Troubleshoot Nexus 7000 with Message Transaction Services

Contents

Introduction

This document describes the Message and Transition Services (MTS) functionality used to troubleshoot Nexus 7000 platforms.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This document is not restricted to specific software and hardware versions.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

Services (also known as Processes) in the NeXus Operating System (NX-OS) communicate with each other by MTS.

If the target service is physically on a different module, Inter Node Communication needs to be used. The communication to the other module is delivered by Advanced Inter Process Communication (AIPC) over the Ethernet Out-of-Band Channel (EOBC) links. The EOBC links are on the MidPlane of the Nexus 7000 chassis.

MTS Provides:

- Messages and high availability (HA) infrastructure and Application Program Interfaces (APIs)
- Buffer management
- Message delivery

AIPC Provides:

- Reliable transport across EOBC
- Delivery of ACK-based retransmission
- Fragmentation and reassembly

MTS mainly consists of destination/source Node/SAP and Opcode.

- Node - an identifier of the physical module and VDC.
- SAP - an identifier of each service. (Each service could have multiple Service Access Points (SAPs) for multiple functions.)
- Opcode - a data type which services use to communicate with other services.

Terminology

MTS : Message and Transaction Services
SAP : Service Access Point

AIPC : Advanced Inter Process Communication / Andiamo IPC
EOBC : Ethernet Out-of-Band Channel

NX-OS : NeXus Operating System

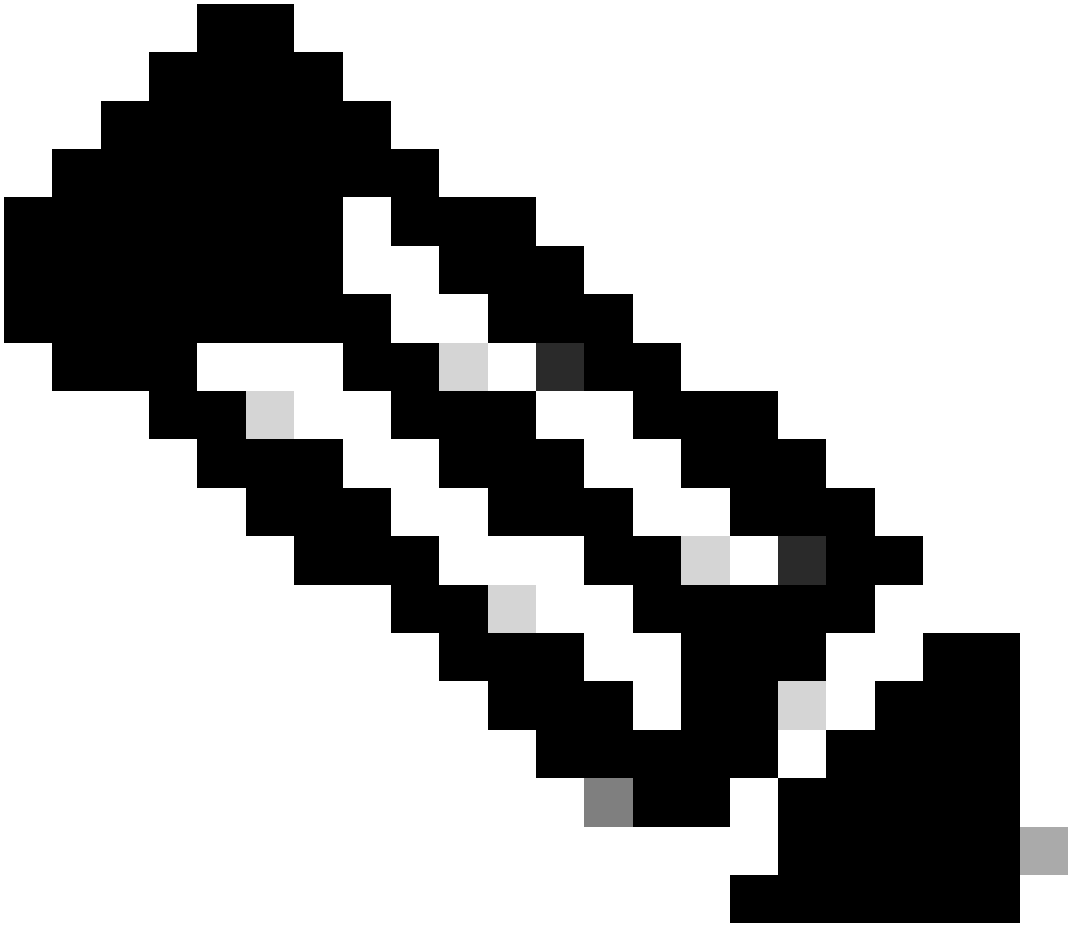
MTS Formats

The MTS destination is represented by SAP (a service) and Node (the module on which the SAP runs).

- **SAP**
  In most cases, the static SAP is used to track the MTS event.

Type	HA	Sap #	Set by
Static SAP (well-known)	Yes	1 - 1023	Caller
Dynamic SAP	No	1024 - 65535	MTS
Registry SAP	Yes	0	Reserved

  SAP can be checked in general or per module (supervisor or linecard).



Note: sup stands for supervisor and lc for linecard.

Get SAP running on any module.

<#root>

`show system internal sysmgr service ~~~~`

```
n7ka# sh system internal sysmgr service all
Name                    UUID     PID
```

SAP

```
      state  Start count               Tag  Plugin ID
----------------  ----------  ------  -----  ---------  -----------  ----------------  -----------
```

aaa

```
 0x000000B5    6942
```

111

```
  s0009             1            N/A           0
cert_enroll       0x0000012B    6941    169  s0009             1            N/A           0
ExceptionLog      0x00000050    7267     92  s0009             1            N/A           0
```

`show system internal mts ~~~~~`

```
n7ka# sh system internal mts sup sap 111 description
Below shows sap on default-VDC, to show saps on non-default VDC, run
show system internal mts node sup-<vnode-id> sap ...
```

AAA Daemon

Get SAP running on specific module only.

<#root>

`Attach to the module`

```
n7ka# attach module 4
Attaching to module 4 ...
To exit type 'exit', to abort type '$.'
Last login: Tue Nov 7 15:42:35 PST 2023 from 127.1.1.2 on pts/0
```

`show system internal sysmgr service ~~~~`

```
n7ka# show system internal sysmgr service all
Name                    UUID     PID
```

SAP

```
      state  Start count               Tag  Plugin ID
---------------  ----------  ------  -----  ---------  -----------  ----------------  -----------
```

aclqo

```
 0x0000016E    1301
```

190

```
  s0009             1            N/A           0
amm               0x00000260    1130    895  s0009             1            N/A           0
bfdc              0x000002C7    1110   1008  s0009             1            N/A           0
```

`show system internal mts ~~~~~`

```
module-4# show system internal mts lc sap
```

190

```
 description
```

Aclqos SAP

- **Opcode Registry**
  Sometimes MTS is sent to destination SAP 0. As a result, the MTS is sent to multiple SAPs that are registered to the SAP registry (opcode registry).
  As an example, Opc 8182 is registered by SAP 175 and 378, confirmed by *show system internal mts sup registry persistent* command. So this MTS is delivered to both SAP175 and SAP 378.

  <#root>

```
n7ka# show system internal ethpm event-history msgs
51) Event:E_MTS_RX, length:60, at 36968 usecs after Thu Sep 18 14:42:15 2014
    [NOT] Opc:MTS_OPC_LINK_EVENT_DOWN(

8182

), Id:0X034960A1, Ret:SUCCESS
    Src:0x00000102/181,

Dst

:0x00000609/

0

, Flags:None
    HA_SEQNO:0X00000000, RRtoken:0x00000000, Sync:UNKNOWN, Payloadsize:82
    Payload:
    0x0000:  00 00 00 07 00 11 00 21 00 00 00 04 00 12 00 04

n7ka# show system internal mts sup registry persistent | i

8182


MTS_OPC_LINK_EVENT_DOWN(

8182

):

175, 378
```

  - **Node ID**
    The Node ID is a 16-bit ID.

    + Slot ID - module slot ID. Starts with 1
    + Vnode ID - virtual node ID. VDC ID starts with 0
    + Lnode ID - logical node ID. SUP:1 , LC:2

    Example: 0x805 = 0x08 + 0b0000_0101 = **0x08** + **0x1** + 0x1 (**slot** + **Vnode** + **Lnode**)
    => **SUP module VDC 2 in slot 8**

- **MTS Address**

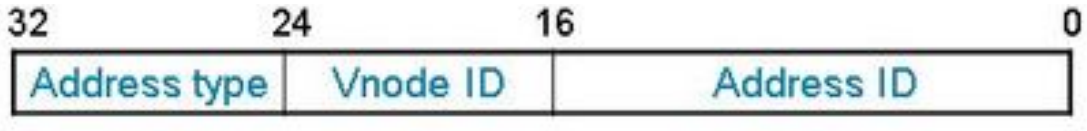
The Node ID is a component of the MTS address. This MTS address indicates the module to which this MTS is to be delivered.

The MTS address (Node ID) is the left part of the output shown next.

<#root>

```
    Src:
0x00000102
/181, Dst:
0x00000609
/0, Flags:None
```

If MTS is unicast, the MTS address is the same as the Node ID.



Address type	Reliable	Bit 31-24 Type bit	Bit 23-16 Vnode	Bit 15-0 Addr-ID	Example
Unicast	Yes	0x00	0x00	Node-id	0x00000801
Alias	Yes	0xFE	Vnode-ID	Alias ID	MTS_NODE_THIS ACTIVE_SUP
Multicast	Yes	0xFF	Vnode-ID	Group ID	MTS_NODE_ACTIVE_SUP _ALL_LC
Dynamic mcast	Yes	0xFF	Vnode-ID	0x0040+	0xFF000040
Broadcast	No	0xFF	0xFF	0xFFFF	MTS_NODE_BROADCAST

Track Process Communication with MTS

ages

You can see the MTS communication history by use of event-history messages. Almost all services have this event history message capability.

This CLI is the MTS event history for eth_port_channel (ethpc).

<#root>

```
n7ka# show port-channel internal event-history msgs
12)
Event:E_MTS_RX
```

```
, length:60, at 15586 usecs after Thu Sep 18 13:13:57 2014


[REQ] Opc:MTS_OPC_ETHPM_PORT_CLEANUP(61444), Id:0X00323B1E

, Ret:SUCCESS


 Src:0x00000601/175, Dst:0x00000601/378

, Flags:None
    HA_SEQNO:0X00000000, RRtoken:0x00323B1E, Sync:UNKNOWN, Payloadsize:26
    Payload:
    0x0000:  00 00 00 02 00 04 00 02 00 01 00 05 00 0c 00 00
```

The fields listed next provide additional details with regard to their purpose in the command:

Event:E_MTS_RX - indicates this MTS was received by this service, ethpc in this case. If it is E_MTS_TX, then ethpc is the sender of this MTS.
Src:0x00000601/175 - MTS_addr/SAP, represents the sender of this MTS.
Dst:0x00000601/378 - MTS_addr/SAP, represents the receiver of this MTS.
Id:0X00323B1E - MTS ID, where both the sender and receiver have the same ID for the same MTS.
Opc:MTS_OPC_ETHPM_PORT_CLEANUP(61444) - Indicates what this event type is. In other words, which opcode this MTS delivers, where 61444 is the opcode number.

Example of an MTS Event

This MTS event example is for a link down event.

```
n7ka# 2014 Sep 18 14:42:15 n7ka %ETHPORT-5-IF_DOWN_LINK_FAILURE: Interface Ethernet1/3 is down (Link fa
```

As a first step, check the ethpm history because it is the comprehensive process for all I/F-related things.

<#root>

```
n7ka# sh system internal ethpm event-history msgs
51) Event:

E_MTS_RX

, length:60, at 36968 usecs after Thu Sep 18 14:42:15 2014
    [NOT] Opc:MTS_OPC_LINK_EVENT_DOWN(8182), Id:

0X034960A1

, Ret:SUCCESS


Src:0x00000102/181

, Dst:0x00000609/0, Flags:None
    HA_SEQNO:0X00000000, RRtoken:0x00000000, Sync:UNKNOWN, Payloadsize:82
    Payload:
    0x0000:  00 00 00 07 00 11 00 21 00 00 00 04 00 12 00 04
```

The result shown indicates the ETHPM received the link down event from Src:0x00000102/181. This MTS address indicates the SAP 181 is on VDC 1 LC on slot 1.

Use the CLI listed next to determine what SAP 181 is on slot 1.

<#root>

```
module-1# attach module 1
module-1# show system internal mts lc sap
```

181

```
 description
```

Port_client SAP

Use the CLI listed next to check the port_client MTS history on slot 1.

<#root>

```
module-1# show system internal
```

port-client

```
 event-history msgs

49) Event:
```

E_MTS_TX

```
, length:60, at 298743 usecs after Thu Sep 18 14:42:14 2014
    [NOT] Opc:MTS_OPC_LINK_EVENT_DOWN(8182), Id:
```

0X034960A1

```
, Ret:SUCCESS
    Src:0x00000102/181, Dst:0x00000609/0, Flags:None
    HA_SEQNO:0X00000000, RRtoken:0x00000000, Sync:UNKNOWN, Payloadsize:82
    Payload:
    0x0000:  00 00 00 07 00 11 00 21 00 00 00 04 00 12 00 04


50) Event:
```

E_MTS_RX

```
, length:60, at 298329 usecs after Thu Sep 18 14:42:14 2014
```

[NOT] Opc:MTS_OPC_LC_LINK_DOWN(8185), Id:0X0349609F, Ret:SUCCESS

Src:0x00000102/536, Dst:0x00000102/0, Flags:None

```
    HA_SEQNO:0X00000000, RRtoken:0x00000000, Sync:UNKNOWN, Payloadsize:17
    Payload:
    0x0000:  00 00 00 02 40 e5 00 07 02 54 1a 70 b6 00 04 84
```

From Id:0X034960A1, we can tell the 49) event is the one of interest. The Port_Client sent the MTS as expected. This must be invoked by the previous MTS event on Port_Client, which is 50). This MTS was sent by Src:0x00000102/536.

Check SAP 536.

<#root>

```
module-1# sh system internal mts lc sap
```

536

```
 de
```

Naxos FPGA

Now we know NAXOS FPGA is the source of this whole event. Use the CLI shown next to find that the local_fault is the root cause of the link down.

<#root>

```
module-1# sh hardware internal
```

naxos

```
 event-history port 3

15) Event E_NAXOS_ISR_DATA length:69, at 170763 usecs after Thu Sep 18 14:42:14 2014
 TO NAXOS_PORT_STATUS_LINK_DOWN: reg_val 0x14
  Status:SUCCESS (0x0)

16) Event E_NAXOS_ISR_DATA length:50, at 170619 usecs after Thu Sep 18 14:42:14 2014
```

LOCAL_FAULT

```
: reg_val 0x14
  Status:SUCCESS (0x0)
```

Related Information

- **Nexus 7000: Troubleshooting MTS Buffer Issues**
- **Cisco Technical Support & Downloads**