Linux Channel Bonding Best Practices and Recommendations
Linux Channel Bonding Best Practices and Recommendations
Before using this information and the product it supports, read the information in "Notices" on page 25.

**Edition notices**

© Copyright International Business Machines Corporation 2018. All rights reserved.

U.S. Government Users Restricted Rights — Use, duplication, or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contents

**Figures** ........................................ v  
**About this publication** ........................ vii 
Notational conventions ............................. vii

**Chapter 1. Overview** .......................... 1  
The Linux channel bonding concept .............. 1  
Linux on IBM Z with channel bonding ............ 2

**Chapter 2. Channel bonding options and recommendations** ............... 5  
Bonding modes ....................................... 5  
Active-backup mode. ................................ 6  
  Option fail_over_mac ................................ 6  
  Option primary/primary_reselect ................. 7  
802.3ad mode ....................................... 8  
  Option lacp_rate .................................. 8  
  Option lxmit_hash_policy ......................... 8  
Link monitoring .................................... 8  
MII monitor ......................................... 9  
ARP monitor ......................................... 9  
MII monitor vs. ARP monitor ...................... 10

**Chapter 3. Setting up channel bonding on different distributions** ....... 11  
Preparation ......................................... 11  
Setting up channel bonding on SLES12 ............ 12  
Setting up channel bonding on RHEL 7 ............ 15  
Setting up channel bonding on Ubuntu 16.04 .... 19

**Chapter 4. Troubleshooting** .................. 21

**References** ..................................... 23

**Notices** ........................................ 25  
Trademarks ......................................... 27  
Terms and conditions .............................. 27

© Copyright IBM Corp. 2018
# Figures

1. The Linux channel bonding concept .......................... 1
2. Typical bonding setup on Linux on Z ...................... 2
3. Example of a bonding setup with shared OSA ............... 3
4. Bonding option fail_over_mac ................................. 7
5. Example: Using the MII monitor ............................... 9
6. Example: Using the ARP monitor .............................. 10
7. MII monitor vs. ARP monitor .................................. 10
8. Example: Loading the bonding module ....................... 11
9. Example: udev rules for persistent network device ........ 11
10. Example: Using the lsqeth command to list qeth devices . 12
11. Example: Using the lsqeth command to show subchannel information ................................................. 11
12. SLES12--adding a bonding slave device ..................... 13
13. SLES12--configuring the bond IP address .................... 13
14. SLES12--choosing bonding slave devices .................... 14
15. Example: SLES12--bonding master configuration file .......... 14
16. Example: SLES 12--bonding slave configuration files .......... 15
17. Example: SLES 12--using Wicked to start a bonding interface .................................................. 15
18. RHEL 7--Adding a bond connection .......................... 16
19. RHEL 7--Editing a bond connection ......................... 17
20. RHEL 7--Adding a slave device .............................. 18
21. Example: RHEL 7--Configuration file of a bonding master .................................................. 18
22. Example: RHEL 7--Configuration file of a slave device .................................................. 19
23. Example: Ubuntu 16.04--Configuration of a bonding master .................................................. 19
24. Example: Ubuntu 16.04--Configuration of bonding slaves .................................................. 20
25. Example: Manipulating configurations in sysfs ............... 21
26. Example: Checking bonding status .......................... 21
27. Example: ip link command results, showing more details ........... 22
About this publication

This white paper describes channel bonding configurations on different Linux distributions on IBM® Z, and explains the recommended options for the Linux bonding driver.

Author

Sa Liu

Remarks

The web links referred to in this paper are up to date as of February 15, 2018.

Notational conventions

The notational conventions used throughout this white paper are described here.

Table 1. Notational conventions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Full name</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>KiB</td>
<td>kibibyte</td>
<td>$2 \times 10 \text{ byte} = 1024 \text{ byte}$</td>
</tr>
<tr>
<td>MiB</td>
<td>mebibyte</td>
<td>$2 \times 20 \text{ byte} = 1048576 \text{ byte}$</td>
</tr>
<tr>
<td>GiB</td>
<td>gibibyte</td>
<td>$2 \times 30 \text{ byte} = 1073741824 \text{ byte}$</td>
</tr>
<tr>
<td>KiB/s</td>
<td>kibibyte per second</td>
<td>$2 \times 10 \text{ byte} / \text{ second}$</td>
</tr>
<tr>
<td>MiB/s</td>
<td>mebibyte per second</td>
<td>$2 \times 20 \text{ byte} / \text{ second}$</td>
</tr>
<tr>
<td>GiB/s</td>
<td>gibibyte per second</td>
<td>$2 \times 30 \text{ byte} / \text{ second}$</td>
</tr>
</tbody>
</table>
Chapter 1. Overview

To support networking high availability on Linux systems, the Linux bonding driver provides a method to aggregate multiple network interfaces into a single logical network interface. Thereby, if a failure occurs on a single link of the network, it does not impact the network availability. It also allows the network to overcome the bandwidth limitation and balance the network traffic on multiple network devices.

This paper covers the Linux channel bonding configurations on different Linux distributions on IBM Z, and explains the recommended options for the Linux bonding driver.

The Linux channel bonding concept

The most important requirements of networking are availability and bandwidth. The modern Linux distributions allow administrators to combine multiple network interfaces together into a single logical interface called a channel bonding interface. The network interfaces that are aggregated together are called slave devices, and the bonded logical interface is called the master device. Using channel bonding, the multiple slave devices act as if there is only one master device that is working with more bandwidth and network redundancy.

The Linux bonding driver is integrated as a loadable kernel module with the latest distributions, and usually it is loaded automatically when the Linux system is booting. The bonding devices are configured by means of module options, in the way that the driver determines how the bonding devices behave. The bonding driver periodically monitors the connectivities of the devices. If link failures are detected, the bonding interface will automatically handle the situation with the other slave devices. This avoids interfering with the whole network availability.

The bonding policy of a single bonding device is specified by the driver mode. The Linux bonding driver provides several modes to support different policies of load balancing and fault toleration. In this paper, setup examples only cover active-backup mode (mode 1) and 802.3ad mode (mode 4). For more details about the driver modes and the other bonding options, refer to "Linux Ethernet Bonding Driver HOWTO" [https://www.kernel.org/doc/Documentation/networking/bonding.txt].
Linux on IBM Z with channel bonding

The IBM Z network controller Open System Adaptor (OSA) provides high standards for networking throughput and availability. Typically when setting up channel bonding interfaces on Linux on Z, the slave devices are configured with OSA devices. The OSA slave interfaces should always be in layer 2 mode.

Figure 2 shows a Linux on Z example of a channel bonding interface consisting of two slave devices: eth1 and eth2, each representing an OSA device. In layer 2 mode, a local MAC address is assigned to each slave device and registered in the OSA address table. The master device (in this example, bond0) inherits the MAC address from a slave device and carries an IP address for the whole bonding interface. For the purpose of network redundancy, the OSA devices are connected to different switches, in order to avoid single link failure on a switch.

In an industrial environment, OSA-Express® devices are often shared between systems. It is also possible to build up a network topology as shown in Figure 3 on page 3.
This figure shows that on each Linux system, one or more bonding interfaces can be defined with slaves representing OSA devices, which are shared between LPARs. Depending on the bonding mode, a bonding interface may define one of the slaves as primary slave and the other one as backup, or it may optimize the traffic load on both.

The setup of a bonding interface is the same on either a native LPAR or on a virtual machine (for example, IBM z®/VM), since control is driven by the bonding driver, which is loaded in the Linux kernel.
Chapter 2. Channel bonding options and recommendations

In this chapter, the most important bonding options are discussed and recommended in terms of a Linux on Z environment.

Bonding modes

The Linux bonding driver allows system administrators to set up bonding interfaces with different modes. A bonding mode specifies the policy indicating how bonding slaves are used during network transmission. To achieve the maximum throughput and fault toleration, it is important to choose the proper bonding mode and the corresponding options for the setup.

The current version of the bonding module supports the following bonding modes:

mode 0 (balance-rr)
Round-robin policy. Transmits packets in sequential order from the first available slave through the last. This mode provides load balancing and fault tolerance.

mode 1 (active-backup)
Active-backup policy. Establishes that only one slave in the bond is active. A different slave becomes active if, and only if, the active slave fails. The bond's MAC address is externally visible on only one port (network adapter) to avoid confusing the switch. This mode provides fault tolerance. The primary option affects the behavior of this mode.

mode 2 (balance-xor)
Transmits based on the selected transmit hash policy, which can be altered via the xmit_hash_policy option. This mode provides load balancing and fault tolerance.

mode 3 (broadcast)
Transmits everything on all slave interfaces. This mode provides fault tolerance.

mode 4 (802.3ad)
IEEE 802.3ad Dynamic link aggregation policy. Creates aggregation groups that share the same speed and duplex settings. Utilizes all slaves in the active aggregator according to the 802.3ad specification.

mode 5 (balance-tlb)
Adaptive transmit load balancing. Establishes channel bonding that does not require any special switch support. The outgoing traffic is distributed according to the current load (computed relative to the speed) on each slave. Incoming traffic is received by the current slave. If the receiving slave fails, another slave takes over the MAC address of the failed receiving slave.

mode 6 (balance-alb)
Adaptive load balancing. Includes balance-transmit load balancing plus receive-load balancing for IPv4 traffic, and does not require any special switch support. The receive-load balancing is achieved by ARP negotiation. The bonding driver intercepts the ARP replies sent by the local system on their way out and overwrites the source hardware address with the unique
hardware address of one of the slaves in the bond. Thus, different peers use different hardware addresses for the server.

For detailed information about bonding modes, refer to this document: ‘Linux Ethernet Bonding Driver HOWTO’ [https://www.kernel.org/doc/Documentation/networking/bonding.txt].

One bonding interface can only specify one mode. Which mode to choose is dependent on the network topology, requirements for the bonding behaviors, and characteristics of the slave devices. This paper only discusses mode 1 (active-backup) and mode 4 (802.3ad).

---

### Active-backup mode

Bonding mode 1 (active-backup) is very often used for fault tolerance. With this mode, only one slave device is active, and if it gets failed, another backup slave will take it over. The `fail_over_mac` option and `primary` option are essential for the active-backup mode.

#### Option `fail_over_mac`

The `fail_over_mac` option is only valid when a bonding is set with active-backup mode. There are three possible values:

- **none (0)**
  
  The default value is `none (0)`. However, this setting disables `fail_over_mac` and sets all of the slaves to the same MAC address. In a typical network environment on Linux on Z, active-backup bonds are set up with shared OSA adapters on layer 2 mode. In this case, without `fail_over_mac` the bonds may not work as expected, and packets will be directed to the inactive slaves. Therefore the value `none (0)` should not be used.

- **active (1)**
  
  With the value `active (1)`, the bond master’s MAC address is always the MAC address of the active slave. If the slave is changed during a failover, the bond’s MAC address is then changed to match the new active slave. Always set `fail_over_mac` to the value `active (1)`: this is especially important for the scenario of using shared OSA adapters (see Figure 3 on page 3).

  **Note:** During a failover, every device on the network must be updated via gratuitous ARP broadcasting the update of the bond’s MAC address. If the gratuitous ARP is lost, the communication might be disrupted. However, considering that failover rarely happens and in most cases the gratuitous ARP is reliable, this should not prevent you from using the value `active`.

- **follow (2)**
  
  The value `follow (2)` specifies the `fail_over_mac` policy that during a failover, the active slave will get the MAC address of the bond and the formerly active slave receives the newly active slave’s MAC address. This is not recommended for Linux on Z.
Option primary/primary_reselect

The option primary defines the primary slave, meaning that it should always be active if it is available. This is useful when one slave is preferred over the others (for example, when one slave has higher throughput or should serve most of the time to one Linux system).

If the primary option is set, the primary_reselect option can be used to specify a policy of selecting an active slave at failover time or when the primary slave is available again. There are three possible values:

always (0)
The primary slave becomes the active slave whenever it is available.

The value always (0) should be chosen if the primary slave is designated to the bond and the other slaves are only considered as backup during the failover. Once the primary slave is available again, it becomes active automatically.

better (1)
Depending on which is better (based on speed and duplex status), either the currently active slave remains active or the primary slave becomes active.

The value better (1) is used only if the backup slave could be better than the primary one, which should not be a normal use case.

failure (2)
Even if the primary slave is up again, it does not become the active slave at once until the current active slave fails.

The value failure (2) is used to avoid flip-flopping between the primary slave and the other slaves.

Note: In order to test the setup of the primary_reselect option, it is recommended to use the command `ip link set dev slave_device_name down/up` to simulate a link failure or recovery. It is not suitable to use a network management tool to disenslave the active slave device and enslave it again because when initially enslaved, the primary_reselect setting is ignored and the primary slave is always made active.
802.3ad mode

Bonding mode 4 (802.3ad), also known as LACP (Link Aggregation Control Protocol) mode, is used for load balancing and fault tolerance. The IEEE 802.3ad specification allows the grouping of Ethernet interfaces at the physical layer to form a single link layer interface. If a bonding interface is set to this mode, it requires that all the slave devices operate at the same speed and are duplex. In this way, the network can benefit from the aggregated bandwidth of all the slaves, and if one of the slaves is down, the whole network will not be affected.

Note: The switch should be configured to support the mode 802.3ad standard and use the LACP protocol. The 802.3ad mode only works with MII link monitor.

Option lacp_rate

The option `lacp_rate` specifies the rate at which link partners should transmit LACPDU (Link Aggregation Control Protocol Data Unit) packets in 802.3ad mode. There are two possible values:

- slow (0) Transmits every 30 seconds. This is the default value.
- fast (1) Transmits every 1 second.

It is recommended to run in fast mode for rapid detection of faults.

Option lxmit_hash_policy

The option `lxmit_hash_policy` is used to select the transmit hash policy for slave selection. There are three possible values:

- Layer 2 Uses XOR of hardware MAC addresses to generate the hash. This algorithm will place all traffic to a particular network peer on the same slave.
- Layer 2+3 Uses a combination of layer2 and layer3 protocol information to generate the hash. Traffic for different remote hosts is mapped to different slaves; one remote host is always mapped to the same slave.
- Layer 3+4 Uses upper-layer protocol information, when available, to generate the hash. Traffic for different connections to the same remote host is mapped to different slaves; each individual connection is always mapped to the same slave. Each connection is identified by IP address plus port number.

Link monitoring

The Linux bonding driver supports two kinds of link monitors: the MII (Media Independent Interface) monitor and the ARP (Address Resolution Protocol) monitor. These two monitors cannot be used in conjunction. You should choose only one monitor at one time, and not all bonding modes support both monitors. The monitor most often used is MII.
MII monitor

The MII monitoring periodically inspects the information provided by the Media Independent Interface. If the interface reflects the failure of a certain network device, the bonding driver interprets that this particular device is down. Options related to MII monitors are:

**miimon**

Specifies the MII link monitoring frequency in milliseconds. The default value is 0, and this will disable the MII monitor. On Linux on Z, it is often set to 1000.

**use_carrier**

Specifies whether or not miimon should use MII or ETHTOOL ioctls vs. netif_carrier_ok() to determine the link status. The default value is 1, which enables the use of netif_carrier_ok(). This is supported by the qeth device driver on Linux on Z.

**updelay**

Specifies the time, in milliseconds, to wait before enabling a slave after a link recovery has been detected. The updelay value should be a multiple of the miimon value.

**downdelay**

Specifies the time, in milliseconds, to wait before disabling a slave after a link failure has been detected. The downdelay value should be a multiple of the miimon value.

```plaintext
miimon=1000
use_carrier=1
updelay=5000
downdelay=2000
```

*Figure 5. Example: Using the MII monitor*

ARP monitor

The ARP monitor checks the links by periodically sending ARP packets to the designated targets. If there is no reply from a certain device, it considers the device to be down. The ARP monitor generates regular traffic by issuing ARP probes. The ARP monitor is not supported by all of the bonding modes. Requested options are:

**arp_interval**

Specifies the ARP link monitoring frequency in milliseconds. The default value is 0, which disables the ARP monitor. A reasonable value could be 1000.

**arp_ip_target**

These are the IP addresses of targets of the ARP request sent to determine the health of the link to the targets. Multiple IP addresses must be separated by a comma. At least one IP address must be given for ARP monitoring to function. The maximum number of targets that can be specified is 16.
MII monitor vs. ARP monitor

The MII monitor is driver-dependent. It monitors the links from the device to the nearest connected switch. If the failure occurs beyond the nearest connected switch, it cannot be detected by MII monitor. However, the ARP monitor is based on the communication to the target hosts designated by their IP addresses. Even if the link is beyond the nearest connected switch, the ARP monitor can detect it.

Figure 7 shows a network typology in which Host 1 is connected to two independent switches. Both of the switches are connected to an external network. If link 2 is broken, both MII and ARP monitors will be able to detect it, since the link is within the nearest connected switch. If link 3 is broken, only ARP monitor can detect it.

The MII monitor is appropriate for bonding when the network communication is within a LAN. The ARP IP target-based monitor is suitable for bonding that is expected to communicate to an outside network.
Chapter 3. Setting up channel bonding on different distributions

There are three distribution partners that support Linux on Z: SUSE Linux Enterprise Server (SLES), Red Hat Enterprise Linux (RHEL), and Ubuntu Server. On each of the distribution systems, the setup of channel bonding is different. This chapter provides three different ways of doing step-by-step configuration of channel bonding:

- “Setting up channel bonding on SLES12” on page 12
- “Setting up channel bonding on RHEL 7” on page 15
- “Setting up channel bonding on Ubuntu 16.04” on page 19

Note: In this paper, we take SLES 12, RHEL 7, and Ubuntu 16.04 as example releases for bonding configuration. The earlier releases of each distribution should also support bonding, but the configuration details might be slightly different. Refer to the documentation provided by the distribution partners.

Preparation

With some Linux releases, the bonding module is not loaded by default. In such cases, issue the following command as root to load the bonding module:

```
# modprobe bonding
```

*Figure 8. Example: Loading the bonding module*

Before setting up a channel bonding interface, the network devices should be ready for the system. To keep the device configuration persistent, it is recommended that you manage network devices using udev rules. The following is an example of a udev rule to add a qeth device with subchannel id 0.0.b230 and preserve the mapping of network interface eth0 to the OSA device 0.0.b230. This rule is defined in /etc/udev/rules.d/70-persistent-net.rules file.

```
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="qeth", KERNELS=="0.0.b230", ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"
```

*Figure 9. Example: udev rules for persistent network device*

You can use the command `lscss` (list subchannels) to display a list of devices with subchannel information, as shown in

```
# lscss
```

*Figure 10. Example: Using the lscss command to show subchannel information*
Here the two OSA devices 0.0.b230 and 0.0.b2a0 are taken as examples for configuration of a channel bonding. Looking at the output of the `lsccs` command in Figure 10 on page 11, we can see both devices with triplet device bus-IDs, connected to different channel paths. This guarantees that if one channel path is in error state, the failover will work with the other one.

You can also use the command `lsqeth` (list qeth-based network devices) to display a summary of information about the qeth group device, as shown in Figure 11.

<table>
<thead>
<tr>
<th># lsqeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device name : eth0</td>
</tr>
<tr>
<td>card_type  : OSD_10GIG</td>
</tr>
<tr>
<td>cdev0      : 0.0.b230</td>
</tr>
<tr>
<td>cdev1      : 0.0.b231</td>
</tr>
<tr>
<td>cdev2      : 0.0.b232</td>
</tr>
<tr>
<td>chpid      : B6</td>
</tr>
<tr>
<td>online     : 1</td>
</tr>
<tr>
<td>portname   : osaport</td>
</tr>
<tr>
<td>portno     : 0</td>
</tr>
<tr>
<td>state      : UP (LAN ONLINE)</td>
</tr>
<tr>
<td>priority_queueing : always queue 0</td>
</tr>
<tr>
<td>buffer_count : 64</td>
</tr>
<tr>
<td>layer2     : 1</td>
</tr>
<tr>
<td>isolation  : none</td>
</tr>
<tr>
<td>switch_attrs : unknown</td>
</tr>
</tbody>
</table>

Figure 11. Example: Using the lsqeth command to list qeth devices

In the example, we can see that the network interface eth0, representing group device 0.0.b230, 0.0.b231, and 0.0.b232, is an OSA-Express adapter with 10-Gigabit Ethernet. For the channel bonding configuration it is important for the OSA device to be defined as a Layer 2 device.

### Setting up channel bonding on SLES12

On SLES 12, it is very convenient to use YaST to configure channel bonding. To begin the bonding setup, access the Network Settings module within YaST.

Add the network cards (in this example, the OSA-Express 5 adapters), with Ethernet device type and select the Address tab. In the Address window (as shown in Figure 12 on page 13), choose the radio button No Link and IP Setup (Bonding Slaves) to define this network interface as a slave device. This selection also allows you to edit the properties of a previously-defined Ethernet-type device in order to define it as a bonding slave device.
After adding all slaves, add a new network interface with type **Bond** and select the **Address** tab again (as shown in [Figure 13]).

![Network Card Setup](image1)

**Figure 12. SLES12--adding a bonding slave device**

In this window, choose the radio button **Statically assign the IP address** to configure the IP address of the bond and the host name and click **Next**. The system displays the **Bond Slaves and Order** window:

![Network Card Setup](image2)

**Figure 13. SLES12--configuring the bond IP address**

In this window, choose the radio button **Statically assign the IP address** to configure the IP address of the bond and the host name and click **Next**. The system displays the **Bond Slaves and Order** window:
In this window, you can select check boxes to add any of your previously-configured slaves to the bond. In the **Bond Driver Options** field, you can specify additional options that define the behavior of the bond. In this example, the SLES 12 default options `mode=active-backup miimon=1000` are specified. These options configure the bond to active-backup mode, with Media Independent Interface (MII) link monitoring of 1000-ms intervals. (As discussed in Chapter 2, “Channel bonding options and recommendations,” on page 5, on Linux on Z the option `fail_over_mac` should always be set to 1 when using active-backup mode.)

To finish the bonding configuration, click the **Next** button. The system automatically generates a configuration file for the bonding interface (`/etc/sysconfig/network/ifcfg-bond0`), as shown in Figure 15:

```plaintext
BONDING_MASTER='yes'
BONDING_MODULE_OPTS='mode=active-backup miimon=1000 fail_over_mac=1'
BONDING_SLAVE0='eth1'
BONDING_SLAVE1='eth2'
IPADDR='192.0.2.0/24'
BOOTPROTO='static'
STARTMODE='auto'
```

**Figure 15. Example: SLES12--bonding master configuration file**

**Note:** The network configuration files are located in the `/etc/sysconfig/network/` directory and the files are named as `ifcfg-NIC_name`. In this example, the configuration file for the bonding master is named `/etc/sysconfig/network/ifcfg-bond0` because the bonding interface is named bond0.

As shown in the example, the bonding master configuration file contains some properties that differentiate it from a normal network device configuration file:

```plaintext
BONDING_MASTER='yes'
   Establishes this as the bonding master.
BONDING_MODULE_OPTS='mode=active-backup miimon=1000 fail_over_mac=1'
   Provides a list of bonding options, separated with blank.
```
BONDING_SLAVE0='eth1'
Defines the name of this slave.

BONDING_SLAVE1='eth2'
Defines the name of this slave.

IPADDR='192.0.2.0/24'
Defines the IP address.

If you use the YaST setup to generate config files for each of the slave devices, the MAC addresses of the slave devices will not show up in these files. With OSA devices as bonding slaves, it is recommended to manually define a virtual MAC address (LLADDR) for each slave device, as shown in Figure 16 so that the MAC addresses keep persistent after rebooting.

/etc/sysconfig/network/ifcfg-eth1:

NAME='OSA Express Network card (0.0.b230)'
BOOTPROTO='none'
STARTMODE='auto'
LLADDR='02:00:00:0a:6c:4b'

/etc/sysconfig/network/ifcfg-eth2:

NAME='OSA Express Network card (0.0.b2a0)'
BOOTPROTO='none'
STARTMODE='auto'
LLADDR='02:0b:0c:0d:0e:02'

Figure 16. Example: SLES 12--bonding slave configuration files

On SLES12, after configuring the bonding interface, you can use the Wicked network management tool to set the interface down and up (as shown in Figure 17) to put the changes in the configuration files into effect.

# wicked ifdown bond0
# wicked ifup bond0

Figure 17. Example: SLES 12--using Wicked to start a bonding interface

**Setting up channel bonding on RHEL 7**

RHEL 7 provides several possibilities for configuring network settings (including channel bonding):

- NetworkManager with the GNOME control-center utility
- NetworkManager text user interface (nmcli) tool
- NetworkManager command line (nmcli) tool

You can also perform manual configuration using the command line interface.

The article "How do I configure a bonding device on Red Hat Enterprise Linux (RHEL)?" [https://access.redhat.com/articles/172483](https://access.redhat.com/articles/172483) provides examples showing how to use nmcli to set up a bond device.

In addition, Red Hat Customer Portal Labs provides a Network Bonding Helper for automatically generating a network bond, based on your environment and deployment goals. (See "Red Hat Network Bonding Helper" [https://](https://])
access.redhat.com/labs/networkbondinghelper/). This is a web application that helps generate configuration files with the setup information given by user.

**Note:** This tool has only limited possibilities for setting bonding options. Be sure to review and manually modify the configuration files before applying to the system.

The examples here use the NetworkManager text user interface (*nmtui*). For more details about the other tools, refer to the corresponding RHEL 7 documentation.

To start the NetworkManager text user interface tool, issue the command *nmtui* in a terminal window on RHEL 7. Within the tool, use the arrow, Tab, Enter, or Shift+Enter keys to step forwards and backwards. To create a new connection select **Edit a connection**, then **<Add>** and **Bond**, as shown in [Figure 18](#).

![Figure 18. RHEL 7--Adding a bond connection](image)

The system displays the Edit Connection screen:
On this screen, you can do these tasks:

- Specify the bond device
- Choose the bonding mode with corresponding parameters
- Configure the IP address of the bond interface

In this example, we set bonding mode to **Active Backup**, the primary slave device is **eth1**, and MII monitor is used for link monitoring with a frequency of 1000ms.

**Note:** This screen does not allow users to configure all possible bonding options. If you need specific configurations that are not listed on this screen, you will have to add them manually later in the bonding device configuration file.

To add additional slave devices for this bond, select **Add** on the right-hand side, and choose **Ethernet** type of connection. The system displays the following screen:
On this screen you can specify the device name of the slave and the unique MAC address. The slave device does not have an IP address.

**Note:** Always add the bond master first, and then add the slave device within the Edit Connection screen of the bond master.

After adding all the slave devices, select **OK** and quit the **nmcli** tool. The corresponding network configuration files are then generated automatically. The network configuration files are located under the directory `/etc/sysconfig/network-scripts/`.

Figure 21 shows the bonding master configuration file, `ifcfg-bond0`.

/etc/sysconfig/network-scripts/ifcfg-bond0:

```
DEVICE=bond0
TYPE=Bond
NAME=bond0
BONDING_MASTER=yes
IPADDR=192.0.2.0
PREFIX=24
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=no
UUID=7099226a-66ac-42a3-a41f-da8284e34838
ONBOOT=yes
BOOTPROTO=None
BONDING_OPTS="<bonding options>"
```

**Figure 21. Example: RHEL 7--Configuration file of a bonding master**

In this example file:

**BONDING_MASTER**

Is set to **yes**.

**IPADDR**

Shows the IP address configured in the bond connection menu.

**UUID**

Identifies a bond connection generated by NetworkManager.

**BONDING_OPTS**

Contains all the default values for bonding options. It is important to go through these options and customize them to fit your own system.
Figure 22 shows a slave device configuration file ifcfg-eth1.

/etc/sysconfig/network-scripts/ifcfg-eth1:

```
MACADDR=02:00:00:D3:FE:A0
SLAVE=yes
DEVICE=enccw0.0.b230
TYPE=Ethernet
NAME=eth1
UUID=4a8e29c7-fb39-457b-8edd-46cbc6e49f9
ONBOOT=yes
MASTER=7099226a-66ac-42a3-a41f-da8284e34838
```

**Figure 22. Example: RHEL 7--Configuration file of a slave device**

In this example file:

**MACADDR**
- Specifies the unique MAC address of the slave device.

**UUID**
- Specifies the unique UUID of the slave device.

**MASTER**
- Specifies the UUID of the bonding master.

### Setting up channel bonding on Ubuntu 16.04

On Ubuntu 16.04, all network interfaces, including the bond master and slaves, are configured manually in one file, /etc/network/interfaces. Figure 23 and Figure 24 on page 20 show the configuration of the bonding master bond0 and the two slaves eth1 and eth2:

```
# bond0 is the bonding NIC and can be used like any other normal NIC.
# bond0 is configured using static network information.
auto bond0
iface bond0 inet static
    address 192.0.2.0
    Netmask 255.255.255.0
    bond-mode active-backup
    bond-miimon 1000
    bond-slaves none
    bond-fail-over-mac 1
```

**Figure 23. Example: Ubuntu 16.04--Configuration of a bonding master**

The bonding master is assigned a static IP address. The bonding options, such as bond-mode and bond-miimon, are added to the bond master’s interface. In this example, the bonding interface is in active-backup mode and has a MII monitor with a 1000-millisecond interval.

The same file also contains the configuration information for the slaves eth1 and eth2, under different interface names, as shown in Figure 24 on page 20.
The slave devices are manually configured. For each slave:

**hwaddress**

Specifies the MAC address.

**bond-master**

Identifies the master device. (This allows the bonding driver to combine the slaves to their master device.)

**bond-primary**

Specifies the primary slave device.

**Note:** If a particular slave device should be set as primary slave, the `bond-primary` directive needs to be part of the slave description rather than part of the master description. (If you include this option in the master section, it will be ignored.)

### Figure 24. Example: Ubuntu 16.04--Configuration of bonding slaves

```text
# eth1 is manually configured, and slave to the "bond0" bonded NIC
auto eth1
iface eth1 inet manual
    hwaddress 02:00:00:87:95:e6
    bond-master bond0
    bond-primary eth1
# eth2 is manually configured, and slave to the "bond0" bonded NIC
auto eth2
iface eth2 inet manual
    hwaddress 02:00:00:0f:ed:eb
    bond-master bond0
```
Chapter 4. Troubleshooting

After setting up the bonding interface, all of the configuration options are recorded in the sysfs file system, under the directory `/sys/class/net/bond<N>/bonding/`. It is possible to change the configuration on the command line (as shown in Figure 25) without reloading the bonding module, and the changes will be in effect immediately. However, these manipulations are not consistent, and after reboot all the changes will disappear.

```
# echo 1000 > /sys/class/net/bond0/bonding/miimon
# echo 2 > /sys/class/net/bond0/bonding/primary_reselect
or
# echo failure > /sys/class/net/bond0/bonding/primary_reselect
```

Figure 25. Example: Manipulating configurations in sysfs

It is always important to check the status of the bonding interface after configuration. The file `/proc/net/bonding/bondName`, as shown in Figure 26, records the current status of the bonding interface, including all the options given in the configuration files.

```
# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)
Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: eth1
Currently Active Slave: eth1
MII Status: up
MII Polling Interval (ms): 1000
Up Delay (ms): 0
Down Delay (ms): 0
Slave Interface: eth1
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 02:00:00:0a:6c:4b
Slave queue ID: 0
Slave Interface: eth2
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 02:0b:0c:0d:0e:02
Slave queue ID: 0
```

Figure 26. Example: Checking bonding status

This example shows a bonding interface with active-backup mode, two slaves with different MAC address, and the MII status of up.

Using command `ip link`, as shown in Figure 27 on page 22, you can see more details about the packet transmission through the bonding interface. This example shows a bonding interface with active-backup mode. Slaves eth1 and eth2 have different MAC addresses, indicating that the option `fail_over_mac` is set to active. The bond master has the same MAC address as eth1, showing that eth1 is the current active slave. (An increasing number of dropped packets would indicate that there might be some issue with the bonding interface that will require more investigation.)
```
# ip -s link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
   mode DEFAULT group default qlen 1000
   link/ether 02:00:00:66:3C:4B brd ff:ff:ff:ff:ff:ff
   RX: bytes packets errors dropped overrun mcast
       391234 4592 0 0 0 3561
   TX: bytes packets errors dropped carrier colls
       78126 279 0 0 0 0
3: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
   mode DEFAULT group default qlen 1000
   link/ether 02:08:00:60:0E:02 brd ff:ff:ff:ff:ff:ff
   RX: bytes packets errors dropped overrun mcast
       166321 3147 0 0 0 2931
   TX: bytes packets errors dropped carrier colls
       39968 176 0 0 0 0
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT group default qlen 1000
   link/ether 02:00:00:66:3C:4B brd ff:ff:ff:ff:ff:ff
   RX: bytes packets errors dropped overrun mcast
       4064 15 0 0 0 0
   TX: bytes packets errors dropped carrier colls
       586 7 0 0 0 0
```

Figure 27. Example: `ip link` command results, showing more details
References

The following documents are referenced in this white paper:

1. "Linux Ethernet Bonding Driver HOWTO":
   https://www.kernel.org/doc/Documentation/networking/bonding.txt
2. "Bonding Parameters Based on Network Layout":
   https://blogs.oracle.com/networking/entry/linux_bonding_tips
   http://www.redbooks.ibm.com/abstracts/sg247995.html
4. "Red Hat Enterprise Linux 7 Networking Guide":
5. "How do I configure a bonding device on Red Hat Enterprise Linux (RHEL)?":
   https://access.redhat.com/articles/172483
6. "Red Hat Network Bonding Helper" (login required):
   https://access.redhat.com/labs/networkbondinghelper/
7. "SUSE Linux Enterprise Server 12 SP2: Administration Guide":
8. "Ubuntu Bonding":
   https://help.ubuntu.com/community/UbuntuBonding
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome
Minato-ku Tokyo 106-0032
Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.
Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation Software Interoperability Coordinator  
Department 49XA  
3605 Highway 52  
N Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM’s suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating
platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

**Trademarks**

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [http://www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

**Terms and conditions**

Permissions for the use of these publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of the manufacturer.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of the manufacturer.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any data, software or other intellectual property contained therein.
The manufacturer reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by the manufacturer, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

THE MANUFACTURER MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THESE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.