



## **IP over InfiniBand (IPoIB)**

# Table of contents

## Upper Layer Protocol (ULP)

---

## Enhanced IPoIB

---

## Port Configuration

---

## IPoIB Configuration

---

### IPoIB Configuration Based on DHCP

---

#### DHCP Server

---

#### DHCP Client (Optional)

---

#### Static IPoIB Configuration

---

#### Manually Configuring IPoIB

---

## Sub-interfaces

---

### Creating a Subinterface

---

### Removing a Subinterface

---

## Verifying IPoIB Functionality

---

## Bonding IPoIB

---

## Dynamic PKey Change

---

## Precision Time Protocol (PTP) over IPoIB

---

## One Pulse Per Second (1PPS) over IPoIB

---

## Upper Layer Protocol (ULP)

The IP over IB (IPoIB) ULP driver is a network interface implementation over InfiniBand. IPoIB encapsulates IP datagrams over an InfiniBand Datagram transport service. The IPoIB driver, `ib_ipoib`, exploits the following capabilities:

- VLAN simulation over an InfiniBand network via child interfaces
- High Availability via Bonding
- Varies MTU values:
  - up to 4k in Datagram mode
- Uses any ConnectX® IB ports (one or two)
- Inserts IP/UDP/TCP checksum on outgoing packets
- Calculates checksum on received packets
- Support net device TSO through ConnectX® LSO capability to defragment large data- grams to MTU quantas.

IPoIB also supports the following software based enhancements:

- Giant Receive Offload
- NAPI
- Ethtool support

## Enhanced IPoIB

Enhanced IPoIB feature enables offloading ULP basic capabilities to a lower vendor specific driver, in order to optimize IPoIB data path. This will allow IPoIB to support multiple stateless offloads, such as RSS/TSS, and better utilize the features supported, enabling IPoIB datagram to reach peak performance in both bandwidth and latency.

Enhanced IPoIB supports/performs the following:

- Stateless offloads (RSS, TSS)
- Multi queues

- Interrupt moderation
- Multi partitions optimizations
- Sharing send/receive Work Queues
- Vendor specific optimizations
- UD mode only

## Port Configuration

The physical port MTU (indicates the port capability) default value is 4k, whereas the IPoIB port MTU ("logical" MTU) default value is 2k as it is set by the OpenSM.

To change the IPoIB MTU to 4k, edit the OpenSM partition file in the section of IPoIB setting as follow:

```
Default=0xffff, ipoib, mtu=5 : ALL=full;
```

where:

"mtu=5" indicates that all IPoIB ports in the fabric are using 4k MTU, ("mtu=4" indicates 2k MTU)

## IPoIB Configuration

Unless you have run the installation script `mlnxofedinstall` with the flag `'-n'`, then IPoIB has not been configured by the installation. The configuration of IPoIB requires assigning an IP address and a subnet mask to each HCA port, like any other network adapter card (i.e., you need to prepare a file called `ifcfg-ib<n>` for each port). The first port on the first HCA in the host is called interface `ib0`, the second port is called `ib1`, and so on.

IPoIB configuration can be based on DHCP or on a static configuration that you need to supply (see below). You can also apply a manual configuration that persists only until the next reboot or driver restart (see below).

## IPoIB Configuration Based on DHCP

Setting an IPoIB interface configuration based on DHCP is performed similarly to the configuration of Ethernet interfaces. In other words, you need to make sure that IPoIB configuration files include the following line:

- For RedHat:

```
BOOTPROTO=dhcp
```

- For SLES:

```
BOOTPROTO='dhcp'
```

### **Note**

If IPoIB configuration files are included, ifcfg-ib<n> files will be installed under:

`/etc/sysconfig/network-scripts/` on a RedHat machine

`/etc/sysconfig/network/` on a SuSE machine.

### **Note**

A patch for DHCP may be required for supporting IPoIB. For further information, please see the README file available under the docs/dhcp/ directory.

### **Note**

Red Hat Enterprise Linux 7 supports assigning static IP addresses to InfiniBand IPoIB interfaces. However, as these interfaces do not have a normal hardware Ethernet address, a different method of specifying a unique identifier for the IPoIB interface must be used. The standard is to use the option `dhcp-client-identifier=` construct to specify the IPoIB interface's `dhcp-client-identifier` field. The DHCP server host construct supports at most one hardware Ethernet and one `dhcp-client-identifier` entry per host stanza. However, there may be more than one `fixed-address` entry and the DHCP server will automatically respond with an address that is appropriate for the network that the DHCP request was received on.

Standard DHCP fields holding MAC addresses are not large enough to contain an IPoIB hardware address. To overcome this problem, DHCP over InfiniBand messages convey a client identifier field used to identify the DHCP session. This client identifier field can be used to associate an IP address with a client identifier value, such that the DHCP server will grant the same IP address to any client that conveys this client identifier.

The length of the client identifier field is not fixed in the specification. For the *NVIDIA OFED for Linux* package, it is recommended to have IPoIB use the same format that FlexBoot uses for this client identifier.

## DHCP Server

In order for the DHCP server to provide configuration records for clients, an appropriate configuration file needs to be created. By default, the DHCP server looks for a configuration file called `dhcpd.conf` under `/etc`. You can either edit this file or create a new one and provide its full path to the DHCP server using the `-cf` flag (See a file example at `docs/dhcpd.conf`).

The DHCP server must run on a machine which has loaded the IPoIB module. To run the DHCP server from the command line, enter:

```
dhcpd <IB network interface name> -d
```

### Example:

```
host1# dhcpcd ib0 -d
```

## DHCP Client (Optional)

### Note

A DHCP client can be used if you need to prepare a diskless machine with an IB driver.

In order to use a DHCP client identifier, you need to first create a configuration file that defines the DHCP client identifier.

Then run the DHCP client with this file using the following command:

```
dhclient -cf <client conf file> <IB network interface name>
```

Example of a configuration file for the ConnectX (PCI Device ID 26428), called `dhclient.conf`:

```
The value indicates a hexadecimal number interface "ib1" {  
send dhcp-client-identifier  
ff:00:00:00:00:00:02:00:00:02:c9:00:00:02:c9:03:00:00:10:39;  
}
```

Example of a configuration file for InfiniHost III Ex (PCI Device ID 25218), called `dhclient.conf`:

```
The value indicates a hexadecimal number interface "ib1" {  
send dhcp-client-identifier  
20:00:55:04:01:fe:80:00:00:00:00:00:00:02:c9:02:00:23:13:92;  
}
```

***In order to use the configuration file, run***



```
host1# dhclient -cf dhclient.conf ib1
```

## Static IPoIB Configuration

If you wish to use an IPoIB configuration that is not based on DHCP, you need to supply the installation script with a configuration file (using the ‘-n’ option) containing the full IP configuration. The IPoIB configuration file can specify either or both of the following data for an IPoIB interface:

- A static IPoIB configuration
- An IPoIB configuration based on an Ethernet configuration

See your Linux distribution documentation for additional information about configuring IP addresses.

The following code lines are an excerpt from a sample IPoIB configuration file:

```
# Static settings; all values provided by this file  
IPADDR_ib0=10.4.3.175  
NETMASK_ib0=255.255.0.0  
NETWORK_ib0=10.4.0.0  
BROADCAST_ib0=10.4.255.255  
ONBOOT_ib0=1
```



```
# Based on eth0; each '*' will be replaced with a corresponding
octet
# from eth0.
LAN_INTERFACE_ib0=eth0
IPADDR_ib0=10.4.*.*
NETMASK_ib0=255.255.0.0
NETWORK_ib0=10.4.0.0
BROADCAST_ib0=10.4.255.255
ONBOOT_ib0=1
# Based on the first eth<n> interface that is found (for n=0,1,...);
# each '*' will be replaced with a corresponding octet from
eth<n>.
LAN_INTERFACE_ib0=
IPADDR_ib0=10.4.*.*
NETMASK_ib0=255.255.0.0
NETWORK_ib0=10.4.0.0
BROADCAST_ib0=10.4.255.255
ONBOOT_ib0=1
```

## Manually Configuring IPoIB

### Note

This manual configuration persists only until the next reboot or driver restart.

 **To manually configure IPoIB for the default IB partition (VLAN), perform the following steps:**

1. Configure the interface by entering the ifconfig command with the following items:
  - The appropriate IB interface (ib0, ib1, etc.)

- The IP address that you want to assign to the interface
- The netmask keyword
- The subnet mask that you want to assign to the interface

The following example shows how to configure an IB interface:

```
host1$ ifconfig ib0 10.4.3.175 netmask 255.255.0.0
```

2. (Optional) Verify the configuration by entering the ifconfig command with the appropriate interface identifier *ib#* argument.

The following example shows how to verify the configuration:

```
host1$ ifconfig ib0
b0  Link encap:UNSPEC  HWaddr 80-00-04-04-FE-80-00-00-00-00-00-00-00-00-00-00
inet addr:10.4.3.175  Bcast:10.4.255.255  Mask:255.255.0.0
UP BROADCAST MULTICAST  MTU:65520  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

3. Repeat the first two steps on the remaining interface(s).

## Sub-interfaces

You can create sub-interfaces for a primary IPoIB interface to provide traffic isolation. Each such sub-interface (also called a child interface) has a different IP and network addresses from the primary (parent) interface. The default Partition Key (PKey), ff:ff, applies to the primary (parent) interface.

This section describes how to:

- Create a subinterface
- Remove a subinterface

## Creating a Subinterface

In the following procedure, ib0 is used as an example of an IB sub-interface.

***To create a child interface (sub-interface), follow this procedure:***



1. Decide on the PKey to be used in the subnet (valid values can be 0 or any 16-bit unsigned value). The actual PKey used is a 16-bit number with the most significant bit set. For example, a value of 1 will give a PKey with the value 0x8001.
2. Create a child interface by running:

```
host1$ echo <PKey> > /sys/class/net/<IB  
subinterface>/create_child
```

### **Example:**

```
host1$ echo 1 > /sys/class/net/ib0/create_child
```

This will create the interface ib0.8001.

3. Verify the configuration of this interface by running:

```
host1$ ifconfig <subinterface>.<subinterface PKey>
```

Using the example of the previous step:

```

host1$ ifconfig ib0.8001
ib0.8001  Link encap:UNSPEC  HWaddr 80-00-00-4A-FE-80-00-00-00-
00-00-00-00-00-00-00
BROADCAST MULTICAST  MTU:2044  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

```

4. As can be seen, the interface does not have IP or network addresses. To configure those, you should follow the manual configuration procedure described in "[Manually Configuring IPoIB](#)" section above.
5. To be able to use this interface, a configuration of the Subnet Manager is needed so that the PKey chosen, which defines a broadcast address, be recognized.

## Removing a Subinterface

***To remove a child interface (subinterface), run:***



```
echo <subinterface PKey> /sys/class/net/<ib_interface>/delete_child
```

Using the example of [the second step](#) from the previous chapter:

```
echo 0x8001 > /sys/class/net/ib0/delete_child
```

Note that when deleting the interface you must use the PKey value with the most significant bit set (e.g., 0x8000 in the example above).

# Verifying IPoIB Functionality

➤ To verify your configuration and IPoIB functionality are successful, perform the following steps:

1. Verify the IPoIB functionality by using the `ifconfig` command.

The following example shows how two IB nodes are used to verify IPoIB functionality. In the following example, IB node 1 is at 10.4.3.175, and IB node 2 is at 10.4.3.176:

```
host1# ifconfig ib0 10.4.3.175 netmask 255.255.0.0
host2# ifconfig ib0 10.4.3.176 netmask 255.255.0.0
```

2. Enter the ping command from 10.4.3.175 to 10.4.3.176.
3. The following example shows how to enter the ping command:

```
host1# ping -c 5 10.4.3.176
PING 10.4.3.176 (10.4.3.176) 56(84) bytes of data.
64 bytes from 10.4.3.176: icmp_seq=0 ttl=64 time=0.079 ms
64 bytes from 10.4.3.176: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 10.4.3.176: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 10.4.3.176: icmp_seq=3 ttl=64 time=0.049 ms
64 bytes from 10.4.3.176: icmp_seq=4 ttl=64 time=0.065 ms
--- 10.4.3.176 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time
3999ms rtt min/avg/max/mdev = 0.044/0.058/0.079/0.014 ms, pipe 2
```

## Bonding IPoIB

To create an interface configuration script for the `ibX` and `bondX` interfaces, you should use the standard syntax (depending on your OS).

Bonding of IPoIB interfaces is accomplished in the same manner as would bonding of Ethernet interfaces: via the Linux Bonding Driver.

- Network Script files for IPoIB slaves are named after the IPoIB interfaces (e.g: ifcfg-ib0)
- The only meaningful bonding policy in IPoIB is High-Availability (bonding mode number 1, or active-backup)
- Bonding parameter "fail\_over\_mac" is meaningless in IPoIB interfaces, hence, the only supported value is the default: 0

For a persistent bonding IPoIB Network configuration, use the same Linux Network Scripts semantics, with the following exceptions/ additions:

- In the bonding master configuration file (e.g: ifcfg-bond0), in addition to Linux bonding semantics, use the following parameter: MTU=65520

COND

### **Note**

For IPoIB slaves, use MTU=2044. If you do not set the correct MTU or do not set MTU at all, performance of the interface might decrease.

Dynamically Connected Transport (DCT)

- In the bonding slave configuration file (e.g: ifcfg-ib0), use the same Linux Network Scripts semantics. In particular: DEVICE=ib0
- In the bonding slave configuration file (e.g: ifcfg-ib0.8003), the line TYPE=InfiniBand is necessary when using bonding over devices configured with partitions (p\_key)
- For RHEL users:

In /etc/modprobe.d/bond.conf add the following lines:

```
alias bond0 bonding
```

- For SLES users:

It is necessary to update the MANDATORY\_DEVICES environment variable in /etc/sysconfig/network/config with the names of the IPoIB slave devices (e.g. ib0, ib1, etc.). Otherwise, bonding master may be created before IPoIB slave interfaces at boot time.

It is possible to have multiple IPoIB bonding masters and a mix of IPoIB bonding master and Ethernet bonding master. However, It is NOT possible to mix Ethernet and IPoIB slaves under the same bonding master.

### **Note**

Restarting openibd does not keep the bonding configuration via Network Scripts. You have to restart the network service in order to bring up the bonding master. After the configuration is saved, restart the network service by running: /etc/init.d/network restart.

## Dynamic PKey Change

Dynamic PKey change means the PKey can be changed (add/removed) in the SM database and the interface that is attached to that PKey is updated immediately without the need to restart the driver.

If the PKey is already configured in the port by the SM, the child-interface can be used immediately. If not, the interface will be ready to use only when SM adds the relevant PKey value to the port after the creation of the child interface. No additional configuration is required once the child-interface is created.

## Precision Time Protocol (PTP) over IPoIB

This feature allows for accurate synchronization between the distributed entities over the network. The synchronization is based on symmetric Round Trip Time (RTT) between the master and slave devices.

This feature is enabled by default, and is also supported over PKey interfaces.

For more on the PTP feature, refer to [Running Linux PTP with ConnectX-4/ConnectX-5/ConnectX-6](#) Community post.

For further information on Time-Stamping, follow the steps in "[Time-Stamping Service](#)".

## One Pulse Per Second (1PPS) over IPoIB

1PPS is a time synchronization feature that allows the adapter to be able to send or receive 1 pulse per second on a dedicated pin on the adapter card using an SMA connector (SubMiniature version A). Only one pin is supported and could be configured as 1PPS in or 1PPS out.

For further information, refer to [HowTo Test 1PPS on NVIDIA Adapters](#) Community post.

**Notice**  
This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.  
NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.  
Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.  
NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.  
NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.  
NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.  
No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.  
Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.  
THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.  
**Trademarks**  
NVIDIA and the NVIDIA logo are



trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.<br/>

© Copyright 2024, NVIDIA. PDF Generated on 01/15/2025