NOTE:
THIS HARDWARE, SOFTWARE OR TEST SUITE PRODUCT ("PRODUCT(S)") AND ITS RELATED DOCUMENTATION ARE PROVIDED BY MELLANOX TECHNOLOGIES "AS IS" WITH ALL FAULTS OF ANY KIND AND SOLELY FOR THE PURPOSE OF AIDING THE CUSTOMER IN TESTING APPLICATIONS THAT USE THE PRODUCTS IN DESIGNATED SOLUTIONS. THE CUSTOMER'S MANUFACTURING TEST ENVIRONMENT HAS NOT MET THE STANDARDS SET BY MELLANOX TECHNOLOGIES TO FULLY QUALIFY THE PRODUCT(S) AND/OR THE SYSTEM USING IT. THEREFORE, MELLANOX TECHNOLOGIES CANNOT AND DOES NOT GUARANTEE OR WARRANT THAT THE PRODUCTS WILL OPERATE WITH THE HIGHEST QUALITY. ANY EXPRESS OR IMPLIED WARRANTIES INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT ARE DISCLAIMED. IN NO EVENT SHALL MELLANOX BE LIABLE TO CUSTOMER OR ANY THIRD PARTIES FOR ANY DIRECT, INDIRECT, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES OF ANY KIND (INCLUDING, BUT NOT LIMITED TO, PAYMENT FOR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY FROM THE USE OF THE PRODUCT(S) AND RELATED DOCUMENTATION EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
# Table of Contents

**Table of Contents** ................................. 3  
**List of Tables** ................................. 6  

**Chapter 1  Overview** .............................. 14  
  1.1 MLNX_EN Package Contents .................. 15  
      1.1.1 Package Images ......................... 15  
      1.1.2 Software Components .................. 15  
      1.1.3 Firmware .............................. 16  
      1.1.4 Directory Structure .................. 16  
      1.1.5 mlx4 VPI Driver ....................... 16  
      1.1.6 mlx5 Driver ........................... 16  
      1.1.7 Unsupported Features in MLNX_EN ...... 17  
  1.2 Module Parameters ............................ 17  
      1.2.1 mlx4 Module Parameters ................. 17  
      1.2.1.1 mlx4_core Parameters .............. 18  
      1.2.1.2 mlx4_en Parameters ................. 20  
      1.2.2 mlx5_core Module Parameters .......... 20  

**Chapter 2  Installation** .......................... 21  
  2.1 Software Dependencies ...................... 21  
  2.2 Downloading MLNX_EN ....................... 21  
  2.3 Installing MLNX_EN .......................... 22  
      2.3.1 Installation Modes .................... 22  
      2.3.2 Installation Procedure ................ 23  
  2.4 Installing MLNX_EN with Innova™ IPsec Adapter Cards ............. 24  
  2.5 Unloading MLNX_EN .......................... 24  
  2.6 Uninstalling MLNX_EN ....................... 24  
  2.7 Recompiling MLNX_EN ......................... 24  
  2.8 Installing MLNX_EN Using YUM .............. 25  
      2.8.1 Setting Up MLNX_EN YUM Repository ... 26  
      2.8.2 Installing MLNX_EN Using YUM Tool ... 27  
  2.9 Installing MLNX_EN Using apt-get ........... 28  
      2.9.1 Setting Up MLNX_EN apt-get Repository 28  
      2.9.2 Installing MLNX_EN Using apt-get Tool ... 29  
  2.10 Updating Firmware After Installation .......... 29  
      2.10.1 Updating the Device Online .......... 29  
      2.10.2 Updating Firmware and FPGA Image on Innova IPsec Cards ... 31  
      2.10.3 Manually Updating the Device .......... 31  
  2.11 Ethernet Driver Usage and Configuration ........... 32
Chapter 3 Features Overview and Configuration

3.1 Ethernet Network

3.1.1 ethtool Counters

3.1.1.1 Counter Groups

3.1.1.2 Counter Types

3.1.1.3 Acceleration Mechanism

3.1.2 Quality of Service (QoS)

3.1.2.1 Mapping Traffic to Traffic Classes

3.1.2.2 Plain Ethernet Quality of Service Mapping

3.1.2.3 Map Priorities with set_egress_map

3.1.2.4 Quality of Service Properties

3.1.2.5 Quality of Service Tools

3.1.2.6 Packet Pacing

3.1.3 Quantized Congestion Notification (QCN)

3.1.3.1 QCN Tool - mlnx_qcn

3.1.3.2 Setting QCN Configuration

3.1.4 Ethtool

3.1.5 Checksum Offload

3.1.6 Ignore Frame Check Sequence (FCS) Errors

3.1.7 RDMA over Converged Ethernet (RoCE)

3.1.8 Flow Control

3.1.8.1 Priority Flow Control (PFC)

3.1.8.2 Dropless Receive Queue (RQ)

3.1.9 Explicit Congestion Notification (ECN)

3.1.9.1 ConnectX-3/ConnectX-3 Pro ECN

3.1.9.2 ConnectX-4 ECN

3.1.10 RSS Support

3.1.10.1 RSS Hash Function

3.1.11 Time-Stamping

3.1.11.1 Time-Stamping Service

3.1.11.2 One Pulse Per Second (1PPS)

3.1.12 Flow Steering

3.1.12.1 Enable/Disable Flow Steering

3.1.12.2 Flow Steering Support

3.1.12.3 A0 Static Device Managed Flow Steering

3.1.12.4 Flow Domains and Priorities

3.1.12.5 Flow Steering Dump Tool

3.1.13 Wake-on-LAN (WoL)

3.1.14 Hardware Accelerated 802.1ad VLAN (Q-in-Q Tunneling)

3.1.15 Ethernet Performance Counters
3.1.16 NVM Express over Fabrics (NVME-oF) ........................................ 80
  3.1.16.1 NVME-oF .................................................. 80
  3.1.16.2 NVME-oF Target Offload .................................. 80

3.2 Virtualization ................................................................. 81
  3.2.1 Single Root IO Virtualization (SR-IOV) .................................. 81
    3.2.1.1 System Requirements ........................................ 81
    3.2.1.2 Setting Up SR-IOV ........................................... 81
    3.2.1.3 Additional SR-IOV Configurations .......................... 88
    3.2.1.4 Uninstalling SR-IOV Driver ................................. 105
  3.2.2 Enabling Para Virtualization ........................................ 106
  3.2.3 VXLAN Hardware Stateless Offloads .................................. 107
    3.2.3.1 Enabling VXLAN Hardware Stateless Offloads for ConnectX-3 Pro ... 108
    3.2.3.2 Enabling VXLAN Hardware Stateless Offloads for ConnectX®-4 Family Devices 108
    3.2.3.3 Important Notes ............................................. 109
  3.2.4 Q-in-Q Encapsulation per VF in Linux (VST) .......................... 109

3.3 Resiliency ................................................................. 113
  3.3.1 Reset Flow ........................................................ 113
    3.3.1.1 Kernel ULPs ............................................... 113
    3.3.1.2 SR-IOV ...................................................... 113
    3.3.1.3 Forcing the VF to Reset .................................. 113
    3.3.1.4 Advanced Error Reporting (AER) in ConnectX-3 and ConnectX-3 Pro ... 114
    3.3.1.5 Extended Error Handling (EEH) .......................... 114
    3.3.1.6 CRDUMP .................................................. 114
  3.4 Fast Driver Unload ..................................................... 115

Chapter 4 Troubleshooting .................................................. 116
  4.1 General Issues ......................................................... 116
  4.2 Ethernet Related Issues ............................................. 117
  4.3 Performance Related Issues ........................................ 118
  4.4 SR-IOV Related Issues .............................................. 119
**List of Tables**

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Document Revision History</td>
<td>7</td>
</tr>
<tr>
<td>Table 2</td>
<td>Common Abbreviations and Acronyms</td>
<td>11</td>
</tr>
<tr>
<td>Table 3</td>
<td>Glossary</td>
<td>12</td>
</tr>
<tr>
<td>Table 4</td>
<td>Related Documentation</td>
<td>13</td>
</tr>
<tr>
<td>Table 5</td>
<td>MLNX_EN Software Components</td>
<td>15</td>
</tr>
<tr>
<td>Table 6</td>
<td>Required Action After Firmware Upgrade</td>
<td>23</td>
</tr>
<tr>
<td>Table 7</td>
<td>ethtool Supported Options</td>
<td>49</td>
</tr>
<tr>
<td>Table 8</td>
<td>Ethernet Performance Counters</td>
<td>77</td>
</tr>
<tr>
<td>Table 9</td>
<td>General Issues</td>
<td>116</td>
</tr>
<tr>
<td>Table 10</td>
<td>Ethernet Related Issues</td>
<td>117</td>
</tr>
<tr>
<td>Table 11</td>
<td>Performance Related Issues</td>
<td>118</td>
</tr>
<tr>
<td>Table 12</td>
<td>SR-IOV Related Issues</td>
<td>119</td>
</tr>
</tbody>
</table>
# Document Revision History

<table>
<thead>
<tr>
<th>Release</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>March 8, 2018</td>
<td>• Updated Section 3.3.1, “Reset Flow”, on page 113</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Removed section 3.4.1.3.8.5 Virtual MAC</td>
</tr>
<tr>
<td>4.2</td>
<td>November 15, 2017</td>
<td>• Added the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• “Configuring Q-in-Q Encapsulation per Virtual Function for ConnectX-5”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 2.4, “Installing MLNX_EN with Innova™ IPsec Adapter Cards”, on page 24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 2.10.2, “Updating Firmware and FPGA Image on Innova IPsec Cards”, on page 31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.7.9, “Force DSCP”, on page 91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.8.1.4, “PFC Storm Prevention”, on page 59</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.6.2, “Configuring VGT+ for ConnectX-4/ConnectX-5”, on page 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.4.1.3.8.5, “Virtual MAC”, on page 262</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.3.1.6, “CRDUMP”, on page 114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.8.2, “Dropless Receive Queue (RQ)”, on page 60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.2.4.4, “Trust State”, on page 38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.2.4.5, “Receive Buffer”, on page 38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.2.4.6, “DCBX Control Mode”, on page 38</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.4, “Q-in-Q Encapsulation per VF in Linux (VST)”, on page 109</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.2.1, “VLAN Guest Tagging (VGT) and VLAN Switch Tagging (VST)”, on page 89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.6, “Virtual Guest Tagging (VGT+)”, on page 99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.2, “Ethernet Virtual Function Configuration when Running SR-IOV”, on page 89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.2.3, “Virtual Function Statistics”, on page 90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.8.2, “Rate Limit and Bandwidth Share Per VF”, on page 103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.2.4, “Quality of Service Properties”, on page 37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.2.5.1, “mlnx_qos”, on page 39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.2.5.2, “Get Current Configuration”, on page 41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 4.1, “General Issues”, on page 116</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 1.2.1.1, “mlx4_core Parameters”, on page 18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 1.2.2, “mlx5_core Module Parameters”, on page 20</td>
</tr>
</tbody>
</table>
Table 1 - Document Revision History

<table>
<thead>
<tr>
<th>Release</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>July 9, 2017</td>
<td>• Added the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.11.2, “One Pulse Per Second (1PPS)”, on page 70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.12.5, “Flow Steering Dump Tool”, on page 75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.16.2, “NVME-of Target Offload”, on page 80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.8.4, “Probed VFs”, on page 104</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.4, “Fast Driver Unload”, on page 115</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.16, “NVM Express over Fabrics (NVME-oF)”, on page 80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.10, “RSS Support”, on page 64</td>
</tr>
<tr>
<td>4.0</td>
<td>April 10, 2017</td>
<td>• Updated the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.8.1, “SR-IOV MAC Anti-Spoofing”, on page 102</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.4, “Q-in-Q Encapsulation per VF in Linux (VST)”, on page 109</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.6, “Virtual Guest Tagging (VGT+)”, on page 99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.9.1, “XOR RSS Hash Function”, on page 56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Table 5, “Supported Uplinks to Servers,” on page 22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.3.8.2, “Rate Limit and Bandwidth Share Per VF”, on page 103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 4.2, “Ethernet Related Issues”, on page 117</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.2.1.2.2, “Configuring SR-IOV for ConnectX-4/ConnectX-5 (Ethernet)”, on page 88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Section 3.1.11.1.4, “Steering PTP Traffic to Single RX Ring”, on page 69</td>
</tr>
</tbody>
</table>
## Table 1 - Document Revision History

<table>
<thead>
<tr>
<th>Release</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
</table>
| 3.40    | November 29, 2016  | • Added the following section:<br>  
  • Section 1.1.7, “Unsupported Features in MLNX_EN”, on page 17                              |
| October 26, 2016 | Added the following sections:<br>  
  • Section 2.8, “Installing MLNX_EN Using YUM”, on page 25<br>  
  • Section 3.2.1.3.8.2, “Rate Limit and Bandwidth Share Per VF”, on page 103<br>  
  • Section 3.2.4, “Q-in-Q Encapsulation per VF in Linux (VST)”, on page 109<br>  
  • Updated the following sections:<br>  
  • Section 1.1, “MLNX_EN Package Contents”, on page 15<br>  
  • Section 2, “Installation”, on page 21<br>  
  • Section 3.1.1, “ethtool Counters”, on page 34<br>  
  • Section 3.1.8.1.3, “Priority Counters”, on page 58<br>  
  • Section 3.1.2.3, “Map Priorities with set_egress_map”, on page 37<br>  
  • Section 3.1.2.5, “Quality of Service Tools”, on page 39 |
| 3.30    | June 13, 2016      | • Added the following sections:<br>  
  • Section 3.2.1.3.8, “SR-IOV Advanced Security Features”, on page 102<br>  
  • Section 3.2.1.3.9, “VF Promiscuous Rx Modes”, on page 105<br>  
  • Section 3.1.2.6, “Packet Pacing”, on page 44<br>  
  • Section 3.1.8.1, “Priority Flow Control (PFC)”, on page 56<br>  
  • Accelerated Receive Flow Steering (aRFS)<br>  
  • Updated the following sections:<br>  
  • Section 3.1.12, “Flow Steering”, on page 70 |
| 3.20    | February 11, 2016  | • Added the following sections:<br>  
  • Section 3.1.8.1.1, “PFC Local Configuration”, on page 56<br>  
  • Section 3.2.3.2, “Enabling VXLAN Hardware Stateless Offloads for ConnectX®-4 Family Devices”, on page 108<br>  
  • Updated the following sections:<br>  
  • Section 3.1.4, “Ethtool”, on page 49: Added the “ethtool -p|--identify DEVNAME” parameter<br>  
  • Section 3.2.3, “VXLAN Hardware Stateless Offloads”, on page 107 |
### Table 1 - Document Revision History

<table>
<thead>
<tr>
<th>Release</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1-1.0.4</td>
<td>October 08, 2015</td>
<td>- Added the following new sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.13, “Wake-on-LAN (WoL)”, on page 76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.14, “Hardware Accelerated 802.1ad VLAN (Q-in-Q Tunneling)”, on page 76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.9.2, “ConnectX-4 ECN”, on page 63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Updated the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.4, “Ethtool”, on page 49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.12.1, “Enable/Disable Flow Steering”, on page 71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.12.3, “A0 Static Device Managed Flow Steering”, on page 72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.2.1.3.2.2, “Additional Ethernet VF Configuration Options”, on page 90</td>
</tr>
<tr>
<td>3.0-1.0.1</td>
<td>June 21, 2015</td>
<td>- Added the following new sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 1.1.5, “mlx4 VPI Driver”, on page 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 1.1.6, “mlx5 Driver”, on page 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 1.2.2, “mlx5_core Module Parameters”, on page 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.6, “Ignore Frame Check Sequence (FCS) Errors”, on page 54</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Updated the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 1.1.2, “Software Components”, on page 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 2.3.1, “Installation Modes”, on page 22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 2.3.2, “Installation Procedure”, on page 23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 2.10.1, “Updating the Device Online”, on page 29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 2.10.3, “Manually Updating the Device”, on page 31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 2.11, “Ethernet Driver Usage and Configuration”, on page 32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.4, “Ethtool”, on page 49</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 3.1.15, “Ethernet Performance Counters”, on page 77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Removed the following sections:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Power Management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Adaptive Interrupt Moderation Algorithm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Virtual Guest Tagging (VGT+)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Installing MLNX_EN on XenServer6.1</td>
</tr>
</tbody>
</table>
About this Manual

This preface provides general information concerning the scope and organization of this User’s Manual.

Intended Audience

This manual is intended for system administrators responsible for the installation, configuration, management and maintenance of the software and hardware of VPI (InfiniBand, Ethernet) adapter cards. It is also intended for application developers.

Common Abbreviations and Acronyms

Table 2 - Common Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation / Acronym</th>
<th>Whole Word / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>(Capital) ‘B’ is used to indicate size in bytes or multiples of bytes (e.g., 1KB = 1024 bytes, and 1MB = 1048576 bytes)</td>
</tr>
<tr>
<td>b</td>
<td>(Small) ‘b’ is used to indicate size in bits or multiples of bits (e.g., 1Kb = 1024 bits)</td>
</tr>
<tr>
<td>FW</td>
<td>Firmware</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware</td>
</tr>
<tr>
<td>LSB</td>
<td>Least significant byte</td>
</tr>
<tr>
<td>lsb</td>
<td>Least significant bit</td>
</tr>
<tr>
<td>MSB</td>
<td>Most significant byte</td>
</tr>
<tr>
<td>msb</td>
<td>Most significant bit</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>PFC</td>
<td>Priority Flow Control</td>
</tr>
<tr>
<td>PR</td>
<td>Path Record</td>
</tr>
<tr>
<td>SL</td>
<td>Service Level</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>ULP</td>
<td>Upper Level Protocol</td>
</tr>
<tr>
<td>VL</td>
<td>Virtual Lane</td>
</tr>
</tbody>
</table>
Glossary

The following is a list of concepts and terms related to InfiniBand in general and to Subnet Managers in particular. It is included here for ease of reference, but the main reference remains the InfiniBand Architecture Specification.

Table 3 - Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Band</td>
<td>A term assigned to administration activities traversing the IB connectivity only.</td>
</tr>
<tr>
<td>Local Identifier (ID)</td>
<td>An address assigned to a port (data sink or source point) by the Subnet Manager, unique within the subnet, used for directing packets within the subnet.</td>
</tr>
<tr>
<td>Local Port</td>
<td>The IB port of the HCA through which IBDIAG tools connect to the IB fabric.</td>
</tr>
<tr>
<td>Master Subnet Manager</td>
<td>The Subnet Manager that is authoritative, that has the reference configuration information for the subnet. See Subnet Manager.</td>
</tr>
<tr>
<td>Multicast Forwarding Tables</td>
<td>A table that exists in every switch providing the list of ports to forward received multicast packet. The table is organized by MLID.</td>
</tr>
<tr>
<td>Network Interface Card (NIC)</td>
<td>A network adapter card that plugs into the PCI Express slot and provides one or more ports to an Ethernet network.</td>
</tr>
<tr>
<td>Unicast Linear Forwarding Tables (LFT)</td>
<td>A table that exists in every switch providing the port through which packets should be sent to each LID.</td>
</tr>
</tbody>
</table>
### Related Documentation

**Table 4 - Related Documentation**

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firmware Release Notes for Mellanox adapter devices</td>
<td>See the Release Notes PDF file relevant to your adapter device under <code>docs/</code> folder of installed package.</td>
</tr>
</tbody>
</table>

### Support and Updates Webpage

Please visit [http://www.mellanox.com](http://www.mellanox.com) > Products > InfiniBand/VPI Drivers > Linux SW/Drivers for downloads, FAQ, troubleshooting, future updates to this manual, etc.
1 Overview

This document provides information about MLNX_EN Linux driver, and instructions on how to install the driver on Mellanox ConnectX® network adapter solutions supporting the following uplinks to servers:

<table>
<thead>
<tr>
<th>Uplink/HCAs</th>
<th>Driver Name</th>
<th>Uplink Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectX®-3/</td>
<td>mlx4</td>
<td>• Ethernet: 10GigE, 40GigE and 56GigE(^a)</td>
</tr>
<tr>
<td>ConnectX®-3 Pro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConnectX®-4</td>
<td>mlx5</td>
<td>• Ethernet: 1GigE, 10GigE, 25GigE, 40GigE, 50GigE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>56GigE(^l), and 100GigE</td>
</tr>
<tr>
<td>ConnectX®-4 Lx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ConnectX®-5</td>
<td></td>
<td>• Ethernet: 1GigE, 10GigE, 25GigE, 40GigE, 50GigE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and 100GigE</td>
</tr>
<tr>
<td>ConnectX®-5 Ex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Innova™ IPsec EN</td>
<td></td>
<td>• Ethernet: 10GigE, 40GigE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. 56 GbE is a Mellanox proprietary link speed and can be achieved while connecting a Mellanox adapter cards to Mellanox SX10XX switch series or connecting a Mellanox adapter card to another Mellanox adapter card.

The MLNX_EN driver release exposes the following capabilities:

- Single/Dual port
- Up to 16 Rx queues per port
- 16 Tx queues per port
- Rx steering mode: Receive Core Affinity (RCA)
- MSI-X or INTx
- Adaptive interrupt moderation
- HW Tx/Rx checksum calculation
- Large Send Offload (i.e., TCP Segmentation Offload)
- Large Receive Offload
- Multi-core NAPI support
- VLAN Tx/Rx acceleration (HW VLAN stripping/insertion)
- Ethtool support
- Net device statistics
- SR-IOV support
- Flow steering
- Ethernet Time Stamping
1.1 MLNX_EN Package Contents

1.1.1 Package Images
MLNX_EN is provided as an ISO image or as a tarball per Linux distribution and CPU architecture that includes source code and binary RPMs, firmware and utilities. The ISO image contains an installation script (called `install`) that performs the necessary steps to accomplish the following:

- Discover the currently installed kernel
- Uninstall any previously installed MLNX_OFED/MLNX_EN packages
- Install the MLNX_EN binary RPMs (if they are available for the current kernel)
- Identify the currently installed HCAs and perform the required firmware updates

1.1.2 Software Components
MLNX_EN contains the following software components:

<table>
<thead>
<tr>
<th>Components</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mlx5 driver</td>
<td>mlx5 is the low level driver implementation for the ConnectX®-4 adapters designed by Mellanox Technologies. ConnectX®-4 operates as a VPI adapter.</td>
</tr>
<tr>
<td>mlx5_core</td>
<td>Acts as a library of common functions (e.g. initializing the device after reset) required by the ConnectX®-4 adapter cards.</td>
</tr>
<tr>
<td>mlx4 driver</td>
<td>mlx4 is the low level driver implementation for the ConnectX adapters designed by Mellanox Technologies. The ConnectX can operate as an InfiniBand adapter and as an Ethernet NIC. To accommodate the two flavors, the driver is split into modules: mlx4_core, mlx4_en, and mlx4_ib. Note: mlx4_ib is not part of this package.</td>
</tr>
<tr>
<td>mlx4_core</td>
<td>Handles low-level functions like device initialization and firmware commands processing. Also controls resource allocation so that the InfiniBand, Ethernet and FC functions can share a device without interfering with each other.</td>
</tr>
<tr>
<td>mlx4_en</td>
<td>Handles Ethernet specific functions and plugs into the netdev mid-layer.</td>
</tr>
<tr>
<td>mstflint</td>
<td>An application to burn a firmware binary image.</td>
</tr>
<tr>
<td>Software modules</td>
<td>Source code for all software modules (for use under conditions mentioned in the modules' LICENSE files)</td>
</tr>
</tbody>
</table>
1.1.3 **Firmware**

The image includes the following firmware item:

- Firmware images (.bin format wrapped in the mlxfwmanager tool) for ConnectX®-2/ConnectX®-3/ConnectX®-3 Pro/ConnectX®-4 and ConnectX®-4 Lx network adapters

1.1.4 **Directory Structure**

The tarball image of MLNX_EN contains the following files and directories:

- install - the MLNX_EN installation script
- uninstall.sh - the MLNX_EN un-installation script
- RPMS/ - directory of binary RPMs for a specific CPU architecture
- src/ - directory of the OFED source tarball
- mlx4_add_kernel_support.sh - a script required to rebuild MLNX_EN for customized kernel version on supported Linux distribution

1.1.5 **mlx4 VPI Driver**

**mlx4** is the low level driver implementation for the ConnectX® family adapters designed by Mellanox Technologies. ConnectX®-3 adapters can operate as an Ethernet NIC. To accommodate the supported configurations, the driver is split into the following modules:

**mlx4_core**

Handles low-level functions like device initialization and firmware commands processing. Also controls resource allocation so that the Ethernet functions can share the device without interfering with each other.

**mlx4_en**

A 10/40GigE driver under drivers/net/ethernet/mellanox/mlx4 that handles Ethernet specific functions and plugs into the netdev mid-layer.

1.1.6 **mlx5 Driver**

**mlx5** is the low level driver implementation for the ConnectX®-4 adapters designed by Mellanox Technologies. ConnectX®-4 operates as a VPI adapter. The mlx5 driver is comprised of the following kernel modules:

**mlx5_core**

Acts as a library of common functions (e.g. initializing the device after reset) required by ConnectX®-4 adapter cards. mlx5_core driver also implements the Ethernet interfaces for ConnectX®-4. Unlike mlx4_en/core, mlx5 drivers does not require the mlx5_en module as the Ethernet functionalities are built-in in the mlx5_core module.
1.1.7 Unsupported Features in MLNX_EN

- InfiniBand protocol
- Remote Direct Memory Access (RDMA)
- Storage protocols that use RDMA, such as:
  - iSCSI Extensions for RDMA (iSER)
  - SCSI RDMA Protocol (SRP)
  - Sockets Direct Protocol (SDP)

1.2 Module Parameters

1.2.1 mlx4 Module Parameters

In order to set mlx4 parameters, add the following line(s) to /etc/modprobe.d/mlx4.conf:

```bash
options mlx4_core parameter=<value>
```

and/or

```bash
options mlx4_en parameter=<value>
```

The following sections list the available mlx4 parameters.

---

1. Only supported with -dpdk and -vma options.
### 1.2.1.1 mlx4_core Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_4k_mtu</td>
<td>(Obsolete) attempt to set 4K MTU to all ConnectX ports (int)</td>
</tr>
<tr>
<td>debug_level</td>
<td>Enable debug tracing if &gt; 0 (int)</td>
</tr>
<tr>
<td>msi_x</td>
<td>0 - don't use MSI-X, 1 - use MSI-X, &gt;1 - limit number of MSI-X irqs to msi_x (non-SRIOV only) (int)</td>
</tr>
<tr>
<td>enable_sys_tune</td>
<td>Tune the cpu's for better performance (default 0) (int)</td>
</tr>
<tr>
<td>block_loopback</td>
<td>Block multicast loopback packets if &gt; 0 (default: 1) (int)</td>
</tr>
<tr>
<td>num_vfs</td>
<td>Either a single value (e.g. '5') to define uniform num_vfs value for all devices functions or a string to map device function numbers to their num_vfs values (e.g. '0000:04:00.0-5,002b:1c:0b.a-15'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for num_vfs value (e.g. 15). (string)</td>
</tr>
<tr>
<td>probe_vf</td>
<td>Either a single value (e.g. '3') to indicate that the Hypervisor driver itself should activate this number of VFs for each HCA on the host, or a string to map device function numbers to their probe_vf values (e.g. '0000:04:00.0-3,002b:1c:0b.a-13'). Hexadecimal digits for the device function (e.g. 002b:1c:0b.a) and decimal for probe_vf value (e.g. 13). (string)</td>
</tr>
<tr>
<td>log_num_mgm_entry_size</td>
<td>log mgm size, that defines the num of qp per mcg, for example: 10 gives 248.range: 7 &lt;= log_num_mgm_entry_size &lt;= 12. To activate device managed flow steering when available, set to -1 (int)</td>
</tr>
<tr>
<td>high_rate_steer</td>
<td>Enable steering mode for higher packet rate (obsolete, set &quot;Enable optimized steering&quot; option in log_num_mgm_entry_size to use this mode). (int)</td>
</tr>
<tr>
<td>fast_drop</td>
<td>Enable fast packet drop when no receive WQEs are posted (int)</td>
</tr>
<tr>
<td>enable_64b_cqe_eqe</td>
<td>Enable 64 byte CQEs/EQEs when the FW supports this if non-zero (default: 1) (int)</td>
</tr>
<tr>
<td>log_num_mac</td>
<td>Log2 max number of MACs per ETH port (1-7) (int)</td>
</tr>
<tr>
<td>log_num_vlan</td>
<td>(Obsolete) Log2 max number of VLANs per ETH port (0-7) (int)</td>
</tr>
<tr>
<td>log_mtt_per_seg</td>
<td>Log2 number of MTT entries per segment (0-7) (default: 0) (int)</td>
</tr>
<tr>
<td>port_type_array</td>
<td>Either pair of values (e.g. '1,2') to define uniform port1/port2 types configuration for all devices functions or a string to map device function numbers to their pair of port types values (e.g. '0000:04:00.0-1;2,002b:1c:0b.a-1;1'). Valid port types: 1-ib, 2-eth, 3-auto, 4-N/A</td>
</tr>
<tr>
<td>log_num_qp</td>
<td>log maximum number of QPs per HCA (default: 19) (int)</td>
</tr>
<tr>
<td>log_num_sq</td>
<td>log maximum number of SRQs per HCA (default: 16) (int)</td>
</tr>
<tr>
<td>log_rdmarc_per_qp</td>
<td>log number of RDMARC buffers per QP (default: 4) (int)</td>
</tr>
<tr>
<td>log_num_cq</td>
<td>log maximum number of CQs per HCA (default: 16) (int)</td>
</tr>
<tr>
<td>log_num_mcg</td>
<td>log maximum number of multicast groups per HCA (default: 13) (int)</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>log_num_mpt</td>
<td>log maximum number of memory protection table entries per HCA (default: 19) (int)</td>
</tr>
<tr>
<td>log_num_mtt</td>
<td>log maximum number of memory translation table segments per HCA (default: max(20, 2*MTTs for register all of the host memory limited to 30)) (int)</td>
</tr>
<tr>
<td>enable_qos</td>
<td>Enable Quality of Service support in the HCA (default: off)</td>
</tr>
<tr>
<td>internal_err_reset</td>
<td>Reset device on internal errors if non-zero (default is 1)</td>
</tr>
<tr>
<td>ingress_parser_mode</td>
<td>Mode of ingress parser for ConnectX3-Pro. 0 - standard. 1 - checksum for non TCP/UDP. (default: standard)</td>
</tr>
<tr>
<td>roce_mode</td>
<td>Set RoCE modes supported by the port</td>
</tr>
<tr>
<td>ud_gid_type</td>
<td>Set gid type for UD QPs</td>
</tr>
<tr>
<td>log_num_mgm_entry_size</td>
<td>log mgm size, that defines the num of qp per mcg, for example: 10 gives 248. range: 7 &lt;= log_num_mgm_entry_size &lt;= 12 (default = -10).</td>
</tr>
<tr>
<td>use_prio</td>
<td>Enable steering by VLAN priority on ETH ports (deprecated)</td>
</tr>
<tr>
<td>enable_vfs_qos</td>
<td>Enable Virtual VFs QoS (default: off)</td>
</tr>
<tr>
<td>mlx4_en_only_mode</td>
<td>Load in Ethernet only mode</td>
</tr>
<tr>
<td>enable_4k_uar</td>
<td>Enable using 4K UAR. Should not be enabled if have VFs which do not support 4K UARs (default: true)</td>
</tr>
<tr>
<td>mlx4_en_only_mode</td>
<td>Load in Ethernet only mode</td>
</tr>
<tr>
<td>rr_proto</td>
<td>IP next protocol for RoCEv1.5 or destination port for RoCEv2. Setting 0 means using driver default values (deprecated)</td>
</tr>
</tbody>
</table>
### 1.2.1.2 mlx4_en Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inline_thold:</td>
<td>Threshold for using inline data (int) Default and max value is 104 bytes. Saves PCI read operation transaction, packet less then threshold size will be copied to hw buffer directly. (range: 17-104)</td>
</tr>
<tr>
<td>udp_rss:</td>
<td>Enable RSS for incoming UDP traffic (uint) On by default. Once disabled no RSS for incoming UDP traffic will be done.</td>
</tr>
<tr>
<td>pfctx:</td>
<td>Priority based Flow Control policy on TX[7:0]. Per priority bit mask (uint)</td>
</tr>
<tr>
<td>pfcrx:</td>
<td>Priority based Flow Control policy on RX[7:0]. Per priority bit mask (uint)</td>
</tr>
<tr>
<td>udev_dev_port_dev_id:</td>
<td>Work with dev_id or dev_port when supported by the kernel. Range: 0 &lt;= udev_dev_port_dev_id &lt;= 2 (default = 0).</td>
</tr>
<tr>
<td>prof_sel</td>
<td>The parameter name for selecting the profile. The supported values for profiles are: 0 - for medium resources, medium performance 1 - for low resources 2 - for high performance (int) (default)</td>
</tr>
</tbody>
</table>

### 1.2.2 mlx5_core Module Parameters

The mlx5_core module supports a single parameter used to select the profile which defines the number of resources supported.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prof_sel</td>
<td>The parameter name for selecting the profile. The supported values for profiles are: 0 - for medium resources, medium performance 1 - for low resources 2 - for high performance (int) (default)</td>
</tr>
<tr>
<td>charp</td>
<td></td>
</tr>
<tr>
<td>node_guid</td>
<td>guid configuration. This module parameter will be obsolete!</td>
</tr>
<tr>
<td>debug_mask</td>
<td>debug_mask: 1 = dump cmd data, 2 = dump cmd exec time, 3 = both. Default=0 (uint)</td>
</tr>
<tr>
<td>probe_vf</td>
<td>probe VFs or not, 0 = not probe, 1 = probe. Default = 1 (bool)</td>
</tr>
</tbody>
</table>
2 Installation

This chapter describes how to install and test the MLNX_EN for Linux package on a single host machine with Mellanox InfiniBand and/or Ethernet adapter hardware installed.

2.1 Software Dependencies

- MLNX_EN driver cannot coexist with OFED software on the same machine. Hence when installing MLNX_EN all OFED packages should be removed (run the install script).

2.2 Downloading MLNX_EN

Step 1. Verify that the system has a Mellanox network adapter (HCA/NIC) installed.

The following example shows a system with an installed Mellanox HCA:

```bash
# lspci -v | grep Mellanox
86:00.0 Network controller [0207]: Mellanox Technologies MT27620 Family
  Subsystem: Mellanox Technologies Device 0014
86:00.1 Network controller [0207]: Mellanox Technologies MT27620 Family
  Subsystem: Mellanox Technologies Device 0014
```

Note: For ConnectX-5 Socket Direct adapters, use `ibdev2netdev` to display the installed card and the mapping of logical ports to physical ports. Example:

```bash
[root@gen-l-vrt-203 ~]# ibdev2netdev -v | grep -i MCX556M-ECAT-S25
0000:84:00.0 mlx5_10 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228
  port 1 (DOWN ) ==> p2p1 (Down)
0000:84:00.1 mlx5_11 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228
  port 1 (DOWN ) ==> p2p2 (Down)
0000:05:00.0 mlx5_2 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228
  port 1 (DOWN ) ==> p5p1 (Down)
0000:05:00.1 mlx5_3 (MT4119 - MCX556M-ECAT-S25SN) CX556M - ConnectX-5 QSFP28 fw 16.22.0228
  port 1 (DOWN ) ==> p5p2 (Down)
```

Note that each PCI card of ConnectX-5 Socket Direct has a different PCI address. In the output example above, the first two rows indicate that one card is installed in a PCI slot with PCI Bus address 84 (hexadecimal), and PCI Device Number 00, and PCI Function Number 0 and 1. RoCE assigned mlx5_10 as the logical port, which is the same as netdevice p2p1, and both are mapped to physical port of PCI function 0000:84:00.0.

Note also that RoCE logical port mlx5_2 of the second PCI card (PCI Bus address 05) and netdevice p5p1 are mapped to physical port of PCI function 0000:05:00.0, which is the same physical port of PCI function 0000:84:00.0.

MT4119 is the PCI Device ID of the Mellanox ConnectX-5 adapters family.


Step 2. Download the ISO or tarball image to your host.
Step 3. Use the md5sum utility to confirm the file integrity of your ISO/tarball image.

2.3 Installing MLNX_EN

The installation script, install, performs the following:

• Discovers the currently installed kernel
• Uninstall any previously installed MLNX_OFED/MLNX_EN packages
• Installs the MLNX_EN binary (if they are available for the current kernel)
• Identifies the currently installed Ethernet network adapters and automatically upgrades the firmware

2.3.1 Installation Modes

mlnx_en installer supports 2 modes of installation. The install script selects the mode of driver installation depending on the running OS/kernel version.

• Kernel Module Packaging (KMP) mode, where the source rpm is rebuilt for each installed flavor of the kernel. This mode is used for RedHat and SUSE distributions.
• Non KMP installation mode, where the sources are rebuilt with the running kernel. This mode is used for vanilla kernels.
• By default, the package will install drivers supporting Ethernet only. In addition, the package will include the following new installation options:
  • Full VMA support which can be installed using the installation option “--vma”.
  • Infrastructure to run DPDK using the installation option “--dpdk”.

Notes:

• DPDK itself is not included in the package. Users would still need to install DPDK separately after the MLNX_EN installation is completed.
• Installing VMA or DPDK infrastructure will allow users to run RoCE.

Installation Results

• For Ethernet only installation mode:
  • The kernel modules are installed under:
    • /lib/modules/`uname -r`/updates on SLES and Fedora Distributions
    • /lib/modules/`uname -r`/extra/mlnx-en on RHEL and other RedHat like Distributions
    • /lib/modules/`uname -r`/updates/dkms/ on Ubuntu
  • The kernel module sources are placed under:
    /usr/src/mlnx-en-<ver>/
  • For VPI installation mode:
2.3.2 Installation Procedure

Step 1. Log in to the installation machine as root.

Step 2.  Mount the ISO image on your machine.

```bash
host1# mount -o ro,loop mlnx-en-<ver>-<OS label>-<CPU arch>.iso /mnt
```

Step 3. Run the installation script.

```bash
/mnt/install
```

Step 4. **Case A:** If the installation script has performed a firmware update on your network adapter, you need to either restart the driver or reboot your system before the firmware update can take effect. Refer to the table below to find the appropriate action for your specific card.

**Table 6 - Required Action After Firmware Upgrade**

<table>
<thead>
<tr>
<th>Action \ Adapter</th>
<th>Driver Restart</th>
<th>Standard Reboot (Soft Reset)</th>
<th>Cold Reboot (Hard Reset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectX-3/ConnectX-3 Pro</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Standard ConnectX-4/ConnectX-4 Lx or higher</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Adapters with Multi-Host Support</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Socket Direct Cards</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

**Case B:** If the installation script has not performed firmware upgrade on your network adapter, restart the driver by running:

- `# /etc/init.d/mlnx-en.d restart` - after Ethernet only installation mode
- `# /etc/init.d/openibd restart` - after VPI installation mode

The “/etc/init.d/openibd” service script will load the mlx4 and/or mlx5 and ULP drivers as set in the “/etc/infiniband/openib.conf” configuration file.

The result is a new net-device appearing in the ‘ifconfig -a’ output.
2.4 Installing MLNX_EN with Innova™ IPsec Adapter Cards

Support for Innova IPsec adapter cards is at beta level.

This type of installation is applicable to RedHat 7.1, 7.2, 7.3 and 7.4 operating systems and Kernel 4.13.

As of version 4.2, MLNX_EN supports Mellanox Innova IPsec 4 Lx EN adapter card that provides security acceleration for IPsec-enabled networks.

For information on the usage of Innova IPsec, please refer to Mellanox Innova IPsec 4 Lx EN Adapter Card documentation on Mellanox official web (mellanox.com --> Products --> Adapters --> Smart Adapters --> Innova IPsec).

Prerequisites

In order to obtain Innova IPsec offload capabilities once MLNX_EN is installed, make sure Kernel v4.13 or newer is installed with the following configuration flags enabled:

- CONFIG_XFRM_OFFLOAD
- CONFIG_INET_ESP_OFFLOAD
- CONFIG_INET6_ESP_OFFLOAD

2.5 Unloading MLNX_EN

➢ To unload the Ethernet driver:

After Ethernet only installation mode:

```bash
#!/etc/init.d/mlnx-en.d stop
Unloading NIC driver: [ OK ]
```

After VPI installation mode:

```bash
#!/etc/init.d/openibd stop
Unloading HCA driver: [ OK ]
```

2.6 Uninstalling MLNX_EN

Use the script `/usr/sbin/ofed_uninstall.sh` to uninstall the Mellanox OFED package.

2.7 Recompiling MLNX_EN

➢ To recompile the driver:

After Ethernet only installation mode:

Step 1. Enter the source directory.

```bash
cp -a /usr/src/mlnx-en-3.0 /tmp
cd /tmp/mlnx-en-3.0
```
Installation

Step 2. Apply kernel backport patch.

```bash
#> scripts/mlnx_en_patch.sh
```

Step 3. Compile the driver sources.

```bash
#> make
```

Step 4. Install the driver kernel modules.

```bash
#> make install
```

After VPI installation mode:

Step 1. Enter the source directory.

```bash
# cp -a /usr/src/ofa_kernel-3.4/ /tmp/ofa_kernel-3.4
# cd /tmp/ofa_kernel-3.4
```

Step 2. Run configure script to define which modules to build and to apply kernel backport patches.

1. Get configure options used to build drivers shipped with the package:

```bash
# /etc/infiniband/info
prefix=/usr
Kernel=3.10.0-229.el7.x86_64
```

Configure options:

```bash
--with-core-mod --with-user_mad-mod --with-user_access-mod --with-addr_trans-mod -
--with-mthca-mod --with-mlx4-mod --with-mlx4_en-mod --with-mlx4_vnic-mod --with-
mlx5-mod --with-cxgb3-mod --with-cxgb4-mod --with-nes-mod --with-ehca-mod --
with-qib-mod --with-ipoib-mod --with-ipath_inf-mod --with-amso1100-mod --with-
ocrdma-mod --with-sdp-mod --with-srp-mod --with-rds-mod --with-iser-mod --with-e_i-
opiib-mod --with-nfsrdma-mod --with-9pnet_rdma-mod --with-9p-mod --with-cxgb3i-
mod --with-cxgb4i-mod --with-isert-mod
```

2. Run configure with options from "Configure options:" line:

```bash
# ./configure -j --with-core-mod --with-user_mad-mod --with-user_access-mod --
with-addr_trans-mod -mod --with-mlx4_en-mod --with-mlx4_vnic-mod --with-mlx5-
mod --with-cxgb3-mod --with-cxgb4-mod --with-nes-mod --with-ehca-mod --with-
qib-mod --with-ipoib-mod --with-ipath_inf-mod --with-amso1100-mod --with-
ocrdma-mod --with-sdp-mod --with-srp-mod --with-rds-mod --with-iser-mod --with-
e_ipoib-mod --with-nfsrdma-mod --with-9pnet_rdma-mod --with-9p-mod --with-
cxgb3i-mod --with-cxgb4i-mod --with-isert-mod
```

Step 3. Compile the driver sources.

```bash
# make -j16
```

Step 4. Install the driver kernel modules.

```bash
# make install_kernel
```

2.8 Installing MLNX_EN Using YUM

This type of installation is applicable to RedHat/OL, Fedora, and XenServer operating systems.
2.8.1 Setting Up MLNX_EN YUM Repository

The package consists of two folders that can be set up as a repository:

- “RPMS_ETH” - provides the Ethernet only installation mode.
- “RPMS” - provides the RDMA support installation mode.

Step 1. Log into the installation machine as root.

Step 2. Mount the ISO image on your machine and copy its content to a shared location in your network.

```
# mount -o ro,loop mlnx-en-<ver>-<OS label>-<CPU arch>.iso /mnt
```

You can download the image from www.mellanox.com --> Products --> Software --> Ethernet Drivers.

Step 3. Download and install Mellanox Technologies GPG-KEY.
The key can be downloaded using the following link: http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox.

Example:

```
# wget http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox
Resolving www.mellanox.com... 72.3.194.0
Connecting to www.mellanox.com|72.3.194.0|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1354 (1.3K) [text/plain]
Saving to: ?RPM-GPG-KEY-Mellanox?
100%[=================================================>] 1,354 --.-K/s in 0s
```

Step 4. Install the key.

Example:

```
# sudo rpm --import RPM-GPG-KEY-Mellanox
warning: rpmts_HdrFromFdno: Header V3 DSA/SHA1 Signature, key ID 6224c050: NOKEY
Importing GPG key 0x6224C050:
Userid: "Mellanox Technologies (Mellanox Technologies - Signing Key v2) <support@mellanox.com>"
From : /repos/MLNX_EN/RPM-GPG-KEY-Mellanox
Is this ok [y/N]:
```

Step 5. Verify that the key was imported successfully.

Example:

```
# rpm -q gpg-pubkey --qf '%{NAME}-%{VERSION}-%{RELEASE}	%{SUMMARY}
' | grep Mellanox gpg-pubkey-a9e4b643-520791ba gpg(Mellanox Technologies <support@mellanox.com>)
Rev 3.30
Mellanox Technologies 45
```

Step 6. Create a YUM repository configuration file called “/etc/yum.repos.d/mlnx_en.repo” with the following content:

```
[mlnx_en]
name=MLNX_EN Repository
```
Step 7. Verify that the repository was added successfully.

```bash
# yum repolist
Loaded plugins: product-id, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-manager to register.
repo id repo name status
mlnx_en MLNX_EN Repository
```

### 2.8.2 Installing MLNX_EN Using YUM Tool

After setting up the YUM repository for MLNX_EN package, install one of the following meta-data packages:

- In case you set up the “RPMS_ETH” folder as the repository (for Ethernet only mode), install:
  ```bash
  # yum install mlnx-en-eth-only
  ```
- In case you set up the “RPMS” folder as the repository (for RDMA mode), install either:
  ```bash
  # yum install mlnx-en-vma
  ```
  Or
  ```bash
  # yum install mlnx-en-dpdk
  ```

Please note the following:

MLNX_EN provides kernel module RPM packages with KMP support for RHEL and SLES. For other operating systems, kernel module RPM packages are provided only for the operating system’s default kernel. In this case, the group RPM packages have the supported kernel version in their packages name.

If you have an operating systems different than RHEL or SLES, or you have installed a kernel that is not supported by default in MLNX_EN, you can use the `mlnx_add_kernel_support.sh` script to build MLNX_EN for your kernel.

The script will automatically build the matching group RPM packages for your kernel so that you can still install MLNX_EN via YUM.

Please note that the resulting MLNX_EN repository will contain unsigned RPMs. Therefore, you should set `gpgcheck=0` in the repository configuration file.

Installing MLNX_EN using the YUM tool does not automatically update the firmware.
To update the firmware to the version included in MLNX_EN package, you can either:

1. Run:

```
# yum install mlnx-fw-updater
```

or

2. Update the firmware to the latest version available on Mellanox website as described in Section 2.10, “Updating Firmware After Installation”, on page 29.

### 2.9 Installing MLNX_EN Using apt-get

This type of installation is applicable to Debian and Ubuntu operating systems.

#### 2.9.1 Setting Up MLNX_EN apt-get Repository

The package consists of two folders that can be set up as a repository:

- “DEBS_ETH” - provides the Ethernet only installation mode.
- “RPMS” - provides the RDMA support installation mode.

**Step 1.** Log into the installation machine as root.

**Step 2.** Extract the MLNX_EN package on a shared location in your network.

You can download the package from www.mellanox.com --> Products --> Software --> Ethernet Drivers.

**Step 3.** Create an apt-get repository configuration file called “/etc/apt/sources.list.d/mlnx_en.list” with the following content:

```
deb file:/<path to extracted MLNX_EN package>/<DEBS FOLDER NAME> ./
```

Replace `<DEBS FOLDER NAME>` with “DEBS_ETH” or “DEBS” depending on the desired installation mode (Ethernet only or RDMA).

**Step 4.** Download and Install Mellanox Technologies GPG-KEY.

**Example:**

```
# wget -qO - http://www.mellanox.com/downloads/ofed/RPM-GPG-KEY-Mellanox | sudo apt-key add -
```

**Step 5.** Verify that the key was imported successfully.

**Example:**

```
# apt-key list
pub 1024D/A9E4B643 2013-08-11
uid Mellanox Technologies <support@mellanox.com>
sub 1024g/09FCC269 2013-08-11
```

**Step 6.** Update the apt-get cache.

```
# sudo apt-get update
```
2.9.2 Installing MLNX_EN Using apt-get Tool

After setting up the apt-get repository for MLNX_EN package, install one of the following metadata packages:

- In case you set up the “DEBS_ETH” folder as the repository (for Ethernet only mode), install:

  ```bash
  # apt-get install minx-en-eth-only
  ```

- In case you set up the “DEBS” folder as the repository (for RDMA mode), install either:

  ```bash
  # apt-get install minx-en-vma
  ```

  Or

  ```bash
  # apt-get install minx-en-dpdk
  ```

Installing MLNX_EN using the apt-get tool does not automatically update the firmware. To update the firmware to the version included in MLNX_EN package, you can either:

1. Run:

   ```bash
   # apt-get install minx-fw-updater
   ```

   or

2. Update the firmware to the latest version available on Mellanox website as described in Section 2.10, “Updating Firmware After Installation”, on page 29.

2.10 Updating Firmware After Installation

The firmware can be updated in one of the following methods.

2.10.1 Updating the Device Online

To update the device online on the machine from Mellanox site, use the following command line:

```bash
mlxfwmanager --online -u -d <device>
```
Example:

mlxfwmanager --online -u -d 0000:09:00.0
Querying Mellanox devices firmware ...

Device #1:
--------
Device Type: ConnectX3
Part Number: MCX354A-FCA_A2-A4
Description: ConnectX-3 VPI adapter card; dual-port QSFP; FDR IB (56Gb/s) and 40GigE; PCIe3.0 x8 8GT/s; RoHS R6
PSID: MT_1020120019
PCI Device Name: 0000:09:00.0
Port1 GUID: 0002c9000100d051
Port2 MAC: 0002c9000002
Versions: Current Available
FW 2.33.5000 2.34.5000
Status: Update required
--------

Found 1 device(s) requiring firmware update. Please use -u flag to perform the update.
2.10.2 Updating Firmware and FPGA Image on Innova IPsec Cards

The firmware and FPGA update package (mlnx-fw-updater) is installed under “/opt/mellanox/mlnx-fw-updater” folder.


You can run the following update script using one of the modes below:

```
/opt/mellanox/mlnx-fw-updater/mlnx_fpga_updater.sh
```

1. With `-u` flag to provide URL to the software package (tarball). Example:

   ```
   ```

2. With `-t` flag to provide the path to the downloaded tarball. Example:

   ```
   ./mlnx_fpga_updater.sh -t <Innova_IPsec_bundle_file.tgz>
   ```

3. With `-p` flag to provide the path to the downloaded and extracted tarball. Example:

   ```
   ./mlnx_fpga_updater.sh -p <Innova_IPsec_extracted_bundle_directory>
   ```

For more information on the script usage, you can run `mlnx_fpga_updater.sh -h`.

- It is recommended to perform firmware and FPGA upgrade on Innova IPsec cards using this script only.

2.10.3 Manually Updating the Device

In case you ran the `install` script with the ‘--without-fw-update’ option or you are using an OEM card and now you wish to (manually) update firmware on your adapter card(s), you need to perform the steps below. The following steps are also appropriate in case you wish to burn newer firmware that you have downloaded from Mellanox Technologies’ Web site (http://www.mellanox.com > Support > Firmware Download).

**Step 1.** Get the device’s PSID.

```
mlxfwmanager_pci | grep PSID
PSID: MT_1210110019
```

**Step 2.** Download the firmware BIN file from the Mellanox website or the OEM website.

**Step 3.** Burn the firmware.

```
mlxfwmanager_pci -i <fw_file.bin>
```

**Step 4.** Reboot your machine after the firmware burning is completed.
2.11 Ethernet Driver Usage and Configuration

➢ To assign an IP address to the interface:

```bash
#> ifconfig eth<x> <ip>
```
1. 'x' is the OS assigned interface number

➢ To check driver and device information:

```bash
#> ethtool -i eth<x>
```

Example:

```bash
#> ethtool -i eth2
driver: mlx4_en
version: 2.1.8 (Oct 06 2013)
firmware-version: 2.30.3110
bus-info: 0000:1a:00.0
```

➢ To query stateless offload status:

```bash
#> ethtool -k eth<x>
```

➢ To set stateless offload status:

```bash
#> ethtool -K eth<x> [rx on|off] [tx on|off] [sg on|off] [tso on|off] [lro on|off]
```

➢ To query interrupt coalescing settings:

```bash
#> ethtool -c eth<x>
```

➢ To enable/disable adaptive interrupt moderation:

```bash
#> ethtool -C eth<x> adaptive-rx on|off
```
By default, the driver uses adaptive interrupt moderation for the receive path, which adjusts the moderation time to the traffic pattern.

➢ To set the values for packet rate limits and for moderation time high and low:

```bash
#> ethtool -C eth<x> [pkt-rate-low N] [pkt-rate-high N] [rx-usecs-low N] [rx-usecs-high N]
```
Above an upper limit of packet rate, adaptive moderation will set the moderation time to its highest value. Below a lower limit of packet rate, the moderation time will be set to its lowest value.

➢ To set interrupt coalescing settings when adaptive moderation is disabled:

```bash
#> ethtool -C eth<x> [rx-usecs N] [rx-frames N]
```

usec settings correspond to the time to wait after the *last* packet is sent/received before triggering an interrupt.

➢ [ConnectX-3/ConnectX-3 Pro] To query pause frame settings:

```bash
#> ethtool -a eth<x>
```

➢ [ConnectX-3/ConnectX-3 Pro] To set pause frame settings:

```bash
#> ethtool -A eth<x> [rx on|off] [tx on|off]
```

➢ To query ring size values:

```bash
#> ethtool -g eth<x>
```
To modify rings size:

```
#> ethtool -G eth<x> [rx <N>] [tx <N>]
```

To obtain additional device statistics:

```
#> ethtool -S eth<x>
```

[ConnectX-3/ConnectX-3 Pro] To perform a self diagnostics test:

```
#> ethtool -t eth<x>
```

The driver defaults to the following parameters:

- Both ports are activated (i.e., a net device is created for each port)
- The number of Rx rings for each port is the nearest power of 2 of number of cpu cores, limited by 16.
- LRO is enabled with 32 concurrent sessions per Rx ring

Some of these values can be changed using module parameters, which can be displayed by running:

```
#> modinfo mlx4_en
```

To set non-default values to module parameters, add to the `/etc/modprobe.conf` file:

```
"options mlx4_en <param_name>=<value> <param_name>=<value> ...
```

Values of all parameters can be observed in `/sys/module/mlx4_en/parameters/`.

### 2.12 Performance Tuning

For further information on Linux performance, please refer to the Performance Tuning Guide for Mellanox Network Adapters.
3 Features Overview and Configuration

3.1 Ethernet Network

3.1.1 ethtool Counters

The ethtool counters are counted in different places, according to which they are divided into groups. Each counters group may also have different counter types.

3.1.1.1 Counter Groups

- Ring counters - software ring counters
- Software port counters - an aggregation of software ring counters
- vPort counters - traffic counters and drops due to steering or no buffers. May indicate NIC issues. These counters include Ethernet traffic counters (including Raw Ethernet) and RDMA/RoCE traffic counters
- Physical port counters - the physical port connecting the NIC to the network. May indicate NIC, link or network issues. This measuring point holds information on standardized counters like IEEE 802.3, RFC2863, RFC 2819, RFC 3635, and additional counters such as flow control, FEC, and more. Physical port counters are not exposed to virtual machines
- Priority port counters - a set of the physical port counters, per priory per port
- PCIe counters - counts physical layer PCIe signal integrity errors
3.1.1.2 Counter Types

Counters are divided into three types:

- **Traffic Informative Counters** - counters that count traffic. These counters can be used for loading estimation for general debugging.

- **Traffic Acceleration Counters** - counters that count traffic accelerated by hardware. These counters form an additional layer to the informative counter set, and the same traffic is counted through both informative and acceleration counters. Acceleration counters are marked with “[A]”

- **Error Counters** - an increment of these counters might indicate a problem. Each of these counters has an explanation and correction action.

Statistic can be fetched via:

- `ip link`:
  
  ```
  ip -s link show <if-name>
  ```

- `ethtool` (provides more detailed information):
  
  ```
  ethtool -S <if-name>
  ```

3.1.1.3 Acceleration Mechanism

The following acceleration mechanisms have dedicated counters:

- **TSO (TCP Segmentation Offload)** - increases outbound throughput and reduces CPU utilization by allowing the kernel to buffer multiple packets in a single large buffer. The NIC splits the buffer into packets and transmits it.

- **LRO (Large Receive Offload)** - increases inbound throughput and reduces CPU utilization by aggregating multiple incoming packets of a single stream in a single buffer.

- **CHECKSUM (Checksum)** - calculates TCP checksum (by the NIC). The following CSUM offloads are available (refer to `skbuff.h` for more detailed explanation):
  
  - CHECKSUM_UNNECESSARY
  - CHECKSUM_NONE - No CSUM acceleration was used
  - CHECKSUM_COMPLETE - Device provided CSUM on the entire packet
  - CHECKSUM_PARTIAL - Device provided CSUM

- **CQE Compress** - compresses Completion Queue Events (CQE) used for sparing bandwidth on PCIe and achieves better performance.

For further details on the counters, refer to the [Understanding mlx5 driver ethtool Counters](https://community.post) Community post.

3.1.2 Quality of Service (QoS)

Quality of Service (QoS) is a mechanism of assigning a priority to a network flow (socket, rdma_cm connection) and manage its guarantees, limitations and its priority over other flows.
This is accomplished by mapping the user's priority to a hardware TC (traffic class) through a 2/3 stage process. The TC is assigned with the QoS attributes and the different flows behave accordingly.

3.1.2.1 Mapping Traffic to Traffic Classes

Mapping traffic to TCs consists of several actions which are user controllable, some controlled by the application itself and others by the system/network administrators.

The following is the general mapping traffic to Traffic Classes flow:
1. The application sets the required Type of Service (ToS).
2. The ToS is translated into a Socket Priority (sk_prio).
3. The sk_prio is mapped to a User Priority (UP) by the system administrator (some applications set sk_prio directly).
4. The UP is mapped to TC by the network/system administrator.
5. TCs hold the actual QoS parameters

QoS can be applied on the following types of traffic. However, the general QoS flow may vary among them:
- **Plain Ethernet** - Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver
- **RoCE** - Applications use the RDMA API to transmit using QPs
- **Raw Ethernet QP** - Application use VERBs API to transmit using a Raw Ethernet QP

3.1.2.2 Plain Ethernet Quality of Service Mapping

Applications use regular inet sockets and the traffic passes via the kernel Ethernet driver.

The following is the Plain Ethernet QoS mapping flow:
1. The application sets the ToS of the socket using setsockopt (IP_TOS, value).
2. ToS is translated into the sk_prio using a fixed translation:

<table>
<thead>
<tr>
<th>TOS</th>
<th>sk_prio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
</tr>
</tbody>
</table>

3. The Socket Priority is mapped to the UP:
   - If the underlying device is a VLAN device, egress_map is used controlled by the vconfig command. This is per VLAN mapping.
   - If the underlying device is not a VLAN device, in ConnectX-3/ConnectX-3 Pro RoCE old kernels, mapping the sk_prio to UP is done by using tc_wrap.py -i <dev name> -u 0,1,2,3,4,5,6,7. Otherwise, the mapping is done in the driver.
4. The UP is mapped to the TC as configured by the `mlnx_qos` tool or by the `lldpad` daemon if DCBX is used.

```plaintext
Socket applications can use `setsockopt(SK_PRIO, value)` to directly set the `sk_prio` of the socket. In this case the ToS to `sk_prio` fixed mapping is not needed. This allows the application and the administrator to utilize more than the 4 values possible via ToS.
```

```plaintext
In case of VLAN interface, the UP obtained according to the above mapping is also used in the VLAN tag of the traffic
```

### 3.1.2.3 Map Priorities with `set_egress_map`

For RoCE old kernels that do not support `set_egress_map`, use the `tc_wrap` script to map between `sk_prio` and UP. Use `tc_wrap` with option `-u`. For example:

```plaintext
tc_wrap -i <ethX> -u <skprio2up mapping>
```

### 3.1.2.4 Quality of Service Properties

The different QoS properties that can be assigned to a TC are:

- **Strict Priority**
- **Enhanced Transmission Selection (ETS)**
- **Rate Limit**
- **Trust State**
- **Receive Buffer**
- **DCBX Control Mode**

#### 3.1.2.4.1 Strict Priority

When setting a TC's transmission algorithm to be 'strict', then this TC has absolute (strict) priority over other TC strict priorities coming before it (as determined by the TC number: TC 7 is highest priority, TC 0 is lowest). It also has an absolute priority over non strict TCs (ETS).

This property needs to be used with care, as it may easily cause starvation of other TCs.

A higher strict priority TC is always given the first chance to transmit. Only if the highest strict priority TC has nothing more to transmit, will the next highest TC be considered.

Non strict priority TCs will be considered last to transmit.

This property is extremely useful for low latency low bandwidth traffic that needs to get immediate service when it exists, but is not of high volume to starve other transmitters in the system.

#### 3.1.2.4.2 Enhanced Transmission Selection (ETS)

Enhanced Transmission Selection standard (ETS) exploits the time periods in which the offered load of a particular Traffic Class (TC) is less than its minimum allocated bandwidth by allowing the difference to be available to other traffic classes.
After servicing the strict priority TCs, the amount of bandwidth (BW) left on the wire may be split among other TCs according to a minimal guarantee policy.

If, for instance, TC0 is set to 80% guarantee and TC1 to 20% (the TCs sum must be 100), then the BW left after servicing all strict priority TCs will be split according to this ratio.

Since this is a minimal guarantee, there is no maximum enforcement. This means, in the same example, that if TC1 did not use its share of 20%, the reminder will be used by TC0.

ETS is configured using the mlnx_qos tool (mlnx_qos) which allows you to:

- Assign a transmission algorithm to each TC (strict or ETS)
- Set minimal BW guarantee to ETS TCs

Usage:

```bash
mlnx_qos -i [options]
```

### 3.1.2.4.3 Rate Limit

Rate limit defines a maximum bandwidth allowed for a TC. Please note that 10% deviation from the requested values is considered acceptable.

### 3.1.2.4.4 Trust State

Trust state enables prioritizing sent/received packets based on packet fields.

The default trust state is PCP. Ethernet packets are prioritized based on the value of the field (PCP/DSCP).

For further information on how to configure Trust mode, please refer to [HowTo Configure Trust State on Mellanox Adapters](community-post).

### 3.1.2.4.5 Receive Buffer

By default, the receive buffer configuration is controlled automatically. Users can override the receive buffer size and receive buffer's xon and xoff thresholds using mlnx_qos tool.

For further information, please refer to [HowTo Tune the Receive buffers on Mellanox Adapters](community-post).

### 3.1.2.4.6 DCBX Control Mode

DCBX settings, such as “ETS” and “strict priority” can be controlled by firmware or software. When DCBX is controlled by firmware, changes of QoS settings cannot be done by the software. The DCBX control mode is configured using the `mlnx_qos -d os-fw` command.

For further information on how to configure the DCBX control mode, please refer to `mlnx_qos` Community post.
3.1.2.5 Quality of Service Tools

3.1.2.5.1 mlnx_qos

mlnx_qos is a centralized tool used to configure QoS features of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.

The mlnx_qos tool enables the administrator of the system to:

- Inspect the current QoS mappings and configuration
  The tool will also display maps configured by TC and vconfig set_egress_map tools, in order to give a centralized view of all QoS mappings.
- Set UP to TC mapping
- Assign a transmission algorithm to each TC (strict or ETS)
- Set minimal BW guarantee to ETS TCs
- Set rate limit to TCs
- Set DCBX control mode
- Set cable length
- Set trust state

For unlimited ratelimit set the ratelimit to 0.
Features Overview and Configuration

Usage

mlnx_qos -i <interface> [options]

Options

- --version
  Show program's version number and exit
- -h, --help
  Show this help message and exit
- -f LIST, --pfc=LIST
  Set priority flow control for each priority. LIST is comma separated value for each priority starting from 0 to 7. Example: 0,0,0,1,1,1 enables PFC on TC4-7
- -p LIST, --prio_tc=LIST
  Maps UPs to TCs. LIST is 8 comma separated TC numbers. Example: 0,0,0,0,1,1,1 maps UPs 0-3 to TC0, and UPs 4-7 to TC1
- -s LIST, --tsa=LIST
  Transmission algorithm for each TC. LIST is comma separated algorithm names for each TC. Possible algorithms: strict, ets, vendor. Example: vendor,strict,ets,ets,ets,ets,ets,ets sets TC0 to vendor, TC1 to strict, TC2-7 to ets.
- -t LIST, --tcbw=LIST
  Set minimal guaranteed %BW for ETS TCs. LIST is comma separated percents for each TC. Values set to TCs that are not configured to ETS algorithm are ignored, but must be present. Example: if TC0,TC2 are set to ETS, then 10,0,90,0,0,0,0,0 will set TC0 to 10% and TC2 to 90%. Percents must sum to 100.
- -r LIST, --ratelimit=LIST
  Rate limit for TCs (in Gbps). LIST is comma separated Gbps limit for each TC. Example: 1,8,8 will limit TC0 to 1Gbps, and TC1,TC2 to 8 Gbps each.
- -d DCBX, --dcbx=DCBX
  Set dcbx mode to firmware controlled(fw) or OS controlled(os). Note, when in OS mode, mlnx_qos should not be used in parallel with other dcbx tools such as lldptool
- --trust=TRUST
  Set priority trust state to pcp or dscp
- --dscp2prio=DSCP2PRIO
  Set/del a (dscp,prio) mapping. Example 'set,30,2' maps dscp 30 to priority 2. 'del,30,2' resets the dscp 30 mapping back to the default setting priority 0.
- --cable_len=CABLE_LEN
  Set cable_len for buffer's xoff and xon thresholds
- -i INTF, --interface=INTF
  Interface name
- -a
  Show all interface's TCs
3.1.2.5.2 Get Current Configuration

ofed_scripts/utils/mlnx_qos -i ens1f0
DCBX mode: OS controlled
Priority trust state: dscp
dscp2prio mapping:
  prio:0 dscp:07,06,05,04,03,02,01,00,
  prio:1 dscp:15,14,13,12,11,10,09,08,
  prio:2 dscp:23,22,21,20,19,18,17,16,
  prio:3 dscp:31,30,29,28,27,26,25,24,
  prio:4 dscp:39,38,37,36,35,34,33,32,
  prio:5 dscp:47,46,45,44,43,42,41,40,
  prio:6 dscp:55,54,53,52,51,50,49,48,
  prio:7 dscp:63,62,61,60,59,58,57,56,
Cable len: 7
PFC configuration:
  priority 0 1 2 3 4 5 6 7
  enabled 0 0 0 0 0 0 0 0
  tc: 0 ratelimit: unlimited, tsa: vendor
      priority: 1
  tc: 1 ratelimit: unlimited, tsa: vendor
      priority: 0
  tc: 2 ratelimit: unlimited, tsa: vendor
      priority: 2
  tc: 3 ratelimit: unlimited, tsa: vendor
      priority: 3
  tc: 4 ratelimit: unlimited, tsa: vendor
      priority: 4
  tc: 5 ratelimit: unlimited, tsa: vendor
      priority: 5
  tc: 6 ratelimit: unlimited, tsa: vendor
      priority: 6
  tc: 7 ratelimit: unlimited, tsa: vendor
      priority: 7
3.1.2.5.3 Set ratelimit. 3Gbps for tc0 4Gbps for tc1 and 2Gbps for tc2

```bash
# mlxq -i <interface> -p 0,1,2 -r 3,4,2
tc: 0 ratelimit: 3 Gbps, tsa: strict
  up: 0
    skprio: 0
    skprio: 1
    skprio: 2 (tos: 8)
    skprio: 3
    skprio: 4 (tos: 24)
    skprio: 5
    skprio: 6 (tos: 16)
    skprio: 7
    skprio: 8
    skprio: 9
    skprio: 10
    skprio: 11
    skprio: 12
    skprio: 13
    skprio: 14
    skprio: 15
  up: 3
  up: 4
  up: 5
  up: 6
  up: 7
tc: 1 ratelimit: 4 Gbps, tsa: strict
  up: 1
tc: 2 ratelimit: 2 Gbps, tsa: strict
  up: 2
```

3.1.2.5.4 Configure QoS. map UP 0,7 to tc0, 1,2,3 to tc1 and 4,5,6 to tc 2. set tc0,tc1 as ets and tc2 as strict. divide ets 30% for tc0 and 70% for tc1

```bash
# mlxq -i <interface> -a ets,ets,strict -p 0,1,1,1,2,2,2 -t 30,70
tc: 0 ratelimit: 3 Gbps, tsa: ets, bw: 30%
  up: 0
    skprio: 0
    skprio: 1
    skprio: 2 (tos: 8)
    skprio: 3
    skprio: 4 (tos: 24)
    skprio: 5
    skprio: 6 (tos: 16)
    skprio: 7
    skprio: 8
    skprio: 9
    skprio: 10
    skprio: 11
    skprio: 12
    skprio: 13
```
3.1.2.5.5 tc and tc_wrap.py

The tc tool is used to create 8 Traffic Classes (TCs).
The tool will either use the sysfs (/sys/class/net/<ethX>/qos/tc_num) or the tc tool to create the TCs.
In case of RoCE ConnectX-3/ConnectX-3 Pro old kernels, the tc_wrap will enable mapping between sk_prio and UP using the sysfs (/sys/class/infiniband/mlx4_0/ports/<port_num>/skprio2up).

Usage

```
tc_wrap.py -i <interface> [options]
```

Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--version</td>
<td>show program's version number and exit</td>
</tr>
<tr>
<td>-h, --help</td>
<td>show this help message and exit</td>
</tr>
<tr>
<td>-u SKPrio_UP, --skprio_up=SKPrio_UP</td>
<td>maps sk_prio to priority for RoCE. LIST is &lt;=16 comma separated priority. index of element is sk_prio.</td>
</tr>
<tr>
<td>-i INTF, --interface=INTF</td>
<td>Interface name</td>
</tr>
</tbody>
</table>

Example

Run:

```
tc_wrap.py -i enp139s0
```
Output:

```
Traffic classes are set to 8

UP 0 skprio: 0 (vlan 5)
UP 1 skprio: 1 (vlan 5)
UP 2 skprio: 2 (vlan 5 tos: 8)
UP 3 skprio: 3 (vlan 5)
UP 4 skprio: 4 (vlan 5 tos: 24)
UP 5 skprio: 5 (vlan 5)
UP 6 skprio: 6 (vlan 5 tos: 16)
UP 7 skprio: 7 (vlan 5)
```

### 3.1.2.5.6 Additional Tools

- tc tool compiled with the `sch_mqprio` module is required to support kernel v2.6.32 or higher. This is a part of `iproute2` package v2.6.32-19 or higher. Otherwise, an alternative custom sysfs interface is available.
- `mlnx_qos tool` (package: `ofed-scripts`) requires python version 2.5 <= X
- `tc_wrap.py` (package: `ofed-scripts`) requires python version 2.5 <= X

### 3.1.2.6 Packet Pacing

ConnectX-4 and ConnectX-4 Lx devices allow packet pacing (traffic shaping) per flow. This capability is achieved by mapping a flow to a dedicated send queue, and setting a rate limit on that send queue.

Note the following:
- Up to 512 send queues are supported
- 16 different rates are supported
- The rates can vary from 1 Mbps to line rate in 1 Mbps resolution
- Multiple queues can be mapped to the same rate (each queue is paced independently)
- It is possible to configure rate limit per CPU and per flow in parallel

**System Requirements**
- MLNX_EN, version 3.3
- Linux kernel 4.1 or higher
- ConnectX-4 or ConnectX-4 Lx adapter cards with a formal firmware version
Packet Pacing Configuration

This configuration is non-persistent and does not survive driver restart.

1. Firmware Activation:

   ➢ **To activate Packet Pacing in the firmware:**

   First, make sure that Mellanox Firmware Tools service (mst) is started:

   ```
   # mst start
   ```

   Then run:

   ```
   #echo "MLNX_RAW_TLV_FILE" > /tmp/mlxconfig_raw.txt
   #echo "0x00000004 0x0000010c 0x00000000 0x00000001" >> /tmp/mlxconfig_raw.txt
   #yes | mlxconfig -d <mst_dev> -f /tmp/mlxconfig_raw.txt set_raw > /dev/null
   #reboot /mlxfwreset
   ```

   ➢ **To deactivate Packet Pacing in the firmware:**

   ```
   #echo "MLNX_RAW_TLV_FILE" > /tmp/mlxconfig_raw.txt
   #echo "0x00000004 0x0000010c 0x00000000 0x00000000" >> /tmp/mlxconfig_raw.txt
   #yes | mlxconfig -d <mst_dev> -f /tmp/mlxconfig_raw.txt set_raw > /dev/null
   #reboot /mlxfwreset
   ```

2. There are two operation modes for Packet Pacing:

   a. Rate limit per CPU core:

      When XPS is enabled, traffic from a CPU core will be sent using the corresponding send queue. By limiting the rate on that queue, the transmit rate on that CPU core will be limited. For example:

      ```
      # echo 300 > /sys/class/net/ens2f1/queues/tx-0/tx_maxrate
      ```

      In this case, the rate on Core 0 (tx-0) is limited to 300Mbit/sec.

   b. Rate limit per flow:

      **Step 1.** The driver allows opening up to 512 additional send queues using the following command:

      ```
      # ethtool -L ens2f1 other 1200
      ```

      In this case, 1200 additional queues are opened.

      **Step 2.** Create flow mapping.

      The user can map a certain destination IP and/or destination layer 4 Port to a specific send queue. The match precedence is as follows:

      • IP + L4 Port
      • IP only
      • L4 Port only
      • No match (the flow would be mapped to default queues)

      To create flow mapping:
Configure the destination IP. Write the IP address in hexadecimal representation to the relevant sysfs entry. For example, to map IP address 192.168.1.1 (0xc0a80101) to send queue 310, run the following command:

```
# echo 0xc0a80101 > /sys/class/net/ens2f1/queues/tx-310/flow_map/dst_ip
```

To map Destination L4 3333 port (either TCP or UDP) to same queue, run:

```
# echo 3333 > /sys/class/net/ens2f1/queues/tx-310/flow_map/dst_port
```

From this point on, all traffic destined to the given IP address and L4 port will be sent using send queue 310. All other traffic will be sent using the original send queue.

**Step 3.** Limit the rate of this flow using the following command:

```
# echo 100 > /sys/class/net/ens2f1/queues/tx-310/tx_maxrate
```

**Note:** Each queue supports only a single IP+Port combination.

### 3.1.3 Quantized Congestion Notification (QCN)

Supported in ConnectX-3 and ConnectX-3 Pro only.

Congestion control is used to reduce packet drops in lossy environments and mitigate congestion spreading and resulting victim flows in lossless environments.

The Quantized Congestion Notification (QCN) IEEE standard (802.1Qau) provides congestion control for long-lived flows in limited bandwidth-delay product Ethernet networks. It is part of the IEEE Data Center Bridging (DCB) protocol suite, which also includes ETS, PFC, and DCBX. QCN is conducted at L2, and is targeted for hardware implementations. QCN applies to all Ethernet packets and all transports, and both the host and switch behavior is detailed in the standard.

QCN user interface allows the user to configure QCN activity. QCN configuration and retrieval of information is done by the `mlnx_qcn` tool. The command interface provides the user with a set of changeable attributes, and with information regarding QCN's counters and statistics. All parameters and statistics are defined per port and priority. QCN command interface is available if and only the hardware supports it.

#### 3.1.3.1 QCN Tool - `mlnx_qcn`

`mlnx_qcn` is a tool used to configure QCN attributes of the local host. It communicates directly with the driver thus does not require setting up a DCBX daemon on the system.

The `mlnx_qcn` enables the user to:

- Inspect the current QCN configurations for a certain port sorted by priority
- Inspect the current QCN statistics and counters for a certain port sorted by priority
- Set values of chosen QCN parameters
Features Overview and Configuration

Usage:

mlnx_qcn -i <interface> [options]

Options:

- --version
- -h, --help
- -i INTF, --interface=INTF
- -g TYPE, --get_type=TYPE
- --rpg_enable=RPG_ENABLE_LIST
- --rppp_max_rps=RPPP_MAX_RPS_LIST
- --rpg_time_reset=RPG_TIME_RESET_LIST
- --rpg_byte_reset=RPG_BYTE_RESET_LIST
- --rpg_threshold=RPG_THRESHOLD_LIST
- --rpg_max_rate=RPG_MAX_RATE_LIST
- --rpg_ai_rate=RPG_AI_RATE_LIST
- --rpg_hai_rate=RPG_HAI_RATE_LIST
- --rpg_gd=RPG_GD_LIST
- --rpg_min_dec_fac=RPG_MIN_DEC_FAC_LIST
- --rpg_min_rate=RPG_MIN_RATE_LIST
- --cndd_state_machine=CNDD_STATE_MACHINE_LIST

➢ To get QCN current configuration sorted by priority:

mlnx_qcn -i eth2 -g parameters
To show QCN's statistics sorted by priority:

```
mlnx_qcn -i eth2 -g statistics
```

Example output when running mlnx_qcn -i eth2 -g parameters:

```
priority 0:
  rpg_enable: 0
  rppp_max_rps: 1000
  rpg_time_reset: 1464
  rpg_byte_reset: 150000
  rpg_threshold: 5
  rpg_max_rate: 40000
  rpg_ai_rate: 10
  rpg_hai_rate: 50
  rpg_gd: 8
  rpg_min_dec_fac: 2
  rpg_min_rate: 10
  cndd_state_machine: 0

priority 1:
  rpg_enable: 0
  rppp_max_rps: 1000
  rpg_time_reset: 1464
  rpg_byte_reset: 150000
  rpg_threshold: 5
  rpg_max_rate: 40000
  rpg_ai_rate: 10
  rpg_hai_rate: 50
  rpg_gd: 8
  rpg_min_dec_fac: 2
  rpg_min_rate: 10
  cndd_state_machine: 0

............................
............................

priority 7:
  rpg_enable: 0
  rppp_max_rps: 1000
  rpg_time_reset: 1464
  rpg_byte_reset: 150000
  rpg_threshold: 5
  rpg_max_rate: 40000
  rpg_ai_rate: 10
  rpg_hai_rate: 50
  rpg_gd: 8
  rpg_min_dec_fac: 2
  rpg_min_rate: 10
  cndd_state_machine: 0
```

### 3.1.3.2 Setting QCN Configuration

Setting the QCN parameters, requires updating its value for each priority. `-1' indicates no change in the current value.
Example for setting 'rp g_enable' in order to enable QCN for priorities 3, 5, 6:
mlnx_qcn -i eth2 --rpg_enable=-1 -1 -1 1 1 1 -1

Example for setting 'rpg_hai_rate' for priorities 1, 6, 7:
mlnx_qcn -i eth2 --rpg_hai_rate=60 -1 -1 -1 -1 -1 60 60

### 3.1.4 Ethtool

Ethtool is a standard Linux utility for controlling network drivers and hardware, particularly for wired Ethernet devices. It can be used to:

- Get identification and diagnostic information
- Get extended device statistics
- Control speed, duplex, auto-negotiation and flow control for Ethernet devices
- Control checksum offload and other hardware offload features
- Control DMA ring sizes and interrupt moderation

The following are the ethtool supported options:

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`ethtool --set-priv-flags eth&lt;x&gt;</td>
<td>Enables/disables driver feature matching the given private flag.</td>
</tr>
<tr>
<td><code>&lt;priv flag&gt;</code> &lt;on/off&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**Table 7 - ethtool Supported Options**
### Table 7 - ethtool Supported Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ethtool --show-priv-flags eth&lt;x&gt;</td>
<td>Shows driver private flags and their states (ON/OFF)</td>
</tr>
<tr>
<td></td>
<td>The private flags are:</td>
</tr>
<tr>
<td></td>
<td>• qcn_disable_32_14_4_e</td>
</tr>
<tr>
<td></td>
<td>• disable_mc_loopback - when this flag is on, multicast traffic is not redirected to the device by loopback.</td>
</tr>
<tr>
<td></td>
<td>• mlx4_flow_steering_ethernet_l2</td>
</tr>
<tr>
<td></td>
<td>• mlx4_flow_steering_ipv4</td>
</tr>
<tr>
<td></td>
<td>• mlx4_flow_steering_tcp</td>
</tr>
<tr>
<td></td>
<td>For further information regarding the last three flags, refer to Flow Steering section.</td>
</tr>
<tr>
<td></td>
<td>The flags below are related to <em>Ignore Frame Check Sequence</em>, and they are active when ethtool -k does not support them:</td>
</tr>
<tr>
<td></td>
<td>• orx-fcs</td>
</tr>
<tr>
<td></td>
<td>• orx-all</td>
</tr>
<tr>
<td></td>
<td>The flag below is relevant for ConnectX-4 family cards only:</td>
</tr>
<tr>
<td></td>
<td>• rx_cqe_compress - used to control CQE compression. It is initialized with the automatic driver decision.</td>
</tr>
<tr>
<td></td>
<td>• per_channel_stats - used to control whether to expose per channel statistics counters in ethtool -S.</td>
</tr>
<tr>
<td>ethtool -a eth&lt;x&gt;</td>
<td><strong>Note:</strong> Supported in ConnectX®-3/ConnectX®-3 Pro cards only.</td>
</tr>
<tr>
<td></td>
<td>Queries the pause frame settings.</td>
</tr>
<tr>
<td>ethtool -A eth&lt;x&gt; [rx on</td>
<td>off] [tx on</td>
</tr>
<tr>
<td></td>
<td>Sets the pause frame settings.</td>
</tr>
<tr>
<td>ethtool -c eth&lt;x&gt;</td>
<td>Queries interrupt coalescing settings.</td>
</tr>
<tr>
<td>ethtool -C eth&lt;x&gt; [pkt-rate-low N] [pkt-rate-high N] [rx-usecs-low N] [rx-usecs-high N]</td>
<td><strong>Note:</strong> Supported in ConnectX®-3/ConnectX®-3 Pro cards only.</td>
</tr>
<tr>
<td></td>
<td>Sets the values for packet rate limits and for moderation time high and low values.</td>
</tr>
<tr>
<td></td>
<td>For further information, please refer to Adaptive Interrupt Moderation section.</td>
</tr>
</tbody>
</table>
### ethtool Supported Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ethtool -C eth&lt;x&gt; [rx-usecs N] [rx-frames N]</code></td>
<td>Sets the interrupt coalescing setting. Rx-frames will be enforced immediately, rx-usecs will be enforced only when adaptive moderation is disabled. <strong>Note:</strong> Usec settings correspond to the time to wait after the <em>last</em> packet is sent/received before triggering an interrupt.</td>
</tr>
<tr>
<td><code>ethtool -C eth&lt;x&gt; adaptive-rx on/off</code></td>
<td><strong>Note:</strong> Supported in ConnectX®-3/ConnectX®-3 Pro cards only. Enables/disables adaptive interrupt moderation. By default, the driver uses adaptive interrupt moderation for the receive path, which adjusts the moderation time to the traffic pattern. For further information, please refer to Adaptive Interrupt Moderation section.</td>
</tr>
<tr>
<td><code>ethtool -g eth&lt;x&gt;</code></td>
<td>Queries the ring size values.</td>
</tr>
<tr>
<td><code>ethtool -G eth&lt;x&gt; [rx &lt;N&gt;] [tx &lt;N&gt;]</code></td>
<td>Modifies the rings size.</td>
</tr>
</tbody>
</table>
| `ethtool -i eth<x>` | Checks driver and device information. For example: #> ethtool -i eth2  
driver: mlx4_en (MT_0DD0120009_CX3)  
version: 2.1.6 (Aug 2013)  
firmware-version: 2.30.3000  
bus-info: 0000:1a:00.0 |
| `ethtool -k eth<x>` | Queries the stateless offload status. |
### ethtool Supported Options

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`ethtool -K eth&lt;x&gt; [rx on</td>
<td>off] [tx on</td>
</tr>
<tr>
<td><code>ethtool -l eth&lt;x&gt;</code></td>
<td>Shows the number of channels</td>
</tr>
<tr>
<td><code>ethtool -L eth&lt;x&gt; [rx &lt;N&gt;] [tx &lt;N&gt;]</code></td>
<td>Sets the number of channels <strong>Note:</strong> This also resets the RSS table to its default distribution, which is uniform across the cores on the NUMA (non-uniform memory access) node that is closer to the NIC. <strong>Note:</strong> For ConnectX®-4 cards, use <code>ethtool -L eth&lt;x&gt;</code> combined <code>&lt;N&gt;</code> to set both RX and TX channels.</td>
</tr>
<tr>
<td>`ethtool -m</td>
<td>--dump-module-eeprom eth&lt;x&gt; [raw on</td>
</tr>
</tbody>
</table>
### 3.1.5 Checksum Offload

MLNX_EN supports the following Receive IP/L4 Checksum Offload modes:

- **CHECKSUM_UNNECESSARY**: By setting this mode the driver indicates to the Linux Networking Stack that the hardware successfully validated the IP and L4 checksum so the Linux Networking Stack does not need to deal with IP/L4 Checksum validation.

Checksum Unnecessary is passed to the OS when all of the following are true:

- `Ethtool -k <DEV> shows rx-checksumming: on`
- `Received TCP/UDP packet and both IP checksum and L4 protocol checksum are correct.`
• **CHECKSUM_COMPLETE**: When the checksum validation cannot be done or fails, the driver still reports to the OS the calculated by hardware checksum value. This allows accelerating checksum validation in Linux Networking Stack, since it does not have to calculate the whole checksum including payload by itself.

Checksum Complete is passed to OS when all of the following are true:
- Ethtool -k <DEV> shows rx-checksumming: on
- Using ConnectX®-3, firmware version 2.31.7000 and up
- Received IPv4/IPv6 non TCP/UDP packet

The ingress parser of the ConnectX®-3-Pro card comes by default without checksum offload support for non TCP/UDP packets.
To change that, please set the value of the module parameter ingress_parser_mode in mlx4_core to 1.
In this mode IPv4/IPv6 non TCP/UDP packets will be passed up to the protocol stack with CHECKSUM_COMPLETE tag.
In this mode of the ingress parser, the following features are unavailable:
- NVGRE stateless offloads
- VXLAN stateless offloads
- RoCE v2 (RoCE over UDP)
Change the default behavior only if non tcp/udp is very common.

• **CHECKSUM_NONE**: By setting this mode the driver indicates to the Linux Networking Stack that the hardware failed to validate the IP or L4 checksum so the Linux Networking Stack must calculate and validate the IP/L4 Checksum.
Checksum None is passed to OS for all other cases.

### 3.1.6 Ignore Frame Check Sequence (FCS) Errors

Supported in ConnectX-3 Pro and ConnectX-4 only.

Upon receiving packets, the packets go through a checksum validation process for the FCS field. If the validation fails, the received packets are dropped.

When FCS is enabled (disabled by default), the device does not validate the FCS field even if the field is invalid.

It is not recommended to enable FCS.

For further information on how to enable/disable FCS, please refer to Table 7, “ethtool Supported Options,” on page 49.
3.1.7 RDMA over Converged Ethernet (RoCE)

To be able to use RoCE, make sure VMA or DPDK infrastructure is installed (refer to Section 2.3.1, “Installation Modes”, on page 22).

Remote Direct Memory Access (RDMA) is the remote memory management capability that allows server-to-server data movement directly between application memory without any CPU involvement. RDMA over Converged Ethernet (RoCE) is a mechanism to provide this efficient data transfer with very low latencies on lossless Ethernet networks. With advances in data center convergence over reliable Ethernet, ConnectX® Ethernet adapter cards family with RoCE uses the proven and efficient RDMA transport to provide the platform for deploying RDMA technology in mainstream data center application at 10GigE and 40GigE link-speed. ConnectX® Ethernet adapter cards family with its hardware offload support takes advantage of this efficient RDMA transport (InfiniBand) services over Ethernet to deliver ultra-low latency for performance-critical and transaction intensive applications such as financial, database, storage, and content delivery networks.

When working with RDMA applications over Ethernet link layer the following points should be noted:

- The presence of a Subnet Manager (SM) is not required in the fabric. Thus, operations that require communication with the SM are managed in a different way in RoCE. This does not affect the API but only the actions such as joining multicast group, that need to be taken when using the API
- Since LID is a layer 2 attribute of the InfiniBand protocol stack, it is not set for a port and is displayed as zero when querying the port
- With RoCE, the alternate path is not set for RC QP. Therefore, APM (another type of High Availability and part of the InfiniBand protocol) is not supported
- Since the SM is not present, querying a path is impossible. Therefore, the path record structure must be filled with the relevant values before establishing a connection. Hence, it is recommended working with RDMA-CM to establish a connection as it takes care of filling the path record structure
- VLAN tagged Ethernet frames carry a 3-bit priority field. The value of this field is derived from the IB SL field by taking the 3 least significant bits of the SL field
- RoCE traffic is not shown in the associated Ethernet device's counters since it is off-loaded by the hardware and does not go through Ethernet network driver. RoCE traffic is counted in the same place where InfiniBand traffic is counted; /sys/class/infiniband/<device>/ports/<port number>/counters/

For further information on RoCE usage, please refer to MLNX_OFED User Manual.
3.1.8 Flow Control

3.1.8.1 Priority Flow Control (PFC)

Priority Flow Control (PFC) IEEE 802.1Qbb applies pause functionality to specific classes of traffic on the Ethernet link. For example, PFC can provide lossless service for the RoCE traffic and best-effort service for the standard Ethernet traffic. PFC can provide different levels of service to specific classes of Ethernet traffic (using IEEE 802.1p traffic classes).

3.1.8.1.1 PFC Local Configuration

3.1.8.1.1.1 Configuring PFC on ConnectX-3

Set the pfctx and pcfrx mlx_en module parameters to the file: /etc/modprobe.d/mlx4_en.conf:

```bash
options mlx4_en pfctx=0x08 pcfrx=0x08
```

**Note:** These parameters are bitmap of 8 bits. In the example above, only priority 3 is enabled (0x8 is 00001000b). 0x16 will enable priority 4 and so on.

3.1.8.1.1.2 Configuring PFC on ConnectX-4

1. Enable PFC on the desired priority:

   ```bash
   # mlnx_qos -i <ethX> --pfc <0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>,<0/1>
   ```

   **Example** (Priority=4):

   ```bash
   # mlnx_qos -i eth1 --pfc 0,0,0,0,1,0,0,0
   ```

2. Create a VLAN interface:

   ```bash
   # vconfig add <ethX> <VLAN_ID>
   ```

   **Example** (VLAN_ID=5):

   ```bash
   # vconfig add eth1 5
   ```

3. Set egress mapping:

   a. For Ethernet traffic:

   ```bash
   # vconfig set_egress_map <vlan_einterface> <skprio> <up>
   ```

   **Example** (skprio=3, up=5):

   ```bash
   # vconfig set_egress_map eth1.5 3 5
   ```

4. Create 8 Traffic Classes (TCs):

   ```bash
   tc_wrap.py -i <interface>
   ```

5. Enable PFC on the switch.

   For information on how to enable PFC on your respective switch, please refer to Switch FC/PFC Configuration sections in the following Mellanox Community page:
   [https://community.mellanox.com/docs/DOC-2283](https://community.mellanox.com/docs/DOC-2283).
3.1.8.1.2 PFC Configuration Using LLDP DCBX

3.1.8.1.2.1 PFC Configuration on Hosts

PFC Auto-Configuration Using LLDP Tool in the OS

Step 1. Start lldpad daemon on host.

```
lldpad -d
service lldpad start
```

Step 2. Send lldpad packets to the switch.

```
lldptool set-lldp -i <ethX> adminStatus=rxtx ;
lldptool -T -i <ethX> -V sysName enableTx=yes;
lldptool -T -i <ethX> -V portDesc enableTx=yes ;
lldptool -T -i <ethX> -V sysDesc enableTx=yes
lldptool -T -i <ethX> -V sysCap enableTx=ysess
lldptool -T -i <ethX> -V mngAddr enableTx=yes
lldptool -T -i <ethX> -V PFC enableTx=yes;
lldptool -T -I <ethX> -V CEE-DCBX enableTx=yes;
```

Step 3. Set the PFC parameters.

- For the CEE protocol, use dcbtool:

```
dcbtool sc <ethX> pfc pfcup:<xxxxxxxxx>
```

Example:

```
dcbtool sc eth6 pfc pfcup:01110001
```

where:

- Enables/disables priority flow control.
- [pfcup:xxxxxxxx] From left to right (priorities 0-7) - x can be equal to either 0 or 1. 1 indicates that the priority is configured to transmit priority pause.

- For IEEE protocol, use lldptool:

```
lldptool -T -i <ethX> -V PFC enabled=x,x,x,x,x,x,x,x
```

Example:

```
lldptool -T -i eth2 -V PFC enabled=1,2,4
```

where:

```
enabled Displays or sets the priorities with PFC enabled. The set attribute takes a comma-separated list of priorities to enable, or the string none to disable all priorities.
```

PFC Auto-Configuration Using LLDP in the Firmware (for mlx5 driver)

There are two ways to configure PFC and ETS on the server:

1. Local Configuration - Configuring each server manually.
2. Remote Configuration - Configuring PFC and ETS on the switch, after which the switch will pass the configuration to the server using LLDP DCBX TLVs.

There are two ways to implement the remote configuration using ConnectX-4 adapters:

a. Configuring the adapter firmware to enable DCBX.
b. Configuring the host to enable DCBX.

For further information on how to auto-configure PFC using LLDP in the firmware, refer to the [HowTo Auto-Config PFC and ETS on ConnectX-4 via LLDP DCBX](#) Community post.

### 3.1.8.1.2 PFC Configuration on Switches

**Step 1.** In order to enable DCBX, LLDP should first be enabled:

```
switch (config) # lldp
show lldp interfaces ethernet remote
```

**Step 2.** Add DCBX to the list of supported TLVs per required interface.

- For IEEE DCBX:
  ```
  switch (config) # interface 1/1
  switch (config interface ethernet 1/1) # lldp tlv-select dcbx
  ```

- For CEE DCBX:
  ```
  switch (config) # interface 1/1
  switch (config interface ethernet 1/1) # lldp tlv-select dcbx-cee
  ```

**Step 3.** [Optional] Application Priority can be configured on the switch, with the required ether-type and priority. For example, IP packet, priority 1:

```
switch (config) # dcb application-priority 0x8100 1
```

**Step 4.** Make sure PFC is enabled on the host (for enabling PFC on the host, refer to Section 3.1.8.1.2.1 above). Once it is enabled, it will be passed in the LLDP TLVs.

**Step 5.** Enable PFC with the desired priority on the Ethernet port.

- `dcb priority-flow-control enable force`
- `dcb priority-flow-control priority <priority> enable`
- `interface ethernet <port> dcb priority-flow-control mode on force`

**Example:** Enabling PFC with priority 3 on port 1/1:

```
dcb priority-flow-control enable force
dcb priority-flow-control priority 3 enable
interface ethernet 1/1 dcb priority-flow-control mode on force
```

### 3.1.8.1.3 Priority Counters

MLNX_EN driver supports several ingress and egress counters per priority. Run `ethtool -S` to get the full list of port counters.

**ConnectX-3 and ConnectX-4 Counters**

- Rx and Tx Counters:
  - Packets
• Bytes
• Octets
• Frames
• Pause
• Pause frames
• Pause Duration
• Pause Transition

**ConnectX-3 Example**

```bash
# ethtool -S eth1 | grep prio_3
```

```
rx_prio_3_packets: 5152
rx_prio_3_bytes: 424080
tx_prio_3_packets: 328209
tx_prio_3_bytes: 361752914
rx_pause_prio_3: 14812
rx_pause_duration_prio_3: 0
rx_pause_transition_prio_3: 0
tx_pause_prio_3: 0
tx_pause_duration_prio_3: 47848
tx_pause_transition_prio_3: 7406
```

**ConnectX-4 Example**

```bash
# ethtool -S eth35 | grep prio4
```

```
prio4_rx_octets: 62147780800
prio4_rx_frames: 14885696
prio4_tx_octets: 0
prio4_tx_frames: 0
prio4_rx_prio4: 0
prio4_rx_prio4_duration: 0
prio4_tx_prio4: 26832
prio4_tx_prio4_duration: 14508
prio4_rx_prio4_transition: 0
```

1. The Pause counters in ConnectX-4 are visible via ethtool only for priorities on which PFC is enabled.

**3.1.8.1.4 PFC Storm Prevention**

This feature is applicable to ConnectX-4/ConnectX-5 adapter cards family only.

PFC storm prevention enables toggling between default and auto modes.
The stall prevention timeout is configured to 8 seconds by default. Auto mode sets the stall prevention timeout to be 100 msec.

The feature can be controlled using sysfs at the following location: /sys/class/net/eth*/settings/pfc_stall_prevention

- To query the PFC stall prevention mode:

  ```bash
  cat /sys/class/net/eth*/settings/pfc_stall_prevention
  ```

**Example**

```bash
$ cat /sys/class/net/ens6/settings/pfc_stall_prevention
default
```

- To configure the PFC stall prevention mode:

  ```bash
  Echo "auto"/"default" > /sys/class/net/eth*/settings/pfc_stall_prevention
  ```

The following two counters were added to the ethtool -S:

- `tx_Pause_storm_warning_events` - when the device is stalled for a period longer than a pre-configured watermark, the counter increases, allowing the debug utility an insight into current device status.

- `tx_pause_storm_error_events` - when the device is stalled for a period longer than a pre-configured timeout, the pause transmission is disabled, and the counter increase.

### 3.1.8.2 Dropless Receive Queue (RQ)

This feature is applicable to ConnectX-4/ConnectX-5 adapter cards family only.

Dropless RQ feature enables the driver to notify the FW when SW receive queues are overloaded. This scenario takes place when the handling of SW receive queue is slower than the handling of the HW receive queues.

When this feature is enabled, a packet that is received while receive queue is full will not be immediately dropped. The FW will accumulate these packets assuming posting of new WQEs will resume shortly. If received WQEs are not posted after a certain period of time, `out_of_buffer` counter will increase, indicating that the packet has been dropped.

This feature is disabled by default. In order to activate it, ensure that Flow Control feature is also enabled.

➢ **To enable the feature, run:**

  ```bash
  ethtool --set-priv-flags ens6 dropless_rq on
  ```

➢ **To get the feature state, run:**

  ```bash
  ethtool --show-priv-flags DEVNAME
  ```
Output example:

<table>
<thead>
<tr>
<th>Private flags for DEVNAME:</th>
</tr>
</thead>
<tbody>
<tr>
<td>rx_cqe_moder : on</td>
</tr>
<tr>
<td>rx_cqe_compress: off</td>
</tr>
<tr>
<td>sniffer : off</td>
</tr>
<tr>
<td>droppless_rq : off</td>
</tr>
<tr>
<td>hw_lro : off</td>
</tr>
</tbody>
</table>

➢ To disable the feature, run:

```bash
ethtool --set-priv-flags ens6 droppless_rq off
```
3.1.9 Explicit Congestion Notification (ECN)

3.1.9.1 ConnectX-3/ConnectX-3 Pro ECN

ECN is an extension to the IP protocol. It allows reliable communication by notifying all ends of communication when a congestion occurs. This is done without dropping packets. Please note that since this feature requires all nodes in the path (nodes, routers etc) between the communicating nodes to support ECN to ensure reliable communication. ECN is marked as 2 bits in the traffic control IP header. This ECN implementation refers to RoCE v2.

ECN command interface is use to configure ECN activity. The access to it is through the file system (mount of debugfs is required). The interface provides a set of changeable attributes, and information regarding ECN’s counters and statistics.

Enabling the ECN command interface is done by setting the en_ecn module parameter of mlx4_ib to 1:

```bash
options mlx4_ib en_ecn=1
```

3.1.9.1.1 Enabling ConnectX-3/ConnectX-3 Pro ECN

➢ To enable ConnectX-3/ConnectX-3 Pro ECN on the hosts

Step 1. Enable ECN in sysfs.

```
/proc/sys/net/ipv4/tcp_ecn = 1
```

Step 2. Enable ECN CLI.

```
options mlx4_ib en_ecn=1
```

Step 3. Restart the driver.

```
/etc/init.d/openibd restart
```

Step 4. Mount debugfs to access ECN attributes.

```
mount -t debugfs none /sys/kernel/debug/
```

Please note, mounting of debugfs is required.

The following is an example for ECN configuration through debugfs (echo 1 to enable attribute):

```
/sys/kernel/debug/mlx4_ib/<device>/ecn/<algo>/ports/1/params/prios/<prio>/<the requested attribute>
```

ECN supports the following algorithms:

- `r_roce_ecn_rp`
- `r_roce_ecn_np`

Each algorithm has a set of relevant parameters and statistics, which are defined per device, per port, per priority.
r_roce_ecn_np has an extra set of general parameters which are defined per device.

ECN and QCN are not compatible. When using ECN, QCN (and all its related daemons/utilities that could enable it, i.e. lldpad) should be turned OFF.

3.1.9.1.2 Various ECN Paths

The following are the paths to ECN algorithm, general parameters and counters.

- The path to an algorithm attribute is (except for general parameters):
  
  `/sys/kernel/debug/mlx4_ib/{DEVICE}/ecn/{algorithm}/ports/{port}/params/prios/{prio}/(attribute)`

- The path to a general parameter is:
  
  `/sys/kernel/debug/mlx4_ib/{DEVICE}/ecn/r_roce_ecn_np/gen_params/{attribute}`

- The path to a counter is:
  
  `/sys/kernel/debug/mlx4_ib/{DEVICE}/ecn/{algorithm}/ports/{port}/statistics/prios/{prio}/{counter}`

3.1.9.2 ConnectX-4 ECN

ECN in ConnectX-4 enables end-to-end congestions notifications between two end-points when a congestion occurs, and works over Layer 3. ECN must be enabled on all nodes in the path (nodes, routers etc) between the two end points and the intermediate devices (switches) between them to ensure reliable communication. ECN handling is supported only in RoCEv2 packets.

3.1.9.2.1 Enabling ConnectX-4 ECN

- To enable ConnectX-4 ECN on the hosts
  
  **Step 1.** Enable ECN in sysfs.
  
  `/sys/class/net/<interface>/<protocol>/ecn_<protocol>_enable =1`

  **Step 2.** Query the attribute.
  
  `cat /sys/class/net/<interface>/ecn/<protocol>/params/<requested attribute>`

  **Step 3.** Modify the attribute.
  
  `echo <value> /sys/class/net/<interface>/ecn/<protocol>/params/<requested attribute>`

ECN supports the following algorithms:

- `r_roce_ecn_rp` - Reaction point
- `r_roce_ecn_np` - Notification point

Each algorithm has a set of relevant parameters and statistics, which are defined per device, per port, per priority.

- To query ECN enable per Priority X:
  
  `cat /sys/class/net/<interface>/ecn/<protocol>/enable/X`
To read ECN configurable parameters:
```bash
cat /sys/class/net/<interface>/ecn/<protocol>/requested attributes
```

To enabled ECN for each priority per protocol:
```bash
echo 1 > /sys/class/net/<interface>/ecn/<protocol>/enable/X
```

To modify ECN configurable parameters:
```bash
echo <value> > /sys/class/net/<interface>/ecn/<protocol>/requested attributes
```

Where:
- X: priority {0..7}
- protocol: roce_rp / roce_np
- requested attributes: Next Slide for each protocol.

3.1.10 RSS Support

3.1.10.1 RSS Hash Function

The device has the ability to use XOR as the RSS distribution function, instead of the default Toplitz function.

The XOR function can be better distributed among driver's receive queues in small number of streams, where it distributes each TCP/UDP stream to a different queue.

3.1.10.1.1 mlx4 RSS Hash Function

MLNX_EN provides one of the following options to change the working RSS hash function from Toplitz to XOR, and vice versa:

- Through ethtool priv-flags, in case mlx4_rss_xor_hash_function is not part of the priv-flags list:
  ```bash
  ethtool --set-priv-flags eth<x> mlx4_rss_xor_hash_function on/off
  ```

- Through ethtool, provided as part of MLNX_EN package, in case mlx4_rss_xor_hash_function is not part of the priv-flags list:
  ```bash
  /opt/mellanox/ethtool# ./sbin/ethtool -X ens4 hfunc xor
  /opt/mellanox/ethtool# ./sbin/ethtool --show-rxfh ens4
  ```

**Output:**

RX flow hash indirection table for ens4 with 8 RX ring(s):

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>

RSS hash key:
```
:b7:6c:c8:7e:d6:e3:14:17
```

RSS hash function:
- toeplitz: off
- xor: on

For further information, please refer to Table 7, “ethtool Supported Options,” on page 49.
3.1.10.1.2 mlx5 RSS Hash Function

MLNX_EN provides the following option to change the working RSS hash function from Tooplitz to XOR, and vice versa:

- Through sysfs, located at: /sys/class/net/eth*/settings/hfunc.

- To query the operational and supported hash functions:

```bash
cat /sys/class/net/eth*/settings/hfunc
```

Example:

```
# cat /sys/class/net/eth2/settings/hfunc
Operational hfunc: toplitz
Supported hfuncs: xor toplitz
```

- To change the operational hash function:

```bash
echo xor > /sys/class/net/eth*/settings/hfunc
```

3.1.10.1.3 RSS Support for IP Fragments

Supported in ConnectX-3 and ConnectX-3 Pro only.

As of MLNX_EN v2.4-.1.0.0, RSS will distribute incoming IP fragmented datagrams according to its hash function, considering the L3 IP header values. Different IP fragmented datagrams flows will be directed to different rings.

- When the first packet in IP fragments chain contains upper layer transport header (e.g. UDP packets larger than MTU), it will be directed to the same target as the proceeding IP fragments that follows it, to prevent out-of-order processing.

3.1.11 Time-Stamping

3.1.11.1 Time-Stamping Service

Time stamping is the process of keeping track of the creation of a packet. A time-stamping service supports assertions of proof that a datum existed before a particular time. Incoming packets are time-stamped before they are distributed on the PCI depending on the congestion in the PCI buffers. Outgoing packets are time-stamped very close to placing them on the wire.

3.1.11.1.1 Enabling Time Stamping

Time-stamping is off by default and should be enabled before use.

- To enable time stamping for a socket:
• Call setsockopt() with SO_TIMESTAMPING and with the following flags:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOF_TIMESTAMPING_TX_HARDWARE</td>
<td>try to obtain send time stamp in hardware</td>
</tr>
<tr>
<td>SOF_TIMESTAMPING_TX_SOFTWARE</td>
<td>if SOF_TIMESTAMPING_TX_HARDWARE is off or fails, then do it in software</td>
</tr>
<tr>
<td>SOF_TIMESTAMPING_RX_HARDWARE</td>
<td>return the original, unmodified time stamp as generated by the hardware</td>
</tr>
<tr>
<td>SOF_TIMESTAMPING_RX_SOFTWARE</td>
<td>if SOF_TIMESTAMPING_RX_HARDWARE is off or fails, then do it in software</td>
</tr>
<tr>
<td>SOF_TIMESTAMPING_RAW_HARDWARE</td>
<td>return original raw hardware time stamp</td>
</tr>
<tr>
<td>SOF_TIMESTAMPING_RAW_SOFTWARE</td>
<td>return hardware time stamp transformed to the system time base</td>
</tr>
<tr>
<td>SOF_TIMESTAMPING_SOFTWARE</td>
<td>return system time stamp generated in software</td>
</tr>
<tr>
<td>SOF_TIMESTAMPING_RAW/RX</td>
<td>determine how time stamps are generated</td>
</tr>
<tr>
<td>SOF_TIMESTAMPING_RAW/SYS</td>
<td>determine how they are reported</td>
</tr>
</tbody>
</table>

➢ To enable time stamping for a net device:

Admin privileged user can enable/disable time stamping through calling ioctl (sock, SIOCSHTIMESTAMP, &ifreq) with following values:

Send side time sampling:
• Enabled by ifreq.hwtstamp_config.tx_type when

```c
/* possible values for hwtstamp_config->tx_type */
enum hwtstamp_tx_types {
    /*
    * No outgoing packet will need hardware time stamping;
    * should a packet arrive which asks for it, no hardware
    * time stamping will be done.
    */
    HWTSTAMP_TX_OFF,

    /*
    * Enables hardware time stamping for outgoing packets;
    * the sender of the packet decides which are to be
    * time stamped by setting %SOF_TIMESTAMPING_TX_SOFTWARE
    * before sending the packet.
    */
    HWTSTAMP_TX_ON,

    /*
    * Enables time stamping for outgoing packets just as
    * HWTSTAMP_TX_ON does, but also enables time stamp insertion
    * directly into Sync packets. In this case, transmitted Sync
    * packets will not received a time stamp via the socket error
    * queue.
    */
    HWTSTAMP_TX_ONESTEP_SYNC,
};
```
Note: for send side time stamping currently only HWTSTAMP_TX_OFF and HWTSTAMP_TX_ON are supported.
Receive side time sampling:

- Enabled by `ifreq.hwtstamp_config.rx_filter` when

```c
/* possible values for hwtstamp_config->rx_filter */
enum hwtstamp_rx_filters {
    /* time stamp no incoming packet at all */
    HWTSTAMP_FILTER_NONE,
    /* time stamp any incoming packet */
    HWTSTAMP_FILTER_ALL,
    /* return value: time stamp all packets requested plus some others */
    HWTSTAMP_FILTER_SOME,
    /* PTP v1, UDP, any kind of event packet */
    HWTSTAMP_FILTER_PTP_V1_L4_EVENT,
    /* PTP v1, UDP, Sync packet */
    HWTSTAMP_FILTER_PTP_V1_L4_SYNC,
    /* PTP v1, UDP, Delay_req packet */
    HWTSTAMP_FILTER_PTP_V1_L4_DELAY_REQ,
    /* PTP v2, UDP, any kind of event packet */
    HWTSTAMP_FILTER_PTP_V2_L4_EVENT,
    /* PTP v2, UDP, Sync packet */
    HWTSTAMP_FILTER_PTP_V2_L4_SYNC,
    /* PTP v2, UDP, Delay_req packet */
    HWTSTAMP_FILTER_PTP_V2_L4_DELAY_REQ,
    /* 802.AS1, Ethernet, any kind of event packet */
    HWTSTAMP_FILTER_PTP_V2_L2_EVENT,
    /* 802.AS1, Ethernet, Sync packet */
    HWTSTAMP_FILTER_PTP_V2_L2_SYNC,
    /* 802.AS1, Ethernet, Delay_req packet */
    HWTSTAMP_FILTER_PTP_V2_L2_DELAY_REQ,
    /* PTP v2/802.AS1, any layer, any kind of event packet */
    HWTSTAMP_FILTER_PTP_V2_EVENT,
    /* PTP v2/802.AS1, any layer, Sync packet */
    HWTSTAMP_FILTER_PTP_V2_SYNC,
    /* PTP v2/802.AS1, any layer, Delay_req packet */
    HWTSTAMP_FILTER_PTP_V2_DELAY_REQ,
};
```

Note: for receive side time stamping currently only HWTSTAMP_FILTER_NONE and HWTSTAMP_FILTER_ALL are supported.

### 3.11.1.2 Getting Time Stamping

Once time stamping is enabled time stamp is placed in the socket Ancillary data. `recvmsg()` can be used to get this control message for regular incoming packets. For send time stamps the outgoing packet is looped back to the socket's error queue with the send time stamp(s) attached. It can be received with `recvmsg()` with `MSG_ERRQUEUE` flag. The call returns the original outgoing packet data including all headers preprended down to and including the link layer, the `scm_time-`
stamping control message and a sock_extended_err control message with ee_errno==ENOMSG and ee_origin==SO_EE_ORIGIN_TIMESTAMPING. A socket with such a pending bounced packet is ready for reading as far as select() is concerned. If the outgoing packet has to be fragmented, then only the first fragment is time stamped and returned to the sending socket.

When time-stamping is enabled, VLAN stripping is disabled. For more info please refer to Documentation/networking/timestamping.txt in kernel.org

3.1.11.3 Querying Time Stamping Capabilities via ethtool

➢ To display Time Stamping capabilities via ethtool:

- Show Time Stamping capabilities

  ```
ethtool -T eth<x>
ethtool -T eth0
```

Example:

```
ethtool -T eth0
Time stamping parameters for p2p1: capabilities:
Capabilities:
  hardware-transmit (SOF_TIMESTAMPING_TX_HARDWARE)
  software-transmit (SOF_TIMESTAMPING_TX_SOFTWARE)
  hardware-receive (SOF_TIMESTAMPING_RX_HARDWARE)
  software-receive (SOF_TIMESTAMPING_RX_SOFTWARE)
  software-system-clock (SOF_TIMESTAMPING_SOFTWARE)
  hardware-raw-clock (SOF_TIMESTAMPING_RAW_HARDWARE)

PTP Hardware Clock: 1
Hardware Transmit Timestamp Modes:
  off (HWTSTAMP_TX_OFF)
  on (HWTSTAMP_TX_ON)

Hardware Receive Filter Modes:
  none (HWTSTAMP_FILTER_NONE)
  all (HWTSTAMP_FILTER_ALL)
```

For more details on PTP Hardware Clock, please refer to:


3.1.11.4 Steering PTP Traffic to Single RX Ring

As a result of Receive Side Steering (RSS) PTP traffic coming to UDP ports 319 and 320, it may reach the user space application in an out of order manner. In order to prevent this, PTP traffic needs to be steered to single RX ring using ethtool.

Example:

```
# ethtool -u ens7
8 RX rings available
```
3.1.11.2 One Pulse Per Second (1PPS)

This feature is supported in ConnectX-4 adapter cards family and above only.

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

For further information, refer to HowTo Test 1PPS on Mellanox Adapters Community post.

3.1.12 Flow Steering

Flow steering is a new model which steers network flows based on flow specifications to specific QPs. Those flows can be either unicast or multicast network flows. In order to maintain flexibility, domains and priorities are used. Flow steering uses a methodology of flow attribute, which is a combination of L2-L4 flow specifications, a destination QP and a priority. Flow steering rules may be inserted either by using ethtool or by using InfiniBand verbs. The verbs abstraction uses a different terminology from the flow attribute (ibv_exp_flow_attr), defined by a combination of specifications (struct ibv_exp_flow_spec_*).
3.1.12.1 Enable/Disable Flow Steering

Flow steering is generally enabled when the `log_num_mgm_entry_size` module parameter is non-positive (e.g., `-log_num_mgm_entry_size`), meaning the absolute value of the parameter, is a bit field. Every bit indicates a condition or an option regarding the flow steering mechanism:

- **b0**: Force device managed Flow Steering
  - When set to 1, it forces HCA to be enabled regardless of whether NC-SI Flow Steering is supported or not.

- **b2**: Enable A0 static DMFS steering (see Section 3.1.12.3, “A0 Static Device Managed Flow Steering”, on page 72)
  - When set to 1, A0 static DMFS steering is enabled. This bit should be set to 0 when "b1- Disable IPoIB Flow Steering" is 0.

- **b3**: Enable DMFS only if the HCA supports more than 64 QPs per MCG entry
  - When set to 1, DMFS is enabled only if the HCA supports more than 64 QPs attached to the same rule. For example, attaching 64 VFs to the same multicast address causes 64 QPs to be attached to the same MCG. If the HCA supports less than 64 QPs per MCG, B0 is used.

- **b4**: Optimize IPoIB steering table for non source IP rules when possible
  - When set to 1, IPoIB steering table will be optimized to support rules ignoring source IP check. This optimization is available only when IPoIB Flow Steering is set.

- **b5**: Optimize steering table for non source IP rules when possible
  - When set to 1, steering table will be optimized to support rules ignoring source IP check. This optimization is possible only when DMFS mode is set.

For example, a value of (-7) means:
- forcing Flow Steering regardless of NC-SI Flow Steering support
- disabling IPoIB Flow Steering support
- enabling A0 static DMFS steering
- steering table is not optimized for rules ignoring source IP check

The default value of `log_num_mgm_entry_size` is -10. Meaning Ethernet Flow Steering (i.e. IPoIB DMFS is disabled by default) is enabled by default if NC-SI DMFS is supported and the HCA supports at least 64 QPs per MCG entry. Otherwise, L2 steering (B0) is used.
When using SR-IOV, flow steering is enabled if there is an adequate amount of space to store the flow steering table for the guest/master.

➢ To enable Flow Steering:

Step 1. Open the /etc/modprobe.d/mlnx.conf file.

Step 2. Set the parameter log_num_mgm_entry_size to a non positive value by writing the option mlx4_core log_num_mgm_entry_size=<value>.

Step 3. Restart the driver

➢ To disable Flow Steering:

Step 1. Open the /etc/modprobe.d/mlnx.conf file.

Step 2. Remove the options mlx4_core log_num_mgm_entry_size= <value>.

Step 3. Restart the driver

3.1.12.2 Flow Steering Support

Flow Steering is supported in ConnectX®-3, ConnectX®-3 Pro, ConnectX®-4 and ConnectX®-4 Lx adapter cards.

➢ [For ConnectX®-3 and ConnectX®-3 Pro only] To determine which Flow Steering features are supported:

```
ethtool --show-priv-flags eth4
```

The following output will be received:

```
mlx4_flow_steering_ethernet_l2: on       Creating Ethernet L2 (MAC) rules is supported
mlx4_flow_steering_ipv4: on              Creating IPv4 rules is supported
mlx4_flow_steering_tcp: on               Creating TCP/UDP rules is supported
```

For ConnectX-4 and ConnectX-4 Lx adapter cards, all supported features are enabled.

Flow Steering support in InfiniBand is determined according to the EXP_MANAGED_-FLOW_STEERING flag.

3.1.12.3 A0 Static Device Managed Flow Steering

This mode is supported in ConnectX®-3 and ConnectX®-3 Pro only.

This mode enables fast steering, however it might impact flexibility. Using it increases the packet rate performance by ~30%, with the following limitations for Ethernet link-layer unicast QPs:
• Limits the number of opened RSS Kernel QPs to 96. MACs should be unique (1 MAC per 1 QP). The number of VFs is limited.

• When creating Flow Steering rules for user QPs, only MAC--> QP rules are allowed. Both MACs and QPs should be unique between rules. Only 62 such rules could be created.

• When creating rules with Ethtool, MAC--> QP rules could be used, where the QP must be the indirection (RSS) QP. Creating rules that indirect traffic to other rings is not allowed. Ethtool MAC rules to drop packets (action -1) are supported.

• RFS is not supported in this mode

• VLAN is not supported in this mode

### 3.1.12.4 Flow Domains and Priorities

ConnectX®-4 and ConnectX®-4 Lx adapter cards support only User Verbs domain with struct ibv_exp_flow_spec_eth flow specification using 4 priorities.

Flow steering defines the concept of domain and priority. Each domain represents a user agent that can attach a flow. The domains are prioritized. A higher priority domain will always supersede a lower priority domain when their flow specifications overlap. Setting a lower priority value will result in higher priority.

In addition to the domain, there is priority within each of the domains. Each domain can have at most 2^12 priorities in accordance with its needs.

The following are the domains at a descending order of priority:

- **Ethtool**

  Ethtool domain is used to attach an RX ring, specifically its QP to a specified flow.

  Please refer to the most recent ethtool manpage for all the ways to specify a flow.

Examples:

- `ethtool –U eth5 flow-type ether dst 00:11:22:33:44:55 loc 5 action 2`
  All packets that contain the above destination MAC address are to be steered into rx-ring 2 (its underlying QP), with priority 5 (within the ethtool domain)

- `ethtool –U eth5 flow-type tcp4 src-ip 1.2.3.4 dst-port 8888 loc 5 action 2`
  All packets that contain the above destination IP address and source port are to be steered into rx-ring 2. When destination MAC is not given, the user's destination MAC is filled automatically.

- `ethtool -U eth5 flow-type ether dst 00:11:22:33:44:55 vlan 45 m 0xf000 loc 5 action 2`
  All packets that contain the above destination MAC address and specific VLAN are steered into ring 2. Please pay attention to the VLAN's mask 0xf000. It is required in order to add such a rule.

- `ethtool –u eth5`
  Shows all of ethtool’s steering rule
When configuring two rules with the same priority, the second rule will overwrite the first one, so this ethtool interface is effectively a table. Inserting Flow Steering rules in the kernel requires support from both the ethtool in the user space and in kernel (v2.6.28).

MLX4 Driver Support

The mlx4 driver supports only a subset of the flow specification the ethtool API defines. Asking for an unsupported flow specification will result with an “invalid value” failure.

The following are the flow specific parameters:

<table>
<thead>
<tr>
<th>ether</th>
<th>tcp/udp</th>
<th>ip4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory</td>
<td>dst</td>
<td>src-ip/dst-ip</td>
</tr>
<tr>
<td>Optional</td>
<td>vlan</td>
<td>src-ip, dst-ip, src-port, dst-port, vlan</td>
</tr>
</tbody>
</table>

• Accelerated Receive Flow Steering (aRFS)

Receive Flow Steering (RFS) and Accelerated Receive Flow Steering (aRFS) are kernel features currently available in most distributions. For RFS, packets are forwarded based on the location of the application consuming the packet. aRFS boosts the speed of RFS by adding the support for the hardware. By using aRFS (unlike RFS), the packets are directed to a CPU that is local to the thread running the application.

aRFS is an in-kernel-logic responsible for load balancing between CPUs by attaching flows to CPUs that are used by flow’s owner applications. This domain allows the aRFS mechanism to use the flow steering infrastructure to support the aRFS logic by implementing the `ndo_rx_flow_steer`, which, in turn, calls the underlying flow steering mechanism with the aRFS domain.

➢ To configure RFS:

Configure the RFS flow table entries (globally and per core).

**Note**: The functionality remains disabled until explicitly configured (by default it is 0).

- The number of entries in the global flow table is set as follows:

  ```
  /proc/sys/net/core/rps_sock_flow_entries
  ```

- The number of entries in the per-queue flow table are set as follows:

  ```
  /sys/class/net/<dev>/queues/rx-<n>/rps_flow_cnt
  ```

**Example**:

```bash
# echo 32768 > /proc/sys/net/core/rps_sock_flow_entries
# for f in /sys/class/net/ens6/queues/rx-*/rps_flow_cnt; do echo 32768 > $f; done
```

➢ To Configure aRFS:

The aRFS feature requires explicit configuration in order to enable it. Enabling the aRFS requires enabling the `ntuple` flag via the ethtool.
For example, to enable ntuple for eth0, run:

```
ethtool -K eth0 ntuple on
```

aRFS requires the kernel to be compiled with the `CONFIG_RFS_ACCEL` option. This option is available in kernels 2.6.39 and above. Furthermore, aRFS requires Device Managed Flow Steering support.

RFS cannot function if LRO is enabled. LRO can be disabled via `ethtool`.

- **All of the rest**

The lowest priority domain serves the following users:

- **The mlx4 Ethernet driver** attaches its unicast and multicast MACs addresses to its QP using L2 flow specifications

- **The mlx4 ipoib driver** when it attaches its QP to his configured GIDS

Fragmented UDP traffic cannot be steered. It is treated as ‘other’ protocol by hardware (from the first packet) and not considered as UDP traffic.

We recommend using `libibverbs v2.0-3.0.0` and `libmlx4 v2.0-3.0.0` and higher as of `MLNX_EN v2.0-3.0.0` due to API changes.

**3.1.12.5 Flow Steering Dump Tool**

This tool is only supported for ConnectX-4 and above adapter cards.

The mlx_fs_dump is a python tool that prints the steering rules in a readable manner. Python v2.7 or above, as well as pip, anytree and termcolor libraries are required to be installed on the host.

Running example:

```
./ofed_scripts/utils/mlx_fs_dump -d /dev/mst/mt4115_pciconf0
```

```
FT: 9 (level: 0x18, type: NIC_RX)
  +-- FG: 0x15 (MISC)
    |-- FTE: 0x0 (FWD) to (TIR:0x7e) out.ethtype:IPv4 out.ip_prot:UDP out.udp_dport:0x140
    |-- FTE: 0x1 (FWD) to (TIR:0x7e) out.ethtype:IPv4 out.ip_prot:UDP out.udp_dport:0x13f
...
```

For further information on the mlx_fs_dump tool, please refer to `mlx_fs_dump` Community post.
3.1.13 Wake-on-LAN (WoL)

Wake-on-LAN (WoL) is a technology that allows a network professional to remotely power on a computer or to wake it up from sleep mode.

- **To enable WoL:**
  
  ```
  # ethtool -s <interface> wol g
  ```

- **To get WoL:**
  
  ```
  ethtool <interface> | grep Wake-on
  Wake-on: g
  ```

  Where:

  “g” is the magic packet activity.

3.1.14 Hardware Accelerated 802.1ad VLAN (Q-in-Q Tunneling)

Q-in-Q tunneling allows the user to create a Layer 2 Ethernet connection between two servers. The user can segregate a different VLAN traffic on a link or bundle different VLANs into a single VLAN. Q-in-Q tunneling adds a service VLAN tag before the user’s 802.1Q VLAN tags.

For Q-in-Q support in virtualized environments (SR-IOV), please refer to Section 3.2.4, “Q-in-Q Encapsulation per VF in Linux (VST)”, on page 109.

3.1.14.1 Requirements

- ConnectX-3/ConnectX-3 Pro without RX offloads
- Firmware version 2.34.8240 and up
- Kernel version 3.10 and up
- iproute-3.10.0-13.el7.x86_64 and up

➢ **To enable device support for accelerated 802.1ad VLAN.**

1. Turn on the new ethtool private flag “phv-bit” (disabled by default).

   ```
   $ ethtool --set-priv-flags eth1 phv-bit on
   ```

   Enabling this flag sets the phv_en port capability.

2. Change the interface device features by turning on the ethtool device feature “tx-vlan-stag-hw-insert” (disabled by default).

   ```
   $ ethtool -K eth1 tx-vlan-stag-hw-insert on
   ```

   Once the private flag and the ethtool device feature are set, the device will be ready for 802.1ad VLAN acceleration.

   The “phv-bit” private flag setting is available for the Physical Function (PF) only. The Virtual Function (VF) can use the VLAN acceleration by setting the “tx-vlan-stag-hw-insert” parameter only if the private flag “phv-bit” is enabled by the PF. If the PF enables/disables the “phv-bit” flag after the VF driver is up, the configuration will take place only after the VF driver is restarted.
3.1.15 Ethernet Performance Counters

Supported in ConnectX-3 and ConnectX-3 Pro only.

Counters are used to provide information about how well an operating system, an application, a service, or a driver is performing. The counter data helps determine system bottlenecks and fine-tune the system and application performance. The operating system, network, and devices provide counter data that an application can consume to provide users with a graphical view of how well the system is performing.

The counter index is a QP attribute given in the QP context. Multiple QPs may be associated with the same counter set. If multiple QPs share the same counter its value represents the cumulative total.

- ConnectX®-3 support 127 different counters which allocated:
  - 4 counters reserved for PF - 2 counters for each port
  - 2 counters reserved for VF - 1 counter for each port
  - All other counters if exist are allocated by demand
- RoCE counters are available only through sysfs located under:
  - # /sys/class/infiniband/mlx4_*/ports/*/counters/
  - # /sys/class/infiniband/mlx4_*/ports/*/counters_ext/
- Physical Function can also read Virtual Functions' port counters through sysfs located under:
  - # /sys/class/net/eth*/vf*_statistics/

To display the network device Ethernet statistics, you can run:

```
Ethtool -S <devname>
```

### Table 8 - Ethernet Performance Counters

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rx_packets</td>
<td>Total packets successfully received.</td>
</tr>
<tr>
<td>rx_bytes</td>
<td>Total bytes in successfully received packets.</td>
</tr>
<tr>
<td>rx_multicast_packets</td>
<td>Total multicast packets successfully received.</td>
</tr>
<tr>
<td>rx_broadcast_packets</td>
<td>Total broadcast packets successfully received.</td>
</tr>
<tr>
<td>rx_errors</td>
<td>Number of receive packets that contained errors preventing them from being deliverable to a higher-layer protocol.</td>
</tr>
<tr>
<td>rx_dropped</td>
<td>Number of receive packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.</td>
</tr>
</tbody>
</table>
### Table 8 - Ethernet Performance Counters

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rx_length_errors</td>
<td>Number of received frames that were dropped due to an error in frame length</td>
</tr>
<tr>
<td>rx_over_errors</td>
<td>Number of received frames that were dropped due to hardware port receive buffer overflow</td>
</tr>
<tr>
<td>rx_crc_errors</td>
<td>Number of received frames with a bad CRC that are not runts, jabbers, or alignment errors</td>
</tr>
<tr>
<td>rx_jabbers</td>
<td>Number of received frames with a length greater than MTU octets and a bad CRC</td>
</tr>
<tr>
<td>rx_in_range_length_error</td>
<td>Number of received frames with a length/type field value in the (decimal) range [1500:46] (42 is also counted for VLAN-tagged frames)</td>
</tr>
<tr>
<td>rx_out_range_length_error</td>
<td>Number of received frames with a length/type field value in the (decimal) range [1535:1501]</td>
</tr>
<tr>
<td>tx_packets</td>
<td>Total packets successfully transmitted.</td>
</tr>
<tr>
<td>tx_bytes</td>
<td>Total bytes in successfully transmitted packets.</td>
</tr>
<tr>
<td>tx_multicast_packets</td>
<td>Total multicast packets successfully transmitted.</td>
</tr>
<tr>
<td>tx_broadcast_packets</td>
<td>Total broadcast packets successfully transmitted.</td>
</tr>
<tr>
<td>tx_errors</td>
<td>Number of frames that failed to transmit</td>
</tr>
<tr>
<td>tx_dropped</td>
<td>Number of transmitted frames that were dropped</td>
</tr>
<tr>
<td>rx_prio_&lt;i&gt;_packets</td>
<td>Total packets successfully received with priority i.</td>
</tr>
<tr>
<td>rx_prio_&lt;i&gt;_bytes</td>
<td>Total bytes in successfully received packets with priority i.</td>
</tr>
<tr>
<td>rx_novlan_packets</td>
<td>Total packets successfully received with no VLAN priority.</td>
</tr>
<tr>
<td>rx_novlan_bytes</td>
<td>Total bytes in successfully received packets with no VLAN priority.</td>
</tr>
<tr>
<td>tx_prio_&lt;i&gt;_packets</td>
<td>Total packets successfully transmitted with priority i.</td>
</tr>
<tr>
<td>tx_prio_&lt;i&gt;_bytes</td>
<td>Total bytes in successfully transmitted packets with priority i.</td>
</tr>
<tr>
<td>tx_novlan_packets</td>
<td>Total packets successfully transmitted with no VLAN priority.</td>
</tr>
<tr>
<td>tx_novlan_bytes</td>
<td>Total bytes in successfully transmitted packets with no VLAN priority.</td>
</tr>
<tr>
<td>rx_pause^1</td>
<td>The total number of PAUSE frames received from the far-end port.</td>
</tr>
<tr>
<td>rx_pause_duration^1</td>
<td>The total time in microseconds that far-end port was requested to pause transmission of packets.</td>
</tr>
<tr>
<td>rx_pause_transition^1</td>
<td>The number of receiver transitions from XON state (paused) to XOFF state (non-paused)</td>
</tr>
</tbody>
</table>
### Table 8 - Ethernet Performance Counters

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tx_pause</td>
<td>The total number of PAUSE frames sent to the far-end port</td>
</tr>
<tr>
<td>tx_pause_duration</td>
<td>The total time in microseconds that transmission of packets has been paused</td>
</tr>
<tr>
<td>tx_pause_transition</td>
<td>The number of transmitter transitions from XON state (paused) to XOFF state (non-paused)</td>
</tr>
<tr>
<td>vport_rx_unicast_packets</td>
<td>Unicast packets received successfully</td>
</tr>
<tr>
<td>vport_rx_unicast_bytes</td>
<td>Unicast packet bytes received successfully</td>
</tr>
<tr>
<td>vport_rx_multicast_packets</td>
<td>Multicast packets received successfully</td>
</tr>
<tr>
<td>vport_rx_multicast_bytes</td>
<td>Multicast packet bytes received successfully</td>
</tr>
<tr>
<td>vport_rx_broadcast_packets</td>
<td>Broadcast packets received successfully</td>
</tr>
<tr>
<td>vport_rx_broadcast_bytes</td>
<td>Broadcast packet bytes received successfully</td>
</tr>
<tr>
<td>vport_rx_dropped</td>
<td>Received packets discarded due to lack of software receive buffers (WQEs). Important indication to weather RX completion routines are keeping up with hardware ingress packet rate</td>
</tr>
<tr>
<td>vport_rx_filtered</td>
<td>Received packets dropped due to packet check that failed. For example: Incorrect VLAN, incorrect Ethertype, unavailable queue/QP or loopback prevention</td>
</tr>
<tr>
<td>vport_tx_unicast_packets</td>
<td>Unicast packets sent successfully</td>
</tr>
<tr>
<td>vport_tx_unicast_bytes</td>
<td>Unicast packet bytes sent successfully</td>
</tr>
<tr>
<td>vport_tx_multicast_packets</td>
<td>Multicast packets sent successfully</td>
</tr>
<tr>
<td>vport_tx_multicast_bytes</td>
<td>Multicast packet bytes sent successfully</td>
</tr>
<tr>
<td>vport_tx_broadcast_packets</td>
<td>Broadcast packets sent successfully</td>
</tr>
<tr>
<td>vport_tx_broadcast_bytes</td>
<td>Broadcast packet bytes sent successfully</td>
</tr>
<tr>
<td>vport_tx_dropped</td>
<td>Packets dropped due to transmit errors</td>
</tr>
<tr>
<td>rx_lro_aggregated</td>
<td>Number of packets processed by the LRO mechanism</td>
</tr>
<tr>
<td>rx_lro_flushed</td>
<td>Number of offloaded packets the LRO mechanism passed to kernel</td>
</tr>
<tr>
<td>rx_lro_no_desc</td>
<td>LRO mechanism has no room to receive packets from the adapter. In normal condition, it should not increase</td>
</tr>
<tr>
<td>rx_alloc_failed</td>
<td>Number of times failed preparing receive descriptor</td>
</tr>
<tr>
<td>rx_csum_good</td>
<td>Number of packets received with good checksum</td>
</tr>
<tr>
<td>rx_csum_none</td>
<td>Number of packets received with no checksum indication</td>
</tr>
<tr>
<td>tx_chksum_offload</td>
<td>Number of packets transmitted with checksum offload</td>
</tr>
<tr>
<td>tx_queue_stopped</td>
<td>Number of times transmit queue queue suspended</td>
</tr>
</tbody>
</table>
3.1.16 NVM Express over Fabrics (NVME-oF)

3.1.16.1 NVME-oF

NVME-oF enables NVMe message-based commands to transfer data between a host computer and a target solid-state storage device or system over a network such as Ethernet, Fibre Channel, and InfiniBand. Tunneling NVMe commands through an RDMA fabric provides a high throughput and low latency.

For information on how to configure NVME-oF, please refer to the HowTo Configure NVMe over Fabrics Community post.

3.1.16.2 NVME-oF Target Offload

This feature is only supported for ConnectX-5 adapter cards family and above.

NVME-oF Target Offload is an implementation of the new NVME-oF standard Target (server) side in hardware. Starting from ConnectX-5 family cards, all regular IO requests can be processed by the HCA, with the HCA sending IO requests directly to a real NVMe PCI device, using...
peer-to-peer PCI communications. This means that excluding connection management and error flows, no CPU utilization will be observed during NVME-oF traffic.

- For instructions on how to configure NVME-oF target offload, refer to [HowTo Configure NVME-oF Target Offload](#) Community post.
- For instructions on how to verify that NVME-oF target offload is working properly, refer to [Simple NVMe-oF Target Offload Benchmark](#) Community post.

### 3.2 Virtualization

#### 3.2.1 Single Root IO Virtualization (SR-IOV)

Single Root IO Virtualization (SR-IOV) is a technology that allows a physical PCIe device to present itself multiple times through the PCIe bus. This technology enables multiple virtual instances of the device with separate resources. Mellanox adapters are capable of exposing in ConnectX®-3 adapter cards up to 126 virtual instances called Virtual Functions (VFs) and ConnectX-4/Connect-IB adapter cards up to 62 virtual instances. These virtual functions can then be provisioned separately. Each VF can be seen as an additional device connected to the Physical Function. It shares the same resources with the Physical Function, and its number of ports equals those of the Physical Function.

SR-IOV is commonly used in conjunction with an SR-IOV enabled hypervisor to provide virtual machines direct hardware access to network resources hence increasing its performance.

In this chapter we will demonstrate setup and configuration of SR-IOV in a Red Hat Linux environment using Mellanox ConnectX® VPI adapter cards family.

#### 3.2.1.1 System Requirements

To set up an SR-IOV environment, the following is required:

- MLNX_EN Driver
- A server/blade with an SR-IOV-capable motherboard BIOS
- Hypervisor that supports SR-IOV such as: Red Hat Enterprise Linux Server Version 6.*
- Mellanox ConnectX® VPI Adapter Card family with SR-IOV capability

#### 3.2.1.2 Setting Up SR-IOV

Depending on your system, perform the steps below to set up your BIOS. The figures used in this section are for illustration purposes only. For further information, please refer to the appropriate BIOS User Manual:
Step 1. Enable "SR-IOV" in the system BIOS.

![BIOS SETUP UTILITY](image1)

Step 2. Enable “Intel Virtualization Technology”.

![BIOS SETUP UTILITY](image2)

Step 3. Install a hypervisor that supports SR-IOV.

Step 4. Depending on your system, update the /boot/grub/grub.conf file to include a similar command line load parameter for the Linux kernel.
For example, to Intel systems, add:

```ini
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-36.x86-645)
   root (hd0,0)
   kernel /vmlinuz-2.6.32-36.x86-64 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
   intel_iommu=on
   initrd /initrd-2.6.32-36.x86-64.img
```

1. Please make sure the parameter "intel_iommu=on" exists when updating the /boot/grub/grub.conf file, otherwise SR-IOV cannot be loaded. Some OSs use /boot/grub2/grub.cfg file. If your server uses such file, please edit this file instead. (add "intel_iommu=on" for the relevant menu entry at the end of the line that starts with "linux16").

### 3.2.1.2.1 Configuring SR-IOV for ConnectX-3/ConnectX-3 Pro

**Step 1.** Install the MLNX_EN driver for Linux that supports SR-IOV.

SR-IOV can be enabled and managed by using one of the following methods:

- Run the mlxconfig tool and set the `SRIOV_EN` parameter to "1" without re-burning the firmware

  To find the mst device run: “mst start” and “mst status”

  ```bash
  mlxconfig -d <mst_device> s SRIOV_EN=1
  # mstflint -dev <PCI Device> dc
  ```

  For further information, please refer to section “mlxconfig - Changing Device Configuration Tool” in the MFT User Manual (www.mellanox.com > Products > Software > Firmware Tools).

**Step 2.** Verify the HCA is configured to support SR-IOV.

```bash
# mstflint -dev <PCI Device> dc
```

1. Verify in the [HCA] section the following fields appear:\n
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Recommended Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>num_pfs</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note:** This field is optional and might not always appear.

---

1. If SR-IOV is supported, to enable SR-IOV (if it is not enabled), it is sufficient to set “sriov_en = true” in the INI.
2. If the HCA does not support SR-IOV, please contact Mellanox Support: support@mellanox.com
2. Add the above fields to the INI if they are missing.
3. Set the `total_vfs` parameter to the desired number if you need to change the number of total VFs.
4. Reburn the firmware using the mlxburn tool if the fields above were added to the INI, or the `total_vfs` parameter was modified.

If the mlxburn is not installed, please download it from the Mellanox website http://www.mellanox.com > products > Firmware tools

```
mlxburn -fw ./fw-ConnectX3-rel.mlx -dev /dev/mst/mt4099_pci_cr0 -conf ./MCX341A-XCG_Ax.ini
```

### Step 3.
Create the text file `/etc/modprobe.d/mlx4_core.conf` if it does not exist.

### Step 4.
Insert an "options" line in the `/etc/modprobe.d/mlx4_core.conf` file to set the number of VFs. The protocol type per port, and the allowed number of virtual functions to be used by the physical function driver (probe_vf).

For example:

```
options mlx4_core num_vfs=5 port_type_array=1,2 probe_vf=1
```

### Parameter | Recommended Value
--- | ---
`total_vfs` | • When using firmware version 2.31.5000 and above, the recommended value is 126.
• When using firmware version 2.30.8000 and below, the recommended value is 63

**Note:** Before setting number of VFs in SR-IOV, please make sure your system can support that amount of VFs. Setting number of VFs larger than what your Hardware and Software can support may cause your system to cease working.

### Parameter | Recommended Value
--- | ---
`sriov_en` | `true`

### Parameter | Recommended Value
--- | ---
`num_vfs` | • If absent, or zero: no VFs will be available
• If its value is a single number in the range of 0-63: The driver will enable the `num_vfs` VFs on the HCA and this will be applied to all ConnectX® HCAs on the host.
• If its a triplet x,y,z (applies only if all ports are configured as Ethernet) the driver creates:
  • `x` single port VFs on physical port 1
  • `y` single port VFs on physical port 2 (applies only if such a port exist)
  • `z` n-port VFs (where n is the number of physical ports on device).
This applies to all ConnectX® HCAs on the host
### num_vfs
- If its format is a string: The string specifies the `num_vfs` parameter separately per installed HCA. The string format is: "bb:dd.f-v,bb:dd.f-v,..."
  - `bb:dd.f = bus:device.function` of the PF of the HCA
  - `v = number of VFs to enable for that HCA which is either a single value or a triplet, as described above.

For example:
- `num_vfs=5` - The driver will enable 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host
- `num_vfs=00:04.0-5,00:07.0-8` - The driver will enable 5 VFs on the HCA positioned in BDF 00:04.0 and 8 on the one in 00:07.0
- `num_vfs=1,2,3` - The driver will enable 1 VF on physical port 1, 2 VFs on physical port 2 and 3 dual port VFs (applies only to dual port HCA when all ports are Ethernet ports).
- `num_vfs=00:04.0-5;6;7,00:07.0-8;9;10` - The driver will enable:
  - HCA positioned in BDF 00:04.0
    - 5 single VFs on port 1
    - 6 single VFs on port 2
    - 7 dual port VFs
  - HCA positioned in BDF 00:07.0
    - 8 single VFs on port 1
    - 9 single VFs on port 2
    - 10 dual port VFs

Notes: 
- PFs not included in the above list will not have SR-IOV enabled.
- Triplets and single port VFs are only valid when all ports are configured as Ethernet. When an InfiniBand port exists, only `num_vfs=a` syntax is valid where “a” is a single value that represents the number of VFs.
- The second parameter in a triplet is valid only when there are more than 1 physical port.
In a triplet, `x+z<=63` and `y+z<=63`, the maximum number of VFs on each physical port must be 63.

### port_type_array
Specifies the protocol type of the ports. It is either one array of 2 port types 't1,t2' for all devices or list of BDF to `port_type_array` 'bb:dd.f-t1;t2,...'. (string)

Valid port types: 1-ib, 2-eth, 3-auto, 4-N/A
If only a single port is available, use the N/A port type for port 2 (e.g '1,4').
Note that this parameter is valid only when `num_vfs` is not zero (i.e., SRIOV is enabled). Otherwise, it is ignored.
Features Overview and Configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Recommended Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>probe_vf</td>
<td></td>
</tr>
</tbody>
</table>
| • If absent or zero: no VF interfaces will be loaded in the Hypervisor/host  
  • If num_vfs is a number in the range of 1-63, the driver running on the Hypervisor will itself activate that number of VFs. All these VFs will run on the Hypervisor. This number will apply to all ConnectX® HCAs on that host.  
  • If its a triplet x,y,z (applies only if all ports are configured as Ethernet), the driver probes:  
    • x single port VFs on physical port 1  
    • y single port VFs on physical port 2 (applies only if such a port exist)  
    • z n-port VFs (where n is the number of physical ports on device). Those VFs are attached to the hypervisor.  
  • If its format is a string: the string specifies the probe_vf parameter separately per installed HCA. The string format is: "bb:dd.f-v,bb:dd.f-v,..."  
    • bb:dd.f = bus:device.function of the PF of the HCA  
    • v = number of VFs to use in the PF driver for that HCA which is either a single value or a triplet, as described above  
  For example:  
  • probe_vfs=5 - The PF driver will activate 5 VFs on the HCA and this will be applied to all ConnectX® HCAs on the host  
  • probe_vfs=00:04.0-5,00:07.0-8 - The PF driver will activate 5 VFs on the HCA positioned in BDF 00:04.0 and 8 for the one in 00:07.0)  
  • probe_vf=1,2,3 - The PF driver will activate 1 VF on physical port 1, 2 VFIs on physical port 2 and 3 dual port VFs (applies only to dual port HCA when all ports are Ethernet ports). This applies to all ConnectX® HCAs in the host.  
  • probe_vf=00:04.0-5;6;7,00:07.0-8;9;10 - The PF driver will activate:  
    • HCA positioned in BDF 00:04.0  
      • 5 single VFs on port 1  
      • 6 single VFs on port 2  
      • 7 dual port VFs  
    • HCA positioned in BDF 00:07.0  
      • 8 single VFs on port 1  
      • 9 single VFs on port 2  
      • 10 dual port VFs  
  Applies when all ports are configure as Ethernet in dual port HCAs. |
The example above loads the driver with 5 VFs (num_vfs). The standard use of a VF is a single VF per a single VM. However, the number of VFs varies upon the working mode requirements. The protocol types are:

- Port 1 = IB
- Port 2 = Ethernet
- port_type_array=2,2  (Ethernet, Ethernet)
- port_type_array=1,1  (IB, IB)
- port_type_array=1,2  (VPI: IB, Ethernet)
- NO port_type_array module parameter: ports are IB

For single port HCAs the possible values are (1,1) or (2,2).

**Step 5.** Reboot the server.

**Step 6.** Load the driver and verify the SR-IOV is supported. Run:

```
lspci | grep Mellanox
03:00.0 InfiniBand: Mellanox Technologies MT26428 [ConnectX VPI PCIe 2.0 5GT/s - IB QDR / 10GigE] (rev b0)
03:00.1 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
03:00.2 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
03:00.3 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
03:00.4 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
03:00.5 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
```

Where:

- “03:00” represents the Physical Function
3.2.1.2 Configuring SR-IOV for ConnectX-4/ConnectX-5 (Ethernet)

To set SR-IOV in Ethernet mode, refer to HowTo Configure SR-IOV for ConnectX-4/ConnectX-5 with KVM (Ethernet) Community Post.

3.2.1.3 Additional SR-IOV Configurations

3.2.1.3.1 Assigning a Virtual Function to a Virtual Machine

This section describes a mechanism for adding a SR-IOV VF to a Virtual Machine.

3.2.1.3.1.1 Assigning the SR-IOV Virtual Function to the Red Hat KVM VM Server

Step 1. Run the virt-manager.

Step 2. Double click on the virtual machine and open its Properties.

Step 3. Go to Details->Add hardware ->PCI host device.

Step 4. Choose a Mellanox virtual function according to its PCI device (e.g., 00:03.1)

Step 5. If the Virtual Machine is up reboot it, otherwise start it.

Step 6. Log into the virtual machine and verify that it recognizes the Mellanox card. Run:

```
lspci | grep Mellanox
```

Example:

```
lspci | grep Mellanox

00:03.0 InfiniBand: Mellanox Technologies MT27500 Family [ConnectX-3 Virtual Function] (rev b0)
```
Step 7. [ConnectX-3/ConnectX-3 Pro] Add the device to the /etc/sysconfig/network-scripts/ifcfg-ethX configuration file. The MAC address for every virtual function is configured randomly, therefore it is not necessary to add it.

3.2.1.3.2 Ethernet Virtual Function Configuration when Running SR-IOV

SR-IOV Virtual function configuration can be done through Hypervisor iproute2/netlink tool, if present. Otherwise, it can be done via sysfs.

```
ip link set { dev DEVICE | group DEVGROUP } [ { up | down } ]
...
[ vf NUM [ mac LLADDR ] [ vlan VLANID [ qos VLAN-QOS ] ]
...
[ spoofchk { on | off } ] ]
...
sysfs configuration (ConnectX-4):
/sys/class/net/enp8s0f0/device/sriov/[VF]
```

```
++- [VF]
 | ++- config
 | ++- link_state
 | ++- mac
 | ++- mac_list
 | ++- max_tx_rate
 | ++- min_tx_rate
 | ++- spoofcheck
 | ++- stats
 | ++- trunk
 | ++- trust
 | ++- vlan
```

3.2.1.3.2.1 VLAN Guest Tagging (VGT) and VLAN Switch Tagging (VST)

When running ETH ports on VGT, the ports may be configured to simply pass through packets as is from VFs (VLAN Guest Tagging), or the administrator may configure the Hypervisor to silently force packets to be associated with a VLAN/Qos (VLAN Switch Tagging).

In the latter case, untagged or priority-tagged outgoing packets from the guest will have the VLAN tag inserted, and incoming packets will have the VLAN tag removed.

The default behavior is VGT.

To configure VF VST mode, run:

```
ip link set dev <PF device> vf <NUM> vlan <vlan_id> [qos <qos>]
```

where:

- **NUM** = 0..max-vf-num
- **vlan_id** = 0..4095
- **qos** = 0..7
For example:

- `ip link set dev eth2 vf 2 vlan 10 qos 3` - sets VST mode for VF #2 belonging to PF eth2, with vlan_id = 10 and qos = 3
- `ip link set dev eth2 vf 2 vlan 0` - sets mode for VF 2 back to VGT

*Note:* In ConnectX-3 adapter cards family, switching to VGT mode can also be done by setting vlan_id to 4095.

### 3.2.1.3.2 Additional Ethernet VF Configuration Options

#### Guest MAC configuration

By default, guest MAC addresses are configured to be all zeroes. If the administrator wishes the guest to always start up with the same MAC, he/she should configure guest MACs before the guest driver comes up.

The guest MAC may be configured by using:

```bash
ip link set dev <PF device> vf <NUM> mac <LLADDR>
```

For legacy and ConnectX-4 guests, which do not generate random MACs, the administrator should always configure their MAC addresses via IP link, as above.

#### Spoof checking

Spoof checking is currently available only on upstream kernels newer than 3.1.

```bash
ip link set dev <PF device> vf <NUM> spoofchk [on | off]
```

#### Guest Link State

```bash
ip link set dev <PF device> vf <UM> state [enable| disable| auto]
```

### 3.2.1.3.3 Virtual Function Statistics

Virtual function statistics can be queried via sysfs:

```bash
cat /sys/class/net/enp8s0f0/device/sriov/1/stats
tx_packets : 0
tx_bytes : 0
tx_dropped : 0
rx_packets : 0
rx_bytes : 0
rx_broadcast : 0
rx_multicast : 0
rx_dropped : 0
```

### 3.2.1.3.4 Mapping VFs to Ports

- To view the VFs mapping to ports:

  Use the `ip link` tool v2.6.34~3 and above.
When a MAC is ff:ff:ff:ff:ff:ff, the VF is not assigned to the port of the net device it is listed under. In the example above, `vf 38` is not assigned to the same port as `p1p1`, in contrast to `vf0`.

However, even VFs that are not assigned to the net device, could be used to set and change its settings. For example, the following is a valid command to change the spoof check:

```
ip link set dev p1p1 vf 38 spoofchk on
```

This command will affect only the `vf 38`. The changes can be seen in `ip link` on the net device that this device is assigned to.

### 3.2.1.3.2.5 Mapping VFs to Ports using the `mlnx_get_vfs.pl` tool

- **To map the PCI representation in BDF to the respective ports:**

  ```
  mlnx_get_vfs.pl
  ``

  The output is as following:

<table>
<thead>
<tr>
<th>BDF 0000:04:00.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1: 2</td>
</tr>
<tr>
<td>vf0 0000:04:00.1</td>
</tr>
<tr>
<td>vf1 0000:04:00.2</td>
</tr>
<tr>
<td>Port 2: 2</td>
</tr>
<tr>
<td>vf2 0000:04:00.3</td>
</tr>
<tr>
<td>vf3 0000:04:00.4</td>
</tr>
<tr>
<td>Both: 1</td>
</tr>
<tr>
<td>vf4 0000:04:00.5</td>
</tr>
</tbody>
</table>

### 3.2.1.3 Configuring Pkeys and GUIDs under SR-IOV in ConnectX-3/ConnectX-3 Pro

### 3.2.1.3.3 Port Type Management

Port Type management is static when enabling SR-IOV (the `connectx_port_config` script will not work). The port type is set on the Host via a module parameter, `port_type_array`, in `mlx4_core`. This parameter may be used to set the port type uniformly for all installed ConnectX® HCAs, or it may specify an individual configuration for each HCA.

This parameter should be specified as an options line in the file `/etc/modprobe.d/mlx4_core.conf.`

For example, to configure all HCAs to have Port1 as IB and Port2 as ETH, insert the following line:

```
options mlx4_core port_type_array=1,2
```
To set HCAs individually, you may use a string of `Domain:bus:device.function=x;y` For example, if you have a pair of HCAs, whose PFs are 0000:04:00.0 and 0000:05:00.0, you may specify that the first will have both ports as IB, and the second will have both ports as ETH as follows:

```text
options mlx4_core port_type_array='0000:04:00.0-1;1,0000:05:00.0-2;2
```

Only the PFs are set via this mechanism. The VFs inherit their port types from their associated PF.

### 3.2.1.3.3.2 Virtual Function InfiniBand Ports

Each VF presents itself as an independent vHCA to the host, while a single HCA is observable by the network which is unaware of the vHCAs. No changes are required by the InfiniBand sub-system, ULPs, and applications to support SR-IOV, and vHCAs are interoperable with any existing (non-virtualized) IB deployments.

Sharing the same physical port(s) among multiple vHCAs is achieved as follows:

- Each vHCA port presents its own virtual GID table
  - For further details, please refer to Section 3.2.1.3.5, on page 94.
- Each vHCA port presents its own virtual PKey table
  - The virtual PKey table (presented to a VF) is a mapping of selected indexes of the physical PKey table. The host admin can control which PKey indexes are mapped to which virtual indexes using a sysfs interface. The physical PKey table may contain both full and partial memberships of the same PKey to allow different membership types in different virtual tables.
- Each vHCA port has its own virtual port state
  - A vHCA port is up if the following conditions apply:
    - The physical port is up
    - The virtual GID table contains the GIDs requested by the host admin
    - The SM has acknowledged the requested GIDs since the last time that the physical port went up
  - Other port attributes are shared, such as: GID prefix, LID, SM LID, LMC mask

To allow the host admin to control the virtual GID and PKey tables of vHCAs, a new sysfs ‘iov sub-tree has been added under the PF InfiniBand device.

If the vHCA comes up without a GUID, make sure you are running the latest version of SM/OpenSM. The SM on QDR switches do not support SR-IOV.
3.2.1.3.3SR-IOV sysfs Administration Interfaces on the Hypervisor

Administration of GUIDs and PKeys is done via the sysfs interface in the Hypervisor (Dom0). This interface is under:

/sys/class/infiniband/<infiniband device>/iov

Under this directory, the following subdirectories can be found:

- **ports** - The actual (physical) port resource tables
  - Port GID tables:
    - ports/<n>/gids/<n> where 0 <= n <= 127 (the physical port gids)
    - ports/<n>/admin_guids/<n> where 0 <= n <= 127 (allows examining or changing the administrative state of a given GUID)
    - ports/<n>/pkeys/<n> where 0 <= n <= 126 (displays the contents of the physical pkey table)
  - <pci id> directories - one for Dom0 and one per guest. Here, you may see the mapping between virtual and physical pkey indices, and the virtual to physical gid 0. Currently, the GID mapping cannot be modified, but the pkey virtual to physical mapping can.

These directories have the structure:

- <pci_id>/port/<m>/gid_idx/0 where m = 1..2 (this is read-only)
  - and
- <pci_id>/port/<m>/pkey_idx/<n>, where m = 1..2 and n = 0..126

For instructions on configuring pkey_idx, please see below.

3.2.1.3.3.4 Configuring an Alias GUID (under ports/<n>/admin_guids)

**Step 1.** Determine the GUID index of the PCI Virtual Function that you want to pass through to a guest.

For example, if you want to pass through PCI function 02:00.3 to a certain guest, you initially need to see which GUID index is used for this function.

To do so:

```bash
cat /sys/class/infiniband/mlx4_0/iov/0000:02:00.3/port/<port_num>/gid_idx/0
```

The value returned will present which guid index to modify on Dom0.

**Step 2.** Modify the physical GUID table via the admin_guids sysfs interface.

To configure the GUID at index <n> on port <port_num>:

```bash
echo NEWGUID > /sys/class/infiniband/mlx4_0/iov/ports/<port_num>/admin_guids/<guid_index>
```

Example:

```bash
echo "0x00000000000000000000000000000000" > /sys/class/infiniband/mlx4_0/iov/ports/1/admin_guids/3
```

**Note:**
Features Overview and Configuration

/\sys/class/infiniband/mlx4_0/iov/ports/<port_num>/admin_guids/0 is read only and cannot be changed.

Step 3. Read the administrative status of the GUID index.

To read the administrative status of GUID index `<guid_index>` on port number `<port_num>`:

```
cat /sys/class/infiniband/mlx4_0/iov/ports/<port_num>/admin_guids/<guid_index>
```

Step 4. Check the operational state of a GUID.

```
/sys/class/infiniband/mlx4_0/iov/ports/<port_num>/gids (where port_num = 1 or 2)
```

The values indicate what gids are actually configured on the firmware/hardware, and all the entries are R/O.

Step 5. Compare the value you read under the "admin_guids" directory at that index with the value under the "gids" directory, to verify the change requested in Step 3 has been accepted by the SM, and programmed into the hardware port GID table.

If the value under `admin_guids/<m>` is different that the value under `gids/<m>`, the request is still in progress.

3.2.1.3.3.5 Alias GUID Support in InfiniBand

Admin VF GUIDs

As of MLNX_EN v3.0, the `query_gid` verb (e.g. `ib_query_gid()`) returns the admin desired value instead of the value that was approved by the SM to prevent a case where the SM is unreachable or a response is delayed, or if the VF is probed into a VM before their GUID is registered with the SM. If one of the above scenarios occurs, the VF sees an incorrect GID (i.e., not the GID that was intended by the admin).

Despite the new behavior, if the SM does not approve the GID, the VF sees its link as down.

On Demand GUIDs

GIDs are requested from the SM on demand, when needed by the VF (e.g. become active), and are released when the GIDs are no longer in use.

Since a GID is assigned to a VF on the destination HCA, while the VF on the source HCA is shut down (but not administratively released), using GIDs on demand eases the GID migrations.

For compatibility reasons, an explicit admin request to set/change a GUID entry is done immediately, regardless of whether the VF is active or not to allow administrators to change the GUID without the need to unbind/bind the VF.

Alias GUIDs Default Mode

Due to the change in the Alias GUID support in InfiniBand behavior, its default mode is now set as HOST assigned instead of SM assigned. To enable out-of-the-box experience, the PF generates random GUIDs as the initial admin values instead of asking the SM.
Initial GUIDs' Values

Initial GUIDs' values depend on the mlx4_ib module parameter 'sm_guid_assign' as follows:

<table>
<thead>
<tr>
<th>Mode Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin assigned</td>
<td>Each admin_guid entry has the random generated GUID value.</td>
</tr>
<tr>
<td>sm assigned</td>
<td>Each admin_guid entry for non-active VFs has a value of 0. Meaning, asking a GUID from the SM upon VF activation. When a VF is active, the returned value from the SM becomes the admin value to be asked later again.</td>
</tr>
</tbody>
</table>

When a VF becomes active, and its admin value is approved, the operational GID entry is changed accordingly. In both modes, the administrator can set/delete the value by using the sysfs Administration Interfaces on the Hypervisor as described above.

Single GUID per VF

Each VF has a single GUID entry in the table based on the VF number. (e.g. VF 1 expects to use GID entry 1). To determine the GUID index of the PCI Virtual Function to pass to a guest, use the sysfs mechanism <gid_idx> directory as described above.

Persistency Support

Once admin request is rejected by the SM, a retry mechanism is set. Retry time is set to 1 second, and for each retry it is multiplied by 2 until reaching the maximum value of 60 seconds. Additionally, when looking for the next record to be updated, the record having the lowest time to be executed is chosen.

Any value reset via the admin_guid interface is immediately executed and it resets the entry’s timer.

Partitioning IPoIB Communication using PKeys

PKeys are used to partition IPoIB communication between the Virtual Machines and the Dom0 by mapping a non-default full-membership PKey to virtual index 0, and mapping the default PKey to a virtual pkey index other than zero.

The below describes how to set up two hosts, each with 2 Virtual Machines. Host-1/vm-1 will be able to communicate via IPoIB only with Host2/vm1, and Host1/vm2 only with Host2/vm2.

In addition, Host1/Dom0 will be able to communicate only with Host2/Dom0 over ib0. vm1 and vm2 will not be able to communicate with each other, nor with Dom0.

This is done by configuring the virtual-to-physical PKey mappings for all the VMs, such that at virtual PKey index 0, both vm-1s will have the same pkey and both vm-2s will have the same PKey (different from the vm-1’s), and the Dom0's will have the default pkey (different from the vm's pkeys at index 0).

OpenSM must be used to configure the physical Pkey tables on both hosts.

- The physical Pkey table on both hosts (Dom0) will be configured by OpenSM to be:

```
index 0 = 0xffff
index 1 = 0xb000
index 2 = 0xb030
```
• The vm1's virt-to-physical PKey mapping will be:

<table>
<thead>
<tr>
<th>pkey_idx</th>
<th>pkey_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0xFFFF</td>
</tr>
<tr>
<td>1</td>
<td>0xB000</td>
</tr>
<tr>
<td>2</td>
<td>0xB030</td>
</tr>
</tbody>
</table>

• The vm2's virt-to-phys pkey mapping will be:

<table>
<thead>
<tr>
<th>pkey_idx</th>
<th>pkey_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x3000</td>
</tr>
<tr>
<td>1</td>
<td>0x3030</td>
</tr>
</tbody>
</table>

so that the default pkey will reside on the vms at index 1 instead of at index 0.

The IPoIB QPs are created to use the PKey at index 0. As a result, the Dom0, vm1 and vm2 IPoIB QPs will all use different PKeys.

To partition IPoIB communication using PKeys:

Step 1. Create a file "/etc/opensm/partitions.conf" on the host on which OpenSM runs, containing lines.

```
Default=0x7fff,ipoib : ALL=full ;
Pkey1=0x3000,ipoib : ALL=full;
Pkey3=0x3030,ipoib : ALL=full;
```

This will cause OpenSM to configure the physical Port Pkey tables on all physical ports on the network as follows:

<table>
<thead>
<tr>
<th>pkey_idx</th>
<th>pkey_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0xFFFF</td>
</tr>
<tr>
<td>1</td>
<td>0xB000</td>
</tr>
<tr>
<td>2</td>
<td>0xB030</td>
</tr>
</tbody>
</table>

(the most significant bit indicates if a PKey is a full PKey).

The ",,ipoib" causes OpenSM to pre-create IPoIB the broadcast group for the indicated PKeys.

Step 2. Configure (on Dom0) the virtual-to-physical PKey mappings for the VMs.

Step a. Check the PCI ID for the Physical Function and the Virtual Functions.

```
lspci | grep Mellanox Technologies
```

Step b. Assuming that on Host1, the physical function displayed by lspci is "0000:02:00.0", and that on Host2 it is "0000:03:00.0"

On Host1 do the following.

```
lsdev | grep mlx4
```

1. 0000:02:00.0 contains the virtual-to-physical mapping tables for the physical function.
   0000:02:00.X contain the virt-to-phys mapping tables for the virtual functions.

Do not touch the Dom0 mapping table (under <nnnn>:<nn>:00.0). Modify only tables under 0000:02:00.1 and/or 0000:02:00.2. We assume that vm1 uses VF 0000:02:00.1 and vm2 uses VF 0000:02:00.2
Step c. Configure the virtual-to-physical PKey mapping for the VMs.

    echo 0 > 0000:02:00.1/ports/1/pkey_idx/1
    echo 1 > 0000:02:00.1/ports/1/pkey_idx/0
    echo 0 > 0000:02:00.2/ports/1/pkey_idx/1
    echo 2 > 0000:02:00.2/ports/1/pkey_idx/0

vm1 pkey index 0 will be mapped to physical pkey-index 1, and vm2 pkey index 0 will be mapped to physical pkey index 2. Both vm1 and vm2 will have their pkey index 1 mapped to the default pkey.

Step d. On Host2 do the following.

    cd /sys/class/infiniband/mlx4_0/iov
    echo 0 > 0000:03:00.1/ports/1/pkey_idx/1
    echo 1 > 0000:03:00.1/ports/1/pkey_idx/0
    echo 0 > 0000:03:00.2/ports/1/pkey_idx/1
    echo 2 > 0000:03:00.2/ports/1/pkey_idx/0

Step e. Once the VMs are running, you can check the VM's virtualized PKey table by doing (on the vm).

    cat /sys/class/infiniband/mlx4_0/ports/[1,2]/pkeys/[0,1]

Step 3. Start up the VMs (and bind VFs to them).

Step 4. Configure IP addresses for ib0 on the host and on the guests.

3.2.1.3.4 Running Network Diagnostic Tools on a Virtual Function in ConnectX-3/ConnectX-3 Pro

Until now, in MLNX_EN, administrators were unable to run network diagnostics from a VF since sending and receiving Subnet Management Packets (SMPs) from a VF was not allowed, for security reasons: SMPs are not restricted by network partitioning and may affect the physical network topology. Moreover, even the SM may be denied access from portions of the network by setting management keys unknown to the SM.

However, it is desirable to grant SMP capability to certain privileged VFs, so certain network management activities may be conducted within virtual machines rather than only on the hypervisor.

3.2.1.3.4.1 Granting SMP Capability to a Virtual Function

To enable SMP capability for a VF, one must enable the Subnet Management Interface (SMI) for that VF. By default, the SMI interface is disabled for VFs. To enable SMI mads for VFs, there are two new sysfs entries per VF per on the Hypervisor (under /sys/class/infiniband/mlx4_X/iov/<b.d.f>/ports/<1 or 2>). These entries are displayed only for VFs (not for the PF), and only for IB ports (not ETH ports).

The first entry, enable_smi_admin, is used to enable SMI on a VF. By default, the value of this entry is zero (disabled). When set to "1", the SMI will be enabled for the VF on the next rebind or openibd restart on the VM that the VF is bound to. If the VF is currently bound, it must be unbound and then re-bind.

The second sysfs entry, smi_enabled, indicates the current enablement state of the SMI. 0 indicates disabled, and 1 indicates enabled. This entry is read-only.
When a VF is initialized (bound), during the initialization sequence, the driver copies the requested `smi_state (enable_smi_admin)` for that VF/port to the operational SMI state (`smi_enabled`) for that VF/port, and operate according to the operational state.

Thus, the sequence of operations on the hypervisor is:

**Step 1.** Enable SMI for any VF/port that you wish.

**Step 2.** Restart the VM that the VF is bound to (or just run `/etc/init.d/openibd restart` on that VM)

The SMI will be enabled for the VF/port combinations that you set in step 2 above. You will then be able to run network diagnostics from that VF.

### 3.2.1.3.4.2 Installing MLNX_EN with Network Diagnostics on a VM

- **To install MLNX_EN on a VF which will be enabled to run the tools, run the following on the VM:**
  ```
  # mlnx_en_install
  ```

### 3.2.1.3.5 MAC Forwarding DataBase (FDB) Management in ConnectX-3/ConnectX-3 Pro

#### 3.2.1.3.5.1 FDB Status Reporting

FDB also known as Forwarding Information Base (FIB) or the forwarding table, is most commonly used in network bridging, routing, and similar functions to find the proper interface to which the input interface should forward a packet.

In the SR-IOV environment, the Ethernet driver can share the existing 128 MACs (for each port) among the Virtual interfaces (VF) and Physical interfaces (PF) that share the same table as follow:

- Each VF gets 2 granted MACs (which are taken from the general pool of the 128 MACs)
- Each VF/PF can ask for up to 128 MACs on the policy of first-asks first-served (meaning, except for the 2 granted MACs, the other MACs in the pool are free to be asked)

To check if there are free MACs for an interface (PF or VF), run: `/sys/class/net/<ethX>/fdb_det`.

**Example:**

```bash
cat /sys/class/net/eth2/fdb_det
device eth2: max: 112, used: 2, free macs: 110
```

- **To add a new MAC to the interface:**
  ```bash
  echo +<MAC> > /sys/class/net/eth<X>/fdb
  ```

  Once running the command above, the interface (VF/PF) verifies if a free MAC exists. If there is a free MAC, the VF/PF takes it from the global pool and allocates it. If there is no free MAC, an error is returned notifying the user of lack of MACs in the pool.

- **To delete a MAC from the interface:**
  ```bash
  echo -<MAC> > /sys/class/net/eth<X>/fdb
  ```
If `/sys/class/net/eth<X>/fdb` does not exist, use the Bridge tool from the ip-route2 package which includes the tool to manage FDB tables as the kernel supports FDB callbacks:

```
bridge fdb add 00:01:02:03:04:05 permanent self dev p3p1
bridge fdb del 00:01:02:03:04:05 permanent self dev p3p1
bridge fdb show dev p3p1
```

If adding a new MAC from the kernel's NDO function fails due to insufficient MACs in the pool, the following error flow will occur:
- If the interface is a PF, it will automatically enter the promiscuous mode
- If the interface is a VF, it will try to enter the promiscuous mode and since it does not support it, the action will fail and an error will be printed in the kernel’s log

### 3.2.1.3.6 Virtual Guest Tagging (VGT+)

VGT+ is an advanced mode of Virtual Guest Tagging (VGT), in which a VF is allowed to tag its own packets as in VGT, but is still subject to an administrative VLAN trunk policy. The policy determines which VLAN IDs are allowed to be transmitted or received. The policy does not determine the user priority, which is left unchanged.

Packets can be sent in one of the following modes: when the VF is allowed to send/receive untagged and priority tagged traffic and when it is not. No default VLAN is defined for VGT+ port. The send packets are passed to the eSwitch only if they match the set, and the received packets are forwarded to the VF only if they match the set.

In some old OSs, such as SLES11 SP4, any VLAN can be created in the VM, regardless of the VGT+ configuration, but traffic will only pass for the allowed VLANs.

### 3.2.1.3.6.1 Configuring VGT+ for ConnectX-3/ConnectX-3 Pro

The following are the current VGT+ limitations:
- The size of the VLAN set is defined to be up to 10 VLANs including the VLAN 0 that is added for untagged/priority-tagged traffic
- This behavior applies to all VF traffic: plain Ethernet, and all RoCE transports
- VGT+ allowed VLAN sets may be only extended when the VF is online
- An operational VLAN set becomes identical as the administration VLAN set only after a VF reset
- VGT+ is available in DMFS mode only

The default operating mode is VGT:

```
cat /sys/class/net/eth5/vf0/vlan_set
oper:
admin:
```
Both states (operational and administrative) are empty.

If you set the `vlan_set` parameter with more than 10 VLAN IDs, the driver chooses the first 10 VLAN IDs provided and ignores all the rest.

➢ To enable VGT+ mode:

**Step 1.** Set the corresponding port/VF (in the example below port eth5 VF0) list of allowed VLANs.

```bash
echo 0 1 2 3 4 5 6 7 8 9 > /sys/class/net/eth5/vf0/vlan_set
```

Where 0 specifies if untagged/priority tagged traffic is allowed.

Meaning if the below command is run, you will not be able to send/receive untagged traffic.

```bash
echo 1 2 3 4 5 6 7 8 9 10 > /sys/class/net/eth5/vf0/vlan_set
```

**Step 2.** Reboot the relevant VM for changes to take effect.

(or run: `/etc/init.d/openibd restart`)

➢ To disable VGT+ mode:

**Step 1.** Set the VLAN.

```bash
echo > /sys/class/net/eth5/vf0/vlan_set
```

**Step 2.** Reboot the relevant VM for changes to take effect.

(or run: `/etc/init.d/openibd restart`)

➢ To add a VLAN:

In the example below, the following state exist:

```bash
# cat /sys/class/net/eth5/vf0/vlan_set
oper: 0 1 2 3
admin: 0 1 2 3
```

**Step 1.** Make an operational VLAN set identical to the administration VLAN.

```bash
echo 2 3 4 5 6 > /sys/class/net/eth5/vf0/vlan_set
```

The delta will be added to the operational state immediately (4 5 6):

```bash
# cat /sys/class/net/eth5/vf0/vlan_set
oper: 0 1 2 3 4 5 6
admin: 2 3 4 5 6
```

**Step 2.** Reset the VF for changes to take effect.

3.2.1.3.6.2 Configuring VGT+ for ConnectX-4/ConnectX-5

➢ To enable VGT+ mode:

When working in SR-IOV, the default operating mode is VGT.
Set the corresponding port/VF (in the example below port eth5, VF0) range of allowed VLANs.

```bash
echo "<add> <start_vid> <end_vid>" > /sys/class/net/eth5/device/sriov/0/trunk
```

**Examples**

- Adding VLAN ID range (4-15) to trunk:
  ```bash
echo add 4 15 > /sys/class/net/eth5/device/sriov/0/trunk
```

- Adding a single VLAN ID to trunk:
  ```bash
echo add 17 17 > /sys/class/net/eth5/device/sriov/0/trunk
```

**Note:** When VLAN ID = 0, it indicates that untagged and priority-tagged traffics are allowed.

- To disable VGT+ mode, make sure to remove all VLANs.
  ```bash
echo rem 0 4095 > /sys/class/net/eth5/device/sriov/0/trunk
```

- To remove selected VLANs:
  - Remove VLAN ID range (4-15) from trunk:
    ```bash
echo rem 4 15 > /sys/class/net/eth5/device/sriov/0/trunk
```
  - Remove a single VLAN ID from trunk:
    ```bash
echo rem 17 17 > /sys/class/net/eth5/device/sriov/0/trunk
```

### 3.2.1.3.7 Virtualized QoS per VF (Rate Limit per VF) in ConnectX-3/ConnectX-3 Pro

Virtualized QoS per VF, (supported in ConnectX®-3/ConnectX®-3 Pro adapter cards only with firmware v2.33.5100 and above), limits the chosen VFs' throughput rate limitations (Maximum throughput). The granularity of the rate limitation is 1Mbits.

The feature is disabled by default. To enable it, set the “enable_vfs_qos” module parameter to “1” and add it to the “options mlx4_core”. When set, and when feature is supported, it will be shown upon PF driver load time (in DEV_CAP in kernel log: Granular QoS Rate limit per VF support), when mlx4_core module parameter debug_level is set to 1. For further information, please refer to Section 1.2.1.1, “mlx4_core Parameters”, on page 18 - debug_level parameter).

When set, and supported by the firmware, running as SR-IOV Master and Ethernet link, the driver also provides information on the number of total available vPort Priority Pair (VPPs) and how many VPPs are allocated per priority. All the available VPPs will be allocated on priority 0.

```bash
mlx4_core 0000:1b:00.0: Port 1 Available VPPs 63
mlx4_core 0000:1b:00.0: Port 1 UP 0 Allocated 63 VPPs
mlx4_core 0000:1b:00.0: Port 1 UP 1 Allocated 0 VPPs
mlx4_core 0000:1b:00.0: Port 1 UP 2 Allocated 0 VPPs
mlx4_core 0000:1b:00.0: Port 1 UP 3 Allocated 0 VPPs
mlx4_core 0000:1b:00.0: Port 1 UP 4 Allocated 0 VPPs
mlx4_core 0000:1b:00.0: Port 1 UP 5 Allocated 0 VPPs
mlx4_core 0000:1b:00.0: Port 1 UP 6 Allocated 0 VPPs
mlx4_core 0000:1b:00.0: Port 1 UP 7 Allocated 0 VPPs
```
3.2.1.3.7.1 Configuring Rate Limit for VFs

Rate limit can be configured using the iproute2/netlink tool.

Please note, the rate limit configuration will take effect only when the VF is in VST mode configured with priority 0.

`ip link set dev <PF device> vf <NUM> rate <TXRATE>`

where

- **NUM** = 0...<Num of VF>
- **<TXRATE>** in units of 1Mbit/s

The rate limit for VF can be configured:

- While setting it to the VST mode

  `ip link set dev <PF device> vf <NUM> vlan <vlan_id> [qos <qos>] rate <TXRATE>`

- Before the VF enters the VST mode with a supported priority

  In this case, the rate limit value is saved and the rate limit configuration is applied when VF state is changed to VST mode.

To disable rate limit configured for a VF set the VF with rate 0. Once the rate limit is set, you cannot switch to VGT or change VST priority.

To view current rate limit configurations for VFs, use the iproute2 tool.

`ip link show dev <PF device>`

Example:

```
89: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether f4:52:14:5e:be:20 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 2, tx rate 1500 (Mbps), spoof checking off, link-state auto
    vf 1 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 2 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 3 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
/sys/class/net/<eth-x>/<vf-i>/tx_rate
```

3.2.1.3.8 SR-IOV Advanced Security Features

3.2.1.3.8.1 SR-IOV MAC Anti-Spoofing

Normally, MAC addresses are unique identifiers assigned to network interfaces, and they are fixed addresses that cannot be changed. MAC address spoofing is a technique for altering the MAC address to serve different purposes. Some of the cases in which a MAC address is altered can be legal, while others can be illegal and abuse security mechanisms or disguises a possible attacker.
The SR-IOV MAC address anti-spoofing feature, also known as MAC Spoof Check provides protection against malicious VM MAC address forging. If the network administrator assigns a MAC address to a VF (through the hypervisor) and enables spoof check on it, this will limit the end user to send traffic only from the assigned MAC address of that VF.

MAC Anti-Spoofing Configuration

MAC anti-spoofing is disabled by default.

In the configuration example below, the VM is located on VF-0 and has the following MAC address: 11:22:33:44:55:66.

There are two ways to enable or disable MAC anti-spoofing:

1. Using the standard IP link commands - available from Kernel 3.10 and above.
   a. To enable MAC anti-spoofing, run:
      ```
      # ip link set ens785f1 vf 0 spoofchk on
      ```
   b. To disable MAC anti-spoofing, run:
      ```
      # ip link set ens785f1 vf 0 spoofchk off
      ```

2. Specify echo “ON” or “OFF” to the file located under /sys/class/net/<ETH_IF_NAME>/device/sriov/<VF index>/spoofchk.
   a. To enable MAC anti-spoofing, run:
      ```
      # echo "ON" > /sys/class/net/ens785f1/vf/0/spoofchk
      ```
   b. To disable MAC anti-spoofing, run:
      ```
      # echo "OFF" > /sys/class/net/ens785f1/vf/0/spoofchk
      ```

This configuration is non-persistent and does not survive driver restart.

In order for spoof-check enabling/disabling to take effect while the VF is up and running, it is required to perform a driver restart on the guest OS.

3.2.1.3.8.2 Rate Limit and Bandwidth Share Per VF

This feature is at beta level.

This feature enables rate limiting traffic per VF in SR-IOV mode for ConnectX-4/ConnectX-4 Lx/ConnectX-5 adapter cards. For details on how to configure rate limit per VF for ConnectX-4/
ConnectX-5, refer to HowTo Configure Rate Limit per VF for ConnectX-4/ConnectX-5 Community post.

### 3.2.1.3.8.3 Privileged VFs

In case a malicious driver is running over one of the VFs, and in case that VF’s permissions are not restricted, this may open security holes. However, VFs can be marked as trusted and can thus receive an exclusive subset of physical function privileges or permissions. For example, in case of allowing all VFs, rather than specific VFs, to enter a promiscuous mode as a privilege, this will enable malicious users to sniff and monitor the entire physical port for incoming traffic, including traffic targeting other VFs, which is considered a severe security hole.

**Privileged VFs Configuration**

In the configuration example below, the VM is located on VF-0 and has the following MAC address: 11:22:33:44:55:66.

There are two ways to enable or disable trust:

1. **Using the standard IP link commands** - available from Kernel 4.5 and above.
   
   a. To enable trust for a specific VF, run:
      ```
      # ip link set ens785f1 vf 0 trust on
      ```
   
   b. To disable trust for a specific VF, run:
      ```
      # ip link set ens785f1 vf 0 trust off
      ```

2. **Specify echo “ON” or “OFF”** to the file located under `/sys/class/net/<ETH_IF_NAME>/device/sriov/<VF index>/trust`.
   
   a. To enable trust for a specific VF, run:
      ```
      # echo "ON" > /sys/class/net/ens785f1/device/sriov/0/trust
      ```
   
   b. To disable trust for a specific VF, run:
      ```
      # echo "OFF" > /sys/class/net/ens785f1/device/sriov/0/trust
      ```

### 3.2.1.3.8.4 Probed VFs

Probing Virtual Functions (VFs) after SR-IOV is enabled might consume the adapter cards’ resources. Therefore, it is recommended not to enable probing of VFs when no monitoring of the VM is needed.

VF probing can be disabled in two ways, depending on the kernel version installed on your server:

1. If the kernel version installed is v4.12 or above, it is recommended to use the PCI sysfs interface `sriov_drivers_autoprobe`. For more information, see linux-next branch.
2. If the kernel version installed is older than v4.12, it is recommended to use the mlx5_core module parameter `probe_vf` with MLNX_EN v4.1 or above.

**Example:**

```bash
# echo 0 > /sys/module/mlx5_core/parameters/probe_vf
```

For more information on how to probe VFs, see HowTo Configure and Probe VFs on mlx5 Drivers Community post.
3.2.1.3.9 VF Promiscuous Rx Modes

3.2.1.3.9.1 VF Promiscuous Mode

VFs can enter a promiscuous mode that enables receiving the unmatched traffic and all the multicast traffic that reaches the physical port in addition to the traffic originally targeted to the VF. The unmatched traffic is any traffic’s DMAC that does not match any of the VFs’ or PFs’ MAC addresses.

**Note:** Only privileged/trusted VFs can enter the VF promiscuous mode.

- **To set the promiscuous mode on for a VF, run:**
  ```
  # ifconfig eth2 promisc
  ```

- **To exit the promiscuous mode, run:**
  ```
  # ifconfig eth2 –promisc
  ```

3.2.1.3.9.2 VF All-Multi Mode

VFs can enter an all-multi mode that enables receiving all the multicast traffic sent from/to the other functions on the same physical port in addition to the traffic originally targeted to the VF.

**Note:** Only privileged/trusted VFs can enter the all-multi RX mode.

- **To set the all-multi mode on for a VF, run:**
  ```
  ifconfig eth2 allmulti
  ```

- **To exit the all-multi mode, run:**
  ```
  #ifconfig eth2 –allmulti
  ```

3.2.1.4 Uninstalling SR-IOV Driver

- **To uninstall SR-IOV driver, perform the following:**

  **Step 1.** For Hypervisors, detach all the Virtual Functions (VF) from all the Virtual Machines (VM) or stop the Virtual Machines that use the Virtual Functions.

  **Please be aware,** stopping the driver when there are VMs that use the VFs, will cause machine to hang.

  **Step 2.** Run the script below. Please be aware, uninstalling the driver deletes the entire driver's file, but does not unload the driver.

  ```
  [root@swl022 ~]#/usr/sbin/ofed_uninstall.sh
  This program will uninstall all OFED packages on your machine.
  Do you want to continue?[y/N]:y
  Running /usr/sbin/vendor_pre_uninstall.sh
  Removing OFED Software installations
  ```
3.2.2 Enabling Para Virtualization

➢ To enable Para Virtualization:

Please note, the example below works on RHEL6.* or RHEL7.* without a Network Manager.

Step 1. Create a bridge.

```bash
vim /etc/sysconfig/network-scripts/ifcfg-bridge0
DEVICE=bridge0
TYPE=Bridge
IPADDR=12.195.15.1
NETMASK=255.255.0.0
BOOTPROTO=static
ONBOOT=yes
NM_CONTROLLED=no
DELAY=0
DEVICE=eth5
BOOTPROTO=none
STARTMODE=on
HWADDR=00:02:c9:2e:66:52
TYPE=Ethernet
NM_CONTROLLED=no
ONBOOT=yes
BRIDGE=bridge0
```

Step 2. Change the related interface (in the example below bridge0 is created over eth5).


warning: /etc/infiniband/openib.conf saved as /etc/infiniband/openib.conf.rpmsave
Running /tmp/2818-ofed_vendor_post_uninstall.sh

Step 3. Restart the service network.
### 3.2.3 VXLAN Hardware Stateless Offloads

VXLAN technology provides scalability and security challenges solutions. It requires extension of the traditional stateless offloads to avoid performance drop. ConnectX-3 Pro and ConnectX-4 family adapter card offer the following stateless offloads for a VXLAN packet, similar to the ones offered to non-encapsulated packets. VXLAN protocol encapsulates its packets using outer UDP header.

Available hardware stateless offloads:

- Checksum generation (Inner IP and Inner TCP/UDP)
- Checksum validation (Inner IP and Inner TCP/UDP). This will allow the use of GRO (in ConnectX-3 Pro card only) for inner TCP packets.
- TSO support for inner TCP packets
- RSS distribution according to inner packets attributes
- Receive queue selection - inner frames may be steered to specific QPs

VXLAN Hardware Stateless Offloads requires the following prerequisites:

- HCA and their minimum firmware required:
  - ConnectX-3 Pro - Firmware v2.32.5100
  - ConnectX-4 - Firmware v12.14.xxxx
  - ConnectX-4 Lx - Firmware v14.14.xxxx

- Operating Systems:
  - RHEL7, Ubuntu 14.04 or upstream kernel 3.12.10 (or higher)

- ConnectX-3 Pro Supported Features:
  - DMFS enabled
  - A0 static mode disabled

---

**Step 4.** Attach a bridge to VM.

```bash
ifconfig -a
...
eth6   Link encap:Ethernet  HWaddr 52:54:00:E7:77:99
       inet addr:13.195.15.5  Bcast:13.195.255.255  Mask:255.255.0.0
       inet6 addr: fe80::5054:ff:fee7:7799/64 Scope:Link
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:481 errors:0 dropped:0 overruns:0 frame:0
       TX packets:450 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:22440 (21.9 KiB)  TX bytes:19232 (18.7 KiB)
       Interrupt:10 Base address:0xa000
...
```
3.2.3.1 Enabling VXLAN Hardware Stateless Offloads for ConnectX-3 Pro

To enable the VXLAN offloads support load the mlx4_core driver with Device-Managed Flow-steering (DMFS) enabled. DMFS is the default steering mode.

> To verify it is enabled by the adapter card:

Step 1. Open the `/etc/modprobe.d/mlnx.conf` file.
Step 2. Set the parameter `debug_level` to “1”.

```
options mlx4_core debug_level=1
```
Step 3. Restart the driver.
Step 4. Verify in the dmesg that the tunneling mode is: `vxlan`.

The net-device will advertise the `tx-udp-tnl-segmentation` flag shown when running “ethtool -k $DEV | grep udp” only when VXLAN is configured in the OpenvSwitch (OVS) with the configured UDP port.

For example:

```
$ ethtool -k eth0 | grep udp_tnl
  tx-udp_tnl-segmentation: on
```

As of firmware version 2.31.5050, VXLAN tunnel can be set on any desired UDP port. If using previous firmware versions, set the VXLAN tunnel over UDP port 4789.

> To add the UDP port to `/etc/modprobe.d/vxlan.conf`:

```
options vxlan udp_port=<number decided above>
```

3.2.3.2 Enabling VXLAN Hardware Stateless Offloads for ConnectX®-4 Family Devices

VXLAN offload is enabled by default for ConnectX-4 family devices running the minimum required firmware version and a kernel version that includes VXLAN support.

To confirm if the current setup supports VXLAN, run:

```
ethtool -k $DEV | grep udp_tnl
```

Example:

```
# ethtool -k ens1f0 | grep udp_tnl
  tx-udp_tnl-segmentation: on
```

ConnectX-4 family devices support configuring multiple UDP ports for VXLAN offload. Ports can be added to the device by configuring a VXLAN device from the OS command line using the "ip" command.

Example:

```
# ip link add vxlan0 type vxlan id 10 group 239.0.0.10 ttl 10 dev ens1f0 dstport 4789
# ip addr add 192.168.4.7/24 dev vxlan0
# ip link set up vxlan0
```

Note: dstport' params are not supported in Ubuntu 14.4

The VXLAN ports can be removed by deleting the VXLAN interfaces.

---

1. If you configure multiple UDP ports for offload and exceed the total number of ports supported by hardware, then those additional ports will still function properly, but will not benefit from any of the stateless offloads.
Example:

```
# ip link delete vxlan0
```

➢ **To verify that the VXLAN ports are offloaded, use `debugfs` (if supported):**

**Step 1.** Mount `debugfs`.

```
# mount -t debugfs nodev /sys/kernel/debug
```

**Step 2.** List the offloaded ports.

```
ls /sys/kernel/debug/mlx5/$PCIDEV/VXLAN
```

Where `$PCIDEV` is the PCI device number of the relevant ConnectX-4 family device.

Example:

```
# ls /sys/kernel/debug/mlx5/0000:81:00.0/VXLAN
4789
```

### 3.2.3.3 Important Notes

- VXLAN tunneling adds 50 bytes (14-eth + 20-ip + 8-udp + 8-vxlan) to the VM Ethernet frame. Please verify that either the MTU of the NIC who sends the packets, e.g. the VM virtio-net NIC or the host side veth device or the uplink takes into account the tunneling overhead. Meaning, the MTU of the sending NIC has to be decremented by 50 bytes (e.g 1450 instead of 1500), or the uplink NIC MTU has to be incremented by 50 bytes (e.g 1550 instead of 1500)

- From upstream 3.15-rc1 and onward, it is possible to use arbitrary UDP port for VXLAN. Note that this requires firmware version 2.31.2800 or higher. Additionally, you need to enable this kernel configuration option `CONFIG_MLX4_EN_VXLAN=y` (ConnectX-3 Pro only).

- On upstream kernels 3.12/3.13 GRO with VXLAN is not supported

### 3.2.4 Q-in-Q Encapsulation per VF in Linux (VST)

This feature is supported for ConnectX-3 Pro and ConnectX-5 adapter cards only.

This section describes the configuration of IEEE 802.1ad QinQ VLAN tag (S-VLAN) to the hypervisor per Virtual Function (VF). The Virtual Machine (VM) attached to the VF (via SR-IOV) can send traffic with or without C-VLAN. Once a VF is configured to VST QinQ encapsulation (VST QinQ), the adapter's hardware will insert S-VLAN to any packet from the VF to the physical port. On the receive side, the adapter hardware will strip the S-VLAN from any packet coming from the wire to that VF.
Features Overview and Configuration

Setup

The setup assumes two servers equipped with ConnectX-3 Pro/ConnectX-5 adapter cards.

Prerequisites

- Kernel must be of v3.10 or higher, or custom/inbox kernel must support vlan-stag
- Firmware version 2.36.5150 or higher must be installed for ConnectX-3 Pro HCAs
- Firmware version 16.21.0458 or higher must be installed for ConnectX-5 HCAs
- The server should be enabled in SR-IOV and the VF should be attached to a VM on the hypervisor.
  - In order to configure SR-IOV in Ethernet mode for ConnectX-3 Pro adapter cards, please refer to Section 3.2.1.2.1, “Configuring SR-IOV for ConnectX-3/ConnectX-3 Pro”, on page 83.
  - In order to configure SR-IOV in Ethernet mode for ConnectX-5 adapter cards, please refer to Section 3.2.1.2.2, “Configuring SR-IOV for ConnectX-4/ConnectX-5 (Ethernet)”, on page 88.

In the following configuration example, the VM is attached to VF0.

Network Considerations

The network switches may require increasing the MTU (to support 1522 MTU size) on the relevant switch ports.

Configuring Q-in-Q Encapsulation per Virtual Function for ConnectX-3 Pro

1. Enable QinQ support in the hardware. Set the phv-bit flag using ethtool (on the hypervisor).
   ```
   # ethtool --set-priv-flags ens2 phv-bit on
   # echo 'vlan 100 proto 802.1ad' > /sys/class/net/ens2/vf0/vlan_info
   ```
2. Add the required S-VLAN (QinQ) tag (on the hypervisor) per port per VF. There are two ways to add the S-VLAN:
   a. By using sysfs only if the Kernel version used is v4.9 or older:
b. By using the `ip link` command (available only when using the latest Kernel version):

```
# ip link set dev ens2 vf 0 vlan 100 proto 802.1ad
```

Check the configuration using the `ip link show` command:

```
# ip link show ens2
2: ens2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 7c:fe:90:19:9e:21 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, vlan protocol 802.1ad , spoof checking off, link-state auto
    vf 1 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 2 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 3 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 4 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
```

3. [Optional] Add S-VLAN priority. Use the `qos` parameter in the `ip link` command (or `sysfs`):

```
# ip link set dev ens2 vf 0 vlan 100 qos 3 proto 802.1ad
```

Check the configuration using the `ip link show` command:

```
# ip link show ens2
2: ens2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
    link/ether 7c:fe:90:19:9e:21 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, qos 3, vlan protocol 802.1ad , spoof checking off, link-state auto
    vf 1 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 2 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 3 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
    vf 4 MAC 00:00:00:00:00:00, vlan 4095, spoof checking off, link-state auto
```

4. Restart the driver in the VM attached to that VF.

```
(VM1)# /etc/init.d/openidb restart
```

5. Create a VLAN interface on the VM and add an IP address.

```
# ip link add link ens5 ens5.40 type vlan protocol 802.1q id 40
# ip addr add 42.134.135.7/16 brd 42.134.255.255 dev ens5.40
# ip link set dev ens5.40 up
```

6. To verify the setup, run `ping` between the two VMs and open Wireshark or `tcpdump` to capture the packet.

For further examples, refer to HowTo Configure QinQ Encapsulation per VF in Linux (VST) for ConnectX-3 Pro Community pots.

**Configuring Q-in-Q Encapsulation per Virtual Function for ConnectX-5**

1. Add the required S-VLAN (QinQ) tag (on the hypervisor) per port per VF. There are two ways to add the S-VLAN:
a. By using sysfs:

```
# echo '100:0:802.1ad' > /sys/class/net/ens1f0/device/sriov/0/vlan
```

b. By using the `ip link` command (available only when using the latest Kernel version):

```
# ip link set dev ens1f0 vf 0 vlan 100 proto 802.1ad
```

Check the configuration using the `ip link show` command:

```
# ip link show ens1f0
ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether ec:0d:9a:44:37:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, vlan protocol 802.1ad, spoof checking off, link-state auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
```

2. **[Optional]** Add S-VLAN priority. Use the `qos` parameter in the `ip link` command (or `sysfs`):

```
# ip link set dev ens1f0 vf 0 vlan 100 qos 3 proto 802.1ad
```

Check the configuration using the `ip link show` command:

```
# ip link show ens1f0
ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether ec:0d:9a:44:37:84 brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 00:00:00:00:00:00, vlan 100, qos 3, vlan protocol 802.1ad, spoof checking off, link-state auto, trust off
    vf 1 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 2 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 3 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
    vf 4 MAC 00:00:00:00:00:00, spoof checking off, link-state auto, trust off
```

3. Create a VLAN interface on the VM and add an IP address.

```
# ip link add link ens5 ens5.40 type vlan protocol 802.1q id 40
# ip addr add 42.134.135.7/16 brd 42.134.255.255 dev ens5.40
# ip link set dev ens5.40 up
```

4. To verify the setup, run `ping` between the two VMs and open Wireshark or tcpdump to capture the packet.
3.3 Resiliency

3.3.1 Reset Flow

Reset Flow is activated by default, once a "fatal device" error is recognized. Both the HCA and the software are reset, the ULPs and user application are notified about it, and a recovery process is performed once the event is raised.

- In mlx4 devices, “Reset Flow” is activated by default. It can be disabled using the mlx-4_core module parameter internal_err_reset (default value is 1).
- In mlx5 devices, “Reset Flow” is activated by default. Currently, it can be triggered by a firmware assert with Recover Flow Request (RFR) only. Firmware RFR support should be enabled explicitly using mlxconfig commands.

➢ To query the current value, run:

```bash
mlxconfig -d /dev/mst/mt4115_pciconf0 query | grep SW_RECOVERY_ON_ERRORS
```

➢ To enable RFR bit support, run:

```bash
mlxconfig -d /dev/mst/mt4115_pciconf0 set SW_RECOVERY_ON_ERRORS=true
```

3.3.1.1 Kernel ULPs

Once a "fatal device" error is recognized, an IB_EVENT_DEVICE_FATAL event is created, ULPs are notified about the incident, and outstanding WQEs are simulated to be returned with "flush in error" message to enable each ULP to close its resources and not get stuck via calling its "remove_one" callback as part of "Reset Flow".

Once the unload part is terminated, each ULP is called with its "add_one" callback, its resources are re-initialized and it is re-activated.

3.3.1.2 SR-IOV

If the Physical Function recognizes the error, it notifies all the VFs about it by marking their communication channel with that information, consequently, all the VFs and the PF are reset.

If the VF encounters an error, only that VF is reset, whereas the PF and other VFs continue to work unaffected.

3.3.1.3 Forcing the VF to Reset

If an outside “reset” is forced by using the PCI sysfs entry for a VF, a reset is executed on that VF once it runs any command over its communication channel.

For example, the below command can be used on a hypervisor to reset a VF defined by 0000:04:00.1:

```bash
echo 1 >/sys/bus/pci/devices/0000\:04\:00.1/reset
```

---

1. For mlx4 devices, a “fatal device” error can be a timeout from a firmware command, an error on a firmware closing command, communication channel not being responsive in a VF, etc.

   For mlx5 devices, a “fatal device” is a firmware assert combined with Recover Flow Request bit.
3.3.1.4 Advanced Error Reporting (AER) in ConnectX-3 and ConnectX-3 Pro

AER, a mechanism used by the driver to get notifications upon PCI errors, is supported only in native mode. ULPs are called with `remove_one/add_one` and expect to continue working properly after that flow. User space application will work in same mode as defined in the "Reset Flow" above.

3.3.1.5 Extended Error Handling (EEH)

Extended Error Handling (EEH) is a PowerPC mechanism that encapsulates AER, thus exposing AER events to the operating system as EEH events.

The behavior of ULPs and user space applications is identical to the behavior of AER.

3.3.1.6 CRDUMP

CRDUMP feature allows for taking an automatic snapshot of the device CR-Space in case the device’s FW/HW fails to function properly. The snapshot is taken in case the driver detects the following issues:
1. Critical event, such as a command timeout
2. Critical FW command failure
3. PCI errors
4. Internal FW error

Example

The following message is printed to the log:

```
[257480.719070] mlx4_core 0000:00:05.0: Internal error detected:
[257480.726019] mlx4_core 0000:00:05.0:   buf[00]: 0xffffffff
[257480.732082] mlx4_core 0000:00:05.0:   buf[01]: 00000000
...  
[257480.806531] mlx4_core 0000:00:05.0:   buf[0f]: 00000000
[257480.811534] mlx4_core 0000:00:05.0: device is going to be reset
```
Snapshot should be copied by Linux standard tool for future investigation.

In mlx4 driver, CRDUMP will not be collected if internal_err_reset module parameter is set to 0.

3.4 Fast Driver Unload

This feature is supported in ConnectX-4 and above adapter cards.

This feature enables optimizing mlx5 driver teardown time in shutdown and kexec flows. The fast driver unload is disabled by default. To enable it, the prof_sel module parameter of mlx5_core module should be set to 3.
4 Troubleshooting

You may be able to easily resolve the issues described in this section. If a problem persists and you are unable to resolve it yourself please contact your Mellanox representative or Mellanox Support at support@mellanox.com.

4.1 General Issues

Table 9 - General Issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
</table>
| The system panics when it is booted with a failed adapter installed. | Malfunction hardware component                                       | 1. Remove the failed adapter.  
2. Reboot the system.                                                        |
| Mellanox adapter is not identified as a PCI device.                  | PCI slot or adapter PCI connector dysfunctionality                     | 1. Run `lspci`.  
2. Reseat the adapter in its PCI slot or insert the adapter to a different PCI slot.  
If the PCI slot confirmed to be functional, the adapter should be replaced. |
| Mellanox adapters are not installed in the system.                   | Misidentification of the Mellanox adapter installed                    | Run the command below and check Mellanox’s MAC to identify the Mellanox adapter installed.  
lspci | grep Mellanox' or 'lspci -d 15b3:  
Mellanox MACs start with:  
| Insufficient memory to be used by udev upon OS boot.                 |udev is designed to fork() new process for each event it receives so it could handle many events in parallel, and each udev instance consumes some RAM memory. | Limit the udev instances running simultaneously per boot by adding udev.children-max=<number> to the kernel command line in grub. |
| Operating system running from root file system located on a remote storage (over Mellanox devices), hang during reboot/shutdown (errors such as “No such file or directory” will appear). | The mlnx-en.d service script is called using the ‘stop’ option by the operating system. This option unloads the driver stack. Therefore, the OS root file system disappears before the reboot/shutdown procedure is completed, leaving the OS in a hang state. | Disable the openibd ‘stop’ option by setting 'ALLOW_STOP=no' in /etc/mlnx-en.conf configuration file. |
## 4.2 Ethernet Related Issues

### Table 10 - Ethernet Related Issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet interfaces renaming fails leaving them with names such as renameXY.</td>
<td>Invalid udev rules.</td>
<td>Review the udev rules inside the &quot;/etc/udev/rules.d/70-persistent-net.rules&quot; file. Modify the rules such that every rule is unique to the target interface, by adding correct unique attribute values to each interface, such as dev_id, dev_port and KERNELS or address). Example of valid udev rules:</td>
</tr>
<tr>
<td>No link.</td>
<td>Misconfiguration of the switch port or using a cable not supporting link rate.</td>
<td>• Ensure the switch port is not down&lt;br&gt;• Ensure the switch port rate is configured to the same rate as the adapter's port</td>
</tr>
<tr>
<td>Degraded performance is measured when having a mixed rate environment (10GbE, 40GbE and 56GbE).</td>
<td>Sending traffic from a node with a higher rate to a node with lower rate.</td>
<td>Enable Flow Control on both switch's ports and nodes:&lt;br&gt;• On the server side run: <code>ethtool -A &lt;interface&gt; rx on tx on</code>&lt;br&gt;• On the switch side run the following command on the relevant interface: <code>send on force and receive on force</code></td>
</tr>
<tr>
<td>No link with break-out cable.</td>
<td>Misuse of the break-out cable or misconfiguration of the switch's split ports</td>
<td>• Use supported ports on the switch with proper configuration. For further information, please refer to the MLNX_OS User Manual.&lt;br&gt;• Make sure the QSFP break-out cable side is connected to the SwitchX.</td>
</tr>
<tr>
<td>Physical link fails to negotiate to maximum supported rate.</td>
<td>The adapter is running an outdated firmware.</td>
<td>Install the latest firmware on the adapter.</td>
</tr>
</tbody>
</table>
4.3 Performance Related Issues

Table 11 - Performance Related Issues

<table>
<thead>
<tr>
<th>Issue</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
</table>
| Physical link fails to come up while port physical state is **Polling** | The cable is not connected to the port or the port on the other end of the cable is disabled. | • Ensure that the cable is connected on both ends or use a known working cable  
• Check the status of the connected port using the `ibportstate` command and enable it if necessary |
| Physical link fails to come up while port physical state is **Disabled** | The port was manually disabled.                                      | Restart the driver: `/etc/init.d/openibd restart`                                                   |

Out of the box throughput performance in Ubuntu 14.04 is not optimal and may achieve results below the line rate in 40GE link speed.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
</table>
| The driver works but the transmit and/or receive data rates are not optimal. | These recommendations may assist with gaining immediate improvement:  
1. Confirm PCI link negotiated uses its maximum capability  
2. Stop the IRQ Balancer service.  
   `/etc/init.d/irq_balancer stop`  
3. Start `mlnx_affinity` service.  
   `mlnx_affinity start`  
For best performance practices, please refer to the Performance Tuning Guide for Mellanox Network Adapters (www.mellanox.com > Products > InfiniBand/VPI Drivers > Linux SW/Drivers). | For additional performance tuning, please refer to Performance Tuning Guide. |

IRQ affinity is not set properly by the `irq_balancer`.
UDP receiver throughput may be lower than expected, when running over mlx4_en Ethernet driver. This is caused by the adaptive interrupt moderation routine, which sets high values of interrupt coalescing, causing the driver to process large number of packets in the same interrupt, leading UDP to drop packets due to overflow in its buffers.

Disable adaptive interrupt moderation and set lower values for the interrupt coalescing manually.

```
ethtool -C <eth>X adaptive-rx off
           rx-usecs 64 rx-frames 24
```

Values above may need tuning, depending on the system, configuration and link speed.

### 4.4 SR-IOV Related Issues

**Table 12 - SR-IOV Related Issues**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failed to enable SR-IOV. The following message is reported in dmesg: mlx4_core 0000:xx:xx.0: Failed to enable, continuing without (err = -22)</td>
<td>The number of VFs configured in the driver is higher than configured in the firmware.</td>
<td>1. Check the firmware SR-IOV configuration, run the mlxconfig tool. 2. Set the same number of VFs for the driver.</td>
</tr>
<tr>
<td>Failed to enable SR-IOV. The following message is reported in dmesg: mlx4_core 0000:xx:xx.0: Failed to enable, continuing without (err = -12)</td>
<td>SR-IOV is disabled in the BIOS.</td>
<td>Check that the SR-IOV is enabled in the BIOS (see Section 3.2.1.2, “Setting Up SR-IOV”, on page 81).</td>
</tr>
<tr>
<td>When assigning a VF to a VM the following message is reported on the screen: &quot;PCI-assgine: error: requires KVM support&quot;</td>
<td>SR-IOV and virtualization are not enabled in the BIOS.</td>
<td>1. Verify they are both enabled in the BIOS 2. Add to the GRUB configuration file to the following kernel parameter: &quot;intel_immun=on&quot; (see Section 3.2.1.2, “Setting Up SR-IOV”, on page 81).</td>
</tr>
</tbody>
</table>