Troubleshooting Layer 2 Switching Issues

Layer 2 is the Data Link Layer of the Open Systems Interconnection model (OSI model) of computer networking.

This chapter describes how to identify and resolve problems that can occur with Layer 2 switching in the Cisco Nexus 5000 Series switch.

This chapter includes the following sections:

MAC Address Table

Data traffic is flooding

Data is not getting forwarded, instead it is being flooded to all ports of a VLAN.

Possible Cause

MAC address learning is disabled because of loop detection. A severity 2 syslog message, STM_LOOP_DETECT, should have been received.

Solution

After waiting for 180 seconds, learning is enabled automatically. A severity 2 syslog message, STM_LEARNING_RE_ENABLE, should be received.

Possible Cause

The MAC address table is full. A severity 2 syslog message, STM_LIMIT_REACHED, should have been received.

Solution

After waiting for 180 seconds, the MAC table is flushed and learning is enabled automatically. Alternatively, wait until some MAC entries are aged out, so that the total learned entries fall below 1500, or enter the **clear mac address-table dynamic [address** <*mac*>**]** command to clear out entries. This creates free space for new MAC entries to be learned. A severity 2 syslog message, STM_LEARNING_RE_ENABLE, should be received.

Possible Cause

MAC address learning is disabled due to a learning overload (that is, too many new addresses in a short time). A severity 4 syslog message, STM_LEARNING_OVERLOAD, should have been received.

Solution

After waiting for 120 seconds, learning is enabled automatically.

MAC address not learned

MAC address is not learned by the switch. This causes the MAC address not to be listed in the MAC table.

Possible Cause

MAC address learning is disabled because of loop detection. A severity 2 syslog message, STM_LOOP_DETECT, should have been received.

Solution

After waiting for 180 seconds, learning is enabled automatically. A severity 2 syslog message, STM_LEARNING_RE_ENABLE, should be received.

Possible Cause

The MAC address table is full. A severity 2 syslog message, STM_LIMIT_REACHED, should have been received.

Solution

After waiting for 180 seconds, the MAC table is flushed and learning is enabled automatically. Alternatively, wait until some MAC entries are aged out, so that the total learned entries fall below 1500, or enter the **clear mac address-table dynamic [address** <*mac*>**]** command to clear out entries. This creates free space for new MAC entries to be learned. A severity 2 syslog message, STM_LEARNING_RE_ENABLE, should be received.

Possible Cause

MAC address learning is disabled due to a learning overload (that is, too many new addresses in a short time). A severity 4 syslog message, STM_LEARNING_OVERLOAD, should have been received.

Solution

After waiting for 120 seconds, learning is enabled automatically.

Possible Cause

No egress path was set for the incoming data traffic. The MAC address from a data stream is not learned if there is no path for that data going out of the switch.

Solution

Configure an outgoing path for the data.

For example, the VLAN may not have been enabled on any of the interfaces other than the one on which data is coming in on. Alternatively, the outgoing interfaces may be down. If this is the case, you need to bring up those interfaces.

Traffic flooding in a VPC setup

Data is not getting forwarded, instead it is being flooded in the presence of a VPC scenario.

Possible Cause

The MAC address is learned on one switch only. Typically, this situation would be a bug regarding the synchronization of the MAC address with the VPC peer.

Solution

Clear the MAC address from the switch where it was learned. This triggers new learning and synchronization of the MAC addresses across the VPC switches.

Spanning Tree Protocol

HIFs go down with the BPDUGuard errDisable message

HIFs go down accompanied with the message, BPDUGuard errDisable.

Possible Cause

By default, the HIFs are in STP edge mode with the BPDU guard enabled. This means that the HIFs are supposed to be connected to hosts or non-switching devices. If they are connected to a non-host device/switch that is sending BPDUs, the HIFs become error-disabled upon receiving a BPDU.

Solution

Enable the BPDU filter on the HIF and on the peer connecting device. With the filter enabled, the HIFs do not send or receive any BPDUs. Use the following commands to confirm the details of the STP port state for the port:

- **show spanning-tree interface** <*id*> **detail**
- **show spanning-tree interface** <*id*>

FWM-2-STM_LOOP_DETECT detected on switch, dynamic learning disabled

When FWM-2-STM_LOOP_DETECT is detected on the switch, dynamic learning is disabled.

Possible Cause

- MAC addresses are moving because of incorrect STP-port state convergence.
- MAC addresses are moving because the source of the data being physically moved across all switches while STP states are converged and in correct states.

Use the following commands to verify the STP port state across VLANS on the switches:

- **show spanning-tree**
- **show spanning-tree vlan** <*id*>

Solution

- Check for a correct STP convergence and for STP port states across all switches in the topography. Also confirm that there are no disputes or incorrect port states.

- If the source of the data frames, which are physically moving, is identified, then control the source to halt rapid and continuous moves.
- By default, dynamic learning is opened after 180 seconds. At that point, any STP disputes or inconsistencies should be resolved.

Port stuck in STP blocking state with BLK*(Type_Inc)

A port is stuck in STP blocking state with BLK*(Type_Inc).

Possible Cause

A type inconsistency might exist on an access port when it is connected to a trunk port on the other end. The port becomes BLK*(Type_Inc) to indicate that there is an incorrect configuration on the link. Use the following commands to confirm the details of the STP port state for the port:

- **show spanning-tree interface** <*id*> **detail**
- **show spanning-tree interface** <*id*>

Solution

Check the switch port modes configured at both ends (ports) of the link. Ensure that they are in the same mode. Both should be in access or trunk mode. Once the modes are synchronized, the port moves out of the inconsistency state.

Port stuck in STP blocking state with BLK*(PVID_Inc)

A port is stuck in STP blocking state with BLK*(PVID_Inc).

Possible Cause

A PVID inconsistency may exist when there is a native VLAN mismatch across a trunk link. When this occurs, the port state becomes BLK* (PVID_Inc). Use the following commands to confirm the details of the STP port state for the port:

- **show spanning-tree interface** <*id*> **detail**
- **show spanning-tree interface** <*id*>

Solution

Check the native VLAN configured at both ends (ports) of the link. Ensure that they have the same native VLAN. Once the native VLANs are synchronized, the port moves out of the inconsistency state.

Port stuck in STP blocking state with BLK*(Loop_Inc)

A port is stuck in STP blocking state with BLK*(Loop_Inc).

Possible Cause

This situation occurs when the loop guard is configured on the port and the port stops receiving BPDUs. This is supposed to prevent loops when unidirectional link failures occur. However, the port is put into a BLK* (Loop_Inc) state. Use the following commands to confirm the details of the STP port state for the port:

- **show spanning-tree interface** <*id*> **detail**
- **show spanning-tree interface** <*id*>

Solution

Check the native VLAN configured at both ends (ports) of the link. Ensure that they have the same native VLAN. Once the native VLANs are synchronized, the port moves out of the inconsistency state.

Multicast

Source MAC addresses of IGMP joins are learned

In this situation, the source MAC addresses of IGMP joins are learned. However, source MAC addresses of IGMP joins are usually not learned by the switch in order to conserve MAC address space.

Possible Cause

Receiving joins and performing an ISSU simultaneously might cause the situation.

Solution

The MAC addresses age out (expire) if the join stops. Alternatively, you can clear the MAC addresses specifically by using the **clear mac address-table dynamic mac** <*mac*> command.

Multicast data traffic not received by host

Host does not receive multicast data traffic.

Possible Cause

The join is not registered.

Solution

- Ensure that the host application is sending the joins.
- Check if the switch port is configured for the VLAN on which the joins are being sent using the **show vlan id** <*vlan*> command.
- Check if the relevant VLAN is active by using the **show vlan id** <*vlan*> command.
- Check if the switch port is in STP forwarding state by using the **show spanning-tree vlan** <*vlan*> command.

Multicast data traffic not received when host is registered for group

Multicast data traffic is not received when the host is registered for the group.

Possible Cause

A bug may exist in exist in the communication between the IGMP and FWM processes.

Review the output from the following commands:

- **show ip igmp snooping groups vlan 1001**
- **show mac address-table multicast vlan 1001 igmp-snooping**
- **show platform fwm info vlan 1001 all_macgs verbose**

Solution

Perform a shut/no-shut operation on the host interface and send the join again.

Multicast traffic is being flooded in a VPC setup

In a VPC setup multicast traffic is being flooded.

Possible Cause

IGMP snooping is disabled on one of the switches.

Solution

Enable IGMP snooping on both switches.

Note Groups for link local IP addresses (that is, 224.0.0.X) are not created.

VLANs

Nexus 5000 does not have the same VLANs as switch running VTP server

VLANs for the Nexus 5000 are not the same as for the switch running the VTP server.

Possible Cause

The Nexus 5000 currently supports VTP only in transparent mode (4.2(1)N1(1) and later releases).

Solution

This situation indicates that VLANs must be configured locally. However a VTP client and server can both communicate through a Nexus 5000 by using the following commands:

```
switch(config)# feature vtp
switch(config)# vtp mode transparent
switch(config)# exit
switch# show vtp status
```

VLAN cannot be created

A VLAN cannot be created.

Possible Cause

An internal VLAN range is used.

Solution

Use a VLAN number that is not being reserved for internal use.

Note The VLAN range of 3968 to 4047 is reserved for internal use.

Example:

```
switch(config)# vlan ?
```

```
<1-3967,4048-4093>  VLAN ID 1-4094 or range(s): 1-5, 10 or 2-5,7-19
```

Interface VLAN is down

The interface VLAN is down.

Possible Cause

The VLAN was not created.

Solution

Although VLAN <###> is not yet created, the NX-OS allows the configuration of the **interface vlan** <###>. As a result, the **interface vlan** <###> does not come up. Use the **show vlan** command to determine if VLAN <###> exists. If it does not exist, use the **vlan** <###> command to create the VLAN. After the VLAN is created, you must bounce the interface VLAN to have it come up.

Example:

```
switch(config)# interface vlan 600
switch(config-if)# no shutdown
switch(config-if)# show interface vlan 600 brief

--------------------------------------------------------------------------------
Interface Secondary VLAN(Type)                    Status Reason
--------------------------------------------------------------------------------
Vlan600   --                                      down   other
switch(config)# show vlan id 600
VLAN 600 not found in current VLAN database
switch(config-if)# vlan 600
switch(config)# show vlan id 600

VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
600  VLAN0600                         active    Po1, Po11, Po30, Po31
switch(config-if)# interface vlan 600
switch(config-if)# shut
switch(config-if)# no shutdown
switch(config-if)# show interface vlan 600 brief

--------------------------------------------------------------------------------
Interface Secondary VLAN(Type)                    Status Reason
--------------------------------------------------------------------------------
Vlan600   --                                      up     --
```

Possible Cause

VLAN was suspended by the vPC configuration on the Nexus 5000 pair.

Solution

Show that the vPC consistency parameters are global and make sure that the VLAN was not suspended. Otherwise, fix the configuration mismatch on the Nexus 5000 pair:

Example:

```
switch# sh vpc consistency-parameters global

    Legend:
        Type 1 : vPC will be suspended in case of mismatch

Name                        Type  Local Value            Peer Value
-------------               ----  ---------------------- -----------------------
```

```
QoS                          1     ([], [3], [], [], [],  ([], [3], [], [], [],
                                   [])                    [])
Network QoS (MTU)            1     (1538, 2240, 0, 0, 0,  (1538, 2240, 0, 0, 0,
                                   0)                     0)
Network Qos (Pause)          1     (F, T, F, F, F, F)     (F, T, F, F, F, F)
Input Queuing (Bandwidth)    1     (50, 50, 0, 0, 0, 0)   (50, 50, 0, 0, 0, 0)
Input Queuing (Absolute      1     (F, F, F, F, F, F)     (F, F, F, F, F, F)
Priority)
Output Queuing (Bandwidth)   1     (50, 50, 0, 0, 0, 0)   (50, 50, 0, 0, 0, 0)
Output Queuing (Absolute     1     (F, F, F, F, F, F)     (F, F, F, F, F, F)
Priority)
STP Mode                     1     Rapid-PVST             Rapid-PVST
STP Disabled                 1     None                   None
STP MST Region Name          1     ""                     ""
STP MST Region Revision      1     0                      0
STP MST Region Instance to   1
 VLAN Mapping
STP Loopguard                1     Disabled               Disabled
STP Bridge Assurance         1     Enabled                Enabled
STP Port Type, Edge          1     Normal, Disabled,      Normal, Disabled,
BPDUFilter, Edge BPDUGuard         Disabled               Disabled
STP MST Simulate PVST        1     Enabled                Enabled
Allowed VLANs                -     1-2                    1-2
Local suspended VLANs              2                      -
switch#
```

Configuring interface to access port does not allow VLAN <###> to go through

After configuring an interface to access a port for allowing VLAN <###>, the VLAN <###> does not go through.

Possible Cause

The VLAN was not created.

Solution

In NX-OS, configuring with the **switchport access vlan** <*###*> command on an interface does not automatically create VLAN <###>. You must specifically create VLAN <###> using the **vlan** <*###*> command. Use the **show vlan** command to determine if VLAN <###> exists. If it does not exist, then use the **vlan** <*###*> command to create the VLAN.

Cannot create VLAN

The VLAN cannot be created.

Possible Cause

All VLAN resources are exhausted.

Solution

For the Nexus 5000, the maximum number of active VLANs and VSANs per switch is 512 (31 reserved for VSAN; remainder reserved for VLAN). Use the **show resource vlan** command to determine the number of available VLANs.

Example:

```
switch(config)# show resource vlan
```

```
 Resource               Min       Max      Used    Unused     Avail
-----------            -----     -----    ------  --------   -------
 vlan                16       512        25         0       487
```

Cannot create SVI

The SVI cannot be created.

Possible Cause

The interface-vlan feature is not enabled.

Solution

The interface-vlan feature must be enabled before configuring the SVI. Use the **show feature** command to determine which features are enabled.

Example:

```
switch(config)# feature interface-vlan
switch(config)# show feature
Feature Name          Instance  State
--------------------  --------  --------
tacacs                1         disabled
lacp                  1         enabled
interface-vlan        1         enabled
private-vlan          1         enabled
udld                  1         enabled
vpc                   1         enabled
fcoe                  1         disabled
fex                   1         enabled
```

Cannot create private VLAN (PVLAN)

The private VLAN (PVLAN) cannot be created.

Possible Cause

The private-vlan feature is not enabled.

Solution

The private-vlan feature must be enabled prior to PVLAN configuration, which makes the PVLAN command available. Use the **show feature** command to determine which features are enabled.

Example:

```
switch(config)# feature private-vlan
switch(config)# show feature
Feature Name          Instance  State
--------------------  --------  --------
tacacs                1         disabled
lacp                  1         enabled
interface-vlan        1         enabled
private-vlan          1         enabled
udld                  1         enabled
vpc                   1         enabled
fcoe                  1         disabled
fex                   1         enabled
```

Registers and Counters

Identifying drops

There are logical and physical causes for the Nexus 5000 to drop a frame. There are also situations when a frame cannot be dropped because of the cut-through nature of the switch architecture. If a drop is necessary, but the frame is being switched in a cut-through path, then the only option is to stomp the Ethernet frame check sequence (FCS). Stomping a frame involves setting the FCS to a known value that does not pass a CRC check. This causes subsequent CRC checks to fail later in the path for this frame. A downstream store-and-forward device, or a host, will be able to drop this frame.

Note When a frame is received on a 10 Gb/s interface, it is considered to be in the cut-through path.

The following example output shows all discards and drops seen on a given interface, except for queuing drops. The queuing drops may be expected or resulting from errors. (Drops are more common than discards.)

Example:

```
switch# show platform fwm info pif ethernet 1/1 ...
Eth1/1 pd: tx stats: bytes 19765995 frames 213263 discard 0 drop 0
Eth1/1 pd: rx stats: bytes 388957 frames 4232 discard 0 drop 126
```

For some commands, you need to know on which chip your port resides.

In the following example, the chip is called Gatos. The example shows which Gatos and which Gatos port is associated with ethernet 1/1.

```
switch# show hardware internal gatos port ethernet 1/1 | include
instance|mac
         gatos instance     : 7 <- Gatos 7
         mac port           : 2 <- Port 2
         fw_instance        : 2
```

Expected/Logical drops

During normal operation, the Nexus 5000 encounters frames that cannot be forwarded based on logical conclusions.

For example, if you learn a MAC address on a given interface and receive traffic on that interface with a destination MAC address on the source interface, then you cannot forward the frame. Because it is a known address and cannot be flooded, you can never send traffic out from where it came. This is a requirement to avoid looping Layer 2 topologies.

The error counter, shown in the following example, increments when the ingress port is the only port in the VLAN.

Example (same Gatos instance as in earlier example):

```
switch# show platform fwm info gatos-errors 7
Printing non zero Gatos error registers:
DROP_SRC_MASK_TO_NULL     9
```

Note The **show platform fwm info gatos-errors** command increments 3 times for a given drop.

Other expected drops

Drop	Description
VLAN_MASK_TO_NULL	Traffic destined for the CPU interfaces; not actually a VLAN.
DROP_NO_FABRIC_SELECTED	Increments with VLAN_MASK_TO_NULL.
DROP_INGRESS_ACL	Increments for an access list matching the frame. If an ACL is not applied, this will increment if a large amount of CPU-bound traffic is being received, and the rate limit in the hardware is enabled to protect NX-OS from Denial of Service.

Queue is full

When a queue is full, you need to increment discards in the respective queue on the ingress interface.

Example:

```
switch# show queuing interface ethernet 1/1
Ethernet1/1 queuing information:
   TX Queuing
     qos-group  sched-type  oper-bandwidth
         0       WRR             50
         1       WRR             50

   RX Queuing
     qos-group 0
     q-size: 243200, HW MTU: 1600 (1500 configured)
     drop-type: drop, xon: 0, xoff: 1520
     Statistics:
         Pkts received over the port             : 0
         Ucast pkts sent to the cross-bar        : 0
         Mcast pkts sent to the cross-bar        : 0
         Ucast pkts received from the cross-bar  : 0
         Pkts sent to the port                   : 0
     <b>    Pkts discarded on ingress              : 0 </b>
         Per-priority-pause status               : Rx (Inactive), Tx
(Inactive)

     qos-group 1
     q-size: 76800, HW MTU: 2240 (2158 configured)
     drop-type: no-drop, xon: 128, xoff: 240
     Statistics:
         Pkts received over the port             : 0
         Ucast pkts sent to the cross-bar        : 0
         Mcast pkts sent to the cross-bar        : 0
         Ucast pkts received from the cross-bar  : 0
         Pkts sent to the port                   : 0
     <b>   Pkts discarded on ingress              : 0 </b>
         Per-priority-pause status               : Rx (Inactive), Tx
(Inactive)
```

```
Total Multicast crossbar statistics:
  Mcast pkts received from the cross-bar      : 0
```

MTU violation

The Nexus 5000 is a cut-through switch at 10 Gb/s. This means that an MTU can be checked, but the frame will already be transmitting before the length is known. Therefore, the frame cannot be dropped. The frame is truncated after the MTU is reached and the CRC value is stomped. The ingress interface increments an Rx Jumbo and the egress interface will increment a Tx CRC and a Tx Jumbo.

- If jumbo frames are seen with the **show interface** or the **show hardware internal gatos port ethernet 1/1 counters rx** commands, this is not an indication that the frames are being dropped. A jumbo frame is just an Ethernet frame, greater than 1500 bytes, that was received or transmitted.
- The <*b*>**show queuing interface** <*i*>**ex/y**</*i*></*b*> command shows the current configured MTU (per class).
- A drop due to an MTU violation can be seen with the **show hardware internal gatos counters interrupt match mtu*** command.
- A counter that matches the Gatos number and fw_instance from the **show hardware internal gatos port ethernet 1/1 | include instance|mac** command is the indicator that an MTU violation has taken place and that the frame has been stomped.

Handling CRC errors

When a CRC error is seen in the FCS on a cut-through port, the Rx CRC counter of the **show interface** command is incremented. However, the frame cannot be dropped because the FCS is at the end of the Ethernet frame on the wire.

The egress interface increments a Tx CRC error and it propagates through to the next device in the path.

You can use the **show hardware internal gatos counters interrupt match stomp** command to determine if the Nexus 5000 is propagating CRCs or generating them.

- If stomp values exist, they should have matching CRC values on that interface.
- If Rx CRC values exist, then you know it entered the switchport with the error already. You can move on to the connected device to trace it back.

MAC Statistics

During normal operation, a Nexus 5000 encounters frames that cannot be forwarded.

Frames are characterized as good frames or bad frames.

- A good frame is a frame that does not have a CRC error or other kind of error
- A bad frame is a frame that has a CRC error or other kind of error

All counters include MAC Control frames where applicable.

MAC TX Statistics

Counter	Description
TX_PKT_LT64	Number of frames (good and bad frames) with transmit packet size less than 64 bytes.
TX_PKT_64	Number of frames (good and bad frames) with transmit packet size equal to 64 bytes.
TX_PKT_65	Number of frames (good and bad frames) with transmit packet size between 65 and 127 bytes.
TX_PKT_128	Number of frames (good and bad frames) with transmit packet size between 128 and 255 bytes.
TX_PKT_256	Number of frames (good and bad frames) with transmit packet size between 256 and 511 bytes.
TX_PKT_512	Number of frames (good and bad frames) with transmit packet size between 512 and 1023 bytes.
TX_PKT_1024	Number of frames (good and bad frames) with transmit packet size between 1024 and 1518 bytes.
TX_PKT_1519	Number of frames (good and bad frames) with transmit packet size between 1519 and 2047 bytes.
TX_PKT_2048	Number of frames (good and bad frames) with transmit packet size between 2048 and 4095 bytes.
TX_PKT_4096	Number of frames (good and bad frames) with transmit packet size between 4096 and 8191 bytes.
TX_PKT_8192	Number of frames (good and bad frames) with transmit packet size between 8192 and 9216 bytes.
TX_PKT_GT9216	Number of frames (good and bad frames) with transmit packet size greater than 9216 bytes.
TX_PKTTOTAL	Total number of frames (good and bad frames) transmitted. This number is the sum of all transmitted packets regardless of frame length.
TX_OCTETS	Total byte count of packet octets (good and bad frames) transmitted.
TX_PKTOK	Number of good frames transmitted.
TX_UCAST	Number of good unicast frames transmitted.
TX_MCAST	Number of good multicast frames transmitted.
TX_BCAST	Number of good broadcast frames transmitted.
TX_VLAN	Number of good 802.1Q tagged VLAN frames transmitted.
TX_PAUSE	Number of 802.3x PAUSE frames transmitted.
TX_USER_PAUSE	Number of transmitted priority flow control frames.
TX_FRM_ERROR	Number of frames with **cl_im_tx_err** set to 1 at EOP.
TX_OCTETSOK	Total byte count of good frames.

MAC RX Statistics

MAC RX Statistic	Description
RX_PKT_LT64	Number of frames (good and bad frames) with receive packet size less than 64 bytes.
RX_PKT_64	Number of frames (good and bad frames) with receive packet size equal to 64 bytes.
RX_PKT_65	Number of frames (good and bad frames) with receive packet size between 65 and 127 bytes.
RX_PKT_128	Number of frames (good and bad frames) with receive packet size between 128 and 255 bytes.
RX_PKT_256	Number of frames (good and bad frames) with receive packet size between 256 and 511 bytes.
RX_PKT_512	Number of frames (good and bad frames) with receive packet size between 512 and 1023 bytes.
RX_PKT_1024	Number of frames (good and bad frames) with receive packet size between 1024 and 1518 bytes.
RX_PKT_1519	Number of frames (good and bad frames) with receive packet size between 1519 and 2047 bytes.
RX_PKT_2048	Number of frames (good and bad frames) with receive packet size between 2048 and 4095 bytes.
RX_PKT_4096	Number of frames (good and bad frames) with receive packet size between 4096 and 8191 bytes.
RX_PKT_8192	Number of frames (good and bad frames) with receive packet size between 8192 and 9216 bytes.
RX_PKT_GT9216	Number of frames (good and bad frames) with receive packet size greater than 9216 bytes.
RX_PKTTOTAL	Total number of frames (good and bad frames) received. This number is the sum of all received packets regardless of frame length.
RX_OCTETS	Total byte count of packet octets (good and bad frames) received.
RX_PKTOK	Number of good frames received.
RX_UCAST	Number of good unicast frames received.
RX_MCAST	Number of good multicast frames received.
RX_BCAST	Number of good broadcast frames received.
RX_VLAN	Number of good 802.1Q tagged VLAN frames received.
RX_OVERSIZE	Number of good frames received that are greater than CFG_xg_rx_stats_max_frame_len.
RX_TOOLONG	Number of frames (good and bad frames) received that are greater than CFG_xg_rx_stats_max_frame_len.
RX_DISCARD	N/A The value of this counter is always 0.

MAC RX Statistic	Description
RX_UNDERSIZE	Number of good frames received that are less than CFG_xg_rx_stats_min_frame_len.
RX_FRAGMENT	Number of bad frames received that are less than CFG_xg_rx_stats_min_frame_len.
RX_CRCERR_NOT_STOMPED	Number of bad frames received that are not stomped.
RX_CRCERR_STOMPED	Number of bad frames received that are stomped.
RX_INRANGEERR	Number of frames with a length field check error, but no CRC error. **Note** Frame length is based on the contents of the ethertype/length field. When the ethertype/length field is less than 0x600, the contents of the field is treated as the length of the frame. This length is compared with the actual frame length. An error is raised when the lengths do not match.
RX_JABBER	Number of bad frames received with frame length greater than CFG_xg_rx_stats_max_frame_len.
RX_PAUSE	Number of good 802.3x MAC control frames received.
RX_USER_PAUSE	Number of good priority flow control frames received.
RX_OCTETSOK	Total byte count of good frames.