

Cisco usNIC Deployment Guide for Cisco UCS C-Series Rack-Mount Standalone Servers

First Published: 2016-01-20

Last Modified: 2018-03-21

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 Libfabric interface instead of sockets or other communication APIs.

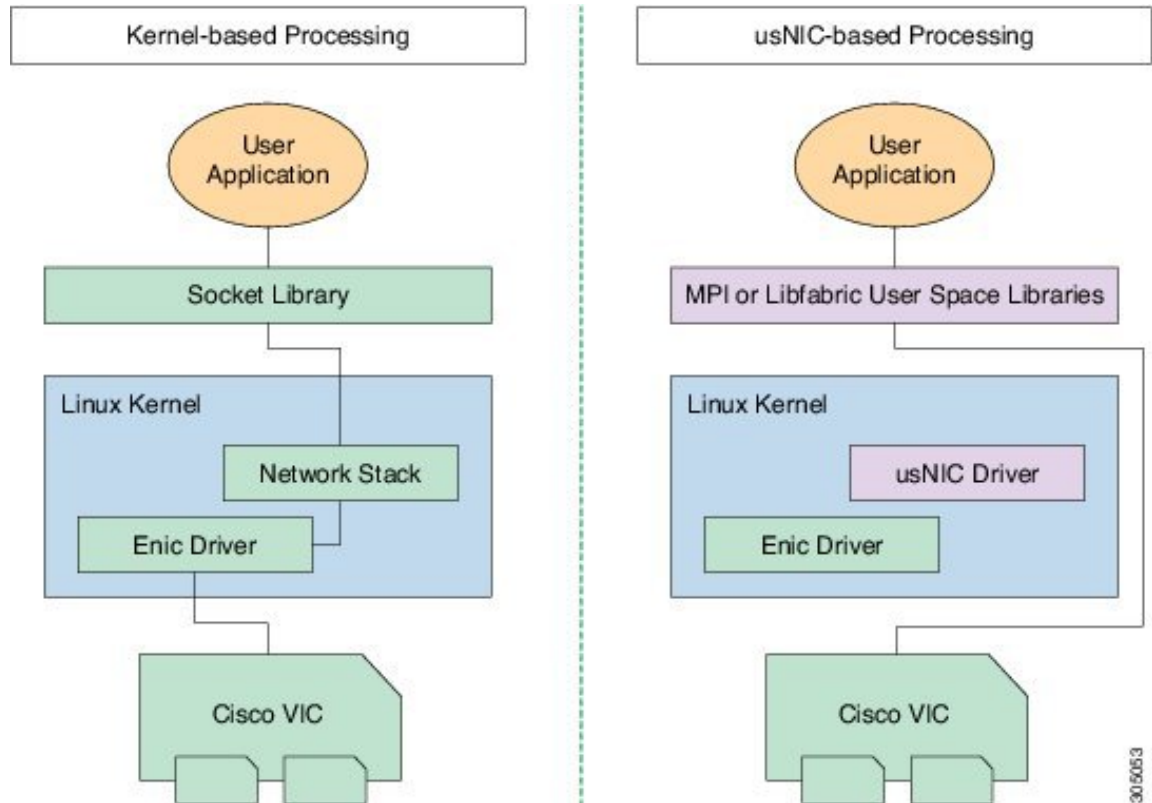
Cisco usNIC offers the following benefits for your applications:

- 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
 - 10 or 40GbE networks

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 the Cisco UCS VIC vNIC configuration to support usNIC
- Install 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

Configuring Cisco usNIC Using the CIMC GUI



Note

Even though several properties are listed for Cisco usNIC in the usNIC properties dialog box, you must configure only the following properties because the other properties are not currently used.

- **Completion Queue Count**
- **Receive Queue Count**
- **Transmit Queue Count**
- **usNIC**

Before You Begin

You must log in to the CIMC GUI with administrator privileges to perform this task. Click Play on this [video](#) to watch how to configure Cisco usNIC in CIMC.

-
- Step 1** Log into the CIMC GUI.
For more information about how to log into CIMC, see [Cisco UCS C-Series Servers Integrated Management Controller GUI Configuration Guide](#).
- Step 2** In the **Navigation** pane, click the **Server** tab.
- Step 3** On the **Server** tab, click **Inventory**.
- Step 4** In the **Inventory** pane, click the **Cisco VIC Adapters** tab.
- Step 5** In the **Adapter Cards** area, select the adapter card.
If the server is powered on, the resources of the selected adapter card appear in the tabbed menu below the **Adapter Cards** area.
- Step 6** In the tabbed menu below the **Adapter Cards** area, click the **vNICs** tab.
- Step 7** In the **Host Ethernet Interfaces** area, select a vNIC from the table.
Note For each vNIC that you want to configure as a usNIC, select the vNIC entry from the table and specify its properties as explained in steps 9 through step 18.
- Step 8** Click **usNIC** to open the **usNIC Properties** dialog box.
- Step 9** In the **usNICs** property, specify the number of Cisco usNICs that you want to create.
Each MPI process that is running on the server requires a dedicated usNIC. You might need to create up to 64 usNICs to sustain 64 MPI processes running simultaneously. We recommend that you create at least as many usNICs, per

usNIC-enabled vNIC, as the number of physical cores on your server. For example, if you have 8 physical cores on your server, create 8 usNICs.

Step 10 In the **Properties** area, update the following fields:

Field Name	Description
Transmit Queue Count	The number of transmit queue resources to allocate. Cisco recommends setting this value to 6.
Receive Queue Count	The number of receive queue resources to allocate. Cisco recommends setting this value to 6.
Completion Queue Count	The number of completion queue resources to allocate. Cisco recommends setting this value to 6.

Step 11 Click **Apply**.

Step 12 In the **Navigation** pane, click the **Server** tab.

Step 13 On the **Server** tab, click **BIOS**.

Step 14 In the **Actions** area, click **Configure BIOS**.

Step 15 In the **Configure BIOS Parameters** dialog box, click the **Advanced** tab.

Step 16 In the **Processor Configuration** area, set the following properties to Enabled:

- **Intel(R) VT-d**
- **Intel(R) VT-d ATS support**
- **Intel(R) VT-d Coherency Support**

Step 17 Click **Save Changes**.

The changes take effect upon the next server reboot.

Creating Cisco usNIC Using the CIMC CLI


Note

Even though several properties are listed for Cisco usNIC in the usNIC properties dialog box, you must configure only the following properties because the other properties are not currently being used.

- **Completion Queue Count**
- **Receive Queue Count**
- **Transmit Queue Count**
- **usNIC**

Before You Begin

You must log in to the CIMC CLI with administrator privileges to perform this task.

DETAILED STEPS

	Command or Action	Purpose
Step 1	server# scope chassis	Enters chassis command mode.
Step 2	server/chassis# scope adapter <i>index</i>	Enters the command mode for the adapter card at the PCI slot number specified by <i>index</i> . Note Make sure that the server is powered on before you attempt to view or change adapter settings. To view the index of the adapters configured on your server, use the show adapter command.
Step 3	server/chassis/adapter# scope host-eth-if {eth0 eth1}	Enters the command mode for the vNIC. Specify the Ethernet ID based on the number of vNICs that you have configured in your environment. For example, specify eth0 if you configured only one vNIC.
Step 4	server/chassis/adapter/host-eth-if# create usnic-config 0	Creates a usNIC config and enters its command mode. Make sure that you always set the index value to 0. Note To create a Cisco usNIC for the first time for a given vNIC using the CLI, you must first create a usnic-config . Subsequently, you only need to scope into the usnic-config and modify the properties for Cisco usNIC. For more information about modifying Cisco usNIC properties, see <i>Modifying a Cisco usNIC value using the CLI</i> .
Step 5	server/chassis/adapter/host-eth-if/usnic-config# set cq-count <i>count</i>	Specifies the number of completion queue resources to allocate. We recommend that you set this value to 6. The number of completion queues equals the number of transmit queues plus the number of receive queues.

	Command or Action	Purpose
Step 6	server/chassis/adapters/host-eth-if/usnic-config# set rq-count <i>count</i>	Specifies the number of receive queue resources to allocate. We recommend that you set this value to 6.
Step 7	server/chassis/adapters/host-eth-if/usnic-config# set tq-count <i>count</i>	Specifies the number of transmit queue resources to allocate. We recommend that you set this value to 6.
Step 8	server/chassis/adapters/host-eth-if/usnic-config# set usnic-count <i>number of usNICs</i> .	Specifies the number of Cisco usNICs to create. Each MPI process that is running on the server requires a dedicated Cisco usNIC. Therefore, you might need to create up to 64 Cisco usNICs to sustain 64 MPI processes running simultaneously. We recommend that you create at least as many Cisco usNICs, per Cisco usNIC-enabled vNIC, as the number of physical cores on your server. For example, if you have 8 physical cores on your server, create 8 Cisco usNICs.
Step 9	server/chassis/adapters/host-eth-if/usnic-config# commit	Commits the transaction to the system configuration. Note The changes take effect when the server is rebooted.
Step 10	server/chassis/adapters/host-eth-if/usnic-config# exit	Exits to host Ethernet interface command mode.
Step 11	server/chassis/adapters/host-eth-if# exit	Exits to adapter interface command mode.
Step 12	server/chassis/adapters# exit	Exits to chassis interface command mode.
Step 13	server/chassis# exit	Exits to server interface command mode.
Step 14	server# scope bios	Enters Bios command mode.
Step 15	server/bios# scope advanced	Enters the advanced settings of BIOS command mode.
Step 16	server/bios/advanced# set IntelVTD Enabled	Enables the Intel Virtualization Technology.
Step 17	server/bios/advanced# set ATS Enabled	Enables the Intel VT-d Address Translation Services (ATS) support for the processor.
Step 18	server/bios/advanced# set CoherencySupport Enabled	Enables Intel VT-d coherency support for the processor.
Step 19	server /bios/advanced# commit	Commits the transaction to the system configuration. Note The changes take effect when the server is rebooted.

This example shows how to configure Cisco usNIC properties:

```

Server # scope chassis
server /chassis # show adapter
server /chassis # scope adapter 2
server /chassis/adapters # scope host-eth-if eth0
server /chassis/adapters/host-eth-if # create usnic-config 0
server /chassis/adapters/host-eth-if/usnic-config *# set usnic-count 64
server /chassis/adapters/host-eth-if/usnic-config *# set cq-count 6
server /chassis/adapters/host-eth-if/usnic-config *# set rq-count 6

```

```

server /chassis/adapter/host-eth-if/usnic-config *# set tq-count 6
server /chassis/adapter/host-eth-if/usnic-config *# commit
Committed settings will take effect upon the next server reset
server /chassis/adapter/host-eth-if/usnic-config # exit
server /chassis/adapter/host-eth-if # exit
server /chassis/adapter # exit
server /chassis # exit
server # exit
server# scope bios
server /bios # scope advanced
server /bios/advanced # set IntelVTD Enabled
server /bios/advanced *# set ATS Enabled*
server /bios/advanced *# set CoherencySupport Enabled
server /bios/advanced *# commit
Changes to BIOS set-up parameters will require a reboot.
Do you want to reboot the system?[y|N]y
A system reboot has been initiated.

```

Modifying a Cisco usNIC value using the CIMC CLI

Before You Begin

You must log in to the CIMC GUI with administrator privileges to perform this task.

DETAILED STEPS

	Command or Action	Purpose
Step 1	server# scope chassis	Enters chassis command mode.
Step 2	server/chassis# scope adapter <i>index</i>	Enters the command mode for the adapter card at the PCI slot number specified by <i>index</i> . Note Make sure that the server is powered on before you attempt to view or change adapter settings. To view the index of the adapters configured on your server, use the show adapter command.
Step 3	server/chassis/adapter# scope host-eth-if { eth0 eth1 }	Enters the command mode for the vNIC. Specify the Ethernet ID based on the number of vNICs that you have configured in your environment. For example, specify eth0 if you configured only one vNIC.
Step 4	server/chassis/adapter/host-eth-if# scope usnic-config 0	Enters the command mode for the usNIC. Make sure that you always set the index value as 0 to configure a Cisco usNIC.
Step 5	server/chassis/adapter/host-eth-if/usnic-config# set usnic-count <i>number of usNICs</i> .	Specifies the number of Cisco usNICs to create. Each MPI process running on the server requires a dedicated Cisco usNIC. Therefore, you might need to create up to 64 Cisco usNIC to sustain 64 MPI processes running simultaneously. We recommend that you create at least as many Cisco usNIC, per Cisco usNIC-enabled vNIC, as the number of physical cores on your server. For example, if you have 8 physical cores on your server, create 8 usNICs.
Step 6	server /chassis/adapter/host-eth-if/usnic-config# commit	Commits the transaction to the system configuration. Note The changes take effect when the server is rebooted.

	Command or Action	Purpose
Step 7	server/chassis/adapter/host-eth-if/usnic-config# exit	Exits to host Ethernet interface command mode.
Step 8	server/chassis/adapter/host-eth-if# exit	Exits to adapter interface command mode.
Step 9	server/chassis/adapter# exit	Exits to chassis interface command mode.
Step 10	server/chassis# exit	Exits to server interface command mode.

This example shows how to configure Cisco usNIC properties:

```
server # scope chassis
server /chassis # show adapter
server /chassis # scope adapter 2
server /chassis/adapter # scope host-eth-if eth0
server /chassis/adapter/host-eth-if # scope usnic-config 0
server /chassis/adapter/host-eth-if/usnic-config # set usnic-count 32
server /chassis/adapter/host-eth-if/usnic-config # commit
Committed settings will take effect upon the next server reset
server /chassis/adapter/host-eth-if/usnic-config # exit
server /chassis/adapter/host-eth-if # exit
server /chassis/adapter # exit
server /chassis # exit
server # exit
```

Deleting Cisco usNIC from a vNIC

Before You Begin

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

DETAILED STEPS

	Command or Action	Purpose
Step 1	server# scope chassis	Enters chassis command mode.
Step 2	server/chassis# scope adapter <i>index</i>	Enters the command mode for the adapter card at the PCI slot number specified by <i>index</i> . Note Make sure that the server is powered on before you attempt to view or change adapter settings. To view the index of the adapters configured on you server, use the show adapter command.
Step 3	server/chassis/adapter# scope host-eth-if {eth0 eth1}	Enters the command mode for the vNIC. Specify the Ethernet ID based on the number of vNICs that you have configured in your environment. For example, specify eth0 if you configured only one vNIC.
Step 4	Server/chassis/adapter/host-eth-if# delete usnic-config 0	Deletes the Cisco usNIC configuration for the vNIC.

	Command or Action	Purpose
Step 5	Server/chassis/adapter/host-eth-if# commit	Commits the transaction to the system configuration Note The changes take effect when the server is rebooted.

This example shows how to delete the Cisco usNIC configuration for a vNIC:

```
server # scope chassis
server/chassis # show adapter
server/chassis # scope adapter 1
server/chassis/adapter # scope host-eth-if eth0
server/chassis/adapter/host-eth-if # delete usnic-config 0
server/chassis/host-eth-if/iscsi-boot *# commit
New host-eth-if settings will take effect upon the next adapter reboot

server/chassis/host-eth-if/usnic-config #
```

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

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, use the grub2 command line tool to add `intel_iommu=on` to the `GRUB_CMDLINE_LINUX_DEFAULT` option found in `/etc/default/grub` configuration file then run the grub2 command below to apply the changes.

```
run grub2-mkconfig -o /boot/grub2/grub.cfg
```

For Ubuntu, use the grub2 command line tool to 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-grub
```

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.
- **libdaplusnic_verbs-{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 Cisco usNIC transport.
- **libfabric-cisco-devel-{version}.x86_64.rpm**— Development headers for the Libfabric package with built-in 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).

**Note**

openmpi-cisco-{version}.x86_64.rpm—Cisco usNIC — Open MPI with the Cisco usNIC BTL MPI transport is available for download at <https://www.open-mpi.org/>.

Before You Begin

Make sure that you have configured the Cisco usNIC properties in CIMC. For more information about how to configure the properties, see *Configuring Cisco usNIC*.

You must also make sure that the host OS distribution on which you want to install Cisco usNIC has a supported version of the Cisco enic driver installed. The Cisco enic driver is the Linux kernel networking driver for the Cisco VIC SR-IOV Ethernet NIC.

-
- Step 1** Upgrade to the latest version of the enic driver included in the Cisco UCS ISO for your Linux distribution.
- Step 2** Install the Cisco usNIC software packages from the Cisco UCS Drivers ISO for your Linux distribution.
- Step 3** **# chkconfig rdma on**
Enables Linux RDMA services. Once enabled, RDMA services will be started automatically after a system reboot.
- 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.
- 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. For more information about how to load the modules, see *Manually Loading the Kernel Modules for Cisco usNIC*.
-

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.

DETAILED STEPS

	Command or Action	Purpose
Step 1	# rmmod enic	Unloads the existing enic driver module.
		Note Make sure that you are not logged into the OS using the enic driver, for example, via SSH to a VIC IP interface. Otherwise, your network connection will get disconnected. Alternatively, you can log in to the server using the KVM to perform this step.

	Command or Action	Purpose
Step 2	# modprobe enic	Loads the enic driver module.
Step 3	# dmesg grep 'Cisco VIC Ethernet NIC Driver'	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.
Step 4	# modprobe usnic_verbs	Loads the usnic_verbs driver module.
Step 5	# lsmod grep usnic_verbs	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.

Upgrading the Linux Software Packages for Cisco usNIC

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

Uninstalling Linux Software Packages for Cisco usNIC

-
- Step 1** Uninstall the following usNIC software packages:
- libusnic_verbs
 - libfabric-cisco-devel
 - libfabric-cisco
 - usd_tools
 - usnic_tools
 - libdaplusnic
 - kmod-usnic_verbs
- Step 2** Reboot your Cisco server.
-

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.

Installing MPI

Installing Community Open MPI

Before You Begin

Install the `kmod-usnic_verbs`, `libfabric-cisco`, and `libfabric-cisco-devel` RPMs.

-
- 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
```
- Note** Parallel builds are fully supported (which can greatly speed up build times) via the "-j" option. For example, on an idle 16-core server, you can "make -j 32" to allow "make" to utilize up to 32 concurrent build commands.
- 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:

- 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:

- <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 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, and no-drop or platinum QoS system Class with default CoS value 5 in Cisco UCS Manager. Please refer to the *Cisco UCS Manager Network Management guide* section on "Quality of Service".

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.

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 C-Series Rack-Mount Standalone 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.

SUMMARY STEPS

1. Search and verify if the **usnic_verbs** kernel module was loaded during the OS driver installation.
2. View the configuration of Cisco usNIC-enabled NICs.
3. Run the **usnic_check** script to view the installed RPMs and their versions.
4. Verify that the Cisco usNIC network packets are being transmitted correctly between the client and server hosts.
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.

DETAILED STEPS

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.

Important **usnic_verbs** is displayed in the output, showing that this kernel module is loaded and active in the kernel.

```
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, make sure that the correct corresponding Ethernet interface is shown for each usNIC interface, that the link state is UP, and that the number of VFs, CQs per VF, and QPs per VF are all the values that you set in the "Create Cisco usNIC using the CIMC GUI" or "Creating Cisco usNIC using the CIMC CLI" sections.

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**.

```
Rack:
usnic_0:
  Interface:          eth3
  MAC Address:        58:ac:78:f2:84:e8
  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 (1d)
  VFs:                64
  CQ per VF:          6
  QP per VF:          6
  Interrupts per VF:  6
```



```

Max CQ:                384
Max CQ Entries:        65535
Max QP:                384
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:         58:ac:78:f2:84:e9
  IP Address:          10.10.10.31
  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(1d)
  VFs:                 64
  CQ per VF:           6
  QP per VF:           6
  Interrupts per VF:   6
  Max CQ:              384
  Max CQ Entries:      65535
  Max QP:              384
  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
enic RPM version 2.3.0.39 installed
usnic_verbs RPM version 1.1.527.0.rhel6u5 installed
libdaplusnic RPM version 2.0.39cisco1.0.527.0 installed
libfabric RPM version 1.4.0cisco1.0.527.0.rhel6u5 installed
libusnic_verbs RPM version 2.0.3.527.0.rhel6u5 installed

```

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: 0000:06:0.0, eth11, 04:62:73:c5:3a:a6, 64 VFs
Per VF: 6 WQ, 6 RQ, 6 CQ, 4 INT

In use:
0 VFs, 0 QPs, 0 CQs

usnic_1: 0000:07:0.0, eth10, 04:62:73:c5:3a:a7, 64 VFs
Per VF: 6 WQ, 6 RQ, 6 CQ, 4 INT

In use:

```

```
0 VFs, 0 QPs, 0 CQs
```

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

```
<server>$ ip addr show dev eth4 | grep "inet[^6]"
    inet 10.1.0.1/16 brd 10.1.255.255 scope global eth4
```

- 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:
/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
/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 The **fi_pingpong** program 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-release/v1.8/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-release/v1.8/examples/ring_c.c
--2015-04-23 10:11:42-- https://raw.githubusercontent.com/open-mpi/ompi-release/v1.8/examples/ring_c.c
Resolving raw.githubusercontent.com... 199.27.74.133
Connecting to raw.githubusercontent.com|199.27.74.133|:443... connected.
WARNING: certificate common name \u201cwww.github.com\u201d doesn\u2019t match requested host name
\u201craw.githubusercontent.com\u201d.
HTTP request sent, awaiting response... 200 OK
Length: 2418 (2.4K) [text/plain]
```

```

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

$ mpiexec --host host1,host2 -n 4 ./ring_c
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 **mpiexec** 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 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

A previously installed version can cause this error.

Solution

- 1 List all the installed versions using the following command: **rpm -qa|grep usnic_verbs**
- 2 Uninstall all versions using the following command: **rpm -e**
- 3 Make sure that the module has been removed.
- 4 Re-install all the RPMs.

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.

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
```

