Chelsio Communications

Accelerate

Chelsio T5/T4 Unified Wire for Linux

Installation and User's Guide

Chelsio Communications (Headquarters)
209 North Fair Oaks Avenue,
Sunnyvale, CA 94085
U.S.A

www.chelsio.com

Tel: 408.962.3600
Fax: 408.962.3661

Chelsio (India) Private Limited
Subramanya Arcade, Floor 3, Tower B
No. 12, Bannerghatta Road,
Bangalore-560029
Karnataka,
India

Tel: +91-80-4039-6800

Chelsio KK (Japan)
Yamato Building 8F,
5-27-3 Sendagaya,
Shibuya-ku,
Tokyo 151-0051,
Japan

Sales
For all sales inquiries please send email to sales@chelsio.com

Support
For all support related questions please send email to support@chelsio.com

Document History

Version	Revision Date
1.0.0	12/08/2011
1.0.1	01/09/2013
1.0.2	01/27/2013
1.0.3	03/26/2013
1.0.4	04/12/2013
1.0.5	06/20/2013
1.0.6	08/17/2013
1.0.7	10/22/2013
1.0.8	03/08/2013
1.0.9	05/15/2013
1.1.0	07/26/2013
1.1.1	08/14/2013
1.1.2	12/06/2013
1.1.3	12/19/2013
1.1.4	03/13/2014
1.1.5	05/02/2014
1.1.6	06/30/2014
1.1.7	10/22/2014
1.1.8	11/04/2014
1.1.9	02/05/2015
1.2.0	03/04/2015
1.2.1	03/25/2015
1.2.2	06/03/2015
1.2.3	08/05/2015
1.2.4	02/29/2016
1.2.5	04/27/2016
1.2.6	08/25/2016
1.2.7	10/07/2016
1.2.8	10/18/2016

TABLE OF CONTENTS

I. Chelsio Unified Wire

1. Introduction

Thank you for choosing Chelsio T5/T4 Unified Wire adapters. These high speed, single chip, single firmware cards provide enterprises and data centers with high performance solutions for various Network and Storage related requirements.

The Terminator 5 (T5) is Chelsio's next generation of highly integrated, hyper-virtualized 40/10GbE controllers. The T5 is built around a programmable protocol-processing engine, with full offload of a complete Unified Wire solution comprising NIC, TOE, iWARP RDMA, iSCSI, FCoE and NAT support. It scales true 40Gb line rate operation from a single TCP connection to thousands of connections, and allows simultaneous low latency and high bandwidth operation thanks to multiple physical channels through the ASIC.

The T4 adapters can fully offload TCP, UDP, iSCSI, iWARP and FCoE over a single Unified Wire. The adapters also fully support SR-IOV, EVB/VNTag, DCB, Traffic Management and Filtering.

Ideal for all data, storage and high performance clustering applications, the T5/T4 Adapters enable a unified fabric over a single wire by simultaneously running all unmodified IP sockets, Fibre Channel and InfiniBand applications over Ethernet at line rate.

Designed for deployment in virtualized data centers, cloud service installations and high performance computing environments, Chelsio T5/T4 adapters bring a new level of performance metrics and functional capabilities to the computer networking industry.

This document describes the installation, use and maintenance of the software and its various components.

1.1. Features

The Chelsio Unified Wire Package uses a single command to install various drivers and utilities. It consists of the following software:

- **Network (NIC/TOE)**
- **iWARP (RDMA)**
- **iSCSI PDU Offload Target**
- **iSCSI PDU Offload Initiator**
- **Offload Bonding driver**
- **Offload IPv6 driver**
- **Classification and Filtering feature**
- **Utility Tools (cop, cxgbtool, t4_perftune, benchmark tools, sniffer & tracer)**
- **libs (iWARP libraries)**

For detailed instructions on loading, unloading and configuring the drivers/tools please refer to their respective sections.

1.2. Hardware Requirements

The Chelsio Unified Wire software supports Chelsio T5 and T4 Series of Unified Wire Adapters. To know more about the list of adapters supported by each driver, please refer to their respective sections.

1.3. Software Requirements

The Chelsio Unified Wire software has been developed to run on 64-bit Linux based platforms and therefore it is a base requirement for running the driver. To know more about the complete list of operating systems supported by each driver, please refer to their respective sections.

1.4. Package Contents

The Chelsio Unified Wire source package consists of the following files/directories:

- **debrules**: This directory contains packaging specification files required for building Debian packages.
- **docs:** This directory contains support documents - README, Release Notes and User's Guide (this document) for the software.
- **libs:** This directory is for libraries required to install the WD-UDP and iWARP drivers. The libibverbs library has implementation of RDMA verbs which will be used by iWARP applications for data transfers. The librdmacm library works as an RDMA connection manager. The libcxgb4 library works as an interface between the above mentioned generic libraries and Chelsio iWARP driver. The libcxgb4_sock library is a LD_PRELOAD-able library that accelerates UDP Socket communications transparently and without recompilation of the user application.
- **OFED**: This directory contains supported OFED packages.
- **scripts:** Support scripts used by the Unified Wire Installer.
- **src:** Source code for different drivers.
- **support:** This directory contains source files for the dialog utility.
- **tools:**
    - **autoconf-x.xx**: This directory contains the source for Autoconf tool needed for WD-UDP and iWARP libraries.
    - **benchmarks:** This directory contains various benchmarking tools to measure throughput and latency of various networks.
    - **chelsio_adapter_config**: This directory contains scripts and binaries needed to configure Chelsio 40G Adapters.

- **cop:** The cop tool compiles offload policies into a simple program form that can be loaded into the kernel and interpreted. These offload policies are used to determine the settings to be used for various connections. The connections to which the settings are applied are based on matching filter specifications. Please find more details on this tool in its manual page (run `man cop` command).
- **cudbg:** Chelsio Unified Debug tool which facilitates collection and viewing of various debug entities like register dump, Devlog, CIM LA, etc.
- **cxgbtool:** The cxgbtool queries or sets various aspects of Chelsio network interface cards. It complements standard tools used to configure network settings and provides functionality not available through such tools. Please find more details on this tool in its manual page (run `man cxgbtool` command).
- **rdma_tools:** This directory contains iWARP benchmarking tools.
- **90-rdma.rules:** This file contains udev rules needed for running RDMA applications as a non-root user.
- **chdebug:** This script collects operating system environment details and debug information which can be sent to the support team, to troubleshoot Chelsio hardware/software related issues.
- **chiscsi_set_affinity.sh**: This shell script is used for mapping iSCSI Worker threads to different CPUs.
- **chsetup:** The chsetup tool loads NIC, TOE and iWARP drivers, and creates WD-UDP configuration file.
- **chstatus:** This utility provides status information on any Chelsio NIC in the system.
- **t4_latencytune.sh**: Script used for latency tuning of Chelsio Adapters.
- **t4_perftune.sh:** This shell script is to tune the system for higher performance. It achieves it through modifying the IRQ-CPU binding. This script can also be used to change Tx coalescing settings.
- **t4-forward.sh:** RFC2544 Forward test tuning script.
- **uname_r:** This file is used by *chstatus* script to verify if the Linux platform is supported or not.

- **install.py, dialog.py:** Python scripts needed for the GUI installer.
- **EULA:** Chelsio's End User License Agreement
- **install.log:** File containing installation summary.
- **Makefile:** The Makefile for building and installing from the source.
- **sample_machinefile:** Sample file used during iWARP installation on cluster nodes.

2. Hardware Installation

Follow these steps to install Chelsio adapter in your system:

1. Shutdown/power off your system.
2. Power off all remaining peripherals attached to your system.
3. Unpack the Chelsio adapter and place it on an anti-static surface.
4. Remove the system case cover according to the system manufacturer's instructions.
5. Remove the PCI filler plate from the slot where you will install the Ethernet adapter.
6. For maximum performance, it is highly recommended to install the adapter into a PCIe x8/x16 slot.
7. Holding the Chelsio adapter by the edges, align the edge connector with the PCI connector on the motherboard. Apply even pressure on both edges until the card is firmly seated. It may be necessary to remove the SFP (transceiver) modules prior to inserting the adapter.
8. Secure the Chelsio adapter with a screw, or other securing mechanism, as described by the system manufacturer's instructions. Replace the case cover.
9. After securing the card, ensure that the card is still fully seated in the PCIE x8 slot as sometimes the process of securing the card causes the card to become unseated.
10. Connect a fiber cable, multi-mode for short range (SR) optics or single-mode for long range (LR) optics, to the 40/10Gb Ethernet adapter or regular Ethernet cable for the 1Gb Ethernet adapter.
11. Power on your system.
12. Run *update-pciids* command to download the current version of PCI ID list

```
[root@host~]# update-pciids
%   Total   %  Received  %  Xferd  Average Speed  Time    Time  Time Current
                                   Dload  Upload Total   Spent Left  Speed
100 198k  100  198k     0   0     491k   0 --:--:-- --:--:-- --:--:-- 626k
Done.
```

13. Verify if the adapter was installed successfully by using the *lspci* command
    a. For T5 adapters:

```
[root@host~]# lspci |grep -i Chelsio
07:00.0 Ethernet controller: Chelsio Communications Inc T520-LL-CR Unified
Wire Ethernet Controller
07:00.1 Ethernet controller: Chelsio Communications Inc T520-LL-CR Unified
Wire Ethernet Controller
07:00.2 Ethernet controller: Chelsio Communications Inc T520-LL-CR Unified
Wire Ethernet Controller
07:00.3 Ethernet controller: Chelsio Communications Inc T520-LL-CR Unified
Wire Ethernet Controller
07:00.4 Ethernet controller: Chelsio Communications Inc T520-LL-CR Unified
```

```
Wire Ethernet Controller
07:00.5 SCSI storage controller: Chelsio Communications Inc T520-LL-CR
Unified Wire Storage Controller
07:00.6 Fibre Channel: Chelsio Communications Inc T520-LL-CR Unified Wire
Storage Controller
```

b. And for T4 adapters:

```
[root@host~]# lspci | grep –i Chelsio
03:00.0 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.1 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.2 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.3 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.4 Ethernet controller: Chelsio Communications Inc T420-CR Unified Wire
Ethernet Controller
03:00.5 SCSI storage controller: Chelsio Communications Inc T420-CR Unified
Wire Storage Controller
03:00.6 Fibre Channel: Chelsio Communications Inc T420-CR Unified Wire
Storage Controller
03:00.7 Ethernet controller: Chelsio Communications Inc Device 0000
```

For Chelsio T5/T4 adapters, the physical functions are currently assigned as:

- Physical functions 0 - 3: for the SR-IOV functions of the adapter
- Physical function 4: for all NIC functions of the adapter
- Physical function 5: for iSCSI
- Physical function 6: for FCoE
- Physical function 7: Currently not assigned

Once Unified Wire package is installed and loaded, examine the output of `dmesg` to see if the card is discovered.

- For T5 adapters:

```
eth2: Chelsio T520-LL rev 1 1000/10GBASE-SFP RNIC MSI-X, Offload capable
0000:07:00.4: S/N: RE12130097, P/N: 11011675004
```

- And, for T4 adapters:

```
eth0: Chelsio T420-CR rev 2 1000/10GBASE-SFP RNIC MSI-X, Offload capable
0000:04:00.4: S/N: PT18111226, P/N: 110112140D0
```

The above outputs indicate the hardware configuration of the adapters as well as the Serial numbers.

Note *Network device names for Chelsio's physical ports are assigned using the following convention: the port farthest from the motherboard will appear as the first network interface. However, for T5 40G and T420-BT adapters, the association of physical Ethernet ports and their corresponding network device names is opposite. For these adapters, the port nearest to the motherboard will appear as the first network interface.*

3. Software/Driver Installation

The following table describes the various *configuration tuning options* available during installation and drivers/software installed with each option by default:

T5/T4 Configuration Tuning Option	Description	Driver/Software installed
Unified Wire	Configures adapters to run multiple protocols like NIC/TOE, iWARP and iSCSI simultaneously.	NIC/TOE, iWARP, iSCSI Target, iSCSI Initiator, IPv6, Bonding, Filtering
Low latency Networking	Configures adapters to run NIC/TOE and iWARP traffic with low latency specially needed for financial applications.	NIC/TOE, iWARP, IPv6, Bonding, Filtering
High capacity RDMA	Configures adapters to establish a large number of RDMA connections.	NIC/TOE, iWARP, Bonding, IPv6, Filtering
RDMA Performance	Improves RDMA performance on T5/T4 adapters.	NIC/TOE, iWARP
High capacity TOE	Configures adapters to establish a large number of TOE connections.	NIC/TOE, Bonding, IPv6, Filtering
iSCSI Performance*	Improves iSCSI performance on T5 adapters.	NIC/TOE, iSCSI Target, iSCSI Initiator, Bonding
T5 Wire Direct Latency*	Configures T5 adapters to provide low Wire Direct latency.	NIC/TOE, iWARP
T5 Hash Filter*	Configures T5 adapters to create more filters.	NIC, Filtering

* Supported only on T5 adapters.

3.1. Pre-requisites

Depending on the component you choose to install, please ensure that the following requirements are met, before proceeding with the installation.

- If you want to install OFED with NFS-RDMA support, please refer “Setting up NFS-RDMA” in **iWARP (RDMA)** (Click here).
- If you’re planning to install iSCSI PDU Offload Initiator, please install openssl-devel package.

3.2. Enabling RDMA on ARM Platforms

RDMA is disabled by default in RHEL 7.2 build of ARM architecture. To enable this feature, follow the steps mentioned below:

i. Download the kernel source package and extract it.
ii. Create a kernel configuration file.

```
[root@host~]# make oldconfig
```

iii. The above command will create a configuration file *.config* in the same location. Edit the file and enable the following parameters as follows:

```
CONFIG_NET_VENDOR_CHELSIO=y
CONFIG_INFINIBAND=y
```

iv. Compile the kernel.
v. During kernel compilation, please ensure that the following parameters are set as follows:

```
CONFIG_CHELSIO_T1=m
CONFIG_CHELSIO_T1_1G=y
CONFIG_CHELSIO_T3=m
CONFIG_CHELSIO_T4=m
CONFIG_CHELSIO_T4VF=m
CONFIG_INFINIBAND_USER_MAD=m
CONFIG_INFINIBAND_USER_ACCESS=m
CONFIG_INFINIBAND_USER_MEM=y
CONFIG_INFINIBAND_CXGB3=m
CONFIG_INFINIBAND_CXGB3_DEBUG=y
CONFIG_INFINIBAND_CXGB4=m
CONFIG_SCSI_CXGB3_ISCSI=m
CONFIG_SCSI_CXGB4_ISCSI=m
```

vi. Install the kernel.
vii. Reboot into the newly installed kernel.

3.3. Installing Chelsio Unified Wire

3.3.1. GUI mode (with Dialog utility)

i. Download the tarball ChelsioUwire-x.xx.x.x.tar.gz from Chelsio Download Center, http://service.chelsio.com/

ii. Untar the tarball using the following command:

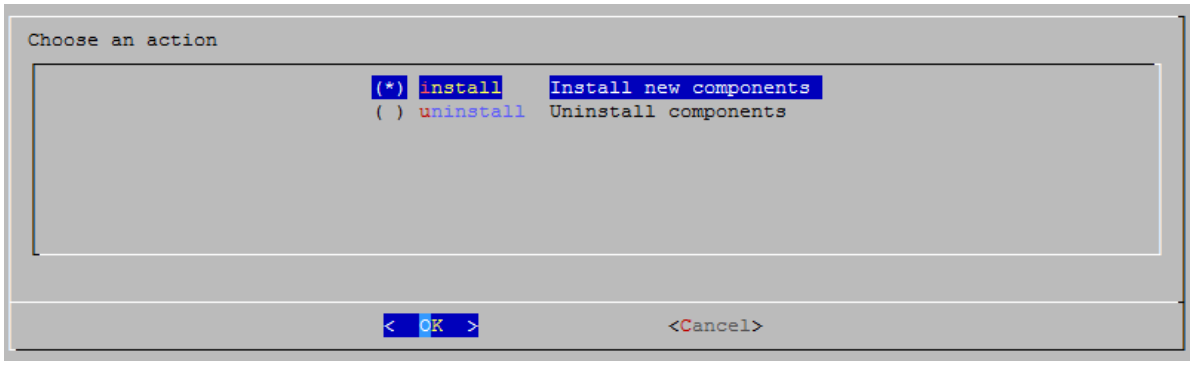
```
[root@host~]# tar zxvfm ChelsioUwire-x.xx.x.x.tar.gz
```

iii. Change your current working directory to Chelsio Unified Wire package directory and run the following script to start the GUI installer:

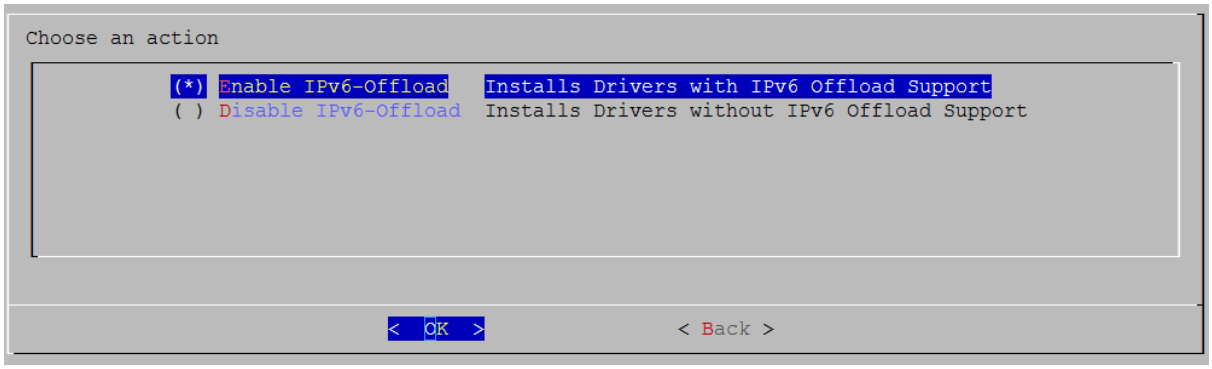
```
[root@host~]# ./install.py
```

iv. If **Dialog** utility is present, you can skip to step (v). If not, press 'y' to install it when the installer prompts for input.

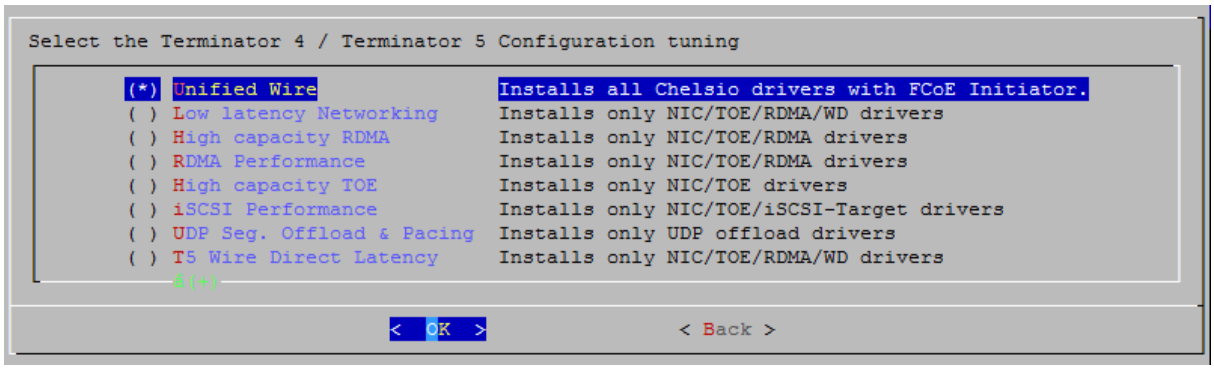
v. Select "install" under "Choose an action"



vi. Select *Enable IPv6-Offload* to install drivers with IPv6 Offload support or *Disable IPv6-offload* to continue installation without IPv6 offload support.

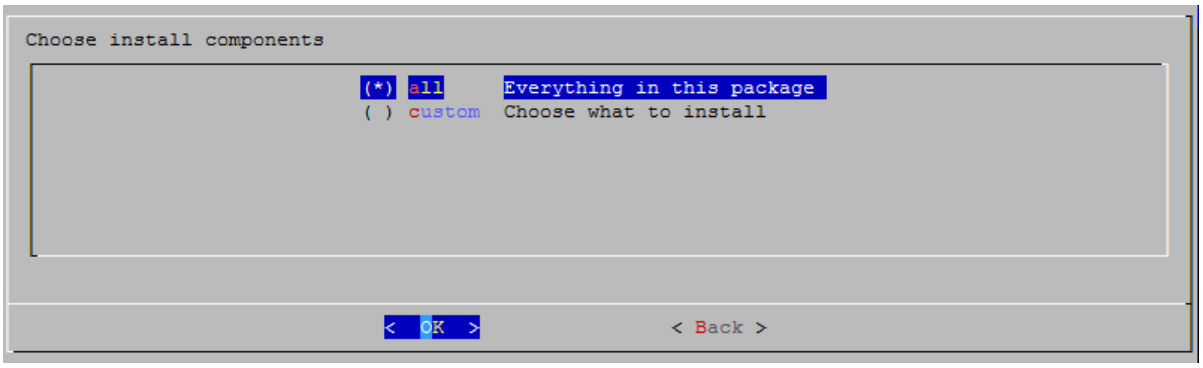


vii. Select the required T5/T4 configuration tuning option:



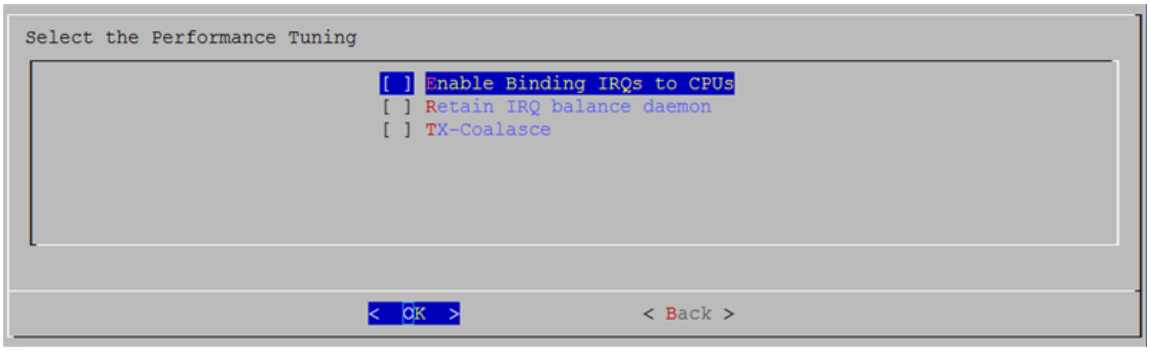
Note *The tuning options may vary depending on the Linux distribution.*

viii. Under “Choose install components”, select “all” to install all the related components for the option chosen in step (vii) or select “custom” to install specific components.



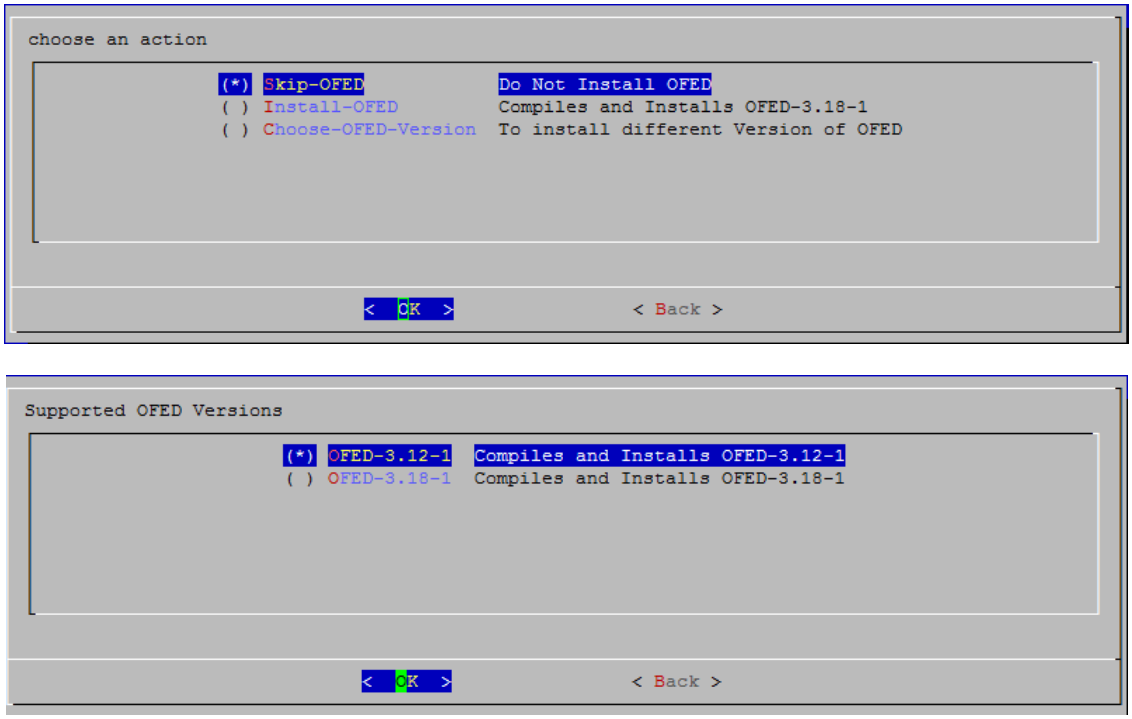
Important *To install benchmark tools, please select “custom option”.*

ix. Select the required performance tuning option.
   a. Enable Binding IRQs to CPUs: Bind MSI-X interrupts to different CPUs and disable IRQ balance daemon.
   b. *Retain IRQ balance daemon*: Do not disable IRQ balance daemon.
   c. *TX-Coalasce*: Write tx_coal=2 to modprobe.d/conf.



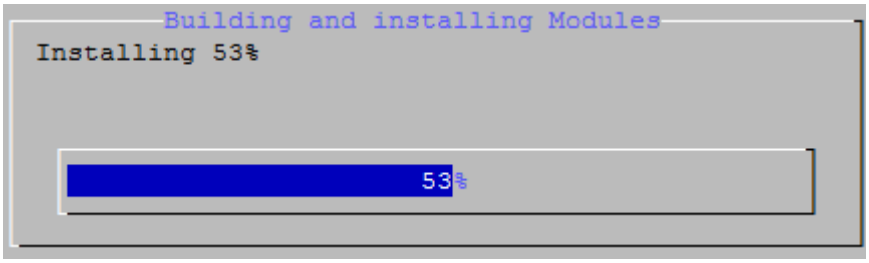
Note *For more information on the Performance tuning options, please refer to* Performance Tuning *section of the* ***Network (NIC/TOE)*** *chapter.*

x. If you already have the required version of OFED software installed, you can skip this step by selecting *Skip-OFED*.
To install OFED-3.18-1 choose the *Install-OFED* option. To install OFED-3.12-1, select *Choose-OFED-Version* and then *OFED-3.12-1*.

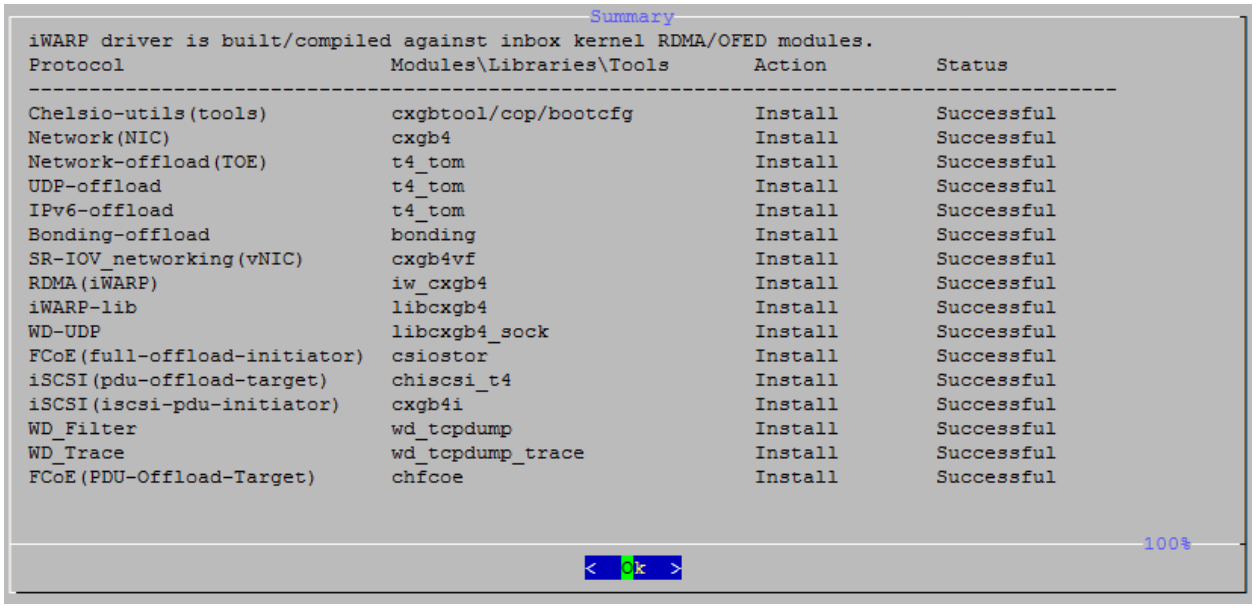


Note *This step will be prompted only for OFED supported platforms.*

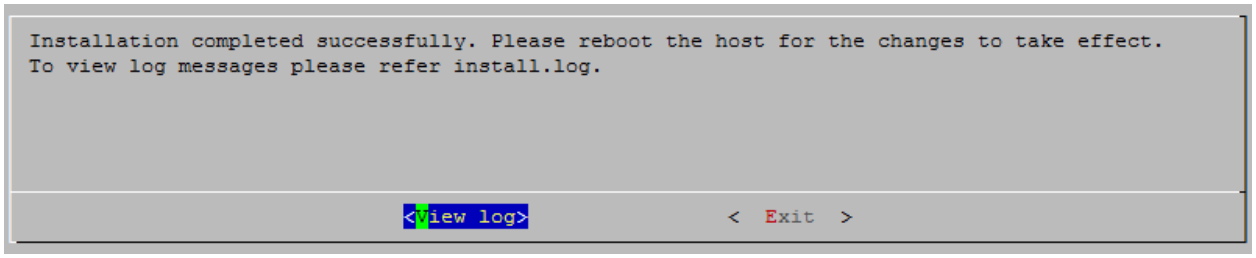
xi. The selected components will now be installed:



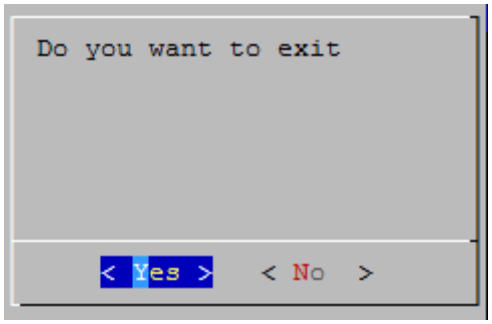
xii. After successful installation, summary of installed components will be displayed.



xiii. Select “View log” to view the installation log or “Exit” to continue.



xiv. Select “Yes” to exit the installer or “No” to go back.



xv. Reboot your machine for changes to take effect.

Note *Press Esc or Ctrl+C to exit the installer at any point of time.*

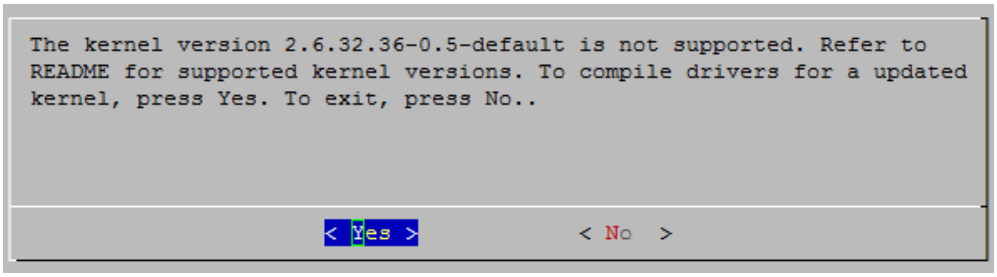
3.3.1.1. Installation on updated kernels

If the kernel version on your Linux distribution is updated, follow the steps mentioned below to install the Unified Wire package:

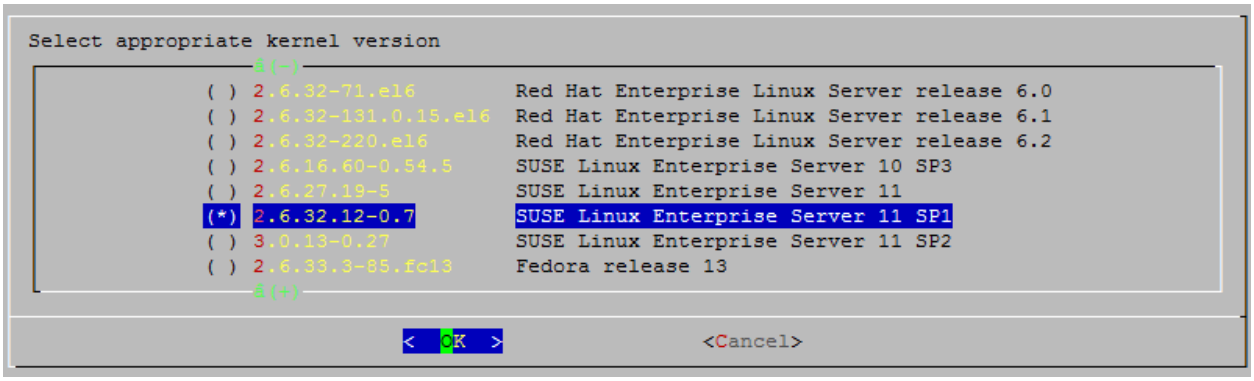
i. Change your current working directory to Chelsio Unified Wire package directory and run the following script to start the GUI installer:

```
[root@host~]# ./install.py
```

ii. Select “Yes” to continue with the installation on the updated kernel or “No” to exit.



iii. Select the nearest supported kernel version from the list and select “OK”.



iv. Follow steps (v) to (xv) mentioned in the previous section.

3.3.2. CLI mode (without Dialog utility)

If your system does not have **Dialog** or you choose not to install it, follow the steps mentioned below to install the Unified Wire package:

i. Download the tarball ChelsioUwire-x.xx.x.x.tar.gz from Chelsio Download Center, http://service.chelsio.com/

ii. Untar the tarball using the following command:

```
[root@host~]# tar zxvfm ChelsioUwire-x.xx.x.x.tar.gz
```

iii. Change your current working directory to Chelsio Unified Wire package directory and run the following script to start the installer:

```
[root@host~]# ./install.py
```

iv. When the installer prompts you for your input, press 'n' to continue installation without the **Dialog** utility.
v. Enter the number corresponding to the Configuration tuning option in the Input field and press Enter.
vi. If you already have the required version of OFED software installed, you can skip this step. To install OFED-3.18-1 choose the *Install-OFED* option. To skip this step, select *Skip-OFED*.

Note *This step will be prompted only for OFED supported platforms.*

vii. The selected components will now be installed.
After successful installation you can press 1 to view the installation log. Press any other key to exit from the installer.

Important *To customize the installation, view the help by typing* `[root@host~]#./install.py –h`

viii. Reboot your machine for changes to take effect.

3.3.2.1. iWARP driver installation on Cluster nodes

Important *Please make sure that you have enabled password less authentication with ssh on the peer nodes for this feature to work.*

Chelsio's Unified Wire package allows installing iWARP drivers on multiple Cluster nodes with a single command. Follow the procedure mentioned below:

i. Create a file (*machinefilename*) containing the IP addresses or hostnames of the nodes in the cluster. You can view the sample file, *sample_machinefile,* provided in the package to view the format in which the nodes have to be listed.
ii. Now, execute the following command:

```
[root@host~]# ./install.py -C  -m <machinefilename>
```

iii. Select the required T5/T4 configuration tuning option. The tuning options may vary depending on the Linux distribution.

iv. Select the required Cluster Configuration.

v. If you already have the required version of OFED software installed, you can skip this step. To install OFED-3.18-1 choose the *Install-OFED* option. To skip this step, select *Skip-OFED*.

vi. The selected components will now be installed.

The above commands will install iWARP (*iw_cxgb4*) and TOE (*t4_tom*) drivers on all the nodes listed in the *machinefilename* file.

3.3.3. CLI mode

i. Download the tarball ChelsioUwire-x.xx.x.x.tar.gz from Chelsio Download Center, http://service.chelsio.com/

ii. Untar the tarball using the following command:

```
[root@host~]# tar zxvfm ChelsioUwire-x.xx.x.x.tar.gz
```

iii. Change your current working directory to Chelsio Unified Wire package directory and build the source using:

```
[root@host~]# make
```

iv. Install the drivers, tools and libraries using the following command:

```
[root@host~]# make install
```

v. The default configuration tuning option is *Unified Wire.* The configuration tuning can be selected using the following commands:

```
[root@host~]# make CONF=<T5/T4 configuration>
[root@host~]# make CONF=<T5/T4 configuration> install
```

Important *Steps (iv) and (v) mentioned above will NOT install benchmark tools. They will have to be installed manually.*

Please refer to section **CLI mode (individual drivers)** *for instructions on installing them.*

Note *To view the different configuration tuning options, view help by typing*
`[root@host~]#make help`

vi. Reboot your machine for changes to take effect.

3.3.3.1. Installation on updated kernels

If the kernel version on your Linux distribution is updated, please execute the following command to install the Unified Wire package:

```
[root@host~]# make UNAME_R=<kernel_version>
```

Where `kernel_version` is the nearest supported kernel version.

For example, if you want to install the package on a RHEL 6 distribution updated to 2.6.32-431.20.3. el6 kernel, run the following commands:

```
[root@host~]# make UNAME_R=2.6.32-431.el6
[root@host~]# make UNAME_R=2.6.32-431.el6 install
```

To view the list of the supported kernel versions, run the following command:

```
[root@host~]# make list_kernels
```

Reboot your machine for changes to take effect.

3.3.4. CLI mode (individual drivers)

You can also choose to install drivers individually. Provided here are steps to build and install NIC, TOE, iWARP drivers and benchmarking tools. To know about other drivers, view help by running `make help`.

- To build and install NIC driver without offload support:

```
[root@host~]# make nic
[root@host~]# make nic_install
```

- To build and install NIC driver with offload support and Offload drivers:

```
[root@host~]# make toe
[root@host~]# make toe_install
```

- To build and install Offload drivers without IPv6 support:

```
[root@host~]# make toe_ipv4
[root@host~]# make toe_ipv4_install
```

- To build and install iWARP driver against outbox OFED:

```
[root@host~]# make iwarp
[root@host~]# make iwarp_install
```

- To build and install all drivers without IPv6 support:

```
[root@host~]# make ipv6_disable=1
[root@host~]# make ipv6_disable=1 install
```

- The default T5/T4 configuration tuning option is *Unified Wire.* The configuration tuning can be selected using the following commands:

```
[root@host~]# make CONF=<T5/T4 configuration> <Build Target>
[root@host~]# make CONF=<T5/T4 configuration> <Install Target>
```

- To build and install drivers along with benchmarks:

```
[root@host~]# make BENCHMARKS=1
[root@host~]# make BENCHMARKS=1 install
```

- The drivers will be installed as RPMs or Debian packages (for ubuntu). To skip this and install drivers:

```
[root@host~]# make SKIP_RPM=1 install
```

Note *To view the different configuration tuning options, view the help by typing* `[root@host~]#make help`

Note *If IPv6 is administratively disabled in the machine, the drivers will be built and installed without IPv6 Offload support by default.*

3.4. Firmware update

The T5 and T4 firmwares are installed on the system, typically in `/lib/firmware/cxgb4`, and the driver will auto-load the firmwares if an update is required. The kernel must be configured to enable userspace firmware loading support:

Device Drivers -> Generic Driver Options -> Userspace firmware loading support

The firmware version can be verified using *ethtool*:

```
[root@host~]# ethtool -i <iface>
```

4. Software/Driver Uninstallation

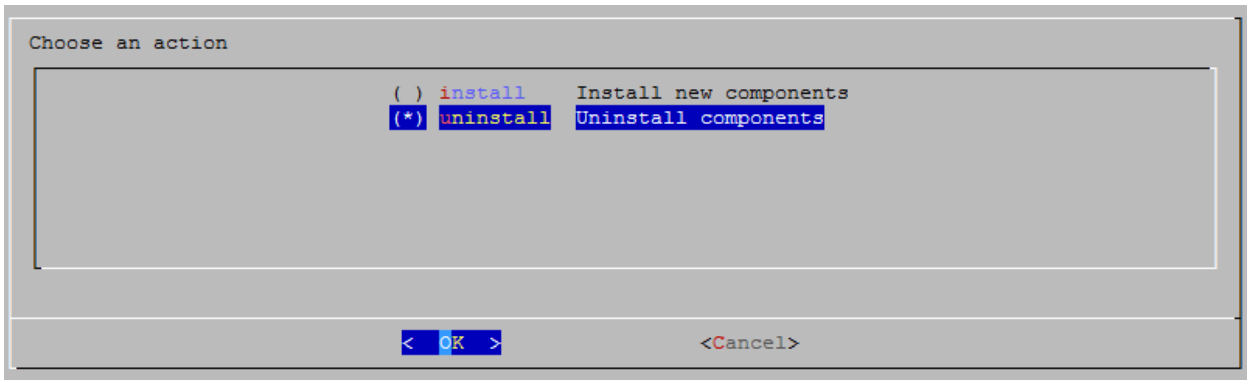
4.1. Uninstalling Chelsio Unified Wire from source

4.1.1. GUI mode (with Dialog utility)

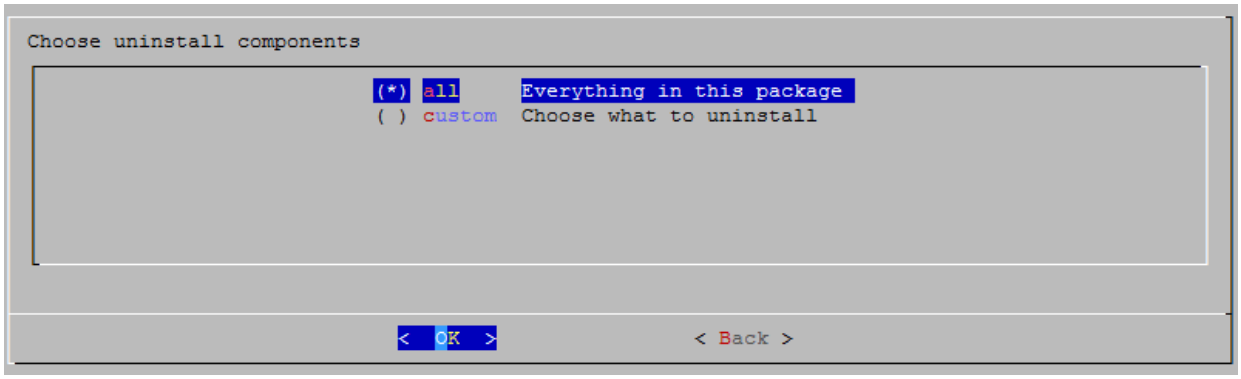
i. Change your current working directory to Chelsio Unified Wire package directory and run the following script to start the GUI installer:

```
[root@host~]# ./install.py
```

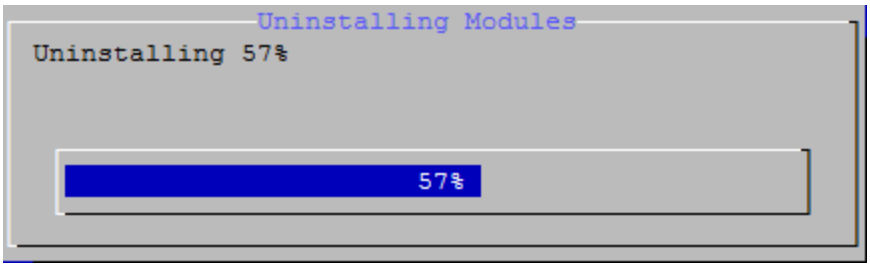
ii. Select “uninstall” , Under “Choose an action”



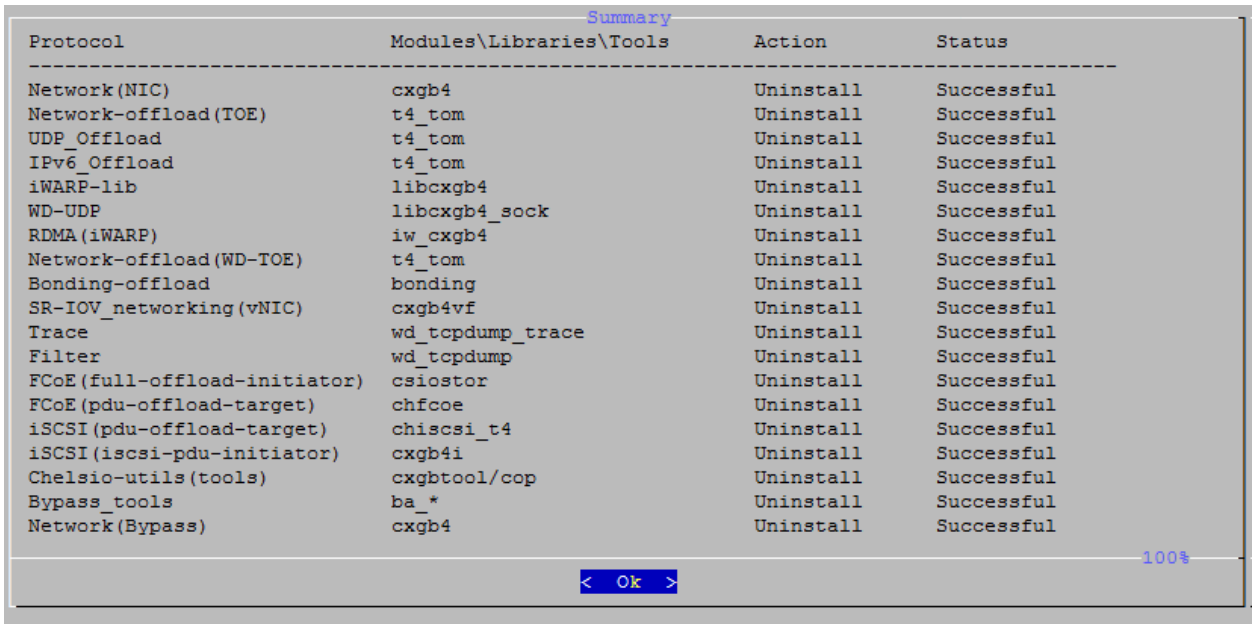
iii. Select “all” to uninstall all the installed drivers, libraries and tools or select “custom” to remove specific components.



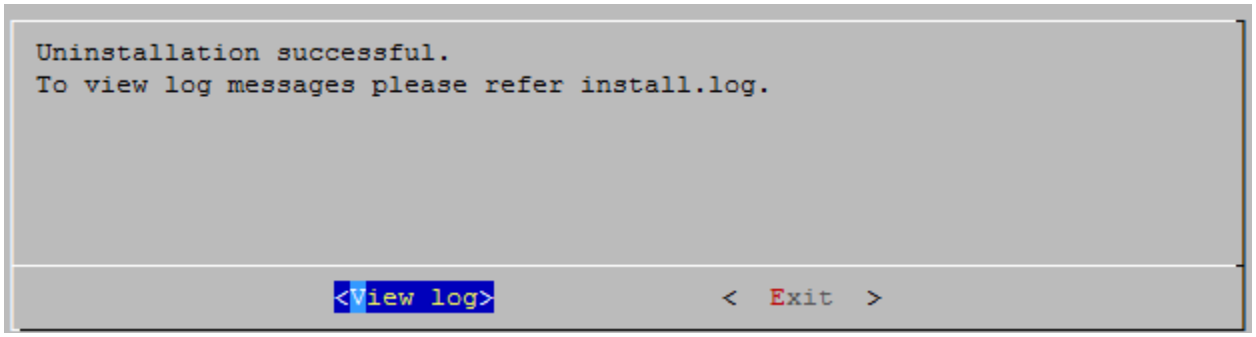
iv. The selected components will now be uninstalled.



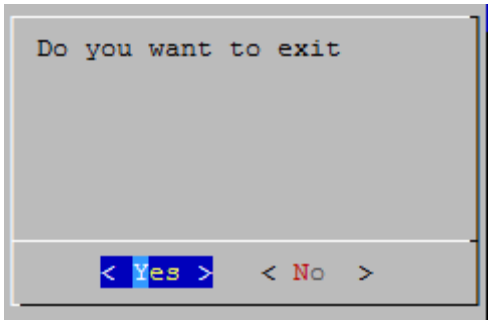
v. After successful uninstalltion, summary of the uninstalled components will be displayed.



vi. Select “View log” to view uninstallation log or “Exit” to continue.



vii. Select “Yes” to exit the installer or “No” to go back.



Note *Press Esc or Ctrl+C to exit the installer at any point of time.*

4.1.2. CLI mode (without Dialog utility)

Run the following script with `-u` option to uninstall the Unified Wire Package:

```
[root@host~]# ./install.py -u <target>
```

Note *View help by typing* `[root@host~]# ./install.py -h` for more information

4.1.3. CLI mode

Change your current working directory to Chelsio Unified Wire package directory and uninstall using the following command:

```
[root@host~]# make uninstall
```

Note *Uninstalling Unified Wire package will not uninstall Unified Wire Manager. Refer to the next section,* **CLI mode (individual drivers)** *to remove the software manually.*

4.1.3.1. iWARP driver uninstallation on Cluster nodes

To uninstal iWARP drivers on multiple Cluster nodes with a single command, run the following command:

```
[root@host~]# ./install.py -C -m <machinefilename> -u all
```

The above command will remove Chelsio iWARP (*iw_cxgb4*) and TOE (*t4_tom*) drivers from all the nodes listed in the *machinefilename* file.

4.1.4. CLI mode (individual drivers/software)

You can also choose to uninstall drivers/software individually. Provided here are steps to uninstall NIC, TOE and iWARP drivers. To know about other drivers, access help by running `make help`

- To uninstall NIC driver:

```
[root@host~]# make nic_uninstall
```

- To uninstall offload driver:

```
[root@host~]# make toe_uninstall
```

- To uninstall iWARP driver:

```
[root@host~]# make iwarp_uninstall
```

5. Configuring Chelsio Network Interfaces

In order to test Chelsio adapters' features it is required to use two machines both with Chelsio's (T5, T4 or both) network adapters installed. These two machines can be connected directly without a switch (back-to-back), or both connected to a switch. The interfaces have to be declared and configured. The configuration files for network interfaces on Red Hat Enterprise Linux (RHEL) distributions are kept under `/etc/sysconfig/network-scripts`.

Note *Some operating systems may attempt to auto-configure the detected hardware and some may not detect all ports on a multi-port adapter. If this happens, please refer to the operating system documentation for manually configuring the network device.*

5.1. Configuring 40G adapters

Chelsio T5 40G adapters can be configured in the following three modes:

i. 2X40Gbps: This is the default mode of operation where each port functions as 40Gbps link. The port nearest to the motherboard will appear as the first network interface (Port 0).
ii. 4X10Gbps: In this mode, port 0 functions as 4 10Gbps links and port 1 is disabled.
iii. QSA: This mode adds support for QSA (QSFP to SFP+) modules, enabling smooth, cost-effective, connections between 40 Gigabit Ethernet adapters and 1 or 10 Gigabit Ethernet networks using existing SFP+ based cabling. The port farthest from the motherboard will appear as the first network interface (Port 0).

To configure/change the mode of operation, use the following procedure:

i. Unload all Chelsio drivers using the *rmmod* command:

```
[root@host~]# rmmod <chelsio_driver>
```

ii. Run the *chelsio_adapter_config* command to detect all T5 40G adapter(s) present in the system.

```
[root@host~]# chelsio_adapter_config
Chelsio T580 card detected
Chelsio T580 PCI devices :
|------------------------------|
| 1 T580-LP-CR    01:00.0      |
| 2 T580-CR       03:00.0      |
| 3 T580-LP-SO-CR 04:00.0      |
|------------------------------|
```

iii. Select the adapter to configure by specifying the adapter index.

iv. Select the required mode:

```
Possible T580 adapter modes:
|------------------------------|
| 1: 2x40G                     |
| 2: 4x10G                     |
| 3: QSA                       |
|------------------------------|
Select mode for adapter (1,2,3):
```

v. Reload the network driver for changes to take effect.

```
[root@host~]# rmmod cxgb4
[root@host~]# modprobe cxgb4
```

Note *In case of T580-SO-CR adapters, reboot the machine for changes to take effect.*

5.2. Configuring network-scripts

A typical interface network-script (e.g. eth0) on RHEL 6.X looks like the following:

```
# file: /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
HWADDR=00:30:48:32:6A:AA
ONBOOT="yes"
NM_CONTROLLED="no"
BOOTPROTO="static"
IPADDR=10.192.167.111
NETMASK=255.255.240.0
```

Note *On earlier versions of RHEL the NETMASK attribute is named IPMASK. Make sure you are using the right attribute name.*

In the case of DHCP addressing the last two lines should be removed and `BOOTPROTO="static"` should be changed to `BOOTPROTO="dhcp"`

The `ifcfg-ethX` files have to be created manually. They are required for bringing the interfaces up and down and attribute the desired IP addresses.

5.3. Creating network-scripts

To spot the new interfaces, make sure the driver is unloaded first. To that point `ifconfig -a | grep HWaddr` should display all non-chelsio interfaces whose drivers are loaded, whether the interfaces are up or not.

```
[root@host~]# ifconfig -a | grep HWaddr
eth0 Link encap:Ethernet HWaddr 00:30:48:32:6A:AA
```

Then load the driver using the `modprobe cxgb4` command (for the moment it does not make any difference whether we are using NIC-only or the TOE-enabling driver). The output of ifconfig should display the T5/T4 interfaces as:

```
[root@host~]# ifconfig -a | grep HWaddr
eth0 Link encap:Ethernet HWaddr 00:30:48:32:6A:AA
eth1 Link encap:Ethernet HWaddr 00:07:43:04:6B:E9
eth2 Link encap:Ethernet HWaddr 00:07:43:04:6B:F1
```

For each interface you can write a configuration file in `/etc/sysconfig/network-scripts`. The `ifcfg-eth1` could look like:

```
# file: /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE="eth1"
HWADDR=00:07:43:04:6B:E9
ONBOOT="no"
NM_CONTROLLED="no"
BOOTPROTO="static"
IPADDR=10.192.167.112
NETMASK=255.255.240.0
```

From now on, the `eth1` interface of the adapter can be brought up and down through the `ifup eth1` and `ifdown eth1` commands respectively. Note that it is of course not compulsory to create a configuration file for every interface if you are not planning to use them all.

5.4. Checking Link

Once the network-scripts are created for the interfaces you should check the link i.e. make sure it is actually connected to the network. First, bring up the interface you want to test using `ifup eth1`. You should now be able to ping any other machine from your network provided it has ping response enabled.

6. Performance Tuning

In order to auto tune the system for best performance, Chelsio recommends:

- Disabling virtualization, c-state technology, VT-d, Intel I/O AT and SR-IOV in the BIOS settings
- Installing the adapter into a PCIe Gen3 x8/x16 slot.
- Installing the **tools** which will copy `t4_perftune.sh` script to **/sbin** directory. Run the script to map the adapter queues to different CPUs:

```
[root@host~]# t4_perftune.sh
```

Also, follow the steps mentioned below to lower your latency:

i. Disable SELinux
ii. Run the following script to disable few services.

```
[root@host~]# t4_latencytune.sh <interface>
```

iii. Set sysctl param *net.ipv4.tcp_low_latency* to 1

```
[root@host~]# sysctl -w net.ipv4.tcp_low_latency=1
```

To optimize your system for different protocols, please refer to their respective chapters.

7. Software/Driver Update

For any distribution specific problems, please check README and Release Notes included in the release for possible workaround.

Please visit Chelsio support web site http://service.chelsio.com/ for regular updates on various software/drivers. You can also subscribe to our newsletter for the latest software updates.

II. Network (NIC/TOE)

1. Introduction

Chelsio's T5 and T4 series of Unified Wire Adapters provide extensive support for NIC operation, including all stateless offload mechanisms for both IPv4 and IPv6 (IP, TCP and UDP checksum offload, LSO - Large Send Offload aka TSO - TCP Segmentation Offload, and assist mechanisms for accelerating LRO - Large Receive Offload).

A high performance fully offloaded and fully featured TCP/IP stack meets or exceeds software implementations in RFC compliance. Chelsio's T5/T4 engine provides unparalleled performance through a specialized data flow processor implementation and a host of features designed for high throughput and low latency in demanding conditions and networking environments.

TCP offload is fully implemented in the hardware, thus freeing the CPU from TCP/IP overhead. The freed CPU can be used for any computing needs. The TCP offload in turn removes network bottlenecks and enables applications to take full advantage of the networking capabilities.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio adapters that are compatible with Chelsio Network driver:

- T580-OCP-SO*
- T520-OCP-SO*
- T520-BT
- T580-CR
- T580-SO-CR*
- T580-LP-CR
- T520-LL-CR
- T520-SO-CR*
- T520-CR
- T540-CR
- T420-CR
- T440-CR
- T422-CR
- T420-SO-CR
- T404-BT
- T420-BCH
- T440-LP-CR
- T420-BT
- T420-LL-CR
- T420-CX

*Only NIC driver supported

1.2. Software Requirements

1.2.1. Linux Requirements

Currently the Network driver is available for the following versions:

- RHEL 7.2, 4.2.0-0.21.el7.aarch64 (ARM64)
- Ubuntu 14.04.3, 3.19.0-25-generic (POWER8)

Other kernel versions have not been tested and are not guaranteed to work.

2. Software/Driver Loading

Important *Please ensure that all inbox drivers are unloaded before proceeding with unified wire drivers.*

The driver must be loaded by the root user. Any attempt to load the driver as a regular user will fail.

2.1. Loading in NIC mode (without full offload support)

To load the Network driver without full offload support, run the following command:

```
[root@host~]# modprobe cxgb4
```

2.2. Loading in TOE mode (with full offload support)

To enable full offload support, run the following command:

```
[root@host~]# modprobe t4_tom
```

Note *Offload support needs to be enabled upon each reboot of the system. This can be done manually as shown above.*

In VMDirect Path environment, it is recommended to load the offload driver using the following command:

```
[root@host~]# modprobe t4_tom vmdirectio=1
```

3. Software/Driver Unloading

3.1. Unloading the NIC driver

To unload the NIC driver, run the following command:

```
[root@host~]# rmmod cxgb4
```

3.2. Unloading the TOE driver

A reboot is required to unload the TOE driver. To avoid rebooting, follow the steps mentioned below:

i. Load *t4_tom* driver with *unsupported_allow_unload* parameter.

```
[root@host~]# modprobe t4_tom unsupported_allow_unload=1
```

ii. Stop all the offloaded traffic, servers and connections. Check for the reference count.

```
[root@host~]# cat /sys/module/t4_tom/refcnt
```

If the reference count is 0, the driver can be directly unloaded. Skip to step (iii)

If the count is non-zero, load a COP policy which disables offload using the following procedure:

a. Create a policy file which will disable offload

```
[root@host~]# cat policy_file
all => !offload
```

b. Compile and apply the output policy file

```
[root@host~]# cop -o no-offload.cop policy_file
[root@host~]# cxgbtool ethX policy no-offload.cop
```

iii. Unload the driver:

```
[root@host~]# rmmod t4_tom
[root@host~]# rmmod toecore
[root@host~]# rmmod cxgb4
```

4. Software/Driver Configuration and Fine-tuning

4.1. Instantiate Virtual Functions (SR-IOV)

To instantiate the Virtual functions, load the cxgb4 driver with `num_vf` parameter with a non-zero value. For example:

```
[root@host~]# modprobe cxgb4 num_vf=1,0,0,0
```

The number(s) provided for `num_vf` parameter specifies the number of Virtual Functions to be instantiated per Physical Function. The Virtual Functions can be assigned to Virtual Machines (Guests). A maximum of 64 Virtual Functions can be instantiated with 16 Virtual Functions per Physical Function. Loading the *cxgb4* driver with `num_vf` parameter loads the *cxgb4vf* module (the driver for Virtual Functions) in the host by default. Hence unload the *cxgb4vf* module (on the host) before assigning Virtual Functions to the Virtual Machines (Guests), using the following command:

```
[root@host~]# rmmod cxgb4vf
```

Note *To get familiar with physical and virtual function terminologies, please refer the PCI Express specification.*

4.2. Enabling Busy waiting

Busy waiting/polling is a technique where a process repeatedly checks to see if an event has occurred, by spinning in a tight loop. By making use of similar technique, Linux kernel provides the ability for the socket layer code to poll directly on an Ethernet device's Rx queue. This eliminates the cost of interrupts and context switching, and with proper tuning allows to achieve latency performance similar to that of hardware.

Chelsio's NIC and TOE drivers support this feature and can be enabled on Chelsio supported devices to attain improved latency.

To make use of BUSY_POLL feature, follow the steps mentioned below:

i. Enable BUSY_POLL support in kernel config file by setting `CONFIG_NET_RX_BUSY_POLL=y`
ii. Enable BUSY_POLL globally in the system by setting the values of following sysctl parameters depending on the number of connections:

```
sysctl -w net.core.busy_read=<value>
sysctl -w net.core.busy_poll=<value>
```

Set the values of the above parameters to 50 for 100 or less connections; and 100 for more than 100 connections.

Note *BUSY_POLL can also be enabled on a per-connection basis by making use of SO_BUSY_POLL option in the socket application code. Refer socket man-page for more details.*

4.3. Performance Tuning

- **Receiver Side Scaling (RSS)**

Receiver Side Scaling enables the receiving network traffic to scale with the available number of processors on a modern networked computer. RSS enables parallel receive processing and dynamically balances the load among multiple processors. Chelsio's T5/T4 network controller fully supports Receiver Side Scaling for IPv4 and IPv6.

This script first determines the number of CPUs on the system and then each receiving queue is bound to an entry in the system interrupt table and assigned to a specific CPU. Thus, each receiving queue interrupts a specific CPU through a specific interrupt now. For example, on a 4-core system, `t4_perftune.sh` gives the following output:

```
[root@host~]# t4_perftune.sh
Discovering Chelsio T4/T5 devices ...
Configuring Chelsio T4/T5 devices ...
Tuning eth7
IRQ table length 4
Writing 1 in /proc/irq/62/smp_affinity
Writing 2 in /proc/irq/63/smp_affinity
Writing 4 in /proc/irq/64/smp_affinity
Writing 8 in /proc/irq/65/smp_affinity
eth7 now up and tuned
...
```

Because there are 4 CPUs on the system, 4 entries of interrupts are assigned. For other T5/T4 network interfaces, you should see similar output message.

Now the receiving traffic is dynamically assigned to one of the system's CPUs through a T5/T4 queue. This achieves a balanced usage among all the processors. This can be verified, for example, by using the **iperf** tool. First set up a server on the receiver host:

```
[root@receiver_host~]# iperf –s
```

Then on the sender host, send data to the server using the iperf client mode. To emulate a moderate traffic workload, use *-P* option to request 20 TCP streams from the server:

```
[root@sender_host~]# iperf -c receiver_host_name_or_IP -P 20
```

Then on the receiver host, look at interrupt rate at `/proc/interrupts`:

```
[root@receiver_host~]# cat /proc/interrupts | grep eth6

Id      CPU0      CPU1      CPU2      CPU3       type              interface
36:   115229         0         0         1   PCI-MSI-edge     eth6 (queue 0)
37:        0    121083         1         0   PCI-MSI-edge     eth6 (queue 1)
38:        0         0    105423         1   PCI-MSI-edge     eth6 (queue 2)
39:        0         0         0    115724   PCI-MSI-edge     eth6 (queue 3)
```

Now interrupts from eth6 are evenly distributed among the 4 CPUs.

Without T5/T4's RSS support, the interrupts caused by network traffic may be distributed unevenly over CPUs. For your information, the traffic produced by the same iperf commands gives the following output in `/proc/interrupts`.

```
[root@receiver_host~]# cat /proc/interrupts | grep eth6

Id      CPU0      CPU1      CPU2      CPU3       type              interface
36:        0         9         0     17418   PCI-MSI-edge     eth6 (queue 0)
37:        0         0     21718      2063   PCI-MSI-edge     eth6 (queue 1)
38:        0         7    391519       222   PCI-MSI-edge     eth6 (queue 2)
39:        1         0        33     17798   PCI-MSI-edge     eth6 (queue 3)
```

Here there are 4 receiving queues from the eth6 interface, but they are not bound to a specific CPU or interrupt entry. Queue 2 has caused a very large number of interrupts on CPU2 while CPU0 and CPU1 are barely used by any of the four queues. Enabling RSS is thus essential for best performance.

Note *Linux's* `irqbalance` *may take charge of distributing interrupts among CPUs on a multiprocessor platform. However,* `irqbalance` *distributes interrupt requests from all hardware devices across processors. For a server with T5/T4 network card constantly receiving large volume of data at 40/10Gbps, the network interrupt demands are significantly high. Under such circumstances, it is necessary to enable RSS to balance the network load across multiple processors and achieve the best performance.*

- **Interrupt Coalescing**

The idea behind Interrupt Coalescing (IC) is to avoid flooding the host CPUs with too many interrupts. Instead of throwing one interrupt per incoming packet, IC waits for 'n' packets to be available in the Rx queues and placed into the host memory through DMA operations before an interrupt is thrown, reducing the CPU load and thus improving latency. It can be changed using the following command:

```
[root@host~]# ethtool -C ethX rx-frames n
```

Note *For more information, run the following command:*
`[root@host~]# ethtool -h`

- **Configuring sysctl, adaptive interrupts, select_queue (NIC)**

i. Turn off *irqbalance*

```
[root@host~]# /etc/init.d/irqbalance stop
```

ii. Add the following sysctl parameters to */etc/sysctl.conf*

```
sysctl -w net.ipv4.tcp_timestamps=0
sysctl -w net.ipv4.tcp_low_latency=1
sysctl -w net.core.netdev_max_backlog=250000
sysctl -w net.core.rmem_max=16777216
sysctl -w net.core.wmem_max=16777216
sysctl -w net.core.rmem_default=16777216
sysctl -w net.core.wmem_default=16777216
sysctl -w net.core.optmem_max=16777216
sysctl -w net.ipv4.tcp_rmem='4096 87380 16777216'
sysctl -w net.ipv4.tcp_wmem='4096 65536 16777216'
```

iii. Bring up the network interfaces and run the following command:

```
[root@host~]# ethtool -C ethXX adaptive-rx on
```

Read back the *ethtool* settings with the following command:

```
[root@host~]# ethtool -c ethXX
```

Output should show *adaptive-rx* as on.

iv. Change *select_queue* parameter's value to 1:

```
[root@host~]# cat /sys/module/cxgb4/parameters/select_queue
0
[root@host~]# echo 1 > /sys/module/cxgb4/parameters/select_queue
[root@host~]# cat /sys/module/cxgb4/parameters/select_queue
1
```

For **TOE** performance, follow the first two steps mentioned above and then set the following *sysctl* parameter:

```
[root@host~]# sysctl -w toe.toe0_tom.delayed_ack=3
```

- **Large Receive Offload / Generic Receive Offload**

Large Receive Offload or Generic Receive Offload is a performance improvement feature at the receiving side. LRO/GRO aggregates the received packets that belong to same stream, and combines them to form a larger packet before pushing them to the receive host network stack. By doing this, rather than processing every small packet, the receiver CPU works on fewer packet headers but with same amount of data. This helps reduce the receive host CPU load and improve throughput in a 40/10Gb network environment where CPU can be the bottleneck.

LRO and GRO are different names to refer to the same receiver packets aggregating feature. LRO and GRO actually differ in their implementation of the feature in the Linux kernel. The feature was first added into the Linux kernel in version 2.6.24 and named Large Receive Offload (LRO). However LRO only works for TCP and IPv4. As from kernel 2.6.29, a new protocol-independent implementation removing the limitation is added to Linux, and it is named Generic Receive Offload (GRO). The old LRO code is still available in the kernel sources but whenever both GRO and LRO are presented GRO is always the preferred one to use.

Please note that if your Linux system has IP forwarding enabled, i.e. acting as a bridge or router, the LRO needs to be disabled. This is due to a known kernel issue.

Chelsio's T5/T4 card supports both hardware assisted GRO/LRO and Linux-based GRO/LRO. `t4_tom` is the kernel module that enables the hardware assisted GRO/LRO. If it is not already in the kernel module list, use the following command to insert it:

```
[root@host~]# lsmod | grep t4_tom
[root@host~]# modprobe t4_tom
[root@host~]# lsmod | grep t4_tom
t4_tom 88378 0 [permanent]
toecore 21618 1 t4_tom
cxgb4 225342 1 t4_tom
```

Then T5/T4's hardware GRO/LRO implementation is enabled.

If you would like to use the Linux GRO/LRO for any reason, first the `t4_tom` kernel module needs to be removed from kernel module list. Please note you might need to reboot your system.

After removing the `t4_tom` module, you can use `ethtool` to check the status of current GRO/LRO settings, for example:

```
[root@host~]# ethtool -k eth6
Offload parameters for eth6:
rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp-segmentation-offload: on
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
```

Now the `generic-receive-offload` option is on. This means GRO is enabled. Please note that there are two offload options here: `generic-receive-offload` and `large-receive-offload`. This is because on this Linux system (RHEL6.0), the kernel supports both GRO and LRO. As mentioned earlier, GRO is always the preferred option when both of them are present. On other systems LRO might be the only available option. Then `ethtool` could be used to switch LRO on and off as well.

When Linux's GRO is enabled, Chelsio's T5/T4 driver provides two GRO-related statistics. They are displayed using the following command:

```
[root@host~]# ethtool -S eth6
...
GROPackets : 0
GROMerged : 897723
...
```

`GROPackets` is the number of held packets. Those are candidate packets held by the kernel to be processed individually or to be merged to larger packets. This number is usually zero. `GROMerged` is the number of packets that merged to larger packets. Usually this number increases if there is any continuous traffic stream present. `ethtool` can also be used to switch off the GRO/LRO options when necessary:

```
[root@host~]# ethtool -K eth6 gro off
[root@host~]# ethtool -k eth6
Offload parameters for eth6:
rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp-segmentation-offload: on
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: off
large-receive-offload: off
```

The output above shows a disabled GRO.

III. iWARP (RDMA)

1. Introduction

Chelsio's T5/T4 engine implements a feature rich RDMA implementation which adheres to the IETF standards with optional markers and MPA CRC-32C.

The iWARP RDMA operation benefits from the virtualization, traffic management and QoS mechanisms provided by T5/T4 engine. It is possible to ACL process iWARP RDMA packets. It is also possible to rate control the iWARP traffic on a per-connection or per-class basis, and to give higher priority to QPs that implement distributed locking mechanisms. The iWARP operation also benefits from the high performance and low latency TCP implementation in the offload engine.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with Chelsio iWARP driver:

- T520-BT
- T580-CR
- T520-LL-CR
- T520-CR
- T580-LP-CR
- T540-CR
- T420-CR
- T440-CR
- T422-CR
- T404-BT
- T440-LP-CR
- T420-LL-CR
- T420-CX

1.2. Software Requirements

1.2.1. Linux Requirements

Currently the iWARP driver is available for the following versions:

- RHEL 7.2, 4.2.0-0.21.el7.aarch64 (ARM64)
- Ubuntu 14.04.3, 3.19.0-25-generic (POWER8)

Other kernel versions have not been tested and are not guaranteed to work

2. Software/Driver Loading

Important *Please ensure that all inbox drivers are unloaded before proceeding with unified wire drivers.*

2.1. Loading iWARP driver

The driver must be loaded by the root user. Any attempt to load the driver as a regular user will fail.

To load the iWARP driver we need to load the NIC driver and core RDMA drivers first. Run the following commands:

```
[root@host~]# modprobe cxgb4
[root@host~]# modprobe iw_cxgb4
[root@host~]# modprobe rdma_ucm
```

3. Software/Driver Unloading

To unload the iWARP driver, run the following command:

```
[root@host~]# rmmod iw_cxgb4
```

Important

openmpi-1.4.3 can cause IMB benchmark stalls due to a shared memory BTL issue. This issue is fixed in openmpi-1.4.5 and later releases. Hence, it is recommended that you download and install the latest stable release from Open MPI's official website, http://www.open-mpi.org

4. Software/Driver Configuration and Fine-tuning

4.1. Testing connectivity with *ping* and *rping*

Load the NIC, iWARP & core RDMA modules as mentioned in Software/Driver Loading section. After which, you will see two or four ethernet interfaces for the T5/T4 device. Configure them with an appropriate ip address, netmask, etc. You can use the Linux *ping* command to test basic connectivity via the T5/T4 interface. To test RDMA, use the *rping* command that is included in the librdmacm-utils RPM:

Run the following command on the server machine:

```
[root@host~]# rping -s -a server_ip_addr -p 9999
```

Run the following command on the client machine:

```
[root@host~]# rping -c –Vv -C10 -a server_ip_addr -p 9999
```

You should see ping data like this on the client:

```
ping data: rdma-ping-0: ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr
ping data: rdma-ping-1: BCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrs
ping data: rdma-ping-2: CDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrst
ping data: rdma-ping-3: DEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstu
ping data: rdma-ping-4: EFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuv
ping data: rdma-ping-5: FGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvw
ping data: rdma-ping-6: GHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwx
ping data: rdma-ping-7: HIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxy
ping data: rdma-ping-8: IJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
ping data: rdma-ping-9: JKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyzA
client DISCONNECT EVENT...
#
```

4.2. Enabling various MPIs

4.2.1. Setting shell for Remote Login

User needs to set up authentication on the user account on all systems in the cluster to allow user to remotely logon or executing commands without password.

Quick steps to set up user authentication:

i. Change to user home directory

```
[root@host~]# cd
```

ii. Generate authentication key

```
[root@host~]# ssh-keygen -t rsa
```

iii. Hit [Enter] upon prompting to accept default setup and empty password phrase
iv. Create authorization file

```
[root@host~]# cd .ssh
[root@host~]# cat *.pub > authorized_keys
[root@host~]# chmod 600 authorized_keys
```

v. Copy directory .ssh to all systems in the cluster

```
[root@host~]# cd
[root@host~]# scp -r /root/.ssh remotehostname-or-ipaddress:
```

4.2.2. Configuration of various MPIs (Installation and Setup)

- **Open MPI (Installation and Setup)**

Open MPI iWARP support is only available in Open MPI version 1.3 or greater.

Open MPI will work without any specific configuration via the openib btl. Users wishing to performance tune the configurable options may wish to inspect the receive queue values. Those can be found in the "Chelsio T4" section of `mca-btl-openib-device-params.ini`. Follow the steps mentioned below to install and configure Open MPI.

i. If not alreay done, install *mpi-selector* tool.

ii. Download the latest stable/feature version of openMPI from OpenMPI website, http://www.open-mpi.org
iii. Untar and change your current working directory to openMPI package directory.
iv. Configure and install as:

```
[root@host~]#./configure --with-openib=/usr  CC=gcc CXX=g++ F77=gfortran
FC=gfortran --enable-mpirun-prefix-by-default --prefix=/usr/mpi/gcc/openmpi-
x.y.z/ --with-openib-libdir=/usr/lib64/ --libdir=/usr/mpi/gcc/openmpi-
x.y.z/lib64/ --with-contrib-vt-flags=--disable-iotrace
[root@host~]# make
[root@host~]# make install
```

The above step will install openMPI in */usr/mpi/gcc/openmpi-x.y.z/*

Note *To enable multithreading, add "--enable-mpi-thread-multiple" and "--with-threads=posix" parameters to the above configure command.*

v. Next, create a shell script , *mpivars.csh,* with the following entry:

```
# path
if ("" == "`echo $path | grep /usr/mpi/gcc/openmpi-x.y.z/bin`") then
   set path=(/usr/mpi/gcc/openmpi-x.y.z/bin $path)
endif


# LD_LIBRARY_PATH
if ("1" == "$?LD_LIBRARY_PATH") then
    if ("$LD_LIBRARY_PATH" !~ */usr/mpi/gcc/openmpi-x.y.z/lib64*) then
    setenv LD_LIBRARY_PATH /usr/mpi/gcc/openmpi-
x.y.z/lib64:${LD_LIBRARY_PATH}
    endif
else
   setenv LD_LIBRARY_PATH /usr/mpi/gcc/openmpi-x.y.z/lib64
endif
# MPI_ROOT
setenv MPI_ROOT /usr/mpi/gcc/openmpi-x.y.z
```

vi. Simlarly, create another shell script, *mpivars.sh*, with the following entry:

```
# PATH
if test -z "`echo $PATH | grep /usr/mpi/gcc/openmpi-x.y.z/bin`"; then
    PATH=/usr/mpi/gcc/openmpi-x.y.z/bin:${PATH}
    export PATH
fi
# LD_LIBRARY_PATH
if test -z "`echo $LD_LIBRARY_PATH | grep /usr/mpi/gcc/openmpi-
   x.y.z/lib64`"; then
   LD_LIBRARY_PATH=/usr/mpi/gcc/openmpi-
   x.y.z/lib64${LD_LIBRARY_PATH:+:}${LD_LIBRARY_PATH}
   export LD_LIBRARY_PATH
fi


# MPI_ROOT
MPI_ROOT=/usr/mpi/gcc/openmpi-x.y.z
export MPI_ROOT
```

vii. Next, copy the two files created in steps (v) and (vi) to */usr/mpi/gcc/openmpi-x.y.z/bin* and */usr/mpi/gcc/openmpi-x.y.z/etc*

viii. Register OpenMPI with MPI-selector:

```
[root@host~]# mpi-selector --register openmpi --source-dir
/usr/mpi/gcc/openmpi-x.y.z/bin
```

ix. Verify if it is listed in mpi-selector:

```
[root@host~]# mpi-selector --l
```

x. Set OpenMPI:

```
[root@host~]# mpi-selector --set openmpi –yes
```

xi. Logut and log back in.

4.2.3. Building MPI tests

i. Download *Intel's MPI Benchmarks* from http://software.intel.com/en-us/articles/intel-mpi-benchmarks
ii. Untar and change your current working directory to *src* directory.
iii. Edit *make_mpich* file and set *MPI_HOME* variable to the MPI which you want to build the benchmarks tool against. For example, in case of openMPI-1.6.4 set the variable as:

```
MPI_HOME=/usr/mpi/gcc/openmpi-1.6.4/
```

iv. Next, build and install the benchmarks using:

```
[root@host~]# gmake -f make_mpich
```

The above step will install IMB-MPI1, IMB-IO and IMB-EXT benchmarks in the current working directory (i.e. *src*).

v. Change your working directory to the MPI installation directory. In case of OpenMPI, it will be */usr/mpi/gcc/openmpi-x.y.z/*
vi. Create a directory called *tests* and then another directory called *imb* under *tests.*
vii. Copy the benchmarks built and installed in step (iv) to the *imb* directory.
viii. Follow steps (v), (vi) and (vii) for all the nodes.

4.2.4. Running MPI applications

- Run Open MPI application as:

```
mpirun --host  node1,node2 -mca btl openib,sm,self /usr/mpi/gcc/openmpi-
x.y.z/tests/imb/IMB-MPI1
```

Note

For OpenMPI/RDMA clusters with node counts greater than or equal to 8 nodes, and process counts greater than or equal to 64, you may experience the following RDMA address resolution error when running MPI jobs with the default OpenMPI settings:

```
The RDMA CM returned an event error while attempting to make a connection.
This type of error usually indicates a network configuration error.
Local host:   core96n3.asicdesigners.com
Local device: Unknown
Error name:   RDMA_CM_EVENT_ADDR_ERROR
Peer:         core96n8
```

***Workaround**: Increase the OpenMPI rdma route resolution timeout. The default is 1000, or 1000ms. Increase it to 30000 with this parameter:*

```
--mca btl_openib_connect_rdmacm_resolve_timeout 30000
```

4.3. Setting up NFS-RDMA

4.3.1. Starting NFS-RDMA

- **Server-side settings**

Follow the steps mentioned below to set up an NFS-RDMA server.

i. Make entry in `/etc/exports` file for the directories you need to export using NFS-RDMA on server as:

```
/share/rdma      *(fsid=0,async,insecure,no_root_squash)
/share/rdma1     *(fsid=1,async,insecure,no_root_squash)
```

Note that for each directory you export, you should have DIFFERENT fsid's.

ii. Load the iwarp modules and make sure peer2peer is set to 1.
iii. Load `xprtrdma` and `svcrdma` modules as:

```
[root@host~]# modprobe xprtrdma
[root@host~]# modprobe svcrdma
```

iv. Start the nfs service as:

```
[root@host~]# service nfs start
```

All services in NFS should start without errors.

v. Now we need to edit the file portlist in the path `/proc/fs/nfsd/`
Include the rdma port 2050 into this file as:

```
[root@host~]# echo rdma 2050 > /proc/fs/nfsd/portlist
```

vi. Run `exportfs` to make local directories available for Network File System (NFS) clients to mount.

```
[root@host~]# exportfs
```

Now the NFS-RDMA server is ready.

- **Client-side settings**

Follow the steps mentioned below at the client side.

i. Load the iwarp modules and make sure peer2peer is set to 1. Make sure you are able to ping and ssh to the server Chelsio interface through which directories will be exported.

ii. Load the `xprtrdma` module.

```
[root@host~]# modprobe xprtrdma
```

iii. Run the `showmount` command to show all directories from server as:

```
[root@host~]# showmount –e <server-chelsio-ip>
```

iv. Once the exported directories are listed, mount them as:

```
[root@host~]# mount.nfs <serverip>:<directory> <mountpoint-on-client> -o
vers=3,rdma,port=2050,wsize=65536,rsize=65536
```

4.4. Performance Tuning

See the **Performance Tuning** section in the **Unified Wire** chapter for generic performance settings.

IV. iSCSI PDU Offload Target

1. Introduction

This section describes how to install and configure iSCSI PDU Offload Target software for use as a key element in your iSCSI SAN. The software runs on Linux-based systems that use Chelsio or non-Chelsio based Ethernet adapters. However, to guarantee highest performance, Chelsio recommends using Chelsio adapters. Chelsio's adapters include offerings that range from stateless offload adapters (regular NIC) to the full line of TCP/IP Offload Engine (TOE) adapters.

The software implements RFC 3720, the iSCSI standard of the IETF. The software has been fully tested for compliance to that RFC and others and it has been exhaustively tested for interoperability with the major iSCSI vendors.

The software implements most of the iSCSI protocol in software running in kernel mode on the host with the remaining portion, which consists of the entire fast data path, in hardware when used with Chelsio's TOE adapters. When standard NIC Adapters are used the entire iSCSI protocol is executed in software.

The performance of this iSCSI stack is outstanding and when used with Chelsio's hardware it is enhanced further. Because of the tight integration with Chelsio's TOE adapters, this software has a distinct performance advantage over the regular NIC. The entire solution, which includes this software, Chelsio TOE hardware, an appropriate base computer system – including a high end disk subsystem, has industry leading performance. This can be seen when the entire solution is compared to others based on other technologies currently available on the market in terms of throughput and IOPS.

1.1. Features

Chelsio's iSCSI driver stack supports the iSCSI protocol in the Target mode. From henceforth "iSCSI Software Entity" term refers to the iSCSI target.

The Chelsio iSCSI PDU Offload Target software provides the following high level features:

- Expanded NIC Support
  - Chelsio TCP Offload Engine (TOE) Support
    - T5/T4 Based HBAs (T5/T4xx Series cards)
  - Non-Chelsio
    - Runs on regular NICs
- Chelsio Terminator ASIC Support
  - Offloads iSCSI Fast Data Path with Direct Data Placement (DDP)
  - Offloads iSCSI Header and Data Digest Calculations
  - Offload Speeds at 1 Gb, 10 Gb and 40Gb
  - Offloads TCP/IP for NAS simultaneously with iSCSI
- Target Specific features

- Full compliance with RFC 3720
- Error Recovery Level 0 (ERL 0)
- CHAP support for both discovery and login including mutual authentication
- Internet Storage Name Service (iSNS) Client
- Target Access Control List (ACL)
- Multiple Connections per Session
- Multiple Targets
- Multiple LUNs per Target
- Multi Path I/O (MPIO)
- Greater than 2 TB Disk Support
- Reserve / Release for Microsoft Cluster© Support
- Persistent Reservation
- Dynamic LUN Resizing
- iSCSI Target Redirection
- Multiple Target device types
    - Block
    - Virtual Block (LVM, Software RAID, EVMS, etc.)
    - Built in RAM Disk
    - Built in zero copy RAM Disk
- Supports iSCSI Boot Initiators
- An Intuitive and Feature Rich Management CLI

This chapter will cover these features in detail.

1.2. Hardware Requirements

1.2.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with iSCSI PDU Offload Target software:

- T520-BT
- T580-CR
- T520-LL-CR
- T520-CR
- T580-LP-CR
- T540-CR
- T420-CR
- T440-CR
- T422-CR
- T404-BT
- T420-BCH
- T440-LP-CR

- T420-BT
- T420-LL-CR
- T420-CX

1.2.2. Adapter Requirements

The Chelsio iSCSI PDU Offload Target software can be used with or without hardware protocol offload technology. There are two modes of operation using the iSCSI PDU Offload Target software on Ethernet-based adapters:

- Regular NIC – The software can be used in non-offloaded (regular NIC) mode. Please note however that this is the least optimal mode of operating the software in terms of performance.
- iSCSI HW Acceleration – In addition to offloading the TCP/IP protocols in hardware (TOE), this mode also takes advantage of Chelsio's ASIC capability of hardware assisted iSCSI data and header digest calculations as well as using the direct data placement (DDP) feature.

1.2.3. Storage Requirements

When using the Chelsio iSCSI target, a minimum of one hardware storage device is required. This device can be any of the device types that are supported (block, virtual block, RAM disk). Multiple storage devices are allowed by configuring the devices to one target or the devices to multiple targets. The software allows multiple targets to share the same device but use caution when doing this.

Chelsio's implementation of the target iSCSI stack has flexibility to accommodate a large range of configurations. For quick testing, using a RAM Disk as the block storage device works nicely. For deployment in a production environment a more sophisticated system would be needed. That typically consists of a system with one or more storage controllers with multiple disk drives attached running software or hardware based RAID.

1.3. Software Requirements

`chiscsi_base.ko` is iSCSI non-offload target mode driver and `chiscsi_t4.ko` is iSCSI PDU offload target mode driver.

cxgb4, toecore, t4_tom and chiscsi_base modules are required by `chiscsi_t4.ko` module to work in offloaded mode. Whereas in iscsi non-offloaded target (NIC) mode, only `cxgb4` is needed by `chiscsi_base.ko` module.

1.3.1. Linux Requirements

Currently the iSCSI PDU Offload Target software is available for the following versions:

- RHEL 7.2, 4.2.0-0.21.el7.aarch64 (ARM64)

- Ubuntu 14.04.3, 3.19.0-25-generic (POWER8)

Other kernel versions have not been tested and are not guaranteed to work.

1.3.2. Requirements for Installing the iSCSI Software

When installing the iSCSI software, it is required that the system have Linux kernel source or its headers installed in order to compile the iSCSI software as a kernel module. The source tree may be only header files, as for RHEL6 as an example, or a complete tree. The source tree needs to be configured and the header files need to be compiled. Additionally, the Linux kernel must be configured to use modules.

2. Software/Driver Loading

Important *Please ensure that all inbox drivers are unloaded before proceeding with unified wire drivers.*

There are two main steps to installing the Chelsio iSCSI PDU Offload Target software. They are:

1. **Installing the iSCSI software** – The majority of this section deals with how to install the iSCSI software.
2. **Configuring the iSCSI software** – Information on configuring the software can be found in a section further into this user's guide.

2.1. Latest iSCSI Software Stack Driver Software

The iSCSI software stack comes bundled in the Chelsio Unified Wire package which can be downloaded from the Chelsio support website (http://service.chelsio.com).

The iSCSI software is available for use with most installations of the Linux kernel . The software is dependent on the underlying NIC adapter driver and thus the limitation on what version of the Linux kernel it can run on is mostly dependent on the NIC driver's limitations.

The iSCSI module will be installed in the `/lib/modules/<linux_kernel_version>/updates/kernel/drivers/scsi/chiscsi` directory. The modules database will be updated by the installer. This allows the iSCSI module to be located when using the `modprobe` utility. The actual module `chiscsi_t4.ko` can be found inside the package under /build/src/chiscsi/t4.

The `iscsictl` tool and the `chisns` tool will be installed in `/sbin`. The `chisns` tool starts the iSNS client. The `iscsictl` tool is provided for configuring and managing the iSCSI targets and iSNS client. It also provides control for iSCSI global settings.

1. Loading the Kernel module

- For RHEL distributions, run modprobe as follows:

```
[root@host~]# modprobe chiscsi_t4
```

- For SLES distributions, run modprobe as follows:

```
[root@host~]# modprobe chiscsi_t4 --allow-unsupported
```

Note *Installation/uninstallation using source-tar-ball will neither remove the conf file nor rename it. It will always be intact. However, it's recommended to always take a backup of your configuration file for both methods of installation.*

A sample iSCSI configuration file will be installed in `/etc/chelsio-iscsi/chiscsi.conf`. This file should be edited using a standard text editor and customized to fit your environment.

2. Set iSCSI service to automatically start at bootup

The `chelsio-target` service scripts are installed to `/etc/init.d` and the parameters for the script are installed at `/etc/sysconfig/chiscsi`. The script is installed as a system service.

To auto-start the iSCSI target service at a certain runlevel, e.g. runlevel 3, `chkconfig` can be used on Red Hat and Novell / SuSE based systems as follows:

```
[root@host~]# chkconfig --level 3 chelsio-target on
```

The `chelsio-target` service scripts do basic checks before starting the iSCSI target service, loads the kernel module, and starts all the targets configured by default. It can also be used to stop the targets, and restart/reload configuration.

Note *For the script to execute properly, make sure the following flag is set on all kernel.org kernels.*

 # CONFIG_MODULE_FORCE_LOAD=y

3. Software/Driver Unloading

Use the following command to unload the module:

```
[root@host~]# rmmod chiscsi_t4
```

4. Software/Driver Configuration and Fine-tuning

The Chelsio iSCSI software needs configuration before it can become useful. The following sections describe how this is done.

There are two main components used in configuring the Chelsio iSCSI software: the **configuration file** and the **iSCSI control tool**. This section describes in some detail what they are and their relationship they have with one another.

4.1. Command Line Tools

There are two command line tools, one for control of the iSNS client and one for control of the iSCSI target nodes.

4.1.1. iscsictl

The Chelsio iSCSI control tool, `iscsictl`, is a Command Line Interface (CLI) user space program that allows administrators to:

- Start/Stop the iSCSI Target
- Start the iSNS client
- Get/Set the iSCSI driver global settings
- Get/Set/Remove the iSCSI Target configuration settings
- Retrieve active sessions' information of an iSCSI Target
- Manually flush data to the iSCSI Target disks
- Reload the iSCSI configuration file
- Update the iSCSI configuration file
- Save the current iSCSI configuration to a file

4.1.2. chisns

The Chelsio iSNS client, `chisns`, can be started independently of `iscsictl`.

4.2. iSCSI Configuration File

The iSCSI configuration file is the place where information about the Chelsio iSCSI software is stored. The information includes global data that pertains to all targets as well as information on each specific iSCSI target node. Most of the information that can be placed in the configuration file has default values that only get overwritten by the values set in the configuration file.There are only a few global configuration items that can be changed.

There are many specific parameters that can be configured, some of which are iSCSI specific and the rest being Chelsio specific. An example of an iSCSI specific item is "HeaderDigest" which is defaulted to "None" but can be overridden to "CRC32C". An example of a Chelsio

specific configurable item is “ACL” (for Access Control List). “ACL” is one of the few items that have no default.

Before starting any iSCSI target, an iSCSI configuration file must be created. An easy way to create this file is to use the provided sample configuration file and modify it. This file can be named anything and placed in any directory but it must be explicitly specified when using iscsictl by using the -f option. To avoid this, put configuration file in the default directory (`/etc/chelsio-iscsi`) and name it the default file name (`chiscsi.conf`).

4.2.1. “On the fly” Configuration Changes

Parameters for the most part can be changed while an iSCSI node is running. However, there are exceptions and restrictions to this rule that are explained in a later section that describes the details of the iSCSI control tool `iscsictl`.

4.3. A Quick Start Guide for Target

This section describes how to get started quickly with a Chelsio iSCSI target. It includes:

- Basic editing of the iSCSI configuration file.
- Basic commands of the iSCSI control tool including how to start and stop a target.

4.3.1. A Sample iSCSI Configuration File

The default Chelsio iSCSI configuration file is located at `/etc/chelsio-iscsi/chiscsi.conf`. If this file doesn’t already exist, then one needs to be created.

To configure an iSCSI target, there are three required parameters (in the form of key=value pairs) needed as follows:

- TargetName – A worldwide unique iSCSI target name.
- PortalGroup – The portal group tag associating with a list of target IP address (es) and port number(s) that service the login request. The format of this field is a Chelsio specific iSCSI driver parameter which is described in detail in the configuration file section.
- TargetDevice – A device served up by the associated target. A device can be:
    - A block device (for example, /dev/sda)
    - A virtual block device (for example, /dev/md0)
    - A RAM disk
    - A regular file

A target can serve multiple devices, each device will be assigned a Logical Unit Number (LUN) according to the order it is specified (i.e., the first device specified is assigned LUN 0, the second one LUN 1, …, and so on and so forth). Multiple TargetDevice key=value pairs are needed to indicate multiple devices.

Here is a sample of a minimum iSCSI target configuration located at `/etc/chelsio-iscsi/chiscsi.conf`:

```
target:
        TargetName=iqn.2006-02.com.chelsio.diskarray.san1
        TargetDevice=/dev/sda
        PortalGroup=1@192.0.2.178:3260
```

The `TargetDevice` value must match with the storage device in the system. The `PortalGroup` value must have a matching IP address of the Ethernet adapter card in the system.

For more information about TargetDevice configuration see **Target Storage Device Configuration**.

4.3.2. Basic iSCSI Control

Control of the Chelsio iSCSI software is done through `iscsictl`, the command line interface control tool. The following are the basic commands needed for effective control of the target.

Start Target: To start all of the iSCSI targets specified in the iSCSI configuration file, execute `iscsictl` with the "`-S`" option followed by "`target=ALL`".

```
[root@host~]# iscsictl -S target=ALL
```

To start a specific target execute `iscsictl` with "`-S`" followed by the target.

```
[root@host~]# iscsictl -S target=iqn.2006-02.com.chelsio.diskarray.san1
```

Stop Target: To stop the all the iSCSI target(s), execute `iscsictl` with "`-s`" option followed by "`target=ALL`".

```
[root@host~]# iscsictl -s target=ALL
```

To stop a specific target execute `iscsictl` with "`-s`" followed by the target name.

```
[root@host~]# iscsictl -s target=iqn.2006-02.com.chelsio.diskarray.san1
```

View Configuration: To see the configuration of all the active iSCSI targets, execute `iscsictl` with "`-c`" option.

```
[root@host~]# iscsictl -c
```

To see the more detailed configuration settings of a specific target, execute `iscsictl` with "`-c`" option followed by the target name.

```
[root@host~]# iscsictl -c target=iqn.2006-02.com.chelsio.diskarray.san1
```

View Global Settings: To see Chelsio global settings, execute `iscsictl` with "`-g`" option.

```
[root@host~]# iscsictl -g
```

Change Global Settings: To change Chelsio global settings, execute `iscsictl` with "`-G`" option.

```
[root@host~]# iscsictl -G iscsi_login_complete_time=300
```

View Help: To print help to stdout, execute `iscsictl` with "`-h`" option.

```
[root@host~]# iscsictl -h
```

4.4. The iSCSI Configuration File

The iSCSI configuration file consists of a series of blocks consisting of the following types of iSCSI entity blocks:

1. global
2. target

There can be only one global entity block whereas multiple target entity blocks are allowed. The global entity block is optional but there must be at least one target entity block.

An entity block begins with a block type (global or target). The content of each entity block is a list of parameters specified in a "key=value" format. An entity block ends at the beginning of the next entity block or at the end-of-file.

The parameter list in an entity block contains both:

- iSCSI parameters that override the default values
- Parameters that facilitate passing of control information to the iSCSI module

All lines in the configuration file that begin with "#" character are treated as comments and will be ignored. White space is not significant except in key=value pairs.

For the "key=value" parameters the <value> portion can be a single value or a list of multiple values. When <value> is a list of multiple values, they must be listed on one line with a comma "," to separate their values. Another way to list the values instead of commas is to list their values as key=value pairs repeatedly, each on a new line, until they are all listed.

There are three categories of "key=value" parameter, the first category belongs to the global entity block whereas the second and third categories belong to target entity block:

1. The Chelsio Global Entity Settings of key=value pairs
2. The iSCSI Entity Settings of key=value pairs
3. The Chelsio Entity Settings of key=value pairs

The following sub-sections describe these three categories and list in tables the details of their key=value parameters.

4.4.1. Chelsio System Wide Global Entity Settings

Description

Chelsio System Wide Global Entity Parameters pass system control information to the iSCSI software which affects all targets in the same way. More detail of these parameters below can be found in a later section entitled "System Wide Parameters".

Table of Chelsio Global Entity Settings

Key	Valid Values	Default Value	Multiple Values	Description
iscsi_auth_order	"ACL" "CHAP"	"CHAP"	No	Authorization order for login verification on the target. Valid only when a target's ACL_Enable=Yes **ACL**: ACL first then CHAP **CHAP**: CHAP first then ACL *Applies to Target(s) Only*
DISC_AuthMethod	"CHAP" "NONE"	None	No	To choose an authentication method for discovery phase.
DISC_Auth_CHAP_Policy	"Oneway" "Mutual"	"Oneway"	No	Oneway or Mutual (two-way) CHAP
DISC_Auth_CHAP_Target	"<user id>" :"<secret>"		Yes	CHAP user id and secret for the target. **<user id>** must be less than 256 characters. Commas "," are not allowed. **<secret>** must be between 6 and 255 characters. Commas "," are not allowed. The target user id and secret are used by the initiator to authenticate the target while doing mutual chap. *NOTE: The double quotes are required as part of the format.*
DISC_Auth_CHAP_Initiator	"<user id>" :"<secret>"		Yes	CHAP user id and secret for the initiator. **<user id>** must be less than 256 characters. Commas "," are not allowed. **<secret>** must be between 6 and 255 characters. Commas "," are not allowed. The initiator user id and secret are used by the target to authenticate the initiator. *NOTE: The double quotes are required as part of the format.*

iscsi_chelsio_ini_idstr	a string of maximum of 255 characters	"cxgb4i"	No	To enable additional optimization when Chelsio Adapters and drivers are used at both ends (initiator and target) systems. Make sure the initiator name contain the substring set in iscsi_chelsio_ini_idstr when using Chelsio iscsi initiator driver.
iscsi_target_vendor_id	a string of maximum of 8 characters	"CHISCSI"	No	The target vendor ID part of the device identification sent by an iSCSI target in response of SCSI Inquiry command.
iscsi_login_complete_time	0 to 3600	300	No	Time allowed (in seconds) for the initiator to complete the login phase. Otherwise, the connection will be closed *NOTE: value zero means this check is NOT performed.*

4.4.2. iSCSI Entity Settings

Description

iSCSI Entity Parameters pass iSCSI protocol control information to the Chelsio iSCSI module. This information is unique for each entity block. The parameters follow the IETF iSCSI standard RFC 3720 in both definition and syntax. The descriptions below are mostly from this RFC.

Table of iSCSI Entity Settings

Key	Valid Values	Default Value	Multiple Values	Description
MaxConnections	1 to 65535	1	No	Initiator and target negotiate the maximum number of connections requested/acceptable.
InitialR2T	"Yes" "No"	"Yes"	No	To turn on or off the default use of R2T for unidirectional and the output part of bidirectional commands.
ImmediateData	"Yes" "No"	"Yes"	No	To turn on or off the immediate data.
FirstBurstLength	512 to 16777215 (2^{24} - 1)	65536	No	The maximum negotiated SCSI data in bytes of unsolicited data that an iSCSI initiator may send to a target during the execution of a single SCSI command.
MaxBurstLength	512 to 16777215 (2^{24} - 1)	262144	No	The maximum negotiated SCSI data in bytes, of a Data-In or a solicited Data-Out iSCSI sequence between the initiator and target.
DefaultTime2Wait	0 to 3600	2	No	The minimum time, in seconds, to wait before attempting an explicit / implicit logout or connection reset between initiator and target.

DefaultTime2Retain	0 to 3600	20	No	The maximum time, in seconds, after an initial wait.
MaxOutstandingR2T	1 to 65535	1	No	The maximum number of outstanding R2Ts per task.
DataPDUInOrder	"Yes" "No"	"Yes"	No	To indicate the data PDUs with sequence must be at continuously increasing order or can be in any order. *Chelsio only supports "Yes".*
DataSequenceInOrder	"Yes" "No"	"Yes"	No	To indicate the Data PDU sequences must be transferred in continuously non-decreasing sequence offsets or can be transferred in any order. *Chelsio only supports "Yes".*
ErrorRecoveryLevel	0 to 2	0	No	To negotiate the recovery level supported by the node. *Chelsio only supports 0.*
HeaderDigest	"None" "CRC32C"	"None"	Yes	To enable or disable iSCSI header Cyclic integrity checksums.
DataDigest	"None" "CRC32C"	"None"	Yes	To enable or disable iSCSI data Cyclic integrity checksums.
AuthMethod	"CHAP" and "None"	"None, CHAP"	Yes	To choose an authentication method during login phase.
TargetName	"<target name>"		No	A worldwide unique iSCSI target name. *Target only.*
TargetAlias	"<target alias>"		No	A human-readable name or description of a target. It is not used as an identifier, nor is it for authentication. *Target only.*
MaxRecvDataSegmentLength	512 to 16777215 (2^{24} - 1)	8192	No	To declare the maximum data segment length in bytes it can receive in an iSCSI PDU.
OFMarker	"Yes" "No"	"No"	No	To turn on or off the initiator to target markers on the connection. *Chelsio only supports "No".*
IFMarker	"Yes" "No"	"No"	No	To turn on or off the target to initiator markers on the connection. *Chelsio only supports "No".*
OFMarkInt	1 to 65535	2048	No	To set the interval for the initiator to target markers on a connection.
IFMarkInt	1 to 65535	2048	No	To set the interval for the target to initiator markers on a connection.

4.4.3. Chelsio Entity Settings

Description

Chelsio Entity Parameters pass control information to the Chelsio iSCSI module. The parameters are specific to Chelsio'simplementation of the iSCSI node (target or initiator) and are unique for each entity block. The parameters consist of information that can be put into three categories:

1. Challenge Handshake Authentication Protocol (CHAP).
2. Target specific settings. All of the following parameters can have multiple instances in one target entity block (i.e., they can be declared multiple times for one particular target).

- Portal Group
- Storage Device
  Access Control List (ACL)

Table of Chelsio Entity Settings

Key	Valid Values	Default Value	Multiple Values	Description
Chelsio CHAP Parameter (Target)				
Auth_CHAP_Target	"<user id>" :"<secret>"		No	CHAP user id and secret for the target. **<user id>** must be less than 256 characters. Commas "," are not allowed. **<secret>** must be between 6 and 255 characters. Commas "," are not allowed. The target user id and secret are used by the initiator to authenticate the target while doing mutual chap. *NOTE: The double quotes are required as part of the format.*
Auth_CHAP_Initiator	"<user id>" :"<secret>"		Yes	CHAP user id and secret for the initiator. **<user id>** must be less than 256 characters. Commas "," are not allowed. **<secret>** must be between 6 and 255 characters. Commas "," are not allowed. The initiator user id and secret are used by the target to authenticate the initiator. *NOTE: The double quotes are required as part of the format.*
Auth_CHAP_Challenge Length	16 to 1024	16	No	CHAP challenge length
Auth_CHAP_Policy	"Oneway" or "Mutual"	"Oneway"	No	Oneway or Mutual (two-way) CHAP
Chelsio Target Specific Parameter				
PortalGroup	<portal group tag> @<target IP address> [:<port number>] . . . [,<target IP		Yes	The portal group name associates the given target with the given list of IP addresses (and optionally, port numbers) for servicing login requests. It"s required to have at least one per target. **<portal group tag>** is a unique tag identifying the portal group. It must be a positive integer.

	address> [:<port number>]] [,timeout= <timeout value in milliseconds>] [,[portalgroutag1, portalgroutag2,… portalgroutagn]			**<target IP address>** is the IP address associated with the portal group tag. **<port number>** is the port number associated with the portal group tag. It is optional and if not specified the well-known iSCSI port number of 3260 is used. **<timeout>** is optional, it applies to all the portals in the group. The timeout value is in milliseconds and needs to be multiple of 100ms. It is used to detect loss of communications at the iSCSI level. *NOTE: There can be multiple target IP address/port numbers per portal group tag. This enables a target to operate on multiple interfaces for instance.* **<portalgrouptagX>**The portalgroup to which login requests should be redirected to. *NOTE: There can be multiple redirection target portalgroups specified for a particular target portal group and the redirection will happen to these in a round robin manner.*
ShadowMode	"Yes" "No"	"No"	No	To turn ShadowMode on or off for iSCSI Target Redirection
TargetSessionMaxCmd	1 to 2048	64	No	The maximum number of outstanding iSCSI commands per session.
TargetDevice*	<path/name> [,FILE|MEM|BLK] [,NULLRW] [,SYNC] [,RO] [,size=xMB] [,ID=xxxxxx] [,WWN=xxxxxxxxxx] [,SN= xxxxxx]		No	A device served up by the associated target. The device mode can be a: • Block Device (e.g. /dev/sda) • Virtual Block Device (e.g. /dev/md0) • RamDisk • Regular File **<path/name>** is the path to the device - with the exception of when a RAM Disk is specified, where it is a unique name given to the device. If multiple RAM Disks are used for a target then each name must be unique within the target. **NULLRW** specifies that random data is returned for reads, and for writes data is dropped. Useful for testing network performance. **SYNC** specifies that the device will function in the write-through mode (i.e.,

				the data will be flushed to the device before the response is returned to the initiator). *NOTE: SYNC is only applicable with FILE mode.* **RO** specifies the device as a read-only device. **FILE** specifies this device should be accessed via the kernel"s VFS layer. This mode is the most versatile, and it is the default mode in the cases where there is no mode specified. **BLK** specifies this device should be accessed via the kernel"s block layer. This mode is suitable for high-speed storage device such as RAID Controllers. **MEM** specifies this device should be created as a RAM Disk. **size**=xMB is used with "MEM", to specify the RamDisk size. If not specified, the default RamDisk size is 16MB (16 Megabytes). The minimum value of x is 1 (1MB) and the maximum value is limited by system memory. **SN** is a 16 character unique value. **ID** is a 24 character unique value. **WWN** is a 16 character unique value. It is recommended when using a multipath aware initiator , the optional ID (short form for SCSI ID), SN and WWN values should be set manually for the TargetDevice. These values will be returned in Inquiry response (VPD 0x83). Multiple TargetDevice key=value pairs are needed to indicate multiple devices. There can be multiple devices for any particular target. Each device will be assigned a Logical Unit Number (LUN) according to the order it is specified (i.e., the first device specified is assigned LUN 0, the second one LUN 1, …, and so on and so forth). *NOTE: **FILE** mode is the most versatile mode, if in doubt use **FILE** mode.*
ACL_Enable	"Yes" "No"	"No"	No	Defines if Chelsio'sAccess Control List (ACL) method will be enforced on the target:

				Yes: ACL is enforced on the target **No:** ACL is not enforced on the target *NOTE*: ACL flag is not allowed to be updated on the fly. Target must be restarted for new ACL flag to take effect.
ACL	[iname=<name1>][;<sip=<sip1>][;dip=<dip1>][;lun=<lun_list:permissions>]		Yes	The ACL specifies which initiators and how they are allowed to access the LUNs on the target. **iname=<Initiator Name>** specifies one or more initiator names, the name must be a fully qualified iSCSI initiator name. **sip=<Source IP address>** specifies one or more IP addresses the initiators are connecting from. **Dip=<Destination IP address>** specifies one or more IP addresses that the iSCSI target is listening on (i.e., the target portal IP addresses). *NOTE: when configuring an ACL at least one of the above three must be provided:* • *iname, and/or* • *sip, and/or* • *dip.* **lun=<lun list>:<permission>** controls how the initiators access the luns. The supported value for **<lun list>** is **ALL**. **<permissions>** can be: **R**: Read Only **RW** or **WR**: Read and Write If permissions are specified then the associated LUN list is required. If no **lun=<lun list>:[R|RW]** is specified then it defaults to **ALL:RW**. *NOTE: For the Chelsio Target Software release with lun-masking included,* ***<lun list>** is in the format of **<0..N | 0~N | ALL>*** *Where:* ***0..N**: only one value from **0** through **N*** ***0~N**: a range of values between **0** through **N*** ***ALL**: all currently supported LUNs.* *Multiple lists of LUN numbers are allowed. When specifying the list separate the LUN ranges by a comma.*
RegisteriSNS	"Yes" "No"	"Yes"	No	To turn on or off exporting of target information via iSNS

4.4.4. Sample iSCSI Configuration File

Following is a sample configuration file. While using iSCSI node (target), irrelevant entity block can be removed or commented.

```
#
# Chelsio iSCSI Global Settings
#
global:
       iscsi_login_complete_time=300
       iscsi_auth_order=CHAP
       DISC_AuthMethod=None
       DISC_Auth_CHAP_Policy=Oneway
       DISC_Auth_CHAP_Target="target_id1":"target_secret1"
       DISC_Auth_CHAP_Initiator="initiator_id1":"initiator_sec1"
#
# an iSCSI Target “iqn.2006-02.com.chelsio.diskarray.san1”
# being served by the portal group "5". Setup as a RAM Disk.
#
target:
       TargetName=iqn.2006-02.com.chelsio.diskarray.san1
       # lun 0: a ramdisk with default size of 16MB
       TargetDevice=ramdisk,MEM
       PortalGroup=5@192.0.2.178:3260
#
# an iSCSI Target “iqn.2005-8.com.chelsio:diskarrays.san.328”
# being served by the portal group "1" and "2"
#
target:
       #
       # iSCSI configuration
       #
       TargetName=iqn.2005-8.com.chelsio:diskarrays.san.328
       TargetAlias=iTarget1
       MaxOutstandingR2T=1
       MaxRecvDataSegmentLength=8192
       HeaderDigest=None,CRC32C
       DataDigest=None,CRC32C
```

```
ImmediateData=Yes
InitialR2T=No
FirstBurstLength=65535
MaxBurstLength=262144

#
# Local block devices being served up
# lun 0 is pointed to /dev/sda
# lun 1 is pointed to /dev/sdb

TargetDevice=/dev/sda,ID=aabbccddeeffgghh,WWN=aaabbbcccdddeeef
TargetDevice=/dev/sdb

#
# Portal groups served this target
#

PortalGroup=1@102.50.50.25:3260
PortalGroup=2@102.60.60.25:3260

#
# CHAP configuration
#
Auth_CHAP_Policy=Mutual
Auth_CHAP_target="iTarget1ID":"iTarget1Secret"

Auth_CHAP_Initiator="iInitiator1":"InitSecret1"
Auth_CHAP_Initiator="iInitiator2":"InitSecret2"
Auth_CHAP_ChallengeLength=16

#
# ACL configuration
#
# initiator "iqn.2006-02.com.chelsio.san1" is allowed full access
# to this target
ACL=iname=iqn.2006-02.com.chelsio.san1

# any initiator from IP address 102.50.50.101 is allowed full access
# of this target
ACL=sip=102.50.50.101
```

```
        # any initiator connected via the target portal 102.60.60.25 is
        # allowed full access to this target
        ACL=dip=102.60.60.25

        # initiator "iqn.2005-09.com.chelsio.san2" from 102.50.50.22 and
        # connected via the target portal 102.50.50.25 is allowed read only
        # access of this target
        ACL=iname=iqn.2006-
02.com.chelsio.san2;sip=102.50.50.22;dip=102.50.50.25;lun=ALL:R
```

4.5. Challenge-Handshake Authenticate Protocol (CHAP)

CHAP is a protocol that is used to authenticate the peer of a connection and uses the notion of a challenge and response, (i.e., the peer is challenged to prove its identity).

The Chelsio iSCSI software supports two CHAP methods: **one-way** and **mutual**. CHAP is supported for both login and discovery sessions.

4.5.1. Normal Session CHAP Authentication

For a normal Session, the CHAP authentication is configured on a per-target basis.

4.5.2. Oneway CHAP authentication

With **one-way** CHAP (also called unidirectional CHAP) the target uses CHAP to authenticate the initiator. The initiator does not authenticate the target. This method is the default method.

For **one-way** CHAP, the initiator CHAP id and secret are configured and stored on a per-initiator with Chelsio Entity parameter "Auth_CHAP_Initiator".

4.5.3. Mutual CHAP authentication

With **mutual** CHAP (also called bidirectional CHAP), the target and initiator use CHAP to authenticate each other.

For **mutual** CHAP, in addition to the initiator CHAP id and secret, the target CHAP id and secret are required. They are configured and stored on a per target basis with Chelsio Entity parameter "Auth_CHAP_Target".

4.5.4. Adding CHAP User ID and Secret

A single `Auth_CHAP_Target` key and multiple `Auth_CHAP_Initiator` keys could be configured per target:

```
target:
      TargetName=iqn.2006-02.com.chelsio.diskarray.san1
      TargetDevice=/dev/sda PortalGroup=1@192.0.2.178:8000
      Auth_CHAP_Policy=Oneway
      Auth_CHAP_Initiator=”remoteuser1”:”remoteuser1_secret”
      Auth_CHAP_Initiator=”remoteuser2”:”remoteuser2_secret”
      Auth_CHAP_Target=“targetid1”:”target1_secret”
```

In the above example, target `iqn.2005-com.chelsio.diskarray.san1` has been configured to authenticate two initiators, and its own id and secret are configured for use in the case of mutual CHAP.

4.5.5. AuthMethod and Auth_CHAP_Policy Keys

By setting the iSCSI keys `AuthMethod` and `Auth_CHAP_Policy`, a user can choose whether to enforce CHAP and if mutual CHAP needs to be performed.

The `AuthMethod` key controls if an initiator needs to be authenticated or not. The default setting of `AuthMethod` is `None,CHAP`

The `Auth_CHAP_Policy` key controls which CHAP authentication (one-way or mutual) needs to be performed if CHAP is used. The default setting of `Auth_CHAP_Policy` is `Oneway`

On an iSCSI node, with `AuthMethod=None,CHAP`

- `Auth_CHAP_Policy=Oneway`, Chelsio iSCSI target will accept a relevant initiator if it does
    a) no CHAP
    b) CHAP Oneway or
    c) CHAP Mutual

- `Auth_CHAP_Policy=Mutual`, the Chelsio iSCSI target will accept a relevant initiator if it does
    a) no CHAP or
    b) CHAP Mutual

With `AuthMethod=None`, regardless the setting of the key Auth_CHAP_Policy, the Chelsio iSCSI target will only accept a relevant initiator if it does no CHAP.

With `AuthMethod=CHAP`, CHAP is enforced on the target:

i. `Auth_CHAP_Policy=Oneway`, the iSCSI target will accept a relevant initiator only if it does
    a) CHAP Oneway or
    b) CHAP Mutual

ii. `Auth_CHAP_Policy=Mutual`, the iSCSI node will accept a relevant initiator only if it does
    a) CHAP Mutual

4.5.6. Discovery Session CHAP

CHAP authentication is also supported for the discovery sessions where an initiator queries of all available targets.

Discovery session CHAP is configured through the global section in the configuration file. List of keys to provision discovery CHAP are:

- DISC_AuthMethod: None or CHAP.
- DISC_Auth_CHAP_Policy: Oneway or Mutual (i.e., two-way) authentication.
- DISC_Auth_CHAP_Target: target CHAP user id and secret
- DISC_Auth_CHAP_Initiator: initiator CHAP user id and secret.

Sample:

```
#
# Chelsio iSCSI Global Settings
#
global:
        DISC_AuthMethod=CHAP
        DISC_Auth_CHAP_Policy=Mutual
        DISC_Auth_CHAP_Target="target_id1":"target_secret1"
        DISC_Auth_CHAP_Initiator="initiator_id1":"initiator_sec1"
```

4.6. Target Access Control List (ACL) Configuration

The Chelsio iSCSI target supports iSCSI initiator authorization via an Access Control List (ACL).

ACL configuration is supported on a per-target basis. The creation of an ACL for a target establishes:

- Which iSCSI initiators are allowed to access it
- The type of the access: read-write or read-only
- Possible SCSI layer associations of LUNs with the initiator

More than one initiator can be allowed to access a target and each initiator‟s access rights can be independently configured.

The format for ACL rule is as follows:

```
ACL=[iname=<initiator name>][;<sip=<source ip addresses>]
    [;dip=<destination ip addresses>][;lun=<lun_list>:<permissions>]

 target:
      TargetName=iqn.2006-02.com.chelsio.diskarray.san1
      TargetDevice=/dev/sda

      PortalGroup=1@102.50.50.25:3260
      PortalGroup=2@102.60.60.25:3260

      # initiator "iqn.2006-02.com.chelsio.san1" is allowed
      # full read-write access to this target
      ACL=iname=iqn.2006-02.com.chelsio.san1

      # any initiator from IP address 102.50.50.101 is allowed full
      # read-write access of this target
      ACL=sip=102.50.50.101

      # any initiator connected via the target portal 102.60.60.25
      # is allowed full read-write access to this target
      ACL=dip=102.60.60.25

      # initiator "iqn.2005-09.com.chelsio.san2" from 102.50.50.22
      # and connected via the target portal 102.50.50.25 is allowed
      # read only access of this target
      ACL=iname=iqn.2006-
      02.com.chelsio.san2;sip=102.50.50.22;dip=102.50.50.25;lun=ALL:R
```

4.6.1. ACL Enforcement

To toggle ACL enforcement on a per-target base, a Chelsio keyword `ACL_Enable` is provided:

- Setting `ACL_Enable=Yes` enables the target to perform initiator authorization checking for all the initiators during login phase. And in addition, once the initiator has been authorized to access the target, the access rights will be checked for each individual LU the initiator trying to access.
- Setting `ACL_Enable=No` disable the target to perform initiator authorization checking.

When a target device is marked as read-only (RO), it takes precedence over ACL's write permission (i.e., all of ACL write permission of an initiator is ignored).

4.7. Target Storage Device Configuration

An iSCSI Target can support one or more storage devices. The storage device can either be the built-in RAM disk or actual backend storage.

Configuration of the storage is done through the Chelsio configuration file via the key-value pair `TargetDevice`.

When option `NULLRW` is specified, on writes the data is dropped without being copied to the storage device, and on reads the data is not actually read from the storage device but instead random data is used. This option is useful for measuring network performance.

The details of the parameters for the key `TargetDevice` are found in the table of Chelsio Entity Settings section earlier in this document.

4.7.1. RAM Disk Details

For the built-in RAM disk:

- The minimum size of the RAM disk is 1 Megabyte (MB) and the maximum is limited by system memory.
- To use a RAM disk with a Windows Initiator, it is recommended to set the size >= 16MB.

To configure an ramdisk specify `MEM` as the device mode:

```
TargetDevice=<name>,MEM,size=xMB
```

Where:

`<name>` Is a unique name given to the RAM disk. This name identifies this particular ramdisk. If multiple RAM disks are configured for the same target, the name must be unique for each RAM Disk.

`x` Is the size of the RAM disk in MB. It's an integer between (1-max), where max is limited by system memory. If this value is not specified the default value is 16 MB.

```
target:
#<snip>
  # 16 Megabytes RAM Disk named ramdisk1
  TargetDevice=ramdisk1,MEM,size=16MB
#<snip>
```

4.7.2. FILE Mode Storage Device Details

The FILE mode storage device is the most common and versatile mode to access the actual storage attached to the target system:

- The FILE mode can accommodate both block devices and virtual block devices.
- The device is accessed in the exclusive mode. The device should not be accessed (or active) in any way on the target system.
- Each device should be used for one and only one iSCSI target.
- "SYNC" can be used with FILE mode to make sure the data is flushed to the storage device before the Target responds back to the Initiator.

To configure a FILE storage device specify `FILE` as the device mode:

```
TargetDevice=<path to the storage device>[,FILE][,SYNC]
```

Where: `<path>` Is the path to the actual storage device, such as /dev/sdb for a block device or /dev/md0 for a software RAID. The path must exist in the system.

`SYNC` When specified, the Target will flush all the data in the system cache to the storage driver before sending response back to the Initiator.

4.7.3. Example Configuration of FILE Mode Storage

Below is an example:

```
target:
#<snip>
 # software raid /dev/md0 is accessed in FILE mode
TargetDevice=/dev/md0,FILE
#<snip>
```

4.7.4. BLK Mode Storage Device Details

The BLK mode storage device is suitable for high-speed storage attached to the target system:

- The BLK mode can accommodate only block devices.
- The device is accessed in the exclusive mode. The device should not be accessed (or active) in any way on the target system.
- Each device should be used for one and only one iSCSI target.

To configure a block storage device specify BLK as the device mode:

```
TargetDevice=<path to the storage device>,BLK
```

Where: `<path>` Is the path to the actual storage device, such as /dev/sdb. The path must exist in the system.

```
target:
#<snip>
 # /dev/sdb is accessed in BLK mode
 TargetDevice=/dev/sdb,BLK
#<snip>
```

4.7.5. Multi-path Support

To enable multi-path support on the initiator, it is highly recommended that the following options are specified:

- *[,ID=xxxxxx]*: SCSI ID, a twenty-four (24) bytes alpha-numeric string
- *[,WWN=xxxxxxxxx]*: SCSI World Wide Name (WWN), a sixteen (16) bytes alpha-numeric string
- *[,SN= xxxxxx]*: SCSI SN, a sixteen (15) bytes alpha-numeric string.

The user should make sure the three values listed above are the same for the target LUNs involved in the multipath.

4.8. Target Redirection Support

An iSCSI Target can redirect an initiator to use a different IP address and port (often called a portal) instead of the current one to connect to the target. The redirected target portal can either be on the same machine, or a different one.

4.8.1. ShadowMode for Local vs. Remote Redirection

The *ShadowMode* setting specifies whether the Redirected portal groups should be present on the same machine or not. If *ShadowMode* is enabled, the redirected portal groups are on a different system. If it is disabled then the redirected portal groups must be present on the same system otherwise the target would fail to start.

Below is an example with *ShadowMode* enabled:

```
target:
 #<snip>
     # any login requests received on 10.193.184.81:3260 will be
     # redirected to 10.193.184.85:3261.

     PortalGroup=1@10.193.184.81:3260,[2]
     PortalGroup=2@10.193.184.85:3261

     # the PortalGroup "2" is NOT presented on the same system.
```

```
    ShadowMode=Yes
 #<snip>
```

Below is an example with *ShadowMode* disabled:

```
target:
 #<snip>
     # any login requests received on 10.193.184.81:3260 will be
     # redirected to 10.193.184.85:3261

    PortalGroup=1@10.193.184.81:3260,[2]
    PortalGroup=2@10.193.184.85:3261
    # the PortalGroup "2" IS present on the same system
    ShadowMode=No
#<snip>
```

4.8.2. Redirecting to Multiple Portal Groups

The Chelsio iSCSI Target Redirection allows redirecting all login requests received on a particular portal group to multiple portal groups in a round robin manner.

Below is an example Redirection to Multiple Portal Groups:

```
target:
 #<snip>
     # any login requests received on 10.193.184.81:3260 will be
     # redirected to 10.193.184.85:3261 and 10.193.184.85:3262 in a
     # Round Robin Manner.

     PortalGroup=1@10.193.184.81:3260,[2,3]
     PortalGroup=2@10.193.184.85:3261
     PortalGroup=3@10.193.184.85:3262
     ShadowMode=No
#<snip>
```

4.9. The command line interface tools "iscsictl" & "chisns"

4.9.1. iscsictl

`iscsictl` is the tool Chelsio provides for controlling the iSCSI target. It is a Command Line Interface (CLI) that is invoked from the console. Its usage is as follows:

```
iscsictl <options> <mandatory parameters> [optional parameters]
```

The mandatory and optional parameters are the **key=value** pair(s) defined in RFC3720, or the **var=const** pair(s) defined for Chelsio iSCSI driver implementation. In this document, the key=value is referred to as "pair", and var=const is referred to as "parameter" to clarify between iSCSI protocol‟s pair value(s), and Chelsio iSCSI driver‟s parameter value(s). Note that all **value** and **const** are case sensitive.

4.9.2. chisns

`chisns` is the command line tool for controlling the iSNS client. This is a simple tool that starts the iSNS client with a client and server parameter.

4.9.3. iscsictl options

Options	Mandatory Parameters	Optional Parameters	Description
`-h`			Display the help messages.
`-v`			Display the version.
`-f`	`<[path/] filename>`		Specifies a pre-written iSCSI configuration text file, used to start, update, save, or reload the iSCSI node(s). This option must be specified with one of the following other options: "**-S**" or "**-W**". For the "**-S**" option "**-f**" must be specified first. All other options will ignore this "**-f**" option. If the "**-f**" option is not specified with the commands above the default configuration file will be used. It‟s name and location is: `/etc/chelsio-iscsi/chiscsi.conf` The configuration file path and filename must conform to Linux standards. For the format of the iSCSI configuration file, please see "Format of The iSCSI Configuration File" section earlier in this document.
`-k`	`<key>[=<val>]`		Specifies an iSCSI Entity or Chelsio Entity parameter. This option can be specified after "**-c**" option to retrieve a parameter setting..
`-c`	`target=<name> [,name2 . . . ,<nameN>]`		Display the Chelsio iSCSI target configuration. **target=<name>** parameter: Where **name** is the name of the node whose information will be returned. **name** can be one or more string of names, separated by a comma, **<name1[,name2,…,nameN] | ALL>** A **name** of **ALL** returns information on all targets. **ALL** is a reserved string that must be uppercase. Example: **iscsictl -c target=iqn.com.cc.it1** **Iscsictl -c target=iqn.com.cc.target1 -k**

			TargetAlias The **<name>** parameter can also be specified as one or more parameter on the same command line, separated by a comma, **target=<name1>, <name2>, … ,<nameN>** The **target=<name>** parameter(s) are optional and if not specified all active Chelsio iSCSI targets(s) configuration(s) will be displayed. If **target=ALL** is specified or no parameters are specified the output will be abbreviated. Specify specific targets to get detailed configuration data. If the **target=<name>** option is specified, the -k <key> option can optionally be specified along with this option to display only the selected entity parameter setting. Example: **iscsictl -c target=iqn.com.cc.target1 -k HeaderDigest**
`-F`		`target=<name>` `-k lun=<value>`	Flush the cached data to the target disk(s). **target=<name>** parameter: Where **name** is the name of the target to be flushed. **name** can be one or more string of names, separated by a comma, **<name1[,name2,…,nameN] | ALL>** A **name** of **ALL** will cause all the target data to be flushed. **ALL** is a reserved string that must be uppercase. The **target=name** parameter is optional. If no **target=name** parameter is specified, it is the same as specifying **target=ALL**. The -k lun=<value> option is optional. It can be used to further specify a particular lun to be flushed. Example: To flush all the targets in the system: **iscsictl -F** To flush a particular target: **iscsictl -F target=iqn.com.cc.it1** To flush only the lun 0 of a particular target: **iscsictl -F target=iqn.com.cc.it1 -k lun=0**
`-g`			Display the Chelsio iSCSI Global Entity Settings.
`-G`	`<var=const>`		Set the Chelsio iSCSI Global Entity Settings. **var=const** parameter: Where **var=const** can be any key listed under Chelsio Global Entity Settings. Example: **iscsictl -G iscsi_auth_order=ACL**

			The **var=const** parameter(s) are mandatory. If the **var=const** parameter is not specified, the command will be denied. If any of the specified **var=const** parameter is invalid, the command will reject only the invalid parameters, but will continue on and complete all other valid parameters if any others are specified.
`-s`	`target=<name>`		Stop the specified active iSCSI targets. **target=<name>** parameter: *See the description of option -c for the target=<name> parameter definition.* The **target=<name>** parameter is mandatory. If no **target=<name>** parameter is specified, the command will be denied. If the **target=<name>** parameter is specified, only the specified targets from the **target=<name>** parameters will be stopped. If **target=ALL** is specified, all active targets will stop.
`-S`	`target=<name>`		Start or reload the iSCSI targets. **target=<name>** parameter: Where **name** is the name of the target(s) that will be started or reloaded. The **target=<name>** parameter can be specified as one or more parameter on the same command line, separated by a space, **target=<name1> target=<name2> … target=<nameN>** The **target=<name>** parameter can also be, **target=ALL** A **name** of **ALL** starts or reloads all targets specified in the configuration file. **ALL** is a reserved string that must be uppercase. The **target=<name>** parameter is optional. If this command line option is specified without the -f option, the default configuration file `/etc/chelsio-iscsi/chiscsi.conf` will be used. Rules, 1. If the **target=<name>** parameter is specified, only the targets from the list will be started or reloaded. 2. If **target=ALL** is specified, all targets specified from the iSCSI configuration file will be started or reloaded. 3. If the **target=<name>** parameter is not specified, all active targets configurations will be reloaded from the configuration file while those targets are running. All non-active targets specified will not be loaded / started.

			For Rules 1-3, if the specified targets are currently active (running), they will get reloaded. For Rules 1 & 2, if the specified targets are not currently active, they will be started. For Rules 2 & 3, please note the differences – they are not the same! The global settings are also reloaded from the configuration file with this option.
-r	target=<name>	-k initiator=<name>	Retrieve active iSCSI sessions under a target. **target=<name>** parameter: Where **name** must be a single target name. If **target=<name>** parameter is specified as target=<name>, the sessions can be further filtered based on the remote node name with optional –k initiator=<name> option. Examples: **iscsictl -r target=iqn.com.cc.it1** **iscsictl -r target=iqn.com.cc.it1 -k initiator=iqn.com.cc.ii1** The first **target=<name>** parameter is mandatory. If it is not specified, the command will be denied.
-D	<Session handle in hex>		Drop initiator session. This option should be specified with the handle of the session (in hex) that needs to be dropped. The session handle can be retrieved using the previous mentioned iscsictl option (-r used to retrieve active iSCSI sessions under a target).
-W			Overwrite the specified iSCSI configuration file with ONLY the current iSCSI global settings and the active iSCSI targets" configuration to the specified iSCSI configuration file. *Will delete any non-active targets' configuration from the specified file.* The -f option MUST be specified along with this option.
-h			Display the help messages.
	server=<IP address> [:<port>]	id=<isns entity id> query=<query interval>	Start the Chelsio iSNS client. **server=<IP address>[:<port>]** where **server** is the iSNS server address. The port is optional and if it"s not specified it defaults to 3205. The server with the ip address is mandatory and if it"s not specified the, the command will be denied. **id=<isns entity id>** where **id** is the iSNS entity ID used to register with the server. It defaults to <hostname>. **query=<query interval>** where **query** is the initiator query interval (in seconds). It defaults to 60 seconds. Examples: chisns server=192.0.2.10

			chisns server=192.0.2.10:3205 id=isnscln2 query=30 In the first example the minimum command set is given where the IP address of the iSNS server is specified. In the second example a fully qualified command is specified by also setting three optional parameters. Here, the mandatory IP address and the corresponding optional port number are specified. Also set is the iSNS entity ID to "`isnscln2`" as well as the query interval to 30 seconds.

4.9.4. chisns options

Options	Mandatory Parameters	Optional Parameters	Description
`-h`			Display the help messages.

	`server=<IP address> [:<port>]`	`id=<isns entity id>`
		
		`query=<query interval>`
			Start the Chelsio iSNS client.
			
			server=<IP address>[:<port>] where **server** is the iSNS server address. The port is optional and if it's not specified it defaults to 3205. The server with the ip address is mandatory and if it's not specified the, the command will be denied.
			
			id=<isns entity id> where **id** is the iSNS entity ID used to register with the server. It defaults to <hostname>.
			
			query=<query interval> where **query** is the initiator query interval (in seconds). It defaults to 60 seconds.
			
			Examples:
			
			chisns server=192.0.2.10
			chisns server=192.0.2.10:3205 id=isnscln2 query=30
			
			In the first example the minimum command set is given where the IP address of the iSNS server is specified.
			
			In the second example a fully qualified command is specified by also setting three optional parameters. Here, the mandatory IP address and the corresponding optional port number are specified. Also set is the iSNS entity ID to 'isnscln2' as well as the query interval to 30 seconds.

4.10. Rules of Target Reload (i.e. "on the fly" changes)

After a target has been started its settings can be modified via reloading of the configuration file (i.e., `iscsictl -S`).

The following parameters cannot be changed once the target is up and running otherwise the target reload would fail:

- TargetName
- TargetSessionMaxCmd
- ACL_Enable
- ACL

The following parameters **CAN** be changed by reloading of the configuration file. The new value will become effective **IMMEDIATELY** for all connections and sessions:

- TargetDevice
- PortalGroup

The following parameter **CAN** be changed by reloading of the configuration file. The new value will **NOT** affect any connections and sessions that already completed login phase:

- TargetAlias
- MaxConnections
- InitialR2T
- ImmediateData
- FirstBurstLength
- MaxBurstLength
- MaxOutstandingR2T
- HeaderDigest
- DataDigest
- MaxRecvDataSegmentLength
- AuthMethod
- Auth_CHAP_Initiator
- Auth_CHAP_Target
- Auth_CHAP_ChallengeLength
- Auth_CHAP_Policy

The following parameters **SHOULD NOT** be changed because only one valid value is supported:

- DataPDUInOrder (support only "Yes")
- DataSequenceInOrder (support only "Yes")
- ErrorRecoveryLevel (support only "0")
- OFMarker (support only "No")
- IFMarker (support only "No")

The following parameters can be changed but would not have any effect because they are either not supported or they are irrelevant:

- DefaultTime2Wait (not supported)
- DefaultTime2Retain (not supported)

- OFMarkInt (irrelevant because OFMarker=No)
- IFMarkInt (irrelevant because IFMarker=No)

4.11. System Wide Parameters

The Chelsio Global Entity Settings are system wide parameters that can be controlled through the configuration file or the use of the command line "`iscsictl -G`". The finer points of some of these parameters are described in detail here:

4.11.1. iscsi_login_complete_time

Options: An integer value between 0 and 3600 (seconds). Default value is 300 (seconds).

This is the login timeout check. The parameter controls the maximum time (in seconds) allowed to the initiator to complete the login phase. If a connection has been in the login phase longer than the set value, the target will drop the connection.

Value zero turns off this login timeout check.

4.11.2. iscsi_auth_order

Options: "ACL" or "CHAP", defaults to "CHAP"

On an iSCSI target when `ACL_Enable` is set to `Yes`, `iscsi_auth_order` decides whether to perform CHAP first then ACL or perform ACL then CHAP.

- **ACL:** When setting `iscsi_auth_order=ACL`, initiator authorization will be performed at the start of the login phase for an iSCSI normal session: upon receiving the first iscsi_login_request, the target will check its ACL. If this iSCSI connection does not match any ACL provisioned, the login attempt will be terminated.

- **CHAP**: When setting `iscsi_auth_order=CHAP`, initiator authorization will be performed at the end of the login phase for an iSCSI normal session: before going to the full feature phase, the target will check its ACL. If this iSCSI connection does not match any ACL provisioned, the login attempt will be terminated.

Note *iscsi_auth_order has no meaning when* `ACL_Enable` *is set to* `No` *on a target.*

4.11.3. iscsi_target_vendor_id

Options: A string of maximum of 8 characters, defaults to `CHISCSI`

The `iscsi_target_vendor_id` is part of the device identification sent by an iSCSI target in response of a SCSI Inquiry request.

4.11.4. iscsi_chelsio_ini_idstr

Options: A string of maximum of 255 characters, defaults to “cxgb4i”.

For an iscsi connection, more optimization can be done when both initiator and target are running Chelsio adapters and drivers.

This string is used to verify the initiator name received, and identify if the initiator is running Chelsio drivers: if the initiator name contains the same substring as `iscsi_chelsio_ini_idstr` it is assumed the initiator is running with the Chelsio iscsi initiator driver and additional offload optimization is performed.

4.12. Performance Tuning

i. **See performance tuning section** in the Unified Wire chapter for generic performance settings.
ii. Next, load the iSCSI PDU offload target driver (*chiscsi_t4*) and run the *chiscsi_set_affinity.sh* script to map iSCSI worker threads to different CPUs.

```
[root@host~]# chiscsi_set_affinity.sh
```

V. iSCSI PDU Offload Initiator

1. Introduction

The Chelsio T5/T4 series Adapters support iSCSI acceleration and iSCSI Direct Data Placement (DDP) where the hardware handles the expensive byte touching operations, such as CRC computation and verification, and direct DMA to the final host memory destination:

- **iSCSI PDU digest generation and verification**
  On transmit -side, Chelsio hardware computes and inserts the Header and Data digest into the PDUs. On receive-side, Chelsio hardware computes and verifies the Header and Data digest of the PDUs.

- **Direct Data Placement (DDP)**
  Chelsio hardware can directly place the iSCSI Data-In or Data-Out PDU's payload into pre-posted final destination host-memory buffers based on the Initiator Task Tag (ITT) in Data-In or Target Task Tag (TTT) in Data-Out PDUs.

- **PDU Transmit and Recovery**
  On transmit-side, Chelsio hardware accepts the complete PDU (header + data) from the host driver, computes and inserts the digests, decomposes the PDU into multiple TCP segments if necessary, and transmit all the TCP segments onto the wire. It handles TCP retransmission if needed.
  On receive-side, Chelsio hardware recovers the iSCSI PDU by reassembling TCP segments, separating the header and data, calculating and verifying the digests, then forwarding the header to the host. The payload data, if possible, will be directly placed into the pre-posted host DDP buffer. Otherwise, the data will be sent to the host too.

The *cxgb4i* driver interfaces with open-iSCSI initiator and provides the iSCSI acceleration through Chelsio hardware wherever applicable.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with iSCSI PDU Offload Initiator Software:

- T520-BT
- T580-CR
- T520-LL-CR
- T520-CR
- T580-LP-CR
- T540-CR
- T420-CR

- T440-CR
- T422-CR
- T404-BT
- T420-BCH
- T440-LP-CR
- T420-BT
- T420-LL-CR
- T420-CX

1.2. Software Requirements

1.2.1. Linux Requirements

Currently the iSCSI PDU Offload Initiator software is available for the following versions:

- RHEL 7.2, 4.2.0-0.21.el7.aarch64 (ARM64)
- Ubuntu 14.04.3, 3.19.0-25-generic (POWER8)

Other kernel versions have not been tested and are not guaranteed to work.

2. Software/Driver Loading

Important *Please ensure that all inbox drivers are unloaded before proceeding with unified wire drivers.*

The driver must be loaded by the root user. Any attempt to load the driver as a regular user will fail.

Run the following command to load the driver:

```
[root@host~]# modprobe cxgb4i
```

On SLES, load the driver with `--allow` option:

```
[root@host~]# modprobe cxgb4i --allow
```

If loading of cxgb4i displays "unkown symbols found" error in dmesg, follow the steps mentioned below:

i. Kill iSCSI daemon *iscsid*
ii. View all the loaded iSCSI modules

```
[root@host~]# lsmod | grep iscsi
```

iii. Now, unload them using the following command:

```
[root@host~]# rmmod <modulename>
```

iv. Finally reload the *cxgb4i* driver.

3. Software/Driver Unloading

To unload the driver, execute the following commands:

```
[root@host~]# rmmod cxgb4i
[root@host~]# rmmod libcxgbi
```

4. Software/Driver Configuration and Fine-tuning

4.1. Accelerating open-iSCSI Initiator

The following steps need to be taken to accelerate the open-iSCSI initiator:

4.1.1. Configuring interface (*iface)* file

Create an interface file located under *iface* directory for the new transport class *cxgb4i* in the following format:

```
iface.iscsi_ifacename = <iface file name>
iface.hwaddress = <MAC address>
iface.transport_name = cxgb4i
iface.net_ifacename = <ethX>
iface.ipaddress = <iscsi ip address>
```

E.g.:-

```
iface.iscsi_ifacename = cxgb4i.00:07:43:04:5b:da
iface.hwaddress = 00:07:43:04:5b:da
iface.transport_name = cxgb4i
iface.net_ifacename = eth3
iface.ipaddress = 102.2.2.137
```

Alternatively, you can create the file automatically by executing the following command:

```
[root@host~]# iscsiadm -m iface
```

Here,

- `iface.iscsi_ifacename` denotes the name of interface file in `/etc/iscsi/ifaces/`.
- `iface.hwaddress` denotes the MAC address of the Chelsio interface via which iSCSI traffic will be running.
- `iface.transport_name` denotes the transport name, which is cxgb4i.
- `iface.net_ifacename` denotes the Chelsio interface via which iSCSI traffic will be running.
- `iface.ipaddress` denotes the IP address which is assigned to the interface.

Note

i. *The interface file needs to be created in* `/etc/iscsi/iscsid.conf`*.*
ii. *If* `iface.ipaddress` *is specified, it needs to be either the same as the ethX's IP address or an address on the same subnet. Make sure the IP address is unique in the network.*

4.1.2. Discovery and Login

i. Starting iSCSI Daemon

Start Daemon from */sbin* by using the following command:

```
[root@host~]# iscsid -f -d 3
```

Note *If* `iscsid` *is already running, then kill the service and start it as shown above after installing the Chelsio Unified Wire package.*

ii. Discovering iSCSI Targets

To discovery an iSCSI target execute a command in the following format:

```
iscsiadm -m discovery -t st -p <target ip address>:<target port no> -I
<cxgb4i iface file name>
```

E.g.:-

```
[root@host~]# iscsiadm -m discovery -t st -p 102.2.2.155:3260 -I
cxgb4i.00:07:43:04:5b:da
```

iii. Logging into an iSCSI Target

Log into an iSCSI target using the following format:

```
iscsiadm -m node -T <iqn name of target> -p <target ip address>:<target port
no> -I  <cxgb4i iface file name> -l
```

E.g.:-

```
[root@host~]# iscsiadm -m node -T iqn.2004-05.com.chelsio.target1 -p
102.2.2.155:3260,1 -I cxgb4i.00:07:43:04:5b:da -l
```

If the login fails with an error message in the format of `ERR! MaxRecvSegmentLength <X> too big. Need to be <= <Y>.` in dmesg, edit the `iscsi/iscsid.conf` file and change the setting for `MaxRecvDataSegmentLength`:

```
node.conn[0].iscsi.MaxRecvDataSegmentLength = 8192
```

Important *Always take a backup of `iscsid.conf` file before installing Chelsio Unified Wire Package.*

iv. Logging out from an iSCSI Target

Log out from an iSCSI Target by executing a command in the following format:

```
iscsiadm -m node -T <iqn name of target> -p <target ip address>:<target port no> -I  <cxgb4i iface file name> -u
```

E.g.:-

```
[root@host~]# iscsiadm -m node -T iqn.2004-05.com.chelsio.target1 -p 102.2.2.155:3260,1 -I cxgb4i.00:07:43:04:5b:da -u
```

Note *Other options can be found by typing* `iscsiadm --help`

4.2. Auto login from cxgb4i initiator at OS bootup

For iSCSI auto login (via *cxgb4i*) to work on OS startup, please add the following line to `start()` in `/etc/rc.d/init.d/iscsid` file on RHEL:

```
modprobe -q cxgb4i
```

E.g.:-

```
force_start() {
    echo -n $"Starting $prog: "
    modprobe -q iscsi_tcpmodprobe -q ib_iser
    modprobe -q cxgb4i
    modprobe -q cxgb3i
    modprobe -q bnx2i
    modprobe -q be2iscsi
    daemon brcm_iscsiuio
    daemon $prog
    retval=$?
    echo
    [ $retval -eq 0 ] && touch $lockfile
    return $retval
}
```

VI. Offload Bonding driver

1. Introduction

The Chelsio Offload bonding driver provides a method to aggregate multiple network interfaces into a single logical bonded interface effectively combining the bandwidth into a single connection. It also provides redundancy in case one of link fails.

The traffic running over the bonded interface can be fully offloaded to the T5/T4 Adapter, thus freeing the CPU from TCP/IP overhead.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with the Chelsio Offload Bonding driver:

- T520-BT
- T580-CR
- T520-LL-CR
- T520-CR
- T580-LP-CR
- T540-CR
- T420-CR
- T440-CR
- T422-CR
- T420-SO-CR
- T404-BT
- T420-BCH
- T440-LP-CR
- T420-BT
- T420-LL-CR
- T420-CX

1.2. Software Requirements

1.2.1. Linux Requirements

Currently the Offload Bonding driver is available for the following versions:

- RHEL 7.2, 4.2.0-0.21.el7.aarch64 (ARM64)
- Ubuntu 14.04.3, 3.19.0-25-generic (POWER8)

Other kernel versions have not been tested and are not guaranteed to work.

2. Software/Driver Loading

Important *Please ensure that all inbox drivers are unloaded before proceeding with unified wire drivers.*

The driver must be loaded by the root user. Any attempt to load the driver as a regular user will fail.

To load the driver (with offload support), run the following command:

```
[root@host~]# modprobe bonding
```

3. Software/Driver Unloading

To unload the driver, run the following command:

```
[root@host~]# rmmod bonding
```

4. Software/Driver Configuration and Fine-tuning

4.1. Offloading TCP traffic over a bonded interface

The Chelsio Offload Bonding driver supports all the bonding modes in NIC Mode. In offload mode (t4_tom loaded) however, only the **balance-rr (mode=0)**, **active-backup (mode=1)**, **balance-xor (mode=2)** and **802.3ad (mode=4)** modes are supported.

To offload TCP traffic over a bonded interface, use the following method:

i. Load the network driver with TOE support:

```
[root@host~]# modprobe t4_tom
```

ii. Create a bonded interface:

```
[root@host~]# modprobe bonding mode=1 miimon=100
```

iii. Bring up the bonded interface and enslave the interfaces to the bond:

```
[root@host~]# ifconfig bond0 up
[root@host~]# ifenslave bond0 ethX ethY
```

Note ***ethX*** *and* ***ethY*** *are interfaces of the same adapter.*

iv. Assign IPv4/IPv6 address to the bonded interface:

```
[root@host~]# ifconfig bond0 X.X.X.X/Y
[root@host~]# ifconfig bond0 inet6 add <128-bit IPv6 Address> up
```

v. Disable FRTO on the PEER:

```
[root@host~]# sysctl -w net.ipv4.tcp_frto=0
```

All TCP traffic will be offloaded over the bonded interface now.

VII. Offload IPv6 driver

1. Introduction

The growth of the Internet has created a need for more addresses than are possible with IPv4. Internet Protocol version 6 (IPv6) is a version of the Internet Protocol (IP) designed to succeed the Internet Protocol version 4 (IPv4).

Chelsio's Offload IPv6 feature provides support to fully offload IPv6 traffic to the T5/T4 adapter.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with Chelsio Offload IPv6 feature:

- T520-BT
- T580-CR
- T520-LL-CR
- T520-CR
- T580-LP-CR
- T540-CR
- T420-CR
- T440-CR
- T422-CR
- T420-SO-CR
- T404-BT
- T420-BCH
- T440-LP-CR
- T420-BT
- T420-LL-CR
- T420-CX

1.2. Software Requirements

1.2.1. Linux Requirements

Currently the Offload IPv6 feature is available for the following versions:

- RHEL 7.2, 4.2.0-0.21.el7.aarch64 (ARM64)
- Ubuntu 14.04.3, 3.19.0-25-generic (POWER8)

Other kernel versions have not been tested and are not guaranteed to work.

2. Software/Driver Loading

Important *Please ensure that all inbox drivers are unloaded before proceeding with unified wire drivers.*

IPv6 must be enabled in your system (enabled by default) to use the Offload IPv6 feature. Also, Unified Wire package must be installed with IPv6 support (see Software/Driver Installation).

After installing Unified Wire package and rebooting the host, load the NIC (*cxgb4*) and TOE (*t4_tom*) drivers. The drivers must be loaded by the root user. Any attempt to load the drivers as a regular user will fail.

```
[root@host~]# modprobe cxgb4
[root@host~]# modprobe t4_tom
```

All the IPv6 traffic will be offloaded now.

3. Software/Driver Unloading

To disable Offload IPv6 feature, unload NIC and TOE drivers:

3.1. Unloading the NIC driver

To unload the NIC driver, run the following command:

```
[root@host~]# rmmod cxgb4
```

3.2. Unloading the TOE driver

Please reboot the system to unload the TOE driver. To unload without rebooting, refer **Unloading the TOE driver** section of **Network (NIC/TOE)** chapter.

VIII. Classification and Filtering

1. Introduction

Classification and Filtering feature enhances network security by controlling incoming traffic as they pass through network interface based on source and destination addresses, protocol, source and receiving ports, or the value of some status bits in the packet. This feature can be used in the ingress path to:

- Steer ingress packets that meet ACL (Access Control List) accept criteria to a particular receive queue.
- Switch (proxy) ingress packets that meet ACL accept criteria to an output port, with optional header rewrite.
- Drop ingress packets that fail ACL accept criteria.

1.1. Hardware Requirements

1.1.1. Supported Adapters

The following are the currently shipping Chelsio Adapters that are compatible with the Classification and Filtering feature:

- T520-BT
- T580-CR
- T520-LL-CR
- T520-SO-CR
- T520-CR
- T540-CR
- T580-LP-CR
- T580-SO-CR
- T420-CR
- T440-CR
- T422-CR
- T420-SO-CR
- T404-BT
- T420-BCH
- T440-LP-CR
- T420-BT
- T420-LL-CR
- T420-CX

1.2. Software Requirements

1.2.1. Linux Requirements

Currently the Classification and Filtering feature is available for the following versions:

- RHEL 7.2, 4.2.0-0.21.el7.aarch64 (ARM64)
- Ubuntu 14.04.3, 3.19.0-25-generic (POWER8)

Other kernel versions have not been tested and are not guaranteed to work.

*Limited QA Peformed.

2. Usage

2.1. Configuration

The Classification and Filtering feature is configured by specifying the filter selection combination set in the firmware configuration (*t5-config.txt for T5; t4-config.txt for T4)* located in */lib/firmware/cxgb4/*

The following combination is set by default and packets will be matched accordingly:

i. For T5:

```
filterMode = fcoemask, srvrsram, fragmentation, mpshittype, protocol, vlan,
port, fcoe
```

ii. For T4:

```
filterMode = fragmentation, mpshittype, protocol, vlan, port, fcoe
```

Where,

srvrsram *:* server-sram
fragmentation: Fragmented IP packets
mpshittype : MAC address "match type" (0=unicast, 1=unicast hash, 2=multicast, 3=multicast hash, 4=PROM, 5=hyper PROM, 6=broadcast, 7=none)
protocol : IP protocol number (ICMP=1, TCP=6, UDP=17, etc)
vlan : Inner VLAN Tag
port : Packet ingress port number
fcoe : Fibre Channel over Ethernet frames

2.2. Creating Filter Rules

Network driver (*cxgb4*) must be installed and loaded before setting the filter rule.

i. If you haven't done already, run the Unified Wire Installer with the appropriate T5/T4 configuration tuning option to install the Network Driver.
ii. Next, run the following command to load the network driver:

```
[root@host~]# modprobe cxgb4
```

iii. Now, create filter rules using *cxgbtool*:

```
[root@host~]# cxgbtool ethx filter <index> action [pass/drop/switch]
```

Where,

ethX : Chelsio interface
index : positive integer set as filter id
action : Ingress packet disposition
pass : Ingress packets will be passed through set ingress queues
switch : Ingress packets will be routed to an output port with optional header rewrite.
drop : Ingress packets will be dropped.

Note *In case of multiple filter rules, the rule with the lowest filter index takes higher priority.*

2.2.1. Examples

- ***Drop* action**

```
[root@host~]# cxgbtool ethX filter 0 action drop fip 192.168.1.5
```

The above filter rule will drop all ingress packets from IP 192.168.1.5

- ***Pass* action**

```
[root@host~]# cxgbtool ethX filter 0 action pass lport 10001 fport 355
queue 2
```

The above filter rule will pass all ingress packets that match local port 10001 and remote port 355 to ingress queue 2 for load balancing.

- ***Switch* action**

```
[root@host~]# cxgbtool ethX filter 0 action switch iport 0 eport 1 vlan 3
```

The above filter rule will route all ingress packets that match VLAN id 3 from port 0 of Chelsio adapter to port 1. Remaining packets will be sent to the host.

For offloaded ingress packets, use the `prio` argument with the above command:

```
[root@host~]# cxgbtool ethx filter <index> action <pass/drop/switch> prio 1
```

Note

- *For more information on additional parameters, refer cxgbtool manual by running the `man cxgbtool` command.*
- `prio` *argument is not supported for LE-TCAM filters when T5 Hash Filter config file is used.*

2.3. Listing Filter Rules

To list the filters set, run the following command:

```
[root@host~]# cxgbtool ethX filter show
```

2.4. Removing Filter Rules

To remove a filter, run the following command with the corresponding filter rule index

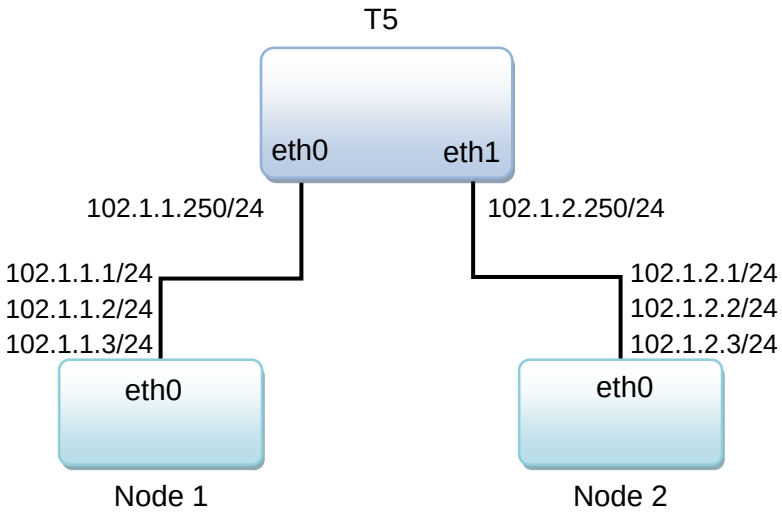
```
[root@host~]# cxgbtool ethX filter index <delete|clear>
```

Note

For more information on additional parameters, refer cxgbtool manual by running the `man cxgbtool` command

2.5. Layer 3 example

Here's an example on how to achieve L3 routing functionality:



- **Follow these steps on Node 1**

i. Configure IP address and enable the 3 interfaces:

```
[root@host~]# ifconfig eth0 102.1.1.1/24 up
[root@host~]# ifconfig eth0:2 102.1.1.2/24 up
[root@host~]# ifconfig eth0:3 102.1.1.3/24 up
[root@host~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:07:43:04:7D:50
          inet addr:102.1.1.1  Bcast:102.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::207:43ff:fe04:7d50/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:14372 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62203 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1585952 (1.5 MiB)  TX bytes:4798122 (4.5 MiB)
          Interrupt:16
```

ii. Setup a static OR default route towards T5 router to reach 102.1.2.0/24 network

```
[root@host~]# route add -net 102.1.2.0/24 gw 102.1.1.250
```

- **Follow these steps on Node 2**

i. Configure IP address and enable the 3 interfaces:

```
[root@host~]# ifconfig eth0 102.1.2.1/24 up
[root@host~]# ifconfig eth0:2 102.1.2.2/24 up
[root@host~]# ifconfig eth0:3 102.1.2.3/24 up
[root@host~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:07:43:12:D4:88
          inet addr:102.1.2.1  Bcast:102.1.2.255  Mask:255.255.255.0
          inet6 addr: fe80::7:4300:112:d488/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1961 errors:0 dropped:2 overruns:0 frame:0
          TX packets:141 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:218606 (213.4 KiB)  TX bytes:17483 (17.0 KiB)
          Interrupt:17
```

ii. Setup a static OR default route towards T5 router to reach 102.1.1.0/24 network

```
[root@host~]# route add -net 102.1.1.0/24 gw 102.1.2.250
```

- **Follow these steps on machine with T5 adapter**

i. Configure IP address and enable the 2 interfaces:

```
[root@host~]# ifconfig eth0 102.1.1.250/24 up
[root@host~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:07:43:04:96:40
          inet addr:102.1.1.250  Bcast:102.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::207:43ff:fe04:9640/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:114 errors:0 dropped:0 overruns:0 frame:0
          TX packets:535 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11880 (11.6 KiB)  TX bytes:61729 (60.2 KiB)
          Interrupt:16
[root@host~]# ifconfig eth1 102.1.2.250/24 up
[root@host~]# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:07:43:04:96:48
          inet addr:102.1.2.250  Bcast:102.1.2.255  Mask:255.255.255.0
          inet6 addr: fe80::7:4300:104:9648/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31 errors:0 dropped:0 overruns:0 frame:0
          TX packets:433 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3181 (3.1 KiB)  TX bytes:49134 (47.9 KiB)
          Interrupt:16
```

ii. Create filter rule to send packets for 102.1.2.0/24 network out via eth1 interface:

```
[root@host~]# cxgbtool eth0 filter 0 lip 102.1.2.0/24 hitcnts 1 action
switch  eport 1 smac 00:07:43:04:96:48 dmac 00:07:43:12:D4:88
```

Where, `smac` is the MAC address of eth1 interface on T5 adapter machine and `dmac` is the MAC address of eth0 interface on Node 2.

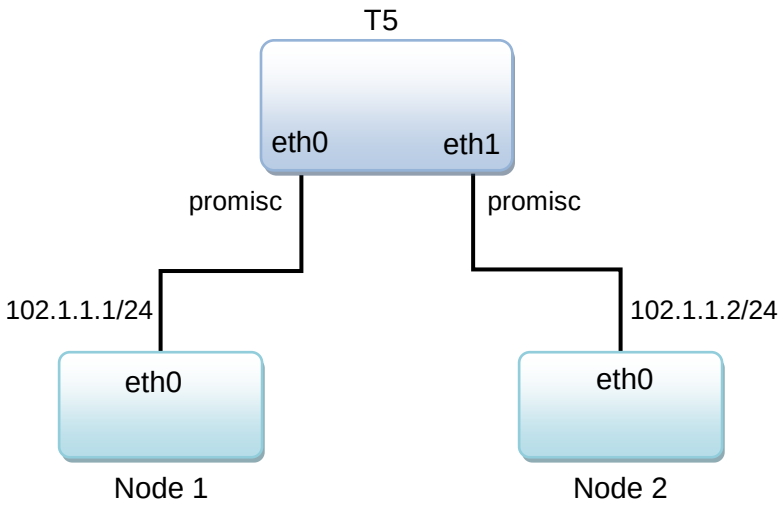
iii. Create filter rule to send packets for 102.1.1.0/24 network out via eth0 inteface

```
[root@host~]# cxgbtool eth0 filter 1 lip 102.1.1.0/24 hitcnts 1 action
switch eport 0 smac 00:07:43:04:96:40 dmac 00:07:43:04:7D:50
```

Where, `smac` is the MAC address of eth0 interface on T5 adapter machine and `dmac` is the MAC address of eth0 interface on Node 1.

2.6. Layer 2 example

Here's an example on how to achieve L2 switching functionality. The following will only work on kernel 3.10 and above.



- **Follow these steps on Node 1**

i. Configure IP address and enable the interface:

```
[root@host~]# ifconfig eth0 102.1.1.1/24 up
[root@host~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:07:43:04:7D:50
          inet addr:102.1.1.1  Bcast:102.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::207:43ff:fe04:7d50/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:14372 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62203 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1585952 (1.5 MiB)  TX bytes:4798122 (4.5 MiB)
          Interrupt:16
```

ii. Setup ARP entry to reach 102.1.1.2

```
[root@host~]# arp -s 102.1.1.2 00:07:43:12:D4:88
```

- **Follow these steps on Node 2**

i. Configure IP address and enable the interface:

```
[root@host~]# ifconfig eth0 102.1.1.2/24 up
[root@host~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:07:43:12:D4:88
          inet addr:102.1.1.2  Bcast:102.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::7:4300:112:d488/64 Scope:Link
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:1961 errors:0 dropped:2 overruns:0 frame:0
          TX packets:141 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:218606 (213.4 KiB)  TX bytes:17483 (17.0 KiB)
          Interrupt:17
```

ii. Setup ARP entry to reach 102.1.1.1

```
[root@host~]# arp -s 102.1.1.1 00:07:43:04:7D:50
```

- **Follow these steps on machine with T5 adapter**

i. Update filtermode value with below combination in */lib/firmware/cxgb4/t5-config.txt* to enable matching based on macidx

```
filterMode = fragmentation, macmatch, mpshittype, protocol, tos, port, fcoe
```

ii. Unload and re-load the *cxgb4* driver.
iii. Enable promiscuous mode on both the interfaces on T5 adapter machine:

```
[root@host~]# ifconfig eth0 up promisc
[root@host~]# ifconfig eth1 up promisc
```

iv. Build and install latest iproute2 package
v. Add fdb entry corresponding to Node-2 on T5's eth0 interface:

```
[root@host~]# bridge fdb add 00:07:43:12:D4:88 dev eth0 self
```

vi. Add fdb entry corresponding to Node-1 on T5's eth1 interface:

```
[root@host~]# bridge fdb add 00:07:43:04:7D:50 dev eth1 self
```

vii. Both MAC entries should show up in MPS table. Run the following command to view the table and note the index (*idx* field) of the entries:

```
[root@host~]# cat /sys/kernel/debug/cxgb4/0000\:01\:00.4/mps_tcam | more
```

viii. Create a filter to match incoming packet's *dst-mac 00:07:43:12:d4:88* with particular mac-idx and switch it out via eport 1:

```
[root@host~]# cxgbtool eth0 filter 0 macidx 5 action switch eport 1 hitcnts
1
```

ix. Create a filter to match incoming packet's *dst-mac 00:07:43:04:7d:50* with particular mac-idx and switch it out via eport 0:

```
[root@host~]# cxgbtool eth0 filter 1 macidx 7 action switch eport 0 hitcnts
1
```

3. Hash/DDR Filters

The default (*Unified Wire)* configuration tuning option allows you to create LE-TCAM filters, which has a limit of 496 filter rules. If you wish to create more, select *T5 Hash Filter* configuration tuning option during installation which allows you to create HASH/DDR filters with a capacity of ~0.5 million filter rules.

Note *Creating Hash/DDR Filters is currently supported only on T5 adapters.*

3.1. Creating Filter Rules

Network driver (*cxgb4*) must be installed and loaded before setting the filter rule.

i. If you haven't done already, run the Unified Wire Installer with the *T5 Hash Filter* configuration tuning option to install the Network Driver.
ii. Load the network driver with DDR filters support :

```
[root@host~]# modprobe cxgb4 use_ddr_filters=1
```

iii. Now, create filter rules using cxgbtool:

```
[root@host~]# cxgbtool ethX filter <index> action [pass/drop/switch] fip
<source ip of incoming packet> lip <destination ip of incoming packet> fport
<source port> lport <destination port> hitcnts 1 cap maskless
```

Where,

ethX : Chelsio interface
index : Filter index. The user must provide a positive integer, which will be ignored and replaced by an automatically computed index, based on the hash (4-tuple).The index will be displayed after the filter rule is created successfully.
action : Ingress packet disposition
pass : Ingress packets will be passed through set ingress queues
switch : Ingress packets will be routed to an output port with optional header rewrite.
drop : Ingress packets will be dropped

Note *"source IP", "destination IP", "source port" and "destination port" are mandatory parameters since Hash filters don't support masks and hence, 4-tuple must always be supplied for Hash filter. "cap maskless" parameter should be appended in order to create Hash/DDR filter rules. Otherwise the above command will create LE-TCAM filter rules.*

3.1.1. Examples

- ***Drop* action**

```
[root@host~]# cxgbtool ethX filter 496 action drop lip 102.1.1.1 fip
102.1.1.2 lport 12865 fport 20000 hitcnts 1 cap maskless iport 1 proto 17
Hash-Filter Index = 61722
```

The above filter rule will drop all UDP packets matching above 4 tuple coming on chelsio port 1.

- ***Pass* action**

```
[root@host~]# cxgbtool ethX filter 496 action pass lip 102.2.2.1 fip
102.2.2.2 lport 12865 fport 12000 hitcnts 1 cap maskless proto 6
Hash-Filter Index = 308184
```

The above filter rule will pass all TCP packets matching above 4 tuple.

- ***Switch* action**

```
[root@host~]# cxgbtool ethX filter 496 action switch lip 102.3.3.1 fip
102.3.3.2 lport 5001 fport 16000 iport 0 eport 1 hitcnts 1 cap maskless
Hash-Filter Index = 489090
```

The above filter rule will switch all the packets matching above 4 tuple from chelsio port 0 to chelsio port 1.

Note *For more information on additional parameters, refer cxgbtool manual by running the `man cxgbtool` command.*

3.2. Listing Filter Rules

- To list the Hash/DDR filters set, run the following command:

```
[root@host~]# cat /sys/kernel/debug/cxgb4/<bus-id>/hash_filters
```

- To list the both LE-TCAM and Hash/DDR filters set, run the following command:

```
[root@host~]# cxgbtool ethX filter show
```

3.3. Removing Filter Rules

To remove a filter, run the following command with *cap maskless* parameter and corresponding filter rule index:

```
[root@host~]# cxgbtool ethX filter index <delete|clear> cap maskless
```

Note

- *Filter rule index can be determined by referring the "hash_filters" file located in /sys/kernel/debug/cxgb4/<bus-id>/.*
- *For more information on additional parameters, refer cxgbtool manual by running the* `man cxgbtool` *command.*

3.4. Swap MAC feature

Chelsio T5's Swap MAC feature swaps packet source MAC and destination MAC addresses. This is applicable only for switch filter rules. Here's an example:

```
[root@host~]# cxgbtool eth2 filter 1 action switch lip 102.2.2.1 fip
102.2.2.2 lport 5001 fport 14000 hitcnts 1 iport 1 eport 0 swapmac 1 proto
17 cap maskless
Hash-Filter Index = 21936
```

The above example will swap source and destination MAC addresses of UDP packets (matching above 4 tuple) received on adapter port 1 and then switch them to port 0.

Note *This feature is currently supported only with Hash/DDR filters.*

3.5. Hit Counters

For LE-TCAM filters, *hit counters* will work simply by adding *hitcnts 1* parameter to the filter rule. However, for Hash/DDR filters, you will have to make use of tracing feature and RSS queues. Here's a step-by-step guide to enable *hit counters* for Hash/DDR filter rules:

i. Enable tracing on T5 adapter.

```
[root@host~]# cxgbtool ethX reg 0x09800=0x13
```

ii. Setup a trace filter

```
[root@host~]# echo tx1 snaplen=40 > /sys/kernel/debug/cxgb4/<bus_id>/trace0
```

Note *Use "snaplen=60" in case of IPV6.*

iii. Configure RSS Queue to send trace packets. Determine the RspQ ID of the queues by looking at *Trace* QType in */sys/kernel/debug/cxgb4/<bus-id>/sge_qinfo* file

```
[root@host~]# cxgbtool ethX reg 0x0a00c=<Trace Queue0-RspQ ID>
```

The above step will trace all the packets transmitting from port1(tx1) to trace filter 0.

- **Multi-tracing**

To enable *hit counters* for multiple chelsio ports in Tx/Rx direction enable Multi-tracing. Using this we can configure 4 different RSS Queues separately corresponding to 4 trace-filters.

i. Enable Tracing as well as MultiRSSFilter

```
[root@host~]# cxgbtool ethX reg 0x09800=0x33
```

ii. Setup a trace filter

```
[root@host~]# echo tx0 snaplen=40 > /sys/kernel/debug/cxgb4/<bus_id>/trace0
```

iii. Configure the RSS Queue corresponding to trace0 filter configured above. Determine the *RspQ ID* of the queues by looking at *Trace* QType in */sys/kernel/debug/cxgb4/<bus-id>/sge_qinfo* file.

```
[root@host~]# cxgbtool ethX reg 0x09808=<Trace-Queue0-RspQ ID>
```

iv. Similarly for other direction and for multiple ports run the follow commands:

```
[root@host~]# echo rx0 snaplen=40 > /sys/kernel/debug/cxgb4/<bus_id>/trace1
[root@host~]# echo tx1 snaplen=40 > /sys/kernel/debug/cxgb4/<bus_id>/trace2
[root@host~]# echo rx1 snaplen=40 > /sys/kernel/debug/cxgb4/<bus_id>/trace3
[root@host~]# cxgbtool ethX reg 0x09ff4=<Trace-Queue1-RspQ ID>
[root@host~]# cxgbtool ethX reg 0x09ffc=<Trace-Queue2-RspQ ID>
[root@host~]# cxgbtool ethX reg 0x0a004=<Trace-Queue3-RspQ ID>
```

Note *Use “snaplen=60” in case of IPV6.*

IX. Appendix A

1. Troubleshooting

- ***Cannot bring up Chelsio interface***

Make sure you have created the corresponding network-script configuration file as stated in **Cheslsio Unified Wire** chapter (See **Creating network-scripts**). If the file does exist, make sure the structure and contents are correct. A sample is given in the **Chelsio Unified Wire** chapter (See **Configuring network-scripts**). Another reason may be that the IP address mentioned in the configuration file is already in use on the network.

- ***Cannot ping through Chelsio interface***

First, make sure the interface was successfully brought up using ifup ethX (where ethX is your interface) and that it is linked to an IP address, either static or obtained through DHCP.

You then may want to check whether the destination host (i.e. the machine you are trying to ping) is up and running and accepts ICMP requests such as ping. If you get a return value of 0 when doing a `cat /proc/sys/net/ipv4/icmp_echo_ignore_all` on the remote host that means it is configured to reply to incoming pings. Change `ipv4` to `ipv6` in the path if you are using IPv6. Note that this is a Linux-only tip.

If you have more than one interface wired to the network, make sure you are using the right one for your outgoing ping requests. This can be done by using the `-I` option of the ping command, as shown in the following example:

```
[root@host~]# ping -I eth1 10.192.167.1
```

Where 10.192.167.1 is the machine you want to ping.

- ***Configuring firewall for your application***

In many cases the firewall software on the systems may prevent the applications from working properly. Please refer to the appropriate documentation for the Linux distribution on how to configure or disable the firewall.

- ***Configuring Ethernet interfaces on Cisco switch***

Always configure Ethernet interfaces on Cisco switch in trunk mode.

- **Cisco nexus switch reporting “pauseRateLimitErrDisable”**

If in any case the switch-port on the Cisco nexus switch is reporting “pauseRateLimitErrDisable”, then perform an Ethernet port shut/no shut.

- ***Removing inbox driver from initramfs of Ubuntu***

The following example describes the method to remove Chelsio inbox network driver (*cxgb4*) from *initramfs* of Ubuntu. Similarly, other inbox drivers can be removed by using this procedure.

i. Ensure that you are the root user.
ii. Determine if driver entry exists in */usr/share/initramfs-tools/modules*.

```
[root@host~] # cat /usr/share/initramfs-tools/modules | grep -i cxgb4
```

iii. Delete the driver entry, if present.
iv. Change your working directory to kernel directory (*/lib/modules/<kernel>/**) and delete the driver (ko) file.

```
[root@host~]# cd /lib/modules/3.13.0-32-
generic/kernel/drivers/net/ethernet/chelsio/cxgb4
[root@host~]# rm -rf cxgb4.ko
```

v. Update the *initramfs* image.

```
[root@host~]# update-initramfs -u -v -k 3.13.0-32-generic
```

vi. Reboot system for changes to take effect.

2. Chelsio End-User License Agreement (EULA)

Installation and use of the driver/software implies acceptance of the terms in the Chelsio End-User License Agreement (EULA).

IMPORTANT: PLEASE READ THIS SOFTWARE LICENSE CAREFULLY BEFORE DOWNLOADING OR OTHERWISE USING THE SOFTWARE OR ANY ASSOCIATED DOCUMENTATION OR OTHER MATERIALS (COLLECTIVELY, THE "SOFTWARE"). BY CLICKING ON THE "OK" OR "ACCEPT" BUTTON YOU AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, CLICK THE "DO NOT ACCEPT" BUTTON TO TERMINATE THE INSTALLATION PROCESS.

1. License. Chelsio Communications, Inc. ("Chelsio") hereby grants you, the Licensee, and you hereby accept, a limited, non-exclusive, non-transferable license to install and use the Software with one or more Chelsio network adapters on a single server computer for use in communicating with one or more other computers over a network. You may also make one copy of the Software in machine readable form solely for back-up purposes, provided you reproduce Chelsio's copyright notice and any proprietary legends included with the Software or as otherwise required by Chelsio.

2. Restrictions. This license granted hereunder does not constitute a sale of the Software or any copy thereof. Except as expressly permitted under this Agreement, you may not:

(i) reproduce, modify, adapt, translate, rent, lease, loan, resell, distribute, or create derivative works of or based upon, the Software or any part thereof; or

(ii) make available the Software, or any portion thereof, in any form, on the Internet. The Software contains trade secrets and, in order to protect them, you may not decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human-perceivable form. You assume full responsibility for the use of the Software and agree to use the Software legally and responsibly.

3. Ownership of Software. As Licensee, you own only the media upon which the Software is recorded or fixed, but Chelsio retains all right, title and interest in and to the Software and all subsequent copies of the Software, regardless of the form or media in or on which the Software may be embedded.

4. Confidentiality. You agree to maintain the Software in confidence and not to disclose the Software, or any information or materials related thereto, to any third party without the express written consent of Chelsio. You further agree to take all reasonable precautions to limit access of the Software only to those of your employees who reasonably require such access to perform their employment obligations and who are bound by confidentiality agreements with you.

5. Term. This license is effective in perpetuity, unless terminated earlier. You may terminate the license at any time by destroying the Software (including the related documentation), together with all copies or modifications in any form. Chelsio may terminate this license, and this license shall be deemed to have automatically terminated, if you fail to comply with any term or condition of this Agreement. Upon any termination, including termination by you, you must destroy the Software (including the related documentation), together with all copies or modifications in any form.

6. Limited Warranty. If Chelsio furnishes the Software to you on media, Chelsio warrants only that the media upon which the Software is furnished will be free from defects in material or workmanship under normal use and service for a period of thirty (30) days from the date of delivery to you.

CHELSIO DOES NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE OR ANY PART THEREOF. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, CHELSIO MAKES NO OTHER WARRANTIES, EXPRESS OR IMPLIED, AND HEREBY DISCLAIMS ALL OTHER WARRANTIES, INCLUDING, BUT NOT LIMITED TO, NON-INFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow the exclusion of implied warranties or limitations on how long an implied warranty may last, so the above limitations may not apply to you. This warranty gives you specific legal rights and you may also have other rights which vary from state to state.

7. Remedy for Breach of Warranty. The sole and exclusive liability of Chelsio and its distributors, and your sole and exclusive remedy, for a breach of the above warranty, shall be the replacement of any media furnished by Chelsio not meeting the above limited warranty and which is returned to Chelsio. If Chelsio or its distributor is unable to deliver replacement media which is free from defects in materials or workmanship, you may terminate this Agreement by returning the Software.

8. Limitation of Liability. IN NO EVENT SHALL CHELSIO HAVE ANY LIABILITY TO YOU OR ANY THIRD PARTY FOR ANY INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, HOWEVER CAUSED, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE LICENSE OR USE OF THE SOFTWARE, INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR LOSS OF ANTICIPATED PROFITS, EVEN IF CHELSIO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL CHELSIO'S LIABILITY ARISING OUT OF OR RELATED TO THE LICENSE OR USE OF THE SOFTWARE EXCEED THE AMOUNTS PAID BY YOU FOR THE LICENSE GRANTED HEREUNDER. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY.

9. High Risk Activities. The Software is not fault-tolerant and is not designed, manufactured or intended for use or resale as online equipment control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software could lead directly to death, personal injury, or severe physical or environmental damage. Chelsio specifically disclaims any express or implied warranty of fitness for any high risk uses listed above.

10. Export. You acknowledge that the Software is of U.S. origin and subject to U.S. export jurisdiction. You acknowledge that the laws and regulations of the United States and other countries may restrict the export and re-export of the Software. You agree that you will not export or re-export the Software or documentation in any form in violation of applicable United States and foreign law. You agree to comply with all applicable international and national laws that apply to the Software, including the U.S.

Export Administration Regulations, as well as end-user, end-use, and destination restrictions issued by U.S. and other governments.

11. Government Restricted Rights. The Software is subject to restricted rights as follows. If the Software is acquired under the terms of a GSA contract: use, reproduction or disclosure is subject to the restrictions set forth in the applicable ADP Schedule contract. If the Software is acquired under the terms of a DoD or civilian agency contract, use, duplication or disclosure by the Government is subject

to the restrictions of this Agreement in accordance with 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors and 49 C.F.R. 227.7202-1 of the DoD FAR Supplement and its successors.

12. General. You acknowledge that you have read this Agreement, understand it, and that by using the Software you agree to be bound by its terms and conditions. You further agree that it is the complete and exclusive statement of the agreement between Chelsio and you, and supersedes any proposal or prior agreement, oral or written, and any other communication between Chelsio and you relating to the subject matter of this Agreement. No additional or any different terms will be enforceable against Chelsio unless Chelsio gives its express consent, including an express waiver of the terms of this Agreement, in writing signed by an officer of Chelsio. This Agreement shall be governed by California law, except as to copyright matters, which are covered by Federal law. You hereby irrevocably submit to the personal jurisdiction of, and irrevocably waive objection to the laying of venue (including a waiver of any argument of forum non conveniens or other principles of like effect) in, the state and federal courts located in Santa Clara County, California, for the purposes of any litigation undertaken in connection with this Agreement. Should any provision of this Agreement be declared unenforceable in any jurisdiction, then such provision shall be deemed severable from this Agreement and shall not affect the remainder hereof. All rights in the Software not specifically granted in this Agreement are reserved by Chelsio. You may not assign or transfer this Agreement (by merger, operation of law or in any other manner) without the prior written consent of Chelsio and any attempt to do so without such consent shall be void and shall constitute a material breach of this Agreement.

Should you have any questions concerning this Agreement, you may contact Chelsio by writing to:

Chelsio Communications, Inc.
209 North Fair Oaks Avenue,
Sunnyvale, CA 94085
U.S.A