing tables when using policy routing rules*

www.suse.com

Contact your local SUSE Solutions Provider, or call SUSE at:

1 800 796 3700 U.S./Canada
1 801 861 4500 Worldwide

SUSE
Maxfeldstrasse 5
90409 Nuremberg
Germany