

Cisco usNIC Deployment Guide for Cisco UCS B-Series Blade Servers

First Published: 2018-08-14

Last Modified: 2019-04-26

Overview of Cisco usNIC

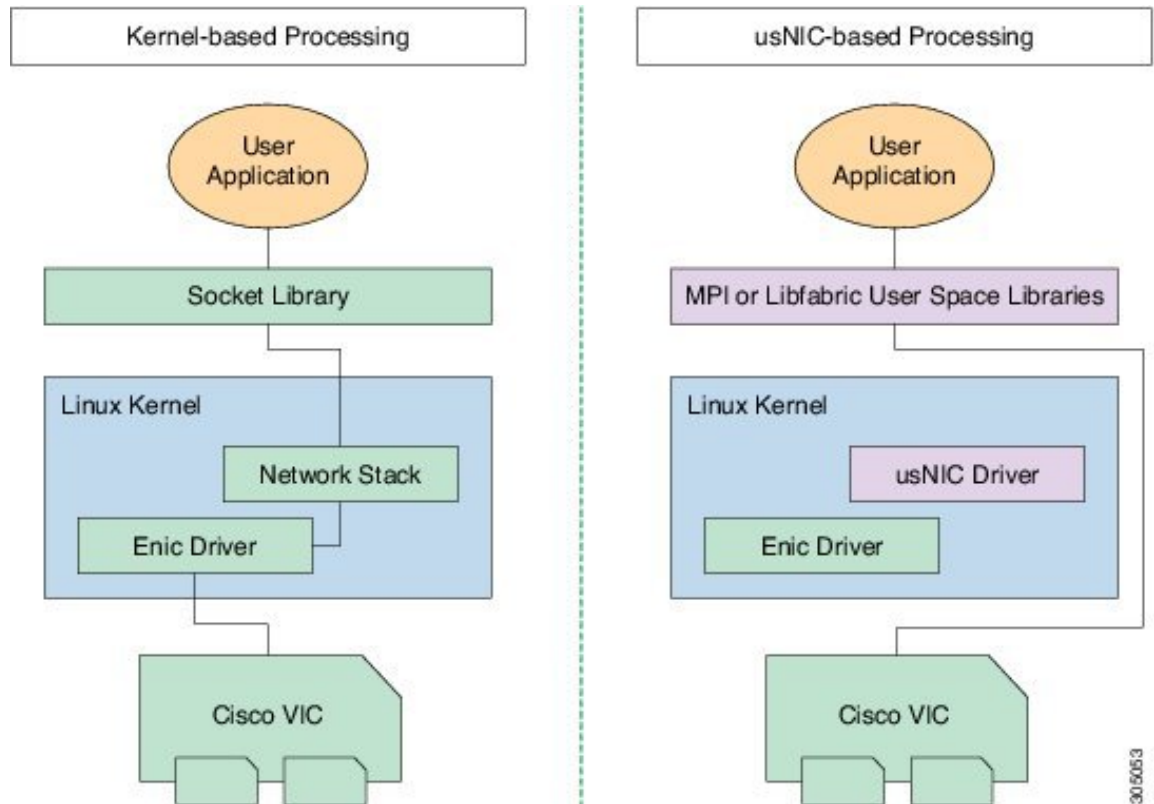
The Cisco user-space NIC (Cisco usNIC) feature improves the performance of software applications that run on the Cisco UCS servers in your data center by bypassing the kernel when sending and receiving networking packets. The applications interact directly with a Cisco UCS VIC second generation or later adapter, which improves the networking performance of your high-performance computing cluster. To benefit from Cisco usNIC, your applications must use the Message Passing Interface (MPI) or the Libfabric interface instead of sockets or other communication APIs.

Cisco usNIC offers the following benefits for your applications:

- Provides a low-latency and high-throughput communication transport.
- Employs the standard and application-independent Ethernet protocol.
- Low jitter, or near constant latency, communications.
- Takes advantage of low latency forwarding, Unified Fabric, and integrated management support in the following Cisco data center platforms:
 - Cisco UCS server
 - Cisco UCS VIC second generation or later generation adapter

Standard Ethernet applications use user-space socket libraries, which invoke the networking stack in the Linux kernel. The networking stack then uses the Cisco eNIC driver to communicate with the Cisco VIC hardware. The following figure shows the contrast between a regular software application and an MPI application that uses usNIC.

Figure 1: Kernel-Based Network Communication versus Cisco usNIC-Based Communication



Cisco usNIC Prerequisites

To benefit from Cisco usNIC, your configuration has the following prerequisites:

- A supported Linux operating system distribution release. For more information on supported Linux operating system releases, please refer to the [UCS Hardware and Software Compatibility Tool](#).
- The UCS Driver ISO corresponding to the UCS server model, selected Linux operating system, and version of UCS firmware installed on the server as identified by the UCS Hardware and Software Compatibility Tool. For more information, see [Downloading Cisco UCS VIC drivers](#).
- A supported MPI implementation, such as IBM Spectrum MPI, the open source Community Open MPI package, or version 4 or 5 of the Intel[®] MPI Library. If the Intel[®] MPI Library is used, the network must be configured with flow control enabled.

Configuring Cisco usNIC

The overall flow to configure Cisco usNIC is as follows:

- Create or Modify a Service Profile to support usNIC
- Install a supported Linux OS (if not already installed)
- Configure Linux kernel and OS to support usNIC

- Install usNIC drivers and utilities
- Install libfabric and MPI software
- Verify the usNIC installation

Creating a Cisco usNIC Connection Policy using the Cisco Manager GUI

You can use the procedure described below or click Play on this [video](#) to watch how a Cisco usNIC Connection policy can be created.

Procedure

- Step 1** In the **Navigation** pane, click **LAN**.
- Step 2** Expand **LAN > Policies**.
- Step 3** Expand the **root** node.
- Step 4** Right-click **usNIC Connection Policies** and choose **Create usNIC Connection Policy**.
- Step 5** In the **Create usNIC Connection Policy** dialog box, complete the following fields:

Name	Description
Name field	The name of the policy. This name can be between 1 and 16 alphanumeric characters. You cannot use spaces or any special characters other than - (hyphen), _ (underscore), : (colon), and . (period), and you cannot change this name after the object is saved.
Description field	A description of the policy. Cisco recommends including information about where and when to use the policy.
Number of usNICs field	The number of usNICs that you want to create. Each MPI process running on the server requires a dedicated usNIC. You can create up to 116 usNICs on one adapter to sustain 116 MPI processes running simultaneously. We recommend that you create at least as many usNICs, per usNIC-enabled vNIC, as there are physical cores on your server. For example, if you have 8 physical cores on your server, create 8 usNICs.
Adapter Policy drop-down list	The adapter policy that you want to specify for the usNIC. Cisco recommends that you choose the usNIC adapter policy, which is created by default.

Configuring a usNIC Ethernet Adapter Policy

Procedure

- Step 1** In the **Navigation** pane, click **Servers**.
- Step 2** On the **Servers** tab, expand **Servers > Policies > root > Adapter Policies**.
- Step 3** Click **Eth Adapter Policy usNIC**.
- Step 4** In the **Work** pane, click the **General** tab.

You can modify the details in the **Resources** and **Options** sections as needed. We recommend that you use the following default values for Resources:

Name	Value
Transmit Queues	6
Ring Size	256
Received Queues	6
Ring Size	512
Completion Queues	6
Interrupts	6

Modifying a usNIC using the Cisco UCS Manager GUI

Procedure

- Step 1** In the **Navigation** pane, click **Servers**.
- Step 2** On the **Servers** tab, expand **Servers > Service Profiles > root**.
- Step 3** Expand the service profile node where you want to configure the usNIC and click **vNICs**.
- Step 4** In the **Work** pane, click the **Network** tab.
- Step 5** In the **vNICs** area, choose a vNIC and click **Modify**.
- Step 6** In the **Adapter Performance Profile** area of the **Modify vNIC** dialog box, choose Linux from the **Adapter Policy** drop-down list.
- Step 7** In the **Connection Policies** area, click the **usNIC** radio button.
- Step 8** Choose the usNIC connection policy that you created from the **usNIC Connection Policy** drop-down list.
- Step 9** Click **OK**.
- Step 10** Click **Save Changes**.
- Step 11** In the **Navigation** pane, click the service profile that you just modified.
- Step 12** In the **Work** pane, click the **Policies** tab.
- Step 13** Expand the **BIOS Policy** bar and choose usNIC in the **BIOS Policy** drop-down list.

Step 14 Click **Save Changes**.

Creating a usNIC using the Cisco UCS Manager CLI

Before you begin

You must log in with admin privileges to perform this task.

Procedure

	Command or Action	Purpose
Step 1	UCS-A # scope service-profile server <i>chassis-id / blade-id or rack_server-id</i>	Enters the service profile for the specified chassis, blade or UCS managed rack server ID.
Step 2	UCS-A /org/service-profile # show vnic	Displays the vnics that are available on the server. A usNIC vNIC is available by default when you upgrade to Cisco UCS Manager, release 2.2.
Step 3	UCS-A /org/service-profile # scope vnic vnic <i>name</i>	Enters the vNIC mode for the specified vNIC.
Step 4	UCS-A /org/service-profile/vnic # set adapter-policy Linux	Specifies Linux and the adapter policy for the usNIC.
Step 5	UCS-A /org/service-profile/vnic # enter usnic-conn-policy-ref <i>usnic connection policy</i> <i>reference name</i>	Creates the usNIC connection policy reference for the vNIC with the specified name. The maximum size for the connection policy name is 16 characters.
Step 6	UCS-A /org/service-profile/vnic/usnic-conn-policy-ref* # commit-buffer	Commits the transaction to the system configuration.
Step 7	UCS-A /org/service-profile/vnic/usnic-conn-policy-ref # top	Enters the top-level mode.
Step 8	UCS-A # scope org	Enters the root organization mode.
Step 9	UCS-A /org # create usnic-conn-policy <i>usnic</i> <i>connection policy name</i>	Creates a usNIC connection policy with the specified name.
Step 10	UCS-A /org/usnic-conn-policy* # set usnic-count <i>number of usnics</i>	Specifies the number of Cisco usNICs to create. It is recommended that you enter 58 for this value.
Step 11	UCS-A /org/usnic-conn-policy* # set adaptor-profile usNIC	Specifies the usNIC Ethernet adaptor profile for the usNIC connection policy. This usNIC adaptor profile is created by default when you upgrade from previous versions of Cisco UCS Manager to release 2.2.

	Command or Action	Purpose
Step 12	UCS-A /org/usnic-conn-policy* # commit-buffer	Commits the transaction to the system configuration.

Example

This example shows how to create a Cisco usNIC and specify its properties:

```
Server # scope org
Server /org # create usnic-conn-policy usnic1
Server /org/usnic-conn-policy* # set usnic-count 58
Server /org/usnic-conn-policy* # set adaptor-profile usNIC
Server /org/usnic-conn-policy* # commit-buffer
Server /org/usnic-conn-policy # top

Server # scope service-profile server 1/1
Server /org/service-profile # show vnic

vNIC:
Name Fabric ID Dynamic MAC Addr Virtualization Preference
-----
eth0 A 00:25:B5:00:00:A1 NONE
eth1 B 00:25:B5:00:00:A2 NONE
eth2 A 00:25:B5:00:00:A3 NONE
Server /org/service-profile # scope vnic eth0
Server /org/service-profile/vnic # set adapter-policy Linux
Server /org/service-profile/vnic # enter usnic-conn-policy-ref usnic1
Server /org/service-profile/vnic/usnic-conn-policy-ref* # commit-buffer
Server /org/service-profile/vnic/usnic-conn-policy-ref # exit
```

Modifying a usNIC using the Cisco UCS Manager CLI

Before you begin

You must log in with admin privileges to perform this task.

Procedure

	Command or Action	Purpose
Step 1	UCS-A # scope service-profile server <i>chassis-id / blade-id or rack_server-id</i>	Enters the service profile for the specified chassis, blade or UCS managed rack server ID.
Step 2	UCS-A /org/service-profile # show vnic	Displays the vnics that are available on the server. A usnic vnic is available by default when you upgrade to Cisco UCS Manager, release 2.2.
Step 3	UCS-A /org/service-profile # scope vnic vnic <i>name</i>	Enters the vnic mode for the specified vNIC.

	Command or Action	Purpose
Step 4	UCS-A /org/service-profile/vnic # enter usnic-conn-policy-ref <i>usnic connection policy reference name</i>	Specifies the usnic connection policy reference for the vNIC that you want to use.
Step 5	UCS-A /org/service-profile/vnic/usnic-conn-policy-ref* # commit-buffer	Commits the transaction to the system configuration.

Example

This example shows how to modify Cisco usNIC properties:

```
Server # scope service-profile server 1/1
Server /org/service-profile # show vnic

vNIC:
Name Fabric ID Dynamic MAC Addr Virtualization Preference
-----
eth0 A 00:25:B5:00:00:A1 SRIOV USNIC
eth1 B 00:25:B5:00:00:A2 NONE
eth2 A 00:25:B5:00:00:A3 NONE
Server /org/service-profile # scope vnic eth0
Server /org/service-profile/vnic # enter usnic-conn-policy-ref usnic2
Server /org/service-profile/vnic/usnic-conn-policy-ref* # commit-buffer
Server /org/service-profile/vnic/usnic-conn-policy-ref # exit
```

Deleting a usNIC using the Cisco UCS Manager CLI

Before you begin

You must log in with admin privileges to perform this task.

Procedure

	Command or Action	Purpose
Step 1	UCS-A # scope service-profile server <i>chassis-id / blade-id or rack_server-id</i>	Enters the service profile for the specified chassis, blade or UCS managed rack server ID.
Step 2	UCS-A /org/service-profile # show vnic	Displays the vNICs that are available on the server. A usNIC vNIC is available by default when you upgrade to Cisco UCS Manager, release 2.2.
Step 3	UCS-A /org/service-profile # scope vnic <i>vnic name</i>	Enters the vNIC mode for the specified vNIC.
Step 4	UCS-A /org/service-profile/vnic # show usnic-conn-policy-ref <i>usnic connection policy reference name</i>	Specifies the usNIC connection policy reference for the vNIC that you want to use.

	Command or Action	Purpose
Step 5	UCS-A /org/service-profile/vnic # delete usnic-conn-policy-ref <i>usnic connection policy reference name</i>	Deletes the specified usNIC connection policy reference.
Step 6	UCS-A /org/service-profile/vnic/usnic-conn-policy-ref* # commit-buffer	Commits the transaction to the system configuration.

Example

This example shows how to modify Cisco usNIC properties:

```
Server # scope service-profile server 1/1
Server /org/service-profile # show vnic

vNIC:
Name Fabric ID Dynamic MAC Addr Virtualization Preference
-----
eth0 A 00:25:B5:00:00:A1 SRIOV USNIC
eth1 B 00:25:B5:00:00:A2 NONE
eth2 A 00:25:B5:00:00:A3 NONE
Server /org/service-profile # scope vnic eth0
Server /org/service-profile/vnic # show usnic-conn-policy-ref

usNIC Connection Policy Reference:
usNIC Connection Policy Name
-----
usnic2
Server /org/service-profile/vnic # delete usnic-conn-policy-ref usnic2
Server /org/service-profile/vnic* # commit-buffer
Server /org/service-profile/vnic # exit
```

Configuring the Linux Kernel for Cisco usNIC

Before you begin

Make sure that the following software and hardware components are installed on the Cisco UCS server:

- A supported Linux operating system distribution release. For more information, see the [UCS Hardware and Software Compatibility Tool](#).
- GCC, G++, and Gfortran
- DAT user library (if using Intel® MPI)
- libnl user library development package (either version 1 or version 3)
- Cisco UCS VIC, second generation or later adapter

Procedure

- Step 1** Ensure the kernel option **CONFIG_INTEL_IOMMU** is selected in the kernel. Enable the Intel IOMMU driver in the Linux kernel by manually adding **'intel_iommu=on'** in the grub.conf file (for example, /boot/grub/grub.conf).
- For example, if your grub.conf file contains a "kernel" line such as `kernel (hd0,0)/vmlinuz LANG=en_US.UTF-8 KEYTABLE=us`, then you will add **'intel_iommu=on'** to the end as shown below:
- ```
kernel (hd0,0)/vmlinuz LANG=en_US.UTF-8 KEYTABLE=us intel_iommu=on
```
- For Red Hat Enterprise Linux use the grubby command line tool to add `intel_iommu=on` to the configuration file.
- ```
# grubby --args="intel_iommu=on" --update-kernel /boot/vmlinuz-`uname -r`
```
- For SLES, add `"intel_iommu=on"` to the GRUB_CMDLINE_LINUX_DEFAULT option found in /etc/default/grub configuration file then run the grub2-mkconfig command below to apply the changes.
- ```
grub2-mkconfig -o /boot/grub2/grub.cfg
```
- For Ubuntu, add `"intel_iommu=on"` to the GRUB\_CMDLINE\_LINUX\_DEFAULT option found in /etc/default/grub configuration file then run the update-grub command below to apply the changes.
- ```
# update-grub2
```
- Step 2** Reboot your Cisco UCS server.
- You must reboot your server for the changes to take after you enable the Intel IOMMU.
- Step 3** Verify that the running kernel has booted with the `intel_iommu=on` option.
- ```
$ cat /proc/cmdline | grep iommu
```
- Step 4** Install the Cisco usNIC Linux drivers.
- For more information about installing the drivers, see "Installing Linux Drivers" section in the guide.
- Note** The Cisco usNIC packages do not support the upgrade or downgrade of an operating system. To update the operating system, first uninstall the usNIC packages, update the operating system, and then reinstall the usNIC drivers.
- Alternatively, you can update the operating system, uninstall the usNIC drivers, and then reinstall the usNIC drivers.

## Installing Linux Software Packages for Cisco usNIC

The following section lists the content of the usNIC folder, specific for each supported Linux operating system distribution that is included in the UCS Drivers ISO bundle. Documentation about known issues and installation instructions are also included in the README file in the usNIC folder.

- **kmod-usnic\_verbs-{version}.x86\_64.rpm**—Linux kernel verbs driver for the usNIC feature of the Cisco VIC SR-IOV Ethernet NIC.



**Note** On systems with SLES 12.1 and later, there is a single RPM that contains both the enic and usnic\_verbs drivers named `cisco-enic-usnic-kmp-default-{version}.x86_64.rpm`. Due to how SLES kernel module dependencies work, this "combo" RPM must be installed instead of installing individual enic and usnic\_verbs RPMs.

- **libdaplusnic-{version}.x86\_64.rpm**— User space library DAPL plugin for usNIC.
- **usnic\_tools-{version}.x86\_64.rpm** — Utility programs for usNIC.
- **usd\_tools-{version}.x86\_64.rpm** — Additional diagnostic tools for usNIC.
- **libfabric-cisco-{version}.x86\_64.rpm**— Libfabric package with built-in support for the Cisco usNIC transport.
- **libfabric-cisco-devel-{version}.x86\_64.rpm** — Development headers for the Libfabric package with built-in support for the Cisco usNIC transport.
- **libusnic\_verbs-{version}.x86\_64.rpm**— A dummy library that causes the libibverbs library to skip Cisco usNIC Linux devices (because Cisco usNIC functionality is exposed through libfabric, not libibverbs). This RPM is only necessary on older Linux distros that have not upgraded to the "rdma-core" packaging of the libibverbs library (e.g., RHEL 6).

## Procedure

- Step 1** Upgrade to the latest version of the enic driver included in the Cisco UCS ISO for your Linux distribution as documented in [Cisco UCS Virtual Interface Card Drivers for Linux Installation Guide](#).
- Step 2** Install the Cisco usNIC software packages from the Cisco UCS Drivers ISO for your Linux distribution.
- Step 3** Enable the Linux RDMA services. Once enabled, RDMA services will be started automatically after a system reboot. The exact command to enable Linux RDMA may vary between each operating system. For example, on Red Hat Enterprise Linux 6.X use the following command:

```
chkconfig rdma on
```

**Note** You may need to perform this step on some Linux operating systems distributions, such as RHEL 6.4.

- Step 4** Reboot your server for the installation changes to take effect automatically. By rebooting at this point and performing the installation validation steps found below, you will be confident your system is configured properly.

**Important** It is recommended that you install all the binary RPMs from the ISO. Building and installing source code packages is intended for advanced users who are familiar with Linux device driver development environments.

## Source code for Linux Cisco usNIC software packages

The source code for the Cisco usNIC software packages is provided on the Cisco UCS Drivers ISO. It is recommended that you do not mix source code and binary package installations.

## Manually Loading the Kernel Modules for Cisco usNIC

Assuming that the operating system was booted with Intel IOMMU support enabled, you can manually load the Cisco usNIC kernel modules with the following steps.

### Before you begin

Ensure you delete all the existing versions of the driver before you load the latest version of the driver. This will help you configure the system successfully.

### Procedure

|               | Command or Action                                           | Purpose                                                                                                                                                                                                                                                                                                                                                                    |
|---------------|-------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Step 1</b> | <code># rmmod enic</code>                                   | Unloads the existing enic driver module.<br><br><b>Note</b> Make sure that you are not logged in to the OS using the enic driver, for example, via SSH to a VIC IP interface, because your network connection will get disconnected when the enic driver is unloaded. Instead, you need to log in to the server using the console, via KVM, or through a LOM IP interface. |
| <b>Step 2</b> | <code># modprobe enic</code>                                | Loads the enic driver module.                                                                                                                                                                                                                                                                                                                                              |
| <b>Step 3</b> | <code># dmesg   grep 'Cisco VIC Ethernet NIC Driver'</code> | Verify that the correct version of the enic driver was loaded. The output from this command should show a version string that matches the version of the enic RPM from the Cisco UCS Driver ISO that you just installed.                                                                                                                                                   |
| <b>Step 4</b> | <code># modprobe usnic_verbs</code>                         | Loads the usnic_verbs driver module.                                                                                                                                                                                                                                                                                                                                       |
| <b>Step 5</b> | <code># lsmod   grep usnic_verbs</code>                     | Verify that the usnic_verbs module loaded successfully. If it did, you should see some output. If the usnic_verbs module did not load, you should see no output.                                                                                                                                                                                                           |

## Uninstalling Linux Software Packages for Cisco usNIC

### Procedure

**Step 1** Uninstall the following usNIC software packages:

- libusnic\_verbs (if applicable)
- libfabric-cisco-devel
- libfabric-cisco
- usd\_tools
- usnic\_tools
- libdaplusnic
- kmod-usnic\_verbs (or cisco-enic-usnic-kmp-default)

**Step 2** Reboot your Cisco server.

## Upgrading the Linux Software Packages for Cisco usNIC

### Procedure

- Step 1** Follow the procedure in "Uninstalling Linux Drivers" to uninstall the previous versions of the usNIC software packages.
- Step 2** Follow the procedure in "Installing Linux Drivers" to install usNIC software packages from the Cisco UCS Drivers ISO for your Linux distribution.

## Installing MPI

### Installing Community Open MPI

#### Before you begin

Install the kmod-usnic\_verbs, libfabric-cisco, and libfabric-cisco-devel RPMs.

### Procedure

- Step 1** Download the latest version of Open MPI from <https://www.open-mpi.org/software/ompi/current/>. This URL will automatically redirect you to the current release.
- Step 2** Extract the Open MPI tarball with the following command:
- ```
$ tar xf openmpi-VERSION.tar.bz2
```
- Step 3** In the directory that is created, run the configure command with the following options:
- ```
$ cd openmpi-VERSION
$./configure \
 --prefix=INSTALL_DIRECTORY \
 --with-usnic \
 --with-libfabric=/opt/cisco/libfabric \
 --without-memory-manager \
 --enable-mpirun-prefix-by-default \
```

```
--enable-mca-no-build=btl-openib,common-verbs,oob-ud \
LDFLAGS="-Wl,-rpath -Wl,/opt/cisco/libfabric/lib -Wl,--enable-new-dtags"
```

Substitute an appropriate directory for "INSTALL\_DIRECTORY" (e.g., /opt/openmpi/VERSION).

**Note** Note that the above configure command will build an Open MPI without verbs support (because older versions of the libibverbs library will issue needless warnings to stderr about usNIC devices). You can remove the --enable-mca-no-build option if you need your Open MPI to support the Linux verbs API.

**Step 4** At the successful conclusion of configure, build Open MPI:

```
$ make -j
```

**Step 5** At the successful conclusion of make, install Open MPI (you may need root or specific user permissions to write to your chosen INSTALL\_DIRECTORY):

```
$ make install
```

## Installing IBM Spectrum MPI

Follow the instructions provided in the IBM Spectrum MPI package at the website below:

### Procedure

- [https://www.ibm.com/support/knowledgecenter/SSZTET\\_10.1.0/smpi\\_welcome/smpi\\_kc\\_install.html](https://www.ibm.com/support/knowledgecenter/SSZTET_10.1.0/smpi_welcome/smpi_kc_install.html)

## Installing the Intel MPI Library

Follow the instructions provided in the Intel MPI package at the website below:

### Procedure

- <https://software.intel.com/en-us/intel-mpi-library/documentation>

## Adding MPI to User Environments

Before MPI applications can be compiled and launched, an MPI implementation must be added to each user's environment. It is recommended that you only add one MPI implementation to a user's environment at a time.

### Environment for IBM Spectrum MPI Library

Spectrum MPI needs very little in terms of environment setup. You can invoke Spectrum MPI's commands either through an absolute path or by adding the directory with the Spectrum MPI commands in to your PATH, and then invoking them without an absolute path.

### Environment for Community Open MPI

Community Open MPI requires that you add its installation binary and library paths to the environment variables. It is usually best to add these paths to the environment in your shell startup files so that they are automatically set upon login to all nodes in your cluster.

Specifically, you should prepend the PATH environment variable with INSTALL\_DIRECTORY/bin, and prefix the LD\_LIBRARY\_PATH environment variable with INSTALL\_DIRECTORY/lib. Optionally, you can also add the INSTALL\_DIRECTORY/share/man to the MANPATH environment variable.

For example, if you configured Community Open MPI with an `INSTALL_DIRECTORY` of `/opt/openmpi`, if you are using Bash as your login shell, you can add these lines to your shell startup file (which is typically `$HOME/.bashrc`):

```
export PATH=/opt/openmpi/bin:$PATH
export LD_LIBRARY_PATH=/opt/openmpi/lib:$LD_LIBRARY_PATH
export MANPATH=/opt/openmpi/share/man:$MANPATH
```

Alternatively, if you're using C shell as your login shell, you can add these lines to your shell startup file (which is typically `$HOME/.cshrc`):

```
set path=(/opt/openmpi/bin $path)
setenv LD_LIBRARY_PATH /opt/openmpi/lib:$LD_LIBRARY_PATH
if ("1" == "$?MANPATH") then
 setenv MANPATH /opt/openmpi/share/man:${MANPATH}
else
 setenv MANPATH /opt/openmpi/share/man:
endif
```

Your system may require slightly different commands. Check the Open MPI Community FAQ (<https://www.open-mpi.org/faq/>) in the "Running MPI Jobs" section for more information about how to set your `PATH`, `LD_LIBRARY_PATH`, `MANPATH` environment variables.

## Environment for the Intel® MPI Library

In addition to the instructions provided by the Intel® MPI Library documentation, additional environment variables must be set in each user's environment to enable Cisco usNIC functionality. Two scripts are installed by the `libdaplusnic` software package to help set the required environment variables. One script is for Bourne shell users; the other script is for C shell users:

- `/opt/cisco/intelmpi-usnic-vars.sh` (For Bourne shell users)
- `/opt/cisco/intelmpi-usnic-vars.csh` (For C shell users)

The appropriate script should be sourced as part of the users's shell startup / login sequence.

Using the Intel® MPI Library with usNIC requires the network to be configured with flow control enabled. This can be either IEEE 802.3x link-level flow control or IEEE 802.1Qbb Priority-based Flow Control (PFC). This feature is sometimes also called "no-drop." Refer to the configuration guide for the switches in your network for information about enabling flow control. If flow control is not enabled in the network, then applications utilizing the Intel® MPI Library may work correctly, but possibly with extremely degraded performance.

In deployments of the Intel® MPI Library, the MPI traffic must have flow control enabled on all Cisco usNIC ports with Class of Service (CoS) 6 in Cisco CIMC.

## Adding Libfabric to User Environments

If you are developing Libfabric-specific applications, you may benefit from having the Libfabric test executables (such as `fi_pingpong`) and/or man pages in your environment. Two scripts are installed by the Cisco `libfabric` package to help set the required environment variables. One script is for Bourne shell users, the other is for C shell users:

- `/opt/cisco/libfabric-vars.sh`

- `/opt/cisco/libfabric-vars.csh`

The appropriate script should be sourced as part of the users's shell startup / login sequence.

### Adding usNIC Tools to User Environments

Adding the usNIC tools to the environment can be accomplished via the following scripts; the first is for Bourne shell users, the second is for C shell users:

- `/opt/cisco/usnic/bin/usnic-vars.sh`
- `/opt/cisco/usnic/bin/usnic-vars.csh`

## Verifying the Cisco usNIC Installation for Cisco UCS B-Series Blade Servers

After you install the required Linux drivers for Cisco usNIC, perform the following procedure at the Linux prompt to make sure that the installation completed successfully.



**Note** The examples shown below are configurations verified on Linux operating system distribution RHEL 6.5.

### Procedure

**Step 1** Search and verify if the **usnic\_verbs** kernel module was loaded during the OS driver installation.

```
$ lsmod | grep usnic_verbs
```

The following details are displayed when you enter the **lsmod | grep usnic\_verbs** command. The kernel modules listed on your console may differ based on the modules that you have currently loaded in your OS.

```
usnic_verbs 73762 2
ib_core 74355 2 ib_uverbs,usnic_verbs
enic 73723 1 usnic_verbs
```

**Step 2** View the configuration of Cisco usNIC-enabled NICs.

```
$ /opt/cisco/usnic/bin/usnic_devinfo
```

The following section is a brief example of the results that are displayed when you execute the **usnic\_devinfo** command. The results may differ based on your current installation. When the results are displayed on your console, ensure that the link state for each of the listed ports are shown as UP.

The following example shows two interfaces (**usnic\_1** and **usnic\_0**) that are configured on a Cisco UCS VIC adapter. If you configured only one Cisco usNIC-enabled vNIC, you will see a listing for only **usnic\_0**.

```
usnic_0:
 Interface: eth3
 MAC Address: 00:25:b5:31:32:10
 IP Address: 10.10.10.2
 Netmask: 255.255.255.0
 Prefix len: 24
 MTU: 9000
 Link State: UP
 Bandwidth: 40 Gb/s
```

```

Device ID: UCSB-MLOM-40G-03 [VIC 1340] [0x012c]
Vendor ID: 4407
Vendor Part ID: 207
Firmware: 4.1 (3S1)
VFs: 58
CQ per VF: 6
QP per VF: 6
Interrupts per VF: 6
Max CQ: 348
Max CQ Entries: 65535
Max QP: 348
Max Send Credits: 4095
Max Recv Credits: 4095
Capabilities:
 Map per res: yes
 PIO sends: yes
 CQ interrupts: no
usnic_1:
 Interface: eth4
 MAC Address: 00:25:b5:31:32:20
 IP Address: 10.20.10.2
 Netmask: 255.255.255.0
 Prefix len: 24
 MTU: 9000
 Link State: UP
 Bandwidth: 40 Gb/s
 Device ID: UCSB-MLOM-40G-03 [VIC 1340] [0x012c]
 Vendor ID: 4407
 Vendor Part ID: 207
 Firmware: 4.1 (3S1)
 VFs: 58
 CQ per VF: 6
 QP per VF: 6
 Interrupts per VF: 6
 Max CQ: 348
 Max CQ Entries: 65535
 Max QP: 348
 Max Send Credits: 4095
 Max Recv Credits: 4095
 Capabilities:
 Map per res: yes
 PIO sends: yes
 CQ interrupts: no

```

**Step 3** Run the **usnic\_check** script to view the installed RPMs and their versions.

```
$ /opt/cisco/usnic/bin/usnic_check
```

**Note** If you installed any components from the source code in the Cisco usNIC software packages, the **usnic\_check** script will report the corresponding RPM as missing. It is recommended that you do not mix source code and binary package installations.

**Step 4** Verify that the Cisco usNIC network packets are being transmitted correctly between the client and server hosts.

a) Determine the name of the Ethernet interface associated with the Cisco usNIC on the server host.

```

[server]$ /opt/cisco/usnic/bin/usnic_status
usnic_0: 0000:07:0.0, eth3, 00:25:b5:31:32:10, 58 VFs
 Per VF: 6 WQ, 6 RQ, 6 CQ, 6 INT

In use:
0 VFs, 0 QPs, 0 CQs

```



```
usnic_1: 0000:0c:0.0, eth4, 00:25:b5:31:32:20, 58 VFs
Per VF: 6 WQ, 6 RQ, 6 CQ, 6 INT
```

```
In use:
0 VFs, 0 QPs, 0 CQs
```

- b) Determine the IP address for the Ethernet interface.

```
[server]$ ip addr show dev eth1 | grep "inet[^6]"
inet 10.10.10.2/24 brd 50.42.110.255 scope global eth1
```

- c) Run the **fi\_pingpong** program on the server host.

```
[server]$ /opt/cisco/libfabric/bin/fi_pingpong -p usnic
```

For more information about the command line options used with the **fi\_pingpong** program, see the output of **fi\_pingpong --help**.

- d) Execute the **fi\_pingpong** program on the client host by using the IP address that corresponds to the Cisco usNIC on the server host.

```
[client]$ /opt/cisco/libfabric/bin/fi_pingpong -p usnic SERVER_IP_ADDRESS
```

The following example shows the results that are displayed when you run the **fi\_pingpong** program.

Server-side:

```
[server]$ /opt/cisco/libfabric/bin/fi_pingpong
```

| bytes | #sent | #ack | total | time  | MB/sec | usec/xfer | Mxfers/sec |
|-------|-------|------|-------|-------|--------|-----------|------------|
| 64    | 10k   | =10k | 1.2m  | 0.07s | 17.84  | 3.59      | 0.28       |
| 256   | 10k   | =10k | 4.8m  | 0.08s | 66.23  | 3.87      | 0.26       |
| 1k    | 10k   | =10k | 19m   | 0.10s | 199.76 | 5.13      | 0.20       |
| 4k    | 10k   | =10k | 78m   | 0.18s | 466.60 | 8.78      | 0.11       |

Client-side:

```
[client]$ /opt/cisco/libfabric/bin/fi_pingpong -p usnic SERVER_IP_ADDRESS
```

| bytes | #sent | #ack | total | time  | MB/sec | usec/xfer | Mxfers/sec |
|-------|-------|------|-------|-------|--------|-----------|------------|
| 64    | 10k   | =10k | 1.2m  | 0.07s | 17.84  | 3.59      | 0.28       |
| 256   | 10k   | =10k | 4.8m  | 0.08s | 66.25  | 3.86      | 0.26       |
| 1k    | 10k   | =10k | 19m   | 0.10s | 199.77 | 5.13      | 0.20       |
| 4k    | 10k   | =10k | 78m   | 0.18s | 466.61 | 8.78      | 0.11       |

**Note** **fi\_pingpong** is not a high-performance benchmark. It shows the general performance levels of your usNIC devices, but it is not highly tuned to show the absolute best performance. The **fi\_pingpong** output should only be used to ensure that performance is in the general neighborhood of expected performance.

**Step 5** Download, compile, and execute the **ring\_c** test program to validate that the MPI traffic is correctly transmitted between the client and server hosts.

You can obtain the **ring\_c** test program from this link: [https://raw.githubusercontent.com/open-mpi/ompi/v2.x/examples/ring\\_c.c](https://raw.githubusercontent.com/open-mpi/ompi/v2.x/examples/ring_c.c).

The following example shows how to use the **wget** utility to obtain, compile, and execute the **ring\_c**. Alternatively, you can use other methods of obtaining and running the test program.

**Note** Run the following commands with a single MPI implementation setup in your environment.

```
$ wget --no-check-certificate
```

```

https://raw.githubusercontent.com/open-mpi/ompi/v2.x/examples/ring_c.c
--2017-03-21 19:46:20-- https://raw.githubusercontent.com/open-mpi/ompi/v2.x/examples/ring_c.c
Resolving raw.githubusercontent.com... 151.101.192.133, 151.101.64.133, 151.101.128.133,
...
Connecting to raw.githubusercontent.com|151.101.192.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2416 (2.4K) [text/plain]
Saving to: 'ring_c.c'

ring_c.c 100%[=====>] 2.36K --.-KB/s in 0s

2017-03-21 19:46:20 (29.5 MB/s) - 'ring_c.c' saved [2416/2416]

$ mpicc ring_c.c -o ring_c
[no output]

IBM Spectrum MPI:
$ /path/to/mpirun --usnic --host host1,host2 -n 2 ./ring_c

Community Open MPI:
$ mpirun --mca btl usnic,vader,self --host host1,host2 -n 2 ./ring_c

The expected output from both IBM Spectrum MPI and Community Open MPI is:
Process 0 sending 10 to 1, tag 201 (4 processes in ring)
Process 0 sent to 1
Process 0 decremented value: 9
Process 0 decremented value: 8
Process 0 decremented value: 7
Process 0 decremented value: 6
Process 0 decremented value: 5
Process 0 decremented value: 4
Process 0 decremented value: 3
Process 0 decremented value: 2
Process 0 decremented value: 1
Process 0 decremented value: 0
Process 0 exiting
Process 2 exiting
Process 1 exiting
Process 3 exiting ...

```

**Note** If desired, setup a different MPI implementation in your environment and re-run the **mpicc** and **mpirun** commands to verify that MPI implementation with Cisco usNIC functionality.

---

If the **fi\_pingpong** program and the **ring\_c** program executed successfully, you should now be able to run general MPI applications over Cisco usNIC.

## Troubleshooting Information

### Problem

Viewing the list of installed RPMs using **usnic\_check** causes the following:

1. A warning such as **No usnic devices found**.
2. A version mismatch error such as **usnic\_verbs\_xxxx does not match installed version**.

### Possible Cause

1. A previously-installed version of `usnic_verbs` can cause this error.
2. The `usnic_verbs` driver may have failed to load if the loaded `enic` driver is not compatible with `usnic_verbs`.

### Solution

This problem is typically caused by one of two things:

1. An old version of the `usnic_verbs` RPM is still installed.
  1. List all installed versions using the following command: `rpm -qa | grep usnic_verbs`
  2. Uninstall all versions using the following command: `rpm -e`
  3. Reboot your system.
  4. Re-install all the RPMs.
2. The `enic` driver loaded by the Linux kernel is not compatible with this version of `usnic_verbs`.
  1. This can happen if the distro-provided `enic` RPM was loaded instead of the `enic` driver on the same UCS Driver ISO from which you obtained the `usnic_verbs` driver.
  2. Specifically: the `enic` and `usnic_verbs` drivers must "match" -- if they don't, it is likely that `usnic_verbs` will fail to load with messages in `"dmesg"` from `usnic_verbs` about missing `enic` symbols.
  3. If this is the case, ensure that both the `enic` and `usnic_verbs` drivers are loaded from the UCS Drivers ISO.
  4. Also ensure that the correct `enic` driver version is being loaded upon bootup: check the output from `"dmesg"` to ensure that the `enic` version number matches that of the `enic` RPM that you installed.
  5. If they do not match, you may need to check `depmod` output and/or make a new `initrd` to ensure that the correct `enic` driver is bootstrapped and loaded into the kernel at boot time.

### Problem

Verifying that Cisco `usNIC` packets are being transmitted correctly between client and server using `fi_pingpong` causes the following errors:

1. "No such address or device" error. See the following example:

```
$ /opt/cisco/libfabric/bin/fi_pingpong -p usnic
fi_getinfo: -61
```

### Possible Cause

1. The Cisco `usNIC` connection policy is not assigned or set as 'not set' in the `vNIC` interface.
2. The server side does not receive packets from the client side.

## Solution

1. Make sure that valid Cisco usNIC connection policy is configured in usNIC Connection Policies and assigned to the vNICs in the Service Profile.
2. Make sure that IP addresses of the Cisco usNIC devices on both the server and client are configured correctly.
3. Make sure that the client pingpong is attempting to send packets to the correct server IP address of Cisco usNIC device.

## Problem

Running the Cisco usNIC traffic using the `mpirun` causes the following errors:

MTU size mismatch error. See the following example:

Example:

```
Enter the command below at the prompt on a single line from mpirun up to Sendrecv.
The backslash is included here as a line continuation and is not needed when the command
 is
entered at the prompt.
$ mpirun --host node05,node06 -np 12 --mca btl usnic,vader,self \
--mca btl_usnic_if_include usnic_1 IMB-MPI1 Sendrecv
```

The MTU does not match on local and remote hosts. All interfaces on all hosts participating in an MPI job must be configured with the same MTU. The usNIC interface listed below will not be used to communicate with this remote host.

```
Local host: node05
usNIC interface: usnic_1
Local MTU: 8958
Remote host: node06
Remote MTU: 1458
```

## Possible Cause

1. The MTU size is incorrectly set on the appropriate VLANs.
2. The MTU size is incorrectly set in the QoS.

## Solution

Make sure that the MTU size has been set correctly on the VLANs and QoS.

See: [Configuring QoS System Classes with the LAN Uplinks Manager](#).

## Problem

Installing a Cisco enic driver causes the following Cisco enic dependency errors:

```
rpm -ivh kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64.rpm
error: Failed dependencies:
 ksym(enic_api_devcmd_proxy_by_index) = 0x107cb661 is needed by
kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
 ksym(vnic_dev_alloc_discover) = 0xfb7e4707 is needed by
kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
 ksym(vnic_dev_get_pdev) = 0xae6ae5c9 is needed by
```

```

kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
ksym(vnic_dev_get_res) = 0xd910c86b is needed by
kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
ksym(vnic_dev_get_res_bar) = 0x31710a7e is needed by
kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
ksym(vnic_dev_get_res_bus_addr) = 0x7be7a062 is needed by
kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
ksym(vnic_dev_get_res_count) = 0x759e4b07 is needed by
kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
ksym(vnic_dev_get_res_type_len) = 0xd122f0a1 is needed by
kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
ksym(vnic_dev_unregister) = 0xd99602a1 is needed by
kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64
#

```

### Possible Cause

1. The enic driver is incorrectly installed.
2. The enic driver is not installed.

### Solution

Ensure that the correct enic driver has been installed. In addition, make sure of the following:

- Specifically, you must ensure the following: the enic and usnic\_verbs drivers must match. If you have a mismatch, you can get the above version errors.
- Specifically, the enic and usnic\_verbs that come in the Cisco UCS drivers ISO must be matched together. If you use an enic from one Cisco UCS driver ISO and usnic\_verbs from another Cisco UCS driver ISO, it will result in the above version errors.



#### Note

When installing the "combo" enic and usnic\_verbs RPM on systems with SLES 12.1 and later, it is guaranteed that the enic and usnic\_verbs drivers will match.

### Problem

Intel IOMMU causes the following warnings:

```

rpm -ivh kmod-usnic_verbs-1.0.4.318.rhel6u5-1.x86_64.rpm
Preparing... ##### [100%]
 1:kmod-usnic_verbs ##### [100%]
WARNING -
Intel IOMMU does not appear to be enabled - please add kernel parameter
intel_iommu=on to your boot configuration for USNIC driver to function.
#

```

### Possible Cause

The Intel IOMMU support is not enabled in the Linux kernel.

### Solution

Enable Intel IOMMU driver in the Linux kernel.

### Problem

Installing DAT user library can cause the following failed dependencies errors:

```
rpm -ivh libdaplusnic-2.0.39cisco1.0.0.317-1.el6.x86_64.rpm
error: Failed dependencies:
 dapl is needed by libdaplusnic-2.0.39cisco1.0.0.317-1.el6.x86_64
#
```

### Possible Cause

The `libdapl` is installed without installing the DAT library.

### Solution

Install the DAT library.

### Problem

When viewing the configuration of Cisco usNIC enabled VICS using `usnic_devinfo`, the command output does not list any usNIC interfaces..

### Possible Cause

The RDMA service is not enabled.

### Solution

Enable RDMA service using the following commands:

```
service rdma start
Or
chkconfig rdma on
```