ORACLE

Oracle® Communications

CGBU Tekelec Platform Development

Technical Reference

Oracle Communication

Performance Intelligence Center

Release 10.5

Troubleshooting Guide

G10600-01

June 2024

Table of Contents

1.0 Introduction **6**
1.1 Purpose and Scope 6
1.2 Product Summary 6
1.3 References 6
1.4 Acronyms 7

2.0 What to do before requesting support **9**
2.1 Basic health check procedure 9
2.2 Identify affected system 9
2.3 Check system alarms 9
2.4 Identify affected DataFlow 9
2.5 Check logs and traces 9
2.6 Collect logs and configuration 9
2.7 Prepare access to the system 10
2.8 Identify support engineer 10

3.0 High level Functional description **11**
3.1 Introduction 11
3.2 IMF Overview 11
3.3 PMF Overview 12
3.4 How does xMF work 13
3.5 Database overview 13
3.6 xMF Process Architecture 15
3.7 xMF Alarms 17
3.8 Shared Memory Allocation on xMF 19
1. Current messages: 20
2. Messages from the past 20
1. Writer cannot flush: 21

4.0 How to write good PR's **22**
4.1 Collect and analyze procedure 22
4.2 Health check 22
4.3 Other information 22

5.0 Logs and Traces **24**
5.1 xMF processes 24
5.2 jmxAgent 28
5.3 DbReplicator 28

6.0 Troubleshooting and monitoring tools **30**
6.1 disp tool 30
6.2 cfgPmia 40
6.3 Pmia module proc files 45
6.4 Af_Pmia module proc files 49

7.0 Solving specific problems **52**
7.1 IDB Database Access Utilities 52
7.2 MSU/Event Access Utilities 52
7.3 Process Management Utilities 53
7.4 COMCOL Tracing Utilities 54
7.5 Platform utilities 54
7.6 Solving lockup or high CPU usage issues 54

7.7 Automatic Failover 54
7.8 End-To-End Flow Verification Procedure 55
7.9 Alarm Management Tips 56

8.0 Eagle monitoring interfaces (IMF) 58
8.1 Eagle connectivity 58
8.2 DHCP 58
8.3 NTP Broadcast 59
8.4 NTP Usage 59
8.5 STC Copy 59
8.6 Fast Copy 59
8.7 IP Troubleshooting 60
8.8 Eagle IMF interface Troubleshooting 60
8.9 E5-APP-B Disk Replacement 67

9.0 SS7 Probed Message feeder (PMF) 68

10.0 Integrated OCDSR monitoring procedures 68
10.1 Bandwidth limitation customization 68
10.2 Load sharing customization 68
10.3 Application Identifier customization 69
10.4 Change period of OCDSR configuration synchronization 69
10.5 Change OCDSR soap log 69

11.0 Web service interface procedures 70
11.1 Change SSL encryption at web service interface 70

12.0 CIF Interface 70

Appendix A. Review Checklist 72

List of Figures

Figure 1: IMF System Overview 11
Figure 2: PMF System Overview 13
Figure 3: xMF Process Architecture 15
Figure 4: Eagle OAM Alarms Handling 19
Figure 5: SS7 Link Alarms Handling 19
Figure 6: IMF-Eagle-ICP/IXP Interface 58
Figure 7 : EAGLE - PIC interface with Fast Copy feature on 60

List of Tables

Table 1: xMF process description 17
Table 2: Listing of the TKLCmf service states 24
Table 3: Tracing mask value 27
Table 4: Column description for fragDisp 31
Table 5: Column description for routeDisp 31
Table 6: Column description for ipDevDisp 32
Table 7: Column description for lev1Disp 33
Table 8: Column description for linkDisp 34
Table 9: Column description for basic linkDisp 35
Table 10: Column description for detailed linkDisp 35
Table 11: Column description for linkDisp-b 36
Table 12: Column description for linkDisp-bc 36
Table 13: Column description for sigtranDisp -t 37
Table 14: Column description for sigtranDisp –p 38
Table 15: Column description for sigtranDisp –u 38
Table 16: Column description for sigtranDisp –s 39
Table 17: Column description for netDisp 40
Table 18: Column description for monitorCounters 41
Table 19: Column description for monitorRcCounters 42
Table 20: Column description for monitorFilters 43
Table 21: Column description for monitorNetwork 44
Table 22: Column description for monitorAfPmiaNetwork 45
Table 23: Column description for global counters 46
Table 24: Column description for rc_cnt 47
Table 25: Column description for encap_lvl_cnt 47
Table 26: Column description for error_cnt 48
Table 27: Error code list for error_cnt counter 48
Table 28: Column description for filter_cnt 49
Table 29: Column description for pmia_dev 50
Table 30: Column description for pmia_sk 50
Table 31: Column description for pmia_kthreads 51

1.0 Introduction

1.1 Purpose and Scope

This Technical Reference document provides a basic overview of the tools and techniques that may be used to investigate and fix/correct issues related to xMF. It also contains procedures useful during writing PRs or capturing customer issues for later investigation.

Based on this document:

- Any Product Development team member should ideally be able to support a product or at least capture data necessary for later investigation. Access to the source code is expected.
- Any Customer Service or PV person should ideally be able to perform basic health check and end-to-end verification of the system base, support a product for first and second level support to resolve issues along with capturing data for level 3 investigations and to write a good PR.

This document is considered a live document that may be updated at any point by anyone. Therefore always get the latest version from the document repository -- do not email this document, as this promotes using stale versions. A desk review is recommended after every Feature Completion.

This document presumes reader's knowledge of xMF system and its components (along with their configuration and usage).

Note: This document is not intended to be a detailed description of the architecture and tools available to troubleshoot. It is meant only as a quick reference used by Customer Service or Engineering persons. No-one would read or properly update long documents. Real-live experience with the product and its troubleshooting cannot be substituted by any document and has to be passed during real support events. Initiative from support persons is expected to evaluate the product in their lab and work with development and FOA team to learn about it.

1.2 Product Summary

An xMF server is a part of the PIC solution offered by Tekelec for real-time monitoring of SS7, IP and GPRS networks in order to provide the necessary business information. xMF shall provide the real-time MSU data feed with the capability to set up specific data flows for a specific customer application.

1.3 References

[1] TEKELEC Acronym Guide, MS005077.DOC, Current Revision, Tekelec.

[2] Formal Peer Review Procedure, PD001866.DOC, Current Revision, Formal Peer Review Committee.

[3] Eagle Monitoring Protocol Technical Reference, TR005008, Revision 7.14, 2010, Tekelec.

[4] NetStat Troubleshooting TR005808.doc, Current Revision, Tekelec.

[5] Message Feeder Technical Reference.doc, TR005562, Current Revision, Tekelec.

[6] PMIA Expert Mode Technical Reference.doc, TR006014.doc, Revision 1.4, Tekelec.

[7] PIC 4.x to 6.0/6.1/6.6 Software-Upgrade Procedure 909-1580-001.doc

[8] SYSTEM INTERCONNECT_ROHS 5/6_IMF_T1000 DC_48 PORT SWITCH, 892-0084-01, Current Version

[9] SYSTEM INTERCONNECT_ROHS 5/6__IMF_T1100 DC_48 PORT SWITCH, 892-0078-01, Current Version

[10] SYSTEM INTERCONNECT ROHS 5/6 PMF T1100 DC 48 PORT SWITCH, 892-0079-01, Current Version

[11] SYSTEM INTERCONNECT ROHS 5/6 IMF T1200 DC 48 PORT SWITCH, 892-0091-08, Current Version

[12] NOC Control Frame IXP/NSP with Cisco 3560, 892-0089-01, Current Version

[13] NOC Control Frame IXP/NSP with Cisco 4948, 892-0089-02, Current Version
[14] NOC Extention Frame IXP/APP with Cisco 3560, 892-0090-01, Current Version
[15] NOC Extention Frame IXP/APP with Cisco 4948, 892-0090-02, Current Version
[16] PMF RACKED SYSTEM-G5, 892-0077-01,Current Version
[17] NOC CONTROL FRAME-G6_AC, 892-0098-01, Current Version
[18] NOC EXTENSION FRAME-G6_AC. 892-0099-01, Current Version
[19] PMF RACKED SYSTEM-G6_AC, 892-0101-01, Current Version
[20] IMF RACKED SYSTEM-G6_DC, 892-0102-01, Current Version
1.

1.4 Acronyms

Acronym	Definition
ADF	Application Data Feed
AIN	Advanced Intelligent Network
BSS	Base Sentinel Server
CCM	Central Configuration Management
DS	Data Server
DHCP	Dynamic Host Configuration Protocol
EMF	ESP Message Feeder
EMP	Eagle Monitoring Protocol
EMR	Event Message Report
ESP	Extended Service Platform
GUI	Graphical User Interface
GSM	Global System for Mobile Communications
PIC	Integrated Application Solution
ICP	Integrated CDR Platform
IXP	Integrated xDR Platform
IMF	Integrated Message Feeder
JMX	Java Management Extensions
MAP	Mobile Application Part
MF	Message Feeder
MFP	Message flow protocol
MLS	Monitor Link Status
NE	Network Element (the network element may be SSP, SCP or interconnect STPs)
NOC	Network Operations Center
NSP	Network Software Platform
NTP	Network Timing Protocol
PMF	Probed Message Feeder
SAMS	Sentinel Alarms Management System
SAP	Service Application Platform
SCMF	Site Collector Message Feeder
SNMP	Simple Network Management Protocol
SP	Signaling Point
SQL	Structured Query Language
STC	Sentinel Transport Card
STP	Signal Transfer Point

Acronym	Definition
TDR	Transaction Detail Record
TEAMPCP	**T**raffic, **E**MR, **A**larms, **M**LS, **P**rotocol Analysis, **C**DR, **P**eg
TPD	Tekelec Platform Distribution
TRP	Tekelec Routing Protocol
xMF	Integrated/Probed Message Feeder Product

2.0 What to do before requesting support

In the case when the person responsible for an xMF subsystem cannot investigate and correct the issue himself, it is recommended to work with the peer and if needed then call the next level of support (i.e. Tier1 would call Tier 2). This section describes what kind of information should be provided and what actions should be performed to speed up investigation and correction process before making such calls.

The first two subsections describes general steps which needs to be taken to ensure correct development team is called. Later subsections describe steps related to the xMF application.

2.1 Basic health check procedure

When experiencing problems with an xMF subsystem, basic health check should be run. See section 4.2.

2.2 Identify affected system

Due to the size and inter-connection of the PIC, it is inevitable that different parts of the PIC influence each other. In many cases the issue needs to be corrected in different part of the PIC system than where it was observed first.

When the issue is encountered in one part of the PIC application, it is necessary to do at least basic check of other parts of the PIC to ensure that support will be asked from correct development team.

2.3 Check system alarms

The alarms are basic identifiers of system health. Check all the recent alarms which are connected to the affected part of the PIC application and note them down. If possible, investigate its cause. The information on alarm conditions should be relayed along with the issue to the support engineer.

2.4 Identify affected DataFlow

Usually, there are more correlation, enrichment and storage DataFlows on one subsystem than on one server in subsystem. In order to speed up investigation, identify which DataFlow chain(s) are affected and on which servers they are distributed. Note this information down.

2.5 Check logs and traces

All the processes of the xMF application use COMCOL traces or other means for logging their progress and health. Check all the logs and traces related to the affected DataFlow chain and note down any recent error or unusual message found there.

Even the information like nothing unusual was found in logs can also help with the investigation.

Note: The logs can contain information which is days, weeks or even months old. Check date and time of any suspicious message to verify if it is recent and can be connected to the current issue.

2.6 Collect logs and configuration

Before raising a problem it is useful to note the system related information (logs and configuration), for example: type of hardware, PIC release number, EAGLE release number, features currently active (e.g.:- fast copy), type of PMF cards etc. The logs and configuration are collected for several purposes:

It can happen that the engineer does not have access to the Tekelec's Customer's network or his access is limited. The logs can be reviewed using just the basic tools.

The collected logs will remain available for later analysis of issue in case something has been missed or in case follow up or more in-depth analysis is required.

The information what was missing in collected data can be used to further enhance collection and analysis utilities.

The person asking for support should have this information prepared so that it can be sent to the engineer immediately as it is requested (via skype, e-mail, ftp …).

For detailed information about collecting procedure see section 4.1.

2.7 Prepare access to the system

In case that the issue requires more in-depth investigation than a simple advice, the support engineer may require access to the system. It is also possible that the engineer may not have access to the Tekelec network and the usual tools used for connection. Consider creating Webex meeting beforehand and supplying the connection information to the engineer when needed.

2.8 Identify support engineer

Use official list of the engineers responsible for the support of different PIC application to determine who should be called. All information about the issue discovered so far and any additional information should be relayed to the support engineer.

3.0 High level Functional description

3.1 Introduction

An xMF server is a part of the PIC solution offered by Tekelec for real-time monitoring of SS7 and IP networks. PIC includes a data acquisition subsystem that monitors SS7 and IP messages exchanged between various network elements in order to provide the necessary business information; the xMF is a part of this subsystem. The xMF server can either connect to Eagle STP (IMF) or to an IP, SS7 or GPRS network using a probe (PMF).

This chapter explains high level architecture of xMF component and then focuses on xMF interfaces with other components.

3.2 IMF Overview

In the figure below, one or more IMFs monitor an Eagle STP using the TR5008 interface. Using the TR5008 interface, the IMF servers receive configuration information, SS7/IP PDUs and alarms/events from the Eagle STP over an IP interface on a private, dedicated LAN eliminating the need for probes to monitor SS7 links on Eagle STP. Since the Eagle STP sends provisioning information to the monitoring system, the IMF automatically discovers and provisions signaling points, linksets and links provisioned in the Eagle STP. This eliminates the need for system administrators to manually provision this information. The PDUs received from Eagle STP are stored in a real-time database for fast shared access by common IMF downstream processes.

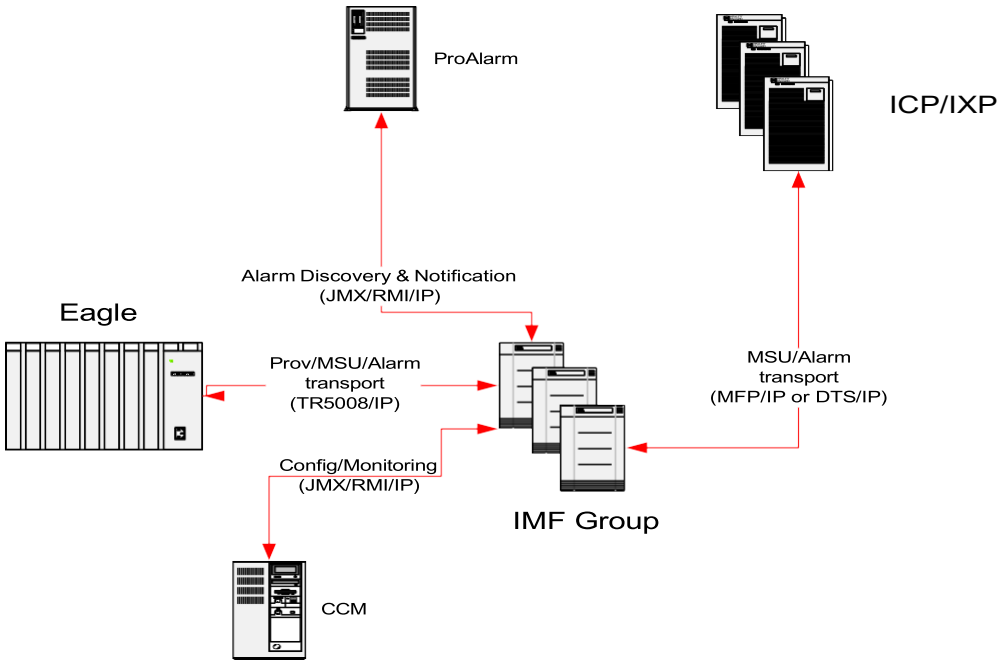


Figure 1: IMF System Overview

The Eagle Monitoring Protocol (EMP) is fully described in the TR005008 technical reference. EMP is briefly described here (from the IMF's perspective) for the convenience of the reader. The Eagle Monitor process on the IMF handles all EMP transactions with the Eagle STP.

When the EISCOPY/FASTCOPY feature is enabled on the Eagle STP, each LIM card broadcasts a "service request" message on both the yellow and blue networks on UDP port 1050. This message is repeated every fifteen seconds until an

IMF responds with a "service accept". Since the message is a broadcast, each IMF receives and parses it. The IMF designated as the IDB primary server checks the provisioning information contained in the message against that in the IMF's database. If the link does not already exist in the database, it is added / updated along with its linkset information. Newly added links and linksets are not assigned to an IMF for monitoring automatically. This information is first stored in the IDB tables having the Ccm prefix for the CCM to use for synchronization. The primary IMF does not send a response, however, unless it happens to be configured to monitor the link.

Each IMF that receives the service request also lookup to the link's information in the shared database. If the link is assigned to that IMF, the IMF responds to the LIM with a UDP unicast "service accept" message containing the connectivity information necessary to establish a TCP connection. On receiving this message LIM card establishes a TCP connection with the IMF and begins to send "link data" (MSU) and "event data" (LSSU) messages across it.

The eagleMonitor process inserts MSUs and LSSUs received from the Eagle STP into the IDB database on the IMF. A separate table is maintained for each link and data is inserted in the order it is received from EAGLE. LSSU event messages sent by the Eagle STP are entered into IDB only on state changes. In other words, repeated events are not entered. This protocol defines a FISU event as well, but the Eagle STP only sends them to the IMF to indicate that an otherwise idle link has changed state to "aligned, in service". As a result, only these FISUs are placed into the database. Generally, MSUs in the flow imply that the link is aligned and in-service as well.

The eagleMonitor process performs minimum processing of MSUs and events because LIM cards have limited buffer space and the eagleMonitor process runs at higher priority than most other applications on the IMF. Data loss is less likely to occur this way, since a busy system continues to acquire MSU data even if the processing falls behind.

The LIM card sends heartbeat messages at regular intervals only if there is no other traffic on the link. Any message from the LIM implies that the connection between it and the IMF is healthy. The IMF, on the other hand, always sends heartbeat messages back to the LIM at regular intervals since there is very little other traffic in that direction. Heartbeat timeouts on either side will cause the connection to be dropped.

Eagle STP sends OAM alarms on a separate connection, established with a different type of service request message. By convention, only one of these connections should be established. To keep the design simple, only the primary IMF will accept the OAM connection. This connection changes when the primary function moves. Traffic on this connection is usually very light and consists only of event messages and heartbeats.

3.3 PMF Overview

For probe monitoring, the PMF servers use interface specific PCI monitoring cards, such as DS0A, V.35, E1, T1 and ATM, to probe SS7 and IP links, using special cables, and patch panels or IPTAPs. The user provisions the Signaling Points/linksets/links and the physical interface parameters, using a web-based GUI. The monitoring probe target cards copy MSUs, HDLC frames, or IP packets and forward them to the Linux based host, using shared-memory based queues. The target cards also report periodic statistics and other events to the host. Processes on the host, copy the messages and events and store them in a real-time database for fast shared access by other downstream processes.

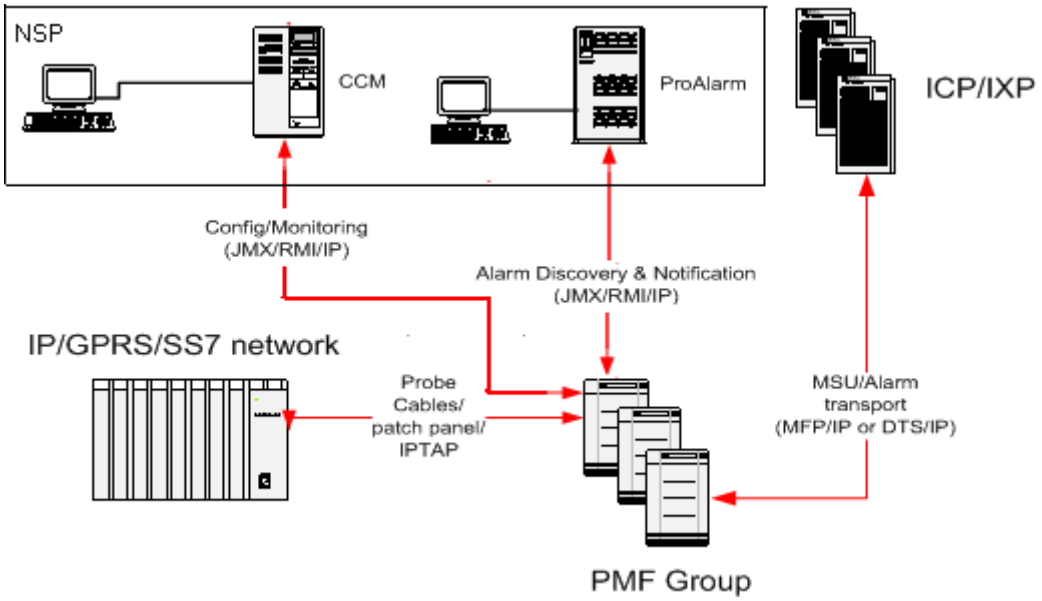


Figure 2: PMF System Overview

3.4 How does xMF work

After PDU's are captured and stored in IDB, they are read from the IDB by various processes residing within the xMF. The processes parse the received PDUs to generate operational measurement pegs and alarms. In addition, selected PDUs are also sent to the Integrated CDR Processors (ICP) / Integrated XDR Platform (IXP) or other servers, after passing through configurable set of filters. Another proprietary protocol, MFP/DTS, is used to transport PDUs from xMF to ICPs/IXPs. The xMF sends alarms to the ProAlarm server for further processing. The xMF system is provisioned and configured from the Central Configuration Management (CCM) using a web-based user interface.

The xMF system generates alarms when certain conditions occur. They are:

1. When the Eagle STP sends an SS7 network alarm message using the TR5008 protocol. These alarm conditions are detected by Eagle STP and sent to the IMF.
2. When the monitoring target card sends an event reporting physical hardware conditions or other monitored link related events.
3. While processing PDU, the xMF discovers certain alarm conditions pertaining to the SS7/IP network being monitored. Some of these alarms are generated following the Q.752 standards.
4. Internal alarms are generated when a process detects certain error conditions.
5. Platform alarms are generated when there is hardware, resource, or operating system-level failures are detected.

The xMF stores the alarms in the IDB database. jmxAgent communicates with the ProAlarm server to notify the system of these alarms. The ProAlarm server can also discover the type of the alarms supported by the system.

3.5 Database overview

3.5.1 MySQL

The MySQL server is an industry-standard open-source SQL database from MySQL, AB. It is a database system with complete SQL support. It also supports access from programs by providing ODBC and JDBC drivers as well as API for C/C++, Perl, PHP and many other programming languages that one can think of.

MySQL supports a pluggable storage engine. A storage engine is a general purpose data storage mechanism with a defined interface for accessing the data. Using the storage engine concept, one can add new types of storage to MySQL. Tekelec has developed IDB, an interface to the MySQL server using the storage engine concept. This enables IDB tables to be accessed from a MySQL interface using standard SQL language supported by MySQL. This make IDB SQL-capable as well as accessible via ODBC, JDBC, and other interfaces MySQL supports. The Java processes in the xMF server (currently OAM, in the diagram) uses the MySQL JDBC interface to access the IDB database.

3.5.2 IDB

The IDB is a database that resides in the Unix/Linux system shared memory. Therefore, multiple processes can access the data at the same time. The data is stored in tables that can be accessed by the xMF processes. IDB also provides backup storage to files for the purpose of providing permanent persistent storage. IDB comes with a low-level API that processes can use to access IDB tables (create, modify, delete, lookup elements) with access times comparable to standard SQL-type access. IDB also supports synchronization of tables across multiple xMF systems using the InetSync protocol as shown in the Figure 3: xMF Process Architecture. IDB also provides a number of command-line utilities to access and manipulate the database tables.

For the xMF system, the IDB provides the following functionality:

1. It acts a central repository for all xMF sub-system configuration information. The data is organized into tables. This includes configuration discovered by the IMF from the Eagle STP as well as information configured by the administrator like DataFlows, routes, destinations, filters and thresholds. All processes have access to this information. This configuration information is also synchronized across multiple xMF servers within the sub-system transparently using the InetSync interface.
2. The PDUs received from the Eagle STP and the probe target cards are stored in the database. This enables one process to write the PDUs into a table and other processes to parse and process the PDUs.
3. Using the COMCOL alarms framework, all alarms generated by the system are stored in the database in IDB system tables. COMCOL provides an API for writing alarms into this table.
4. All real-time monitoring pegs and measurements generated by various message processing processes are also stored in the database. The OAM process reads these measurements and sends them to the CCM on demand.

3.6 xMF Process Architecture

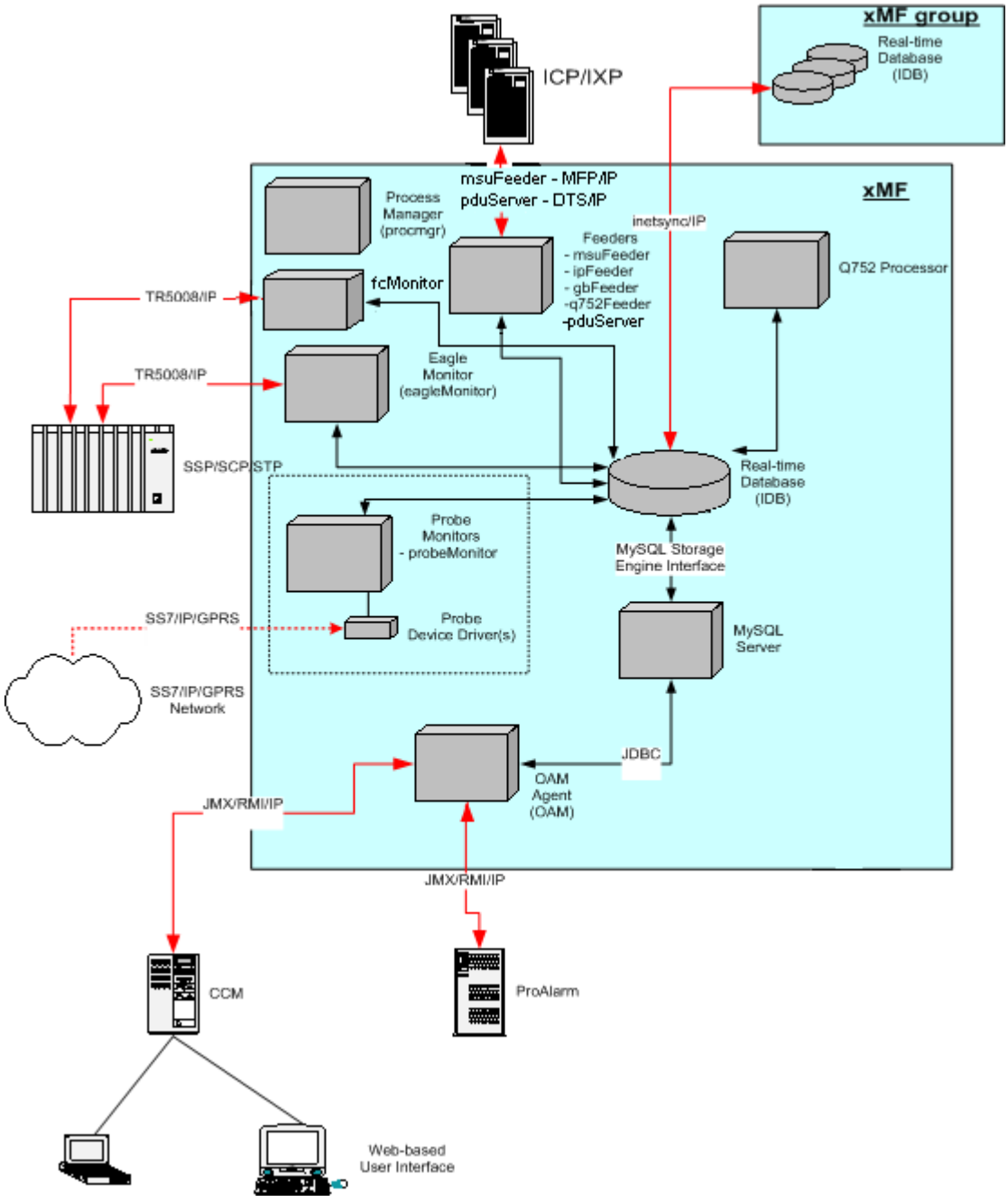


Figure 3: xMF Process Architecture

Process	Description
Procmgr	The process manager is a part of the COMCOL framework. This process starts and stops processes in an orderly manner based on the system configuration. It also acts as a watchdog and restarts processes that are terminated abnormally.
eagleMonitor (IMF)	The eagleMonitor process communicates with the Eagle STP using the TR5008 protocol interface. EM receives Eagle STP configuration information and uses this information to populate the links and linksets tables in the database. It also receives all the PDUs for the monitored links. The PDUs are stored in the database in the PDU table. In addition, it also receives alarms/events from the Eagle STP. It stores the alarms/events into the database for further processing. The IMF is designed to receive and store PDUs at a very fast rate by de-coupling the message receive logic from the message processing logic. The EM implements the message receive logic. The database acts as the PDU queue and other processes described below handles the message processing.
probeMonitor pmiaMonitor (PMF)	The Probe Monitor process shown in the above diagram receives PDUs and events from the monitoring target cards. This process reads PDUs from the SS7 hardware by reading from Shared Memory and writes the data to the database. The pmiaMonitor process receives IP packets from network cards. The process reads packets from the Linux network device using the libpcap interface and writes the data into the IDB database. This process is also able to do a basic filtering of messages based on source and destination IP address and port.
q752Gen	The q752Gen is one of the message processing processes. Its primary function is to generate Q.752 operational measurements counters and alarms. It reads PDUs from the database, and parses them. Based on the parsed PDU, it can determine if an alarm condition is set or an alarm condition is cleared. The alarms being monitored are defined by the Q.752 standards. If an alarm is set or cleared, it writes the information to the database for further processing. In addition, the q752Gen also maintains a set of operational measurements counters defines by the Q.752 standard. The counters are also written to the database.
msuFeeder pduServer ipFeeder gbFeeder(PMF) q752Feeder	The msuFeeder and pduServer are other message processing processes. Their primary function is to route selected MSUs to one or more ICPs/IXPs. They read the MSUs from the database and parse them. Then they apply a set of filter conditions defined in the database to determine if the MSU is to be forwarded to an ICP/IXP or not. If the MSU is to be forwarded, they determine the ICP(s)/IXP(s) that a MSU needs to be forwarded to and finally routes the MSU to one or more ICPs/IXPs. msuFeeder uses MFP protocol to feed ICP and pduServer uses DTS protocol to feed IXP. The ipFeeder, gbFeeder and q752Feeder are analogous processes to the MSU Feeder. ipFeeder and gbFeeder processes provide access to IP and GPRS traffic. Whereas, q752Feeder is used to send the measurements calculated using q752Gen to one or more ICP/IXP based on the DataFlow defined. Their structure and functionality is very similar to MSU Feeder, and the interfaces of these processes to the ICP/IXP are identical.

Process	Description
jmxAgent	The OAM Agent process (OAM) is a central point of contact for the ProAlarm systems. The OAM implements a Java Management Extensions (JMX) agent. The JMX architecture is defined by Sun Microsystems. It provides a framework that enables management applications to access network elements. The access may be in the form of: 1. Read/write access to "exposed" attributes. 2. Remote invocation of "exposed" operations 3. Notification of events to the management application. For more details, visit the JMX home page at http://java.sun.com/products/JavaManagement/. Tekelec has implemented a standard JMX agent as a part of the prOcess7 product and is used in multiple sub-systems. The OAM agent is basically the same as the Tekelec JMX agent with custom "MBEANS" that provide the xMF configuration, monitoring and alarm functionality. As shown in the diagram, the ProAlarm application accesses the OAM using the standard JMX/RMI interface. Using this interface, at connection time, the ProAlarm application requests the OAM agent for all the configured alarms. This is called the *Alarms Discovery Process*. The OAM agent reads the alarm information from the database and sends it out to the ProAlarm server. After that, it wakes up periodically, looks for new alarms/events from the database and reports them to the ProAlarm application. Similarly, the CCM connects to the OAM using the standard JMX/RMI interface for the purpose of viewing the xMF system configuration, modifying the system configurations and for live monitoring of link, route, process and transport measurements pegs. The OAM and the CCM communicate using remote invocations method supported by JMX. On receiving a request, the OAM accesses the database in order to process the request and depending on the request, it sends an appropriate response. For configuration operations, the user submits a HTML form to specify the parameters that is then translated into JMX invocations by the CCM. Depending on the operation, the OAM agent reads or writes data to the database and returns a result. For monitoring operations, the ProMonitor queries the A-Node periodically to refresh the user interface screen with latest values of the parameters being monitored.

Table 1: xMF process description

3.7 xMF Alarms

The IMF is designed to generate alarms or events from any process residing in the xMF. The COMCOL alarms API is a set of C++ classes that any process can invoke to generate alarms/events. The alarms are stored in the database and the OAM agent periodically reports the alarms to the ProAlarm server.

There are three major categories of alarms generated from IMF:

1) Tekserver Platform Alarms,
2) Eagle OAM Alarms,
3) SS7 Link Related Alarms.

xMF applications are running under Linux on Tekserver machine. Tekserver Platform Alarms are detected by the platform software – "syscheck". The syscheck daemon periodically checks the status of the Tekserver platform hardware/software and generates three alarm bitmasks. The bitmasks are:

- Critical Alarm Bitmask
- Major Alarm Bitmask
- Minor Alarm Bitmask

Each bitmask contains 64 bits. The meanings of the first 4 bits are given below:

- 0001 (0x1): Critical Alarms
- 0011 (0x3): Major Alarms
- 0101 (0x5): Minor Alarms

Logs generated by the syscheck utility are placed in /var/TKLC/log/syscheck/fail_log location. Example for "syscheck" is provided in the section 4.2.1

After each alarm is processed, it will be logged in the Event Log IDB table. The xMF MBean will be accessing the log using JDBC interface to retrieve the events and forward them to ProAlarm as RMI notifications.

Eagle OAM Alarms come to IMF through the TR5008 interface, and they are processed by the Eagle Monitor. The alarm definitions are stored in the Event Definition IDB table. Based on the information in the Event Definition table, the Eagle Monitor generates the events and stores them in the Event Log IDB table. All the events are forwarded to ProAlarm through xMF MBean in the JMX Agent.

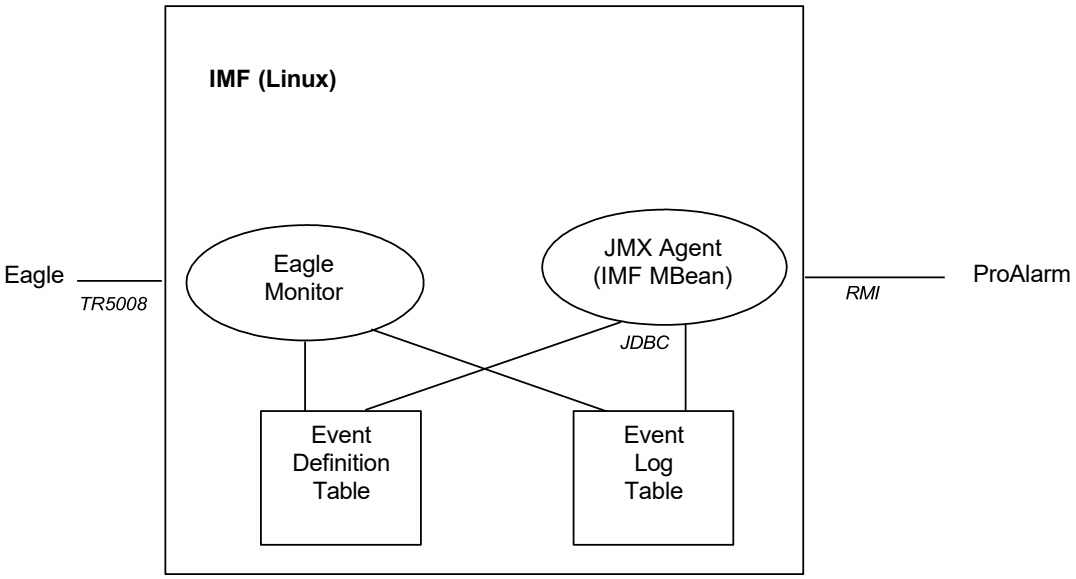


Figure 4: Eagle OAM Alarms Handling

SS7 Link Related Alarms are detected by checking the SS7 messages coming from Eagle STP. The SS7 messages are sent to IMF through the TR5008 interface. They get stored in the IDB tables by the Eagle Monitor. The Q752 Processor reads the messages and generates an entry in the Event Log IDB table when it detects an alarm condition.

With the same mechanism, alarms will be forwarded to ProAlarm by the xMF MBean in the JMX Agent through the RMI interface.

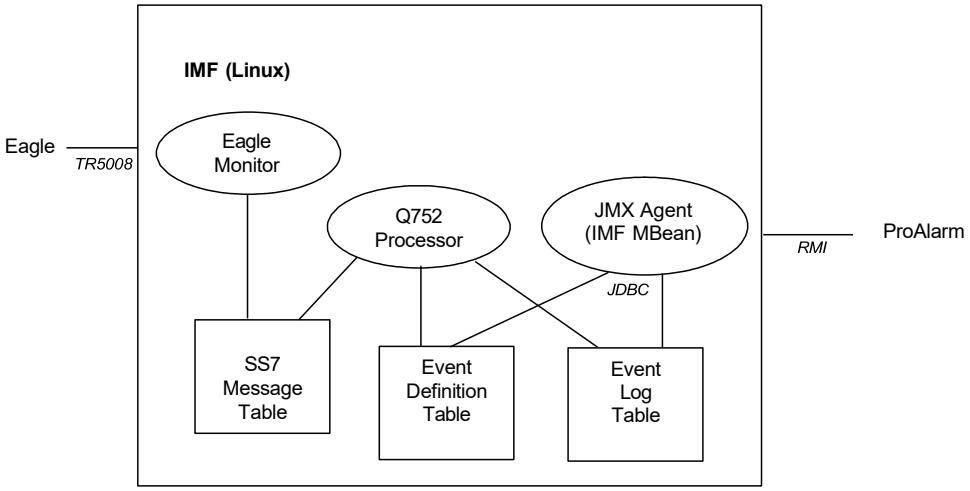


Figure 5: SS7 Link Alarms Handling

3.8 Shared Memory Allocation on xMF

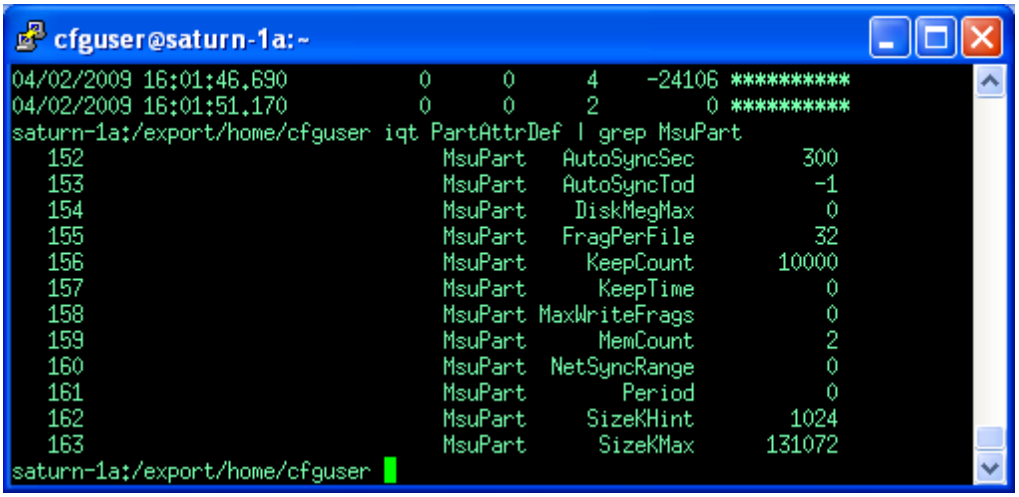
The xMF product uses the COMCOL library for managing shared memory. Shared memory is used for storing:

1. configuration of the server

2. the actual messages on the links being monitored

For monitored link there is one COMCOL table allocated for storing the messages, called the DbMsuBlock.*n* (where *n* = 0...50)

Each COMCOL table keeps shared memory fragments and the number is determined by MemCount.

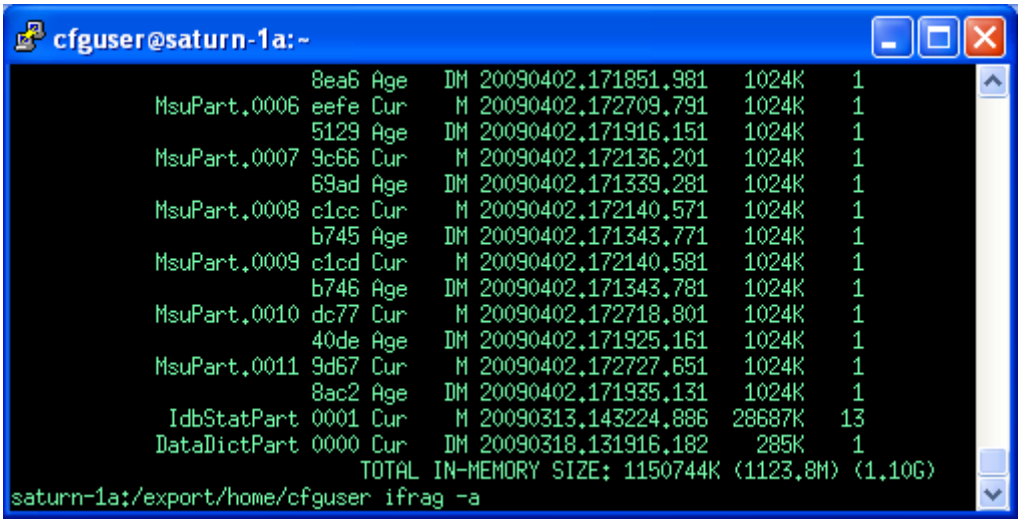


Each link being monitored writes its data to the associated shared memory fragment till it is full and then the shared memory gets flushed to disk.

Each reader reading from the link can either be looking for:

1. ***Current messages:***
   In this case the reader reads from the same memory fragment that the writer is writing.
2. ***Messages from the past***
   In this case the reader loads fragments from disk, one at a time

At any given time the command “ifrag –a “can show us the fragments in memory and the total memory used:



Problems that may appear:

1. ***Writer cannot flush:***
   If the writer cannot flush the written fragments to disk then there will be aged fragments in the memory (Age) beyond the allowed MemCount and these will stay till the writing process can flush these fragments to disk therefore increasing the memory usage. Process starvation, long IO waits … can cause this issue.

4.0 How to write good PR's

This section provides necessary information needed when submitting a PR against xMF.

It's essential to provide the PR with accurate information in order to perform successful reproduction (during investigation) or to find proper fix. Therefore when a problem is encountered, collect as much information as possible about the subsystem. It is also important to explicitly identify the difference between corrupted behavior and expected behavior and support it by screenshots and copy/paste text.

Note that it is NOT appropriate to include the customer name or any customer-specific information in the PR "Title", "Description", or "Customer Impact" fields because these PR fields may be seen by ANY customer in the Customer Known Report or Release Notes. The customer name should appear only in the "Company" field. Customer-identifiable, proprietary, confidential, or otherwise sensitive information must be put only in the fields of the "Investigation" page or in the "Notes" (Eng. Comments) field.

When problem is encountered collect as much information as possible about the subsystem.

4.1 Collect and analyze procedure

Since xMF 6.x this task is little less automated.

dumpScript (placed at /usr/TKLC/TKLCmf/bin /) is used for collecting all the logs and all the data that can be required to analyze the state and the operating environment of the server. For this dumpScript has to be executed on each of the server with root permissions on which data is to be collected and then the collected data has to be manually transferred to a safe location.

4.2 Health check

There is already basic pre and post upgrade health check procedure in Install/Upgrade documentation, but that is more upgrade related. The procedure in this document is supposed to be more complex and also include proposed actions for most common problems (or refer to section 8.0).

Corrupted behavior can occur due to many reasons, but it is important to check following things before considering it as a defect.

Login to server as cfguser	Login to server as **cfguser**

Run analyze tool	**`$ analyze_subsystem.sh`**
	The tool displays all details verification and should summarize the setup status:
	`IMF-1A TPD: 7.0.2.0.0-86.25.0 XMF: 10.1.5.0.0-4.13.0 0 test(s) failed`
	`IMF-1B TPD: 7.0.2.0.0-86.25.0 XMF: 10.1.5.0.0-4.13.0 0 test(s) failed`
	Test(s) failed status should return 0.

4.3 Other information

- Check process information
  - Use *`netstat -anp`* to verify that TCP queues are not full.

  - On xMF, execute *pm.getprocs* and verify that no process gets restarted.

- Check shared memory
  - On xMF, execute *ifrag -a* and *iaudit –ck* and verify that tables are OK.

- Check properly running traffic
  - Run `linkDisp -` and check, that traffic is properly received.
  - Run `routeDisp -` and check, that traffic is sent to IXP.

5.0 Logs and Traces

xMF processes generate log and trace messages.

The trace messages are used to debug the system. It is mainly intended for software developers and customer support personnel. The trace messages are not printed by default because of high system overhead. Only vital tracing is enabled. Different level of tracing can be turned on at a process level by setting the trace masks. The tracing infrastructure is provided by the COMCOL framework and an API is provided to the applications for printing trace messages. The setting of the masks and viewing the trace messages are done using command-line COMCOL utilities. The user has to create a Linux command line session in order to perform these operations.

Important system information and events are also logged. The COMCOL infrastructure is used for this purpose. Applications print log messages by using a COMCOL API. The log messages are stored in COMCOL internal tables and are available to the primary xMF server.

5.1 xMF processes

5.1.1 Overview

The xMF application is built using COMCOL framework and for successful operation it requires running instance of the IDB and the **procmgr** process. If any of them is not running, some commands will not be available.

You can use **prod.state** command for retrieving state of the TKLCmf service. Table 2: Listing of the TKLCmf service states contains list of possible states of the TKLCmf service and the description of active key processes in every state.

```
$ prod.state [↵ ]

               ...prod.state  (RUNID=00)...
               ...getting current state...
Current state: A (product under procmgr)
```

State	Description
NoDb	There is no disk version of IDB (and IDB is not loaded and procmgr is not running)
DbUP	IDB is evidently loaded but procmgr is not running
DbDown	IDB is evidently on disk but not loaded (and procmgr is not running)
A	Active - (application is up and running and the server is active from an HA perspective)
B	Standby - (application is up and running, but server is standby from an HA perspective)
O	Out-of-service (e.g., only procmgr, raclerk, and idbsvc run)

Table 2: Listing of the TKLCmf service states

5.1.2 Listing processes running under procmgr

Important xMF processes are managed by COMCOL procmgr process which starts and stops processes when moving between product states or starts processes which have died unexpectedly.

The processes on xMF server can be listed from terminal session using **cfguser** by running command:

```
# pm.getprocs –W [ ↵ ]
```

Example for PMF:

```
[cfguser@PMF0804-0A ~]$ pm.getprocs
S   pid procTag          $1     stat  spawnTime      N cmd
A  7201 Imysqld                Up    06/04 04:17:56 1 Imysqld.start -force
A  7200 MsuFeeder              Up    06/04 04:17:56 1 MsuFeeder -r 15
A  7199 NTPDeamon              Up    06/04 04:17:56 1 NTPDeamon
A  7195 ProcWatch              Up    06/04 04:17:56 1 ProcWatch
A  7205 chkLinkShm.pl          Up    06/04 04:17:56 1 chkLinkShm.pl
A  7211 cmha                   Up    06/04 04:17:56 1 cmha
A  7213 cmsoapa                Up    06/04 04:17:56 1 cmsoapa
A  7203 daqConfig              Up    06/04 04:17:56 1 daqConfig
A  7406 discoverPMFCards       Up    06/04 04:17:58 1 discoverPMFCards
A  7405 gbFeeder               Up    06/04 04:17:58 1 gbFeeder -r 15
A 18795 idbsvc                 Up    06/15 00:01:04 1 idbsvc -j -D75 -W5 -S50 -e -l0 -p10 -o 'Msu*'
A 31583 inetmerge              Up    06/15 01:29:27 1 inetmerge -C
A  7212 inetrep                Up    06/04 04:17:56 1 inetrep
A  7404 ipFeeder               Up    06/04 04:17:58 1 ipFeeder -r 15
A  7772 jmxAgent               Up    06/04 04:18:14 2 /opt/TKLCjmxagent/bin/startJmxAgentXmf.sh
A  7202 mfFailOver             Up    06/04 04:17:56 1 mfFailOver
A  7208 monStat                Up    06/04 04:17:56 1 monStat
A  7210 pduServer0             Up    06/04 04:17:56 1 pduServer -i 0 -I DTS -t TCP
A  7197 pm.watchdog            Up    06/04 04:17:56 1 pm.watchdog
A  7403 pmiaMonitor            Up    06/04 04:17:58 1 setRealTime -P20 pmiaMonitor
A  7207 q752Feeder             Up    06/04 04:17:56 1 q752Feeder -r 15
A  7206 q752Gen                Up    06/04 04:17:56 1 q752Gen
A  7193 raclerk                Up    06/04 04:17:56 1 raclerk -a0 -c30 -r1000
A  7196 re.portmap             Up    06/04 04:17:56 1 re.portmap -c100
A  7204 statMonitor            Up    06/04 04:17:56 1 statMonitor
A  7198 statclerk              Up    06/04 04:17:56 1 statclerk -s -0
A  7214 vipmgr                 Up    06/04 04:17:56 1 vipmgr
[cfguser@PMF0804-0A ~]$
```

Example for IMF:

```
[cfguser@IMF1002-1A ~]$ pm.getprocs
S   pid procTag          $1     stat  spawnTime      N cmd
A  1217 Imysqld                Up    06/02 07:12:18 1 Imysqld.start -force
A  3354 MsuFeeder              Up    06/02 07:18:48 1 MsuFeeder -r 15
A  1213 NTPDeamon              Up    06/02 07:12:18 1 NTPDeamon
A  1207 ProcWatch              Up    06/02 07:12:18 1 ProcWatch
A  1229 chkLinkShm.pl          Up    06/02 07:12:19 1 chkLinkShm.pl
A  1245 cmha                   Up    06/02 07:12:19 1 cmha
A  1254 cmsoapa                Up    06/02 07:12:19 1 cmsoapa
A  3356 daqConfig              Up    06/02 07:18:48 1 daqConfig
A  3358 eagleMonitor           Up    06/02 07:18:48 1 eagleMonitor --eagletimestamp
A  3361 fcMonitor              Up    06/02 07:18:48 1 setRealTime -P20 fcMonitor
A 21628 idbsvc                 Up    06/15 00:01:25 1 idbsvc -j -D75 -W5 -S50 -e -l0 -p10 -o 'Msu*'
A  1206 inetmerge              Up    06/02 07:12:18 1 inetmerge -C
A  3369 inetrep                Up    06/02 07:18:48 1 inetrep
A  3362 ipFeeder               Up    06/02 07:18:48 1 ipFeeder -r 15
A  1237 jmxAgent               Up    06/02 07:12:19 1 /opt/TKLCjmxagent/bin/startJmxAgentXmf.sh
A  3355 mfFailOver             Up    06/02 07:18:48 1 mfFailOver
A  3367 monStat                Up    06/02 07:18:48 1 monStat
A  3368 pduServer0             Up    06/02 07:18:48 1 pduServer -i 0 -I DTS -t TCP
A  1210 pm.watchdog            Up    06/02 07:12:18 1 pm.watchdog
A  3365 q752Feeder             Up    06/02 07:18:48 1 q752Feeder -r 15
A  3363 q752Gen                Up    06/02 07:18:48 1 q752Gen
A  1204 raclerk                Up    06/02 07:12:18 1 raclerk -a0 -c30 -r1000
A  1209 re.portmap             Up    06/02 07:12:18 1 re.portmap -c100
A  3360 statMonitor            Up    06/02 07:18:48 1 statMonitor
A  1212 statclerk              Up    06/02 07:12:18 1 statclerk -s -0
A  1223 trpd                   Up    06/02 07:12:19 1 setRealTime -P20 trpd
A  1256 vipmgr                 Up    06/02 07:12:19 1 vipmgr
[cfguser@IMF1002-1A ~]$
```

In command output, two most important columns are **spawnTime** and **N**. First one records time of last process restart, second one tracks how many times process was restarted since xMF service was started.

Note: Basically, processes restarts because of errors. If N column indicates process was restarted and if **spawnTime** column contains recent date and you should check process logs and traces.

5.1.3 Tracing

The latest trace messages can be displayed using **tr.tail <process name>** command:

```
# tr.tail eagleMonitor [ ↵ ]
```

```
+ cf.follow -10 /tekelec/TKLCmf/runtime/run/proc/eagleMonitor/trace.dat
0512:171503.007  TR10  M3UA SI not found at fourth offset
[17313/MfLinkWriter.C:1291]
0512:171503.007 TR5    [tid=27519920] ESPAppHandler::rxMsu -- <Exit
[17313/ESPAppHandler.C:346]
0512:171503.007 TR5    [tid=27519920] EMPPortMonitor::processLinkDataMsg -- <Exit
[17313/EMPPortMonitor.C:1464]
0512:171503.007 TR5    [tid=27519920] EMPPortMonitor::processRxMessage -- <Exit
[17313/EMPPortMonitor.C:1157]
0512:171503.007 TR5    [tid=27519920] EMPPortMonitor::handleData -- <Exit
[17313/EMPPortMonitor.C:967]
```

Amount of information written into traces by process depends on trace mask set for each informational area in that process. Trace mask setting can be displayed using **tr.show** command, using **grep** for filtering only important lines is highly recommended:

```
# tr.show | grep eagleMonitor [ ↵ ]
```

```
+ iqt <opts> TraceCntl where "traceName like '*'"
    eagleMonitor.APP          {}           No      eagleMonitor
    eagleMonitor.CML          {}           No      eagleMonitor
    eagleMonitor.COM          {}           No      eagleMonitor
     eagleMonitor.EV          {}           No      eagleMonitor
    eagleMonitor.IDB          {}           No      eagleMonitor
     eagleMonitor.TR          {}           No      eagleMonitor
```

First column lists different informational areas in one process and second column displays trace mask currently set for area.

5.1.4 Setting trace mas k and levels

Enabling all traces in one area can be done by **tr.setmask -1 <process name>.<area name>** command.

- Replacing **-1** with **0** in command will turn off all tracing for given area. Only the default information i.e. “VITAL TRACES” will be written in the trace files.
- Replacing **-1** with **4** in command will set trace mask to level **4**, which is default for most areas.

```
# tr.setmask none eagleMonitor.TR [ ↵ ]
```

```
+ iset -ftraceMask='0x0' TraceCntl where "traceName like'eagleMonitor.TR'"
  === changed 1 records ===
+ iqt <opts> TraceCntl where "traceName like 'eagleMonitor.TR'"
           traceName      traceMask       chg file bin
     eagleMonitor.TR          {}           No      eagleMonitor
```

To view specific set of traces following table can be referred to identify the mask corresponding to it.

Type of Trace Message	Mask Value	Description

TRMASK_ERROR	0x00000001	error
TRMASK_INFO	0x00000002	non-error information
TRMASK_WARN	0x00000004	unexpected warning
TRMASK_STATE	0x00000008	state transitions
TRMASK_FUNC_IN	0x00000010	function entry
TRMASK_FUNC_OUT	0x00000020	function exit
TRMASK_MSG_IN	0x00000040	incoming message
TRMASK_MSG_OUT	0x00000080	outgoing message
TRMASK_CONST	0x00000100	constructors
TRMASK_DEST	0x00000200	destructors
TRMASK_TPS	0x00000400	TPS information
TRMASK_TIMER	0x00000800	timer related information
TRMASK_PA_FILTER	0x00001000	protocolAnalysis filtering
TRMASK_TEST	0x80000000	for temporary special test

Table 3: Tracing mask value

The trace masks stated in the table above can also be combined together to get the desired logs. For example mask value for logs having messages corresponding to TR_MASK_ERROR, TRMASK_INFO and TRMASK_TPS would be 0x00000403.

5.1.5 Restarting processes

All xMF processes managed by **procmgr** can be restarted by using **pm.kill <procTag>** command.

If the invocation is successful, there will be no output to the command line to indicate that. You can verify the successful restart by running **pm.getprocs** and checking **spawnTime** column. It shall contain time of process restart. In case of unsuccessful invocation, error will be displayed into terminal. Most common error is mistyped **procTag.**

Following example shows successful restart with verification.

```
$ pm.getprocs
S   pid procTag          $1     stat spawnTime      N cmd
A 17292 MsuFeeder              Up   05/06 16:51:27 1 MsuFeeder -r 15
A 17293 eagleMonitor           Up   05/06 16:51:27 1 eagleMonitor --eagletimestamp
A 17294 fcMonitor              Up   05/06 16:51:27 1 setRealTime -P20 fcMonitor
A 17295 ipFeeder               Up   05/06 16:51:27 1 ipFeeder -r 15

$ pm.kill eagleMonitor

$ pm.getprocs
S   pid procTag          $1     stat spawnTime      N cmd
A 17292 MsuFeeder              Up   05/06 16:51:27 1 MsuFeeder -r 15
A  7937 eagleMonitor           Up   05/12 17:44:27 2 eagleMonitor --eagletimestamp
A 17294 fcMonitor              Up   05/06 16:51:27 1 setRealTime -P20 fcMonitor
A 17295 ipFeeder               Up   05/06 16:51:27 1 ipFeeder -r 15
```

The next command illustrates what happens when incorrect **procTag** is supplied on the command input – the error is emitted and no process is restarted**.**

```
$ pm.kill WrongProcTag

05/12/2010 17:46:00  pm.kill#31000{S/W Fault}
-F GN_NOTFOUND/FTL item not found in collection
  ^^ PmCtl::kill(sig=15, tag=WrongProcTag, actLine=-1)
```

```
^^ [8290:talkToPM.cxx:122]
```

5.2 jmxAgent

jmxAgent provides interface to xMF server for System Management GUI and for NSP discovery and management processes. Alarm forwarding is also done by jmxAgent. When it is not working properly, xMF server can't be configured through System Management.

The jmxAgent logs can be displayed from web interface and from command line interface.

To display logs using command line interface, you need to be logged in as **cfguser** or **root**. Log messages are stored in **/var/TKLC/log/jmxagent** directory in files **Agent.xml.<0-9>** and **/tekelec/TKLCmf/runtime/run/proc/jmx_agent.log** contains output of the process. The xml files are formatted for displaying in web interface, **jmxagent.log** is using COMCOL rotation files. The cf.follow command will display last 10 rows the log and new lines.

```
$ cf.follow -10 /tekelec/TKLCmf/runtime/run/proc/jmx_agent.log [↵ ]
```

```
May 12, 2010 5:44:39 PM Sending alarm: Event=57 EventData={ManagedObject:
name=.1.3.6.1.4.1.4404.2.1.6.1.1.16974595.16986625.0}
RootOID=.1.3.6.1.4.1.4404.2.1.0.
SourceOID=.1.3.6.1.4.1.4404.2.1.6.1.1.16974595.16986625.0 AdditionalText=Monitoring
is On theItu=24 Severity=5
May 12, 2010 5:44:39 PM Sending notification com.steleus.alarm with message
"Monitoring is On" and user data
"source=.1.3.6.1.4.1.4404.2.1.6.1.1.16974595.16986625.0 probableCause=24
perceivedSeverity=5 oid=.1.3.6.1.4.1.4404.2.1.0.57" from
.1.3.6.1.4.1.4404.2.1.6.1.1.16974595.16986625.0
May 12, 2010 5:45:12 PM Got 0 alarms when quering database for alarms with seqNum
>= 50946 and timeStamp >= 1304269494475
May 12, 2010 5:45:33 PM Got 0 alarms when quering database for alarms with seqNum
>= 50948 and timeStamp >= 1304269494475
```

5.3 DbReplicator

DbReplicator is a process responsible for synchronizing information about streams, servers and Q752 linksets from NSP. This information then can be imported by CCM. DbReplicator is running only on server designated as primary during xMF subsystem installation. Usually it is 1A server.

DbReplicator logs can be displayed only from command line interface.

Log messages are stored in **/tekelec/TKLCmf/runtime/run/proc/** directory. Other log files present in this directory are the **sshKeyXMF.log** and **jmx_agent.log**. Apart from this there are trace.dat files for all the running processes on the server.

The **tail** command can be used for displaying last messages in those files:

```
[root@imf66-1a proc]# tail –f /tekelec/TKLCmf/runtime/run/proc/dbreplicator.log
```

```
[2010-03-22 23:43:59,775] [RMI TCP Connection(1103)-10.240.23.239] INFO - View:
DbUserSSN: 0 rows changed since 2010-03-22 23:43:59.699, 0 rows added, 0 rows
modified, 0 rows deleted in 76 milliseconds
```

```
[2010-03-22 23:43:59,781] [RMI TCP Connection(1103)-10.240.23.239] INFO - View:
LongParam: 0 rows changed since 2010-03-22 23:43:59.778, 0 rows added, 0 rows
modified, 0 rows deleted in 3 milliseconds
```

```
[2010-03-22 23:43:59,785] [RMI TCP Connection(1103)-10.240.23.239] INFO - View:
DbHost: 0 rows changed since 2010-03-22 23:43:59.783, 0 rows added, 0 rows
modified, 0 rows deleted in 2 milliseconds

[2010-03-22 23:43:59,826] [RMI TCP Connection(1103)-10.240.23.239] INFO - View:
DbDsToLinkMap: 0 rows changed since 2010-03-22 23:43:59.787, 0 rows added, 0 rows
modified, 0 rows deleted in 39 milliseconds

[2010-03-22 23:43:59,838] [RMI TCP Connection(1103)-10.240.23.239] INFO - View:
DbGroup: 0 rows changed since 2010-03-22 23:43:59.829, 0 rows added, 0 rows
modified, 0 rows deleted in 9 milliseconds

[2010-03-22 23:43:59,844] [RMI TCP Connection(1103)-10.240.23.239] INFO - View:
DbIpLinkIntfMap: 0 rows changed since 2010-03-22 23:43:59.84, 0 rows added, 0 rows
modified, 0 rows deleted in 4 milliseconds

[2010-03-22 23:43:59,849] [RMI TCP Connection(1103)-10.240.23.239] INFO - View:
DbAssociation: 0 rows changed since 2010-03-22 23:43:59.846, 0 rows added, 0 rows
modified, 0 rows deleted in 3 milliseconds

[2010-05-22 02:38:33,437] [JMXAgent] INFO - Starting DBREPLICATOR...

[2010-05-22 02:38:33,451] [JMXAgent] INFO - Starting DB Replicator...

[2010-05-22 02:38:36,330] [JMXAgent] INFO - View Group: xmf-config: Started
replication task xmf-config
```

6.0 Troubleshooting and monitoring tools

6.1 disp tool

The tool contains menu with all *Disp tools in the xMF. You can run a specific tool and by pressing key regarding to following list.

NOTE: The disp tool was added to the PMF 2.0 so if you are using older release (IMF 2.2 or earlier or PMF 1.0) you need to run the *Disp tool from command line.

Run:

disp

Key :

- **f** - fragDisp - Message Feeder IP traffic fragmentation
- **r** - routeDisp - Data destinations MSU traffic and error rate
- **p** - ipInfoDisp - Traffic on IP Data Flows with XOR (IP addresses, XOR type, packet counts)
- **i** - ipDevDisp - List of IP devices and their traffic statistics (actual state, packets, frames, errors)
- **v** - lev1Disp - Low level statistics for Mistral cards
- **l** - linkDisp - Detailed statistics for the data links assigned to Message Feeder (signaling traffic - MSUs, SCCP, ISUP, ...)
- **s -** sigtranDisp - Detailed statistics for Sigtran(SCTP,M2PA,M3UA,SUA) assigned to Message Feeder
- **m** - mfDisp - Message Feeder statistics (resources usage, MSU traffic)
- **g** - avgInDisp - Overall input averages
- **n** - netDisp - Message Feeder statistics (The traffic statistics on Internet protocol layers - TCP, UDP, IP, ICMP)
- **a** - lastMsuDisp - Last MSU processed for each application
- **w** - wdDisp - Watchdog enabled process status
- **x,q** - Exit

6.1.1 frag Dis p

It is only for PMF 2.0 and newer versions. It shows the fragmentation of the traffic on the PMF frame.

Following columns are displayed on execution :

MsgFeeder	msgFeederId	show the hostname of the MF
Received	totIPFrags	Total IP fragments received by ipFeeder
Reassembled	totIPReas	Total IP fragments reassembled by ipFeeder
Failed	totIPBadFrags	Total bad IP fragments received by ipFeeder

Table 4: Column description for fragDisp

6.1.2 route Dis p

It shows the statistic for all destinations for the particular server. The overall statictics are on the 1A server only.

Following columns are displayed on execution :

DataSource	destId	unique ID for each record and pointer to DbDestination
St	state	values from "LinkStateType" (mentioned below the table)
cong	congestion	connection is congested or not
PduLost	pduRecLost	PDU records lost count
MsuSend	msuCount	MSU transmitted count
Fail	msuFailCount	MSU transmitted failure count
Msu/Sec	msuPerSec	MSU per second
Byt/sec	dataSndPerSec	data transmitted per second
Last State Time	stateTime	last link state time
mfId	msgFeederId	message feeder Id
tId	tid	thread id of the running thread for the destination

Table 5: Column description for routeDisp

The congestion information mentioned in the output actually tells about whether there is a congestion in the IXP-xMF network or not. This network congestion can be there due to following reasons :-

- IXPBuilder process is proceessing the packets at a slower rate as compared to the rate at which packets are being transmitted by xMF.
- There is a heavy traffic flowing on the network

Following are the values that LinkStateType (St) can take :-

UA - Link is not connected
AD - Link is administratively disabled (not used)
ND - Link is in the "No Data" state
OS - Link is Out of Service
O - Link is Out of Alignment
N - Link has entered Normal Alignment proving period
E - Link has entered Emergency Alignment proving period
A - Link is In Service
PO - Link is reporting Processor Outage
B - Link is reporting Busy
CONN - Link is connected but has not sent status

6.1.3 ipDevDis p

It shows a list of IP devices and their traffic statistics. All the IP devices are stored in the DbIPDevice table.

Following columns are displayed on execution :

Interface	id	Unique ID for IP device
State	rxState	state of receiving side (values at table "IpDevState" follows)
OK	rxPcapOk	Frame received (pcap library)
Lost	rxPcatDrop	Frame dropped (pcap library)
rxOk	rxOk	Rx OK (packet)
rxErr	rxErr	Rx ERROR (packet)
rxDrop	rxDrop	Rx DROP (packet)
rxFifo	rxFifo	Rx FIFO buffer error
rxFrame	rxFrame	Rx framing error

Table 6: Column description for ipDevDisp

Following are the values that IpDevState (state) can take :

Disconnect - Cable disconnected
Connect - Cable connected

6.1.4 lev1Dis p

The current level 1 statistics of a SS7 span is defined in an E1/T1 channel card or in an ATM card.

Level 1 alarms are RAI, LFA, AIS, LOS. Where acronym ends with "IS" - the current state of the level1 alarm for each way (RX or TX). Only two values are possible (yes: alarm active, no: alarm cleared). Rest of alarm‘s columns are counters of appearance of the level1 alarm for each way (RX or TX).

Following columns are displayed on execution :

MsgFeeder	msgFeederId	show the hostname of the MF
Slot	slotId	slot ID of the record
sId	subId	spanId for DbPMF_Channel and portId for DbPMF_ATM
RAI	raiRx	Remote alarm identification Rx counter
RAI_RX	raiRxIS	Current state of remote alarm Rx
RAI_Tx	raiTx	Remote alarm identification Tx counter
RAI_TX	raiTxIS	Current state of remote alarm Tx

LFA	lfaRx	Loss of Frame Alignment (or LOF) Rx counter
LFA_RX	lfaRxIS	Current state of Loss of Frame Alignment Rx
LFA_TX	lfaTx	Loss of Frame Alignment (or LOF) Tx counter
LFA_TX	lfaTxIS	Current state of Loss of Frame Alignment Tx
AIS	aisRx	Alarm identification Signal (only logical 1 on the span) Rx counter
AIS_RX	aisRxIS	Current state of Alarm identification Signal Rx
AIS_TX	aisTx	Alarm identification Signal (only logical 1 on the span) Tx counter
AIS_TX	aisTxIS	Current state of Alarm identification Signal Tx
LOS	losRx	Loss of Signal (appears when cable is not connected) Rx counter
LOS_RX	losRxIS	Current state of Loss of Signal Rx
LOS_TX	losTx	Loss of Signal (appears when cable is not connected) Tx counter
LOS_TX	losTxIS	Current state of Loss of Signal Tx
CRCi	crc4Rx	Input CRC error rate (CRC = Cyclic Redundancy Check)
CRCo	crc4Tx	Output CRC error rate (CRC = Cyclic Redundancy Check)
BERi	berRx	Input bit error rate
BERo	ber4Tx	Output bit error rate
_resetTime	time	last time the level1 stat were reset
FrmRx	frameRx	Input frame rate
FrmTx	frameTx	Output frame rate
Fi/S	avgFrameRx	Input average frame rate
Fo/S	avgFrameTx	Output average frame rate

Table 7: Column description for lev1Disp

6.1.5 linkDis p

It shows detailed statistics for the links assigned to Message Feeder (signaling traffic - MSUs, SCCP, ISUP, ...). The linkDisp has options like:

```
linkDisp -h
Usage: /opt/TKLCmf/bin/linkDisp [-m show merged | -M show merged with node] [-s
show stat] [-r show raw linkId] [<xMF Name> | -ls Linkset Names | -as
Association Names | -df PDU Dataflow Names | -d Destination Names | -c Card
Names] [-b bandwidth counters | -bc show bandwidth counters for cards | -dc
show disconnected links]
```

The linkDisp menu is based on iqt command (COMCOL tool). To get any information about its usage, use '**`man iqt`**' command.

The disp has a submenu for the linkDisp and the meaning of the menu items is:

```
linkDisp -r all # is equivalent for 'a' option in the disp submenu
linkDisp -r    # is equivalent for 'r' option in the disp submenu
linkDisp -s all # is equivalent for 'd' option in the disp submenu
linkDisp -s     # is equivalent for 's' option in the disp submenu
```

Following columns are displayed on execution

linkName	linkId	show the linkName or raw link ID (if is user '-r')
tbl	msuTable	table number for this link
stRx	stateRx	state of receiving side ("LinkStateType" value defined in6.1.2)
stTx	stateTx	state of transmission side ("LinkStateType" value defined in 6.1.2)
ocpRx	occupRx	percent occupancy received
ocpTx	occupTx	percent occupancy transmitted

Table 8: Column description for linkDisp

- **basic:** Columns from previous part and following columns are displayed on executing `linkDisp -r`

msuRx	msuRx	MSU received per second
msuTx	msuTx	MSU transmitted per second
sccpRx	sccpRx	SCCP received per second
sccpTx	sccpTx	SCCP transmitted per second
isupRx	isupRx	ISUP received per second
isupTx	isupTx	ISUP transmitted per second
sigRx	signetRx	network mgmt received per second
sigTx	signetTx	network mgmt received per second
losRx	flagLossRx	Flag Losses received counter
losTx	flagLossTx	Flag Losses transmitted counter
sErRx	suErrorRx	SU error received counter (SU = signaling unit)
sErTx	suErrorTx	SU error transmitted counter

rtmRx	retransRx	Retransmitted SU received counter
rtmTx	retransTx	Retransmitted SU transmitted counter
monitorTime	monitorTime	the time the monitoring data last updated

Table 9: Column description for basic linkDisp

- **detailed:** Columns from the top of this part and the folloing columns are observed on executing `linkDisp -s`

octetRx	octetsRx	Bytes received
octetTx	octetsTx	Bytes transmitted
msuRx	msuRxAccu	MSU received
msuTx	msuTxAccu	MSU transmitted
OOC	chgOver	change over counter
oSvc	outSvc	out of Service counter
oAln	outAln	out of aligned counter
nAln	nrmAln	normal aligned counter
eAln	emgAln	emergency aligned counter
Aln	aln	aligned counter
pOuR	proOutRx	processor outage transmit counter
pOuT	proOutTx	processor outage receive counter
bsyR	bsyRx	busy transmit counter
bsyT	bsyTx	busy receive counter
connect	connectStatus	possible states are DISC (disconnected), CONN (connected) and NA (not available)
monitorTime	monitorTime	the time the monitoring data last updated

Table 10: Column description for detailed linkDisp

- `linkDisp -b`

Columns displayed as part of linkDisp –b option.

This option displays bandwidth counters for all the links present in monitoring group.

msuRx	_msuRx	MSU received per second
msuTx	_msuTx	MSU transmitted per second

bytesRx	_bandwidthUnitRx	Number of bytes received per second
bytesTx	_bandwidthUnitTx	Number of bytes transmitted per second

Table 11: Column description for linkDisp-b

- `linkDisp -bc`

This option displays bandwidth counters for all the links present on the given card.

Usage: linkDisp –bc <cardName1, cardName2…cardNamen>

Columns displayed as part of linkDisp –bc option

msuRx	_msuRx	MSU received per second
msuTx	_msuTx	MSU transmitted per second
bytesRx	_bandwidthUnitRx	Number of bytes received per second
bytesTx	_bandwidthUnitTx	Number of bytes transmitted per second
bandwidthRx	Calculated using _ bandwidthUnitRx	Mbps received
bandwidthTx	Calculated using _ bandwidthUnitTx	Mbps transmitted

Table 12: Column description for linkDisp-bc

- `linkDisp -c`

Usage: linkDisp –c <cardName>

This option display all the link counters for all the links on the given card.

- `linkDisp -m`

  Usage: linkDisp –m

  This option display the merged data for the links.

- `linkDisp -M`

  Usage: linkDisp –M

  This option display the merged data for the links with the nodes.

6.1.6 s igtranDis p

It shows the detailed statistics for the associations assigned to the message feeder. The state & status corresponding to SCTP, M2PA, M3UA & SUA.

sigtranDisp –h or sigtranDisp

Usage: /opt/TKLCmf/bin/sigtranDisp {-t(SCTP)|-p(M2PA)|-u(M3UA)|-s(SUA)} [<xMF Name> | -ls Linkset Names | -as Association Names | -df PDU Dataflow Names | -d Destination Names]

sigtranDisp –t

It displays the state & status counters at the SCTP level for the associations. Some of the important fields are given below:

Status	Status of SCTP association
CtrlChunkRx	Number of control chunks received
CtrlChunkTx	Number of control chunks transmitted
CtrlBytesRx	Number of control bytes received
CtrlBytesTx	Number of control bytes transmitted
DataChunkRx	Number of data chunks received
DataChunkTx	Number of data chunks transmitted
DataBytesRx	Number of data bytes received
DataBytesTx	Number of data bytes transmitted
TotalRx	Number of total packet received
TotalTx	Number of total packet transmitted
TotalBytesRx	Number of total bytes received
TotalBytesTx	Number of total bytes transmitted

Table 13: Column description for sigtranDisp -t

sigtranDisp –p

It displays the state & status counters at the M2PAlevel for the associations. Some of the important fields are given below:

Status	Status of M2PA association
UDMRx	Number of data messages received
UDMTx	Number of data messages transmitted
UDMBytesRx	Number of data bytes received
UDMBytesTx	Number of data bytes transmitted
SccpRx	Number of SCCP messages received
SccpTx	Number of SCCP messages transmitted
IsupRx	Number of ISUP messages received
IsupTx	Number of ISUP messages transmitted
TotalMsgRx	Number of total packet received
TotalMsgTx	Number of total packet transmitted
TPSRx	Transaction per second received

TPSTx	Transaction per second transmitted

Table 14: Column description for sigtranDisp –p

sigtranDisp –u

It displays the state & status counters at the M3UAlevel for the associations. Some of the important fields are given below:

Status	Status of M3UA association
UDMRx	Number of data messages received
UDMTx	Number of data messages transmitted
UDMBytesRx	Number of data bytes received
UDMBytesTx	Number of data bytes transmitted
SccpRx	Number of SCCP messages received
SccpTx	Number of SCCP messages transmitted
IsupRx	Number of ISUP messages received
IsupTx	Number of ISUP messages transmitted
TotalMsgRx	Number of total packet received
TotalMsgTx	Number of total packet transmitted
TPSRx	Transaction per second received
TPSTx	Transaction per second transmitted
NdmRx	Number of non data messages received
NdmTx	Number of non data messages transmitted
MgmtMsgsRx	Number of management messages received
MgmtMsgsTx	Number of management messages transmitted

Table 15: Column description for sigtranDisp –u

sigtranDisp –s

It displays the state & status counters at the SUAlevel for the associations. Some of the important fields are given below:

Status	Status of SUA association
MgmtRx	Number of management messages received
MgmtTx	Number of management messages transmitted
MgmtBytesRx	Number of management bytes received
MgmtBytesTx	Number of management bytes transmitted
DmRx	Number of data messages received
DmTx	Number of data messages transmitted
DmBytesRx	Number of data bytes received
DmBytesTx	Number of data bytes transmitted
TotalRx	Number of total packet received
TotalTx	Number of total packet transmitted
TotalBytesRx	Number of total bytes received
TotalBytesTx	Number of total bytes transmitted
TpsRx	Transaction per second received
TpsTx	Transaction per second transmitted

Table 16: Column description for sigtranDisp –s

6.1.7 mfDisp

It shows resources usage and MSU traffic for the Message Feeder.

MsgFeeder	msgFeederId	show the hostname of the MF

System statistics

CPU %Usage	cpuUsage	CPU usage
Mem Free K	memUsage	Memory usage
Disk %Usage	diskUsage	Disk usage

Other statistics

MsuSend	msuCount	MSU transmitted count
Fail	msuFailCount	MSU transmitted failure count
Msu/Sec	msuPerSec	MSU per second
InKbps	inThroughput	input throughput Kbps per second
OutKbps	outThroughput	output throughput Kbps per second

6.1.8 avgInDisp

It shows estimations of MSU traffic received on customer network. Depending of the type of packets, some extract bytes are added to the receive packet length in order to make this estimation:

Packet type	Overhead bytes
MTP2	3
MTP2 enhancement	6
M2PA	17
M3UA	8

Statistics:

Kbps	msuCount	throughput Kbps per second
KBytes/s	msuFailCount	throughput KBytes per second
PDUs/s	msuPerSec	MSU per second
AvgSize	inThroughput	estimation of the average packets size

6.1.9 netDisp

It shows the count of the packets for the protocols TCP, UDP, IP and ICMP. The shown MFs are divided by line with '=' characters

Following columns are displayed on execution :

msgFeederId	shows the hostname of the MF
ipPktRcv	total IP packets received
ipForward	IP packet forwarded
ipInPktDiscard	incoming IP packets discarded
ipInPktSent	incoming IP packets delivered
ipReqSent	IP requests sent out
icmpRcv	ICMP messages received
icmpRcvFailed	input ICMP message receive failed
icmpInUnReach	ICMP input histogram: destination unreachable
icmpInEchoReq	ICPM input histogram: echo requests
icmpInEchoRep	ICMP input histogram: echo replies
icmpSent	ICMP messages sent
icmpFail	ICMP messages failed
icmpOutUnReach	ICMP output histogram: destination unreachable
icmpOutEchoRep	ICMP output histogram: echo replies
tcpActConn	active TCP connection openings
tcpPasConn	passive TCP connection openings
tcpConnFail	failed TCP connection attempts
tcpConnReset	TCP connection resets received
tcpConnEst	TCP connections established
tcpSegRcv	TCP segments received
tcpSegSend	TCP segments send out
tcpSegReTran	TCP segments retransmitted
tcpBadSegRcv	bad TCP segments received
tcpResetSent	TCP resets sent
udpPktRcv	UDP packets received
udpUnkwnPort	UDP packets to unknown port received
udpRcvError	UDP packet receive errors
udpPktSent	UDP packets sent

Table 17: Column description for netDisp

6.2 cfgPmia

This tool provides real-time monitoring on IP networks.

NOTE: The cfgPmia tool was added to the PMF 6.0

Units: F/s frames per second, kB/s kilo bytes per second (1000)

6.2.1 monitorCounters

It shows detailed counters used by current configuration. This list of counters is mentioned below.

Command:

```
cfgPmia monitorCounters    # display counters with default refresh time 2 sec
```

```
cfgPmia monitorCounters 5 # display counters with refresh time set at 5 sec.
```

Following columns are displayed on execution :

current time , last refresh and refresh time

Global input counters of PMIA module

itf	Interface identifier (0-7)
eth_too_long	Number of packets too long
eth_too_short	Number of packets too small
eth_not_ip	Number of packets not IPV4
packets_rcv	Number of packets received
bytes_rcv	Number of bytes received
average_size	Average size of packets received
packets_fwd	Number of packets forwarded
frag_decision	Number of packets routed with decision already done (session use)

Table 18: Column description for monitorCounters

Input/Routed/Drop frames per ports in AF_PMIA module

PmiaId	Pmia Identifier
PortName	Number of packets too long
SysId	system Identifier
RefCnt	reference counter
RcvTot	total packets received
RcvRt	total packets routed
RcvDrp	total packets dropped

Traffic counters for each traffic classification

Filter	Traffic classification identifier
Rx (F/s)	Received frames per sec.
Rx (kB/s)	Received bytes per sec.
Drop (F/s)	Dropped frames per sec.
Drop (kB/s)	Dropped bytes per sec.

Network counters per ports

I/F_name	Interface name
Rx (F/s)	Input frames per sec.
Rx (kB/s)	Input bytes per sec.
Tx (F/s)	Output frames per sec.
Tx (kB/s)	Output bytes per sec.
RxErrs/RxDrop/ RxFifo/RxFrame	errors since last tool start

Interrupt counters per ports

Itf	Interface name
IRQ	Interrupt number
CPUX (I/s)	Interruptions per sec for each processor

Traffic counters processed by processor :- Display number of frames and bandwidth filtered by PMIA for each processor.

Example

```
2009-08-19 10:49:16.626    refresh time   2.002
------------------- PMIA ------------------
itf                      0|      Total|Units
---------------- PMIA Input ---------------
eth_too_long             0|          0|  F/s
eth_too_short            0|          0|  F/s
eth_not_ip               0|          0|  F/s
packets_rcv         120575|     120575|  F/s
bytes_rcv            27629|      27629| kB/s
average_size           229|        229|  B
packets_fwd              0|          0|  F/s
frag_decision            0|          0|  F/s
------------------------------------------------ AF_PMIA DEV -------------------------------------------------
PmiaId   0      eth41 SysId  11 RefCnt  1 RcvTot    120929 (F/s) RcvRt    120929 (F/s) RcvDrp         0 (F/s)
--------------------------------------------------------------------------------------------------------------
---------------------- AF_PMIA SK ----------------------
Filter    Rx (F/s)    Rx (kB/s)  Drop (F/s) Drop (kB/s)
     0       66492        15300       53962           0
   254           0            0           0           0
-------------------------------------------------------
 Total       66492        15300       53962           0
-------------------------------------------------------
------------------------------------------ PROC_NET_DEV -------------------------------------------
I/F name       Rx(F/s)     Rx(kB/s)      Tx(F/s)     Tx(kB/s)       RxErrs       RxDrop       RxFifo      RxFrame
     eth41      118605        27651            0            0            0            0            0            0
---------------------------------------------------------------------------------------------------------------
---------------------------------------- PROC_INTERRUPTS ----------------------------------------
Itf    IRQ  CPU0 (I/s)  CPU1 (I/s)  CPU2 (I/s)  CPU3 (I/s)  CPU4 (I/s)  CPU5 (I/s)  CPU6 (I/s)  CPU7 (I/s)
eth41   67           0       19882           0           0           0           0           0           0
---------------------------------------------------------------------------------------------------------
---------------------------------------- AF_PMIA KTREADS ----------------------------------------
            CPU0       CPU1       CPU2       CPU3       CPU4       CPU5       CPU6       CPU7|        ALL
 F/s:     120565          0          0          0          0          0          0          0|     120565
kB/s:      27626          0          0          0          0          0          0          0|      27626
-----------------------------------------------------------------------------------------------------
```

6.2.2 monitorRc Counters

It shows detailed counters for each traffic classification at the PMIA output, used by current configuration.

Command:

```
cfgPmia monitorRcCounters   # display counters with default refresh time 2 sec.

cfgPmia monitorRcCounters 5 # display counters with refresh time set at 5 sec.
```

Following columns are displayed on execution :

current time and last refresh

itf	Interface identifier (0-7)
rc X	Number of packets for traffic classification id X
rc X	Number of bytes for traffic classification id X

Table 19: Column description for monitorRcCounters

Example

```
2009-08-19 10:52:11.451    refresh time   2.001
------------------- PMIA ------------------
itf                     0|      Total|Units
---------------- PMIA Output --------------
rc   0             120272|     120272|  F/s
rc   0              27559|      27559| kB/s
-------------------------------------------
```

6.2.3 monitorFilters

It shows detailed average counters of number filter lines used by current configuration.

Command:

```
cfgPmia monitorFilters   # display counters with default refresh time 2 sec.

cfgPmia monitorFilters 5 # display counters with refresh time set at 5 sec.
```

Following columns are displayed on execution:

current time and last refresh time

UsedLines	number of PMIA lines ID used
ActiveLines	Average number of PMIA lines used by each frame (L/F)
SumPackets	Number of packets per second
SumUsedLines	Number of PMIA lines used per second

Table 20: Column description for monitorFilters

Example

```
2009-08-19 10:53:04.585    refresh time   2.001
-------------------------------------------------------------------------------------------
UsedLines    2 ActiveLines    2 SumPackets(F/s)      119020 SumUsedLines(L/s)      238041
-------------------------------------------------------------------------------------------
```

6.2.4 monitorNetwork

It shows detailed network counters of xMF.

Command:

```
cfgPmia monitorNetwork   # display counters with default refresh time 2 sec.

cfgPmia monitorNetwork 5 # display counters with refresh time set at 5 sec.
```

Following columns are displayed on execution:

current time and last refresh time

I/F name	Interface name
Rx(F/s)	received packets per sec.
Rx(kB/s)	received kilobytes per sec.
Tx(F/s)	transmitted packets per sec.
Tx(kB/s)	transmitted kilo bytes per sec.
RxErrs/RxDrop/RxFifo/RxFrame	error counters

Table 21: Column description for monitorNetwork

Example

```
2009-08-19 10:53:43.497    refresh time   2.001
------------------------------------------------ PROC_NET_DEV -------------------------------------------------
I/F name       Rx(F/s)    Rx(kB/s)     Tx(F/s)    Tx(kB/s)      RxErrs      RxDrop      RxFifo     RxFrame
        lo          17           6          17           6           0           0           0           0
     eth01           8           0           7           1           0           0           0           0
     eth02           0           0           0           0           0           0           0           0
     eth54           0           0           0           0           0           0           0           0
     eth53           0           0           0           0           0           0           0           0
     eth52           0           0           0           0           0           0           0           0
     eth51           0           0           0           0           0           0           0           0
     eth44           0           0           0           0           0           0           0           0
     eth43           0           0           0           0           0           0           0           0
     eth42           0           0           0           0           0           0           0           0
     eth41      122944       28663           0           0           0           0           0           0
-------------------------------------------------------------------------------------------------------------
Total           122969       28669          24           7
-------------------------------------------------------------------------------------------------------------
```

6.2.5 monitorAfPmiaNetwork

It shows detailed network counters of input ports used by current configuration.

Command:

```
cfgPmia monitorAfPmiaNetwork   # display counters with default refresh time 2 sec.

cfgPmia monitorAfPmiaNetwork 5 # display counters with refresh time set at 5 sec.
```

Following columns are displayed on execution:

current time and last refresh time

I/F name	Interface name
Rx(F/s)	received packets per sec.
Rx(kB/s)	received kilo bytes per sec.
Tx(F/s)	transmitted packets per sec.
Tx(kB/s)	transmitted kilo bytes per sec.

RxErrs/RxDrop/RxFifo/RxFrame	error counters

Table 22: Column description for monitorAfPmiaNetwork

Example

```
2009-08-19 10:55:04.169    refresh time   2.001
------------------------------------------- PROC_NET_DEV ---------------------------------------------
I/F name       Rx(F/s)    Rx(kB/s)     Tx(F/s)    Tx(kB/s)      RxErrs      RxDrop      RxFifo     RxFrame
     eth41      113148       26379           0           0           0           0           0           0
-----------------------------------------------------------------------------------------------------
```

6.2.6 getCounters

This command permits to get all the absolute counters.

Command:

```
cfgPmia getCounters
```

6.2.7 res etCounters

This command permits to reset all counters of PMIA module (remark: this command doesn't reset counters from AF_PMIA module).

Command:

```
cfgPmia resetCounters
```

6.3 Pmia module proc files

These files provide PMIA module counters. This module implements the PMIA filtering/routing functions.

Proc files location: /proc/driver/pmia

6.3.1 global

This proc file provides global counters from PMIA module.

Command:

```
cat /proc/driver/pmia/global
```

Following columns are displayed on execution:

itf	Interface identifier (0-7)
eth_too_long	Number of packets too long
eth_too_short	Number of packets too small
eth_not_ip	Number of packets not IPV4
packets_rcv	Number of packets received
bytes_rcv	Number of bytes received
average_size	Average size of packets received
packets_fwd	Number of packets forwarded
frag_decision	Number of packets routed with decision already done (session use)
frag_coll	Number of IP fragments that could not be inserted into the fragmentation table
frag_ovl	Number of opened context in the fragmentation table that has been removed before timeout is reached (reach max. limit size)
session_coll	Number of IP packets that could not be inserted into the session table
session_ovl	Number of opened context in the session table that has been removed before timeout is reached (reach max. limit size)
evt_ref_id	number of events sent to socket 254 (this event is attached to a packet sent to socket 254 to identify a re-assembly context)
evt_snd_ref_id	This event is sent to socket 254 to inform PMIA level 2 storage in which destinations send packets attached to a given reference id
stack_used	Stack used in bytes by PMIA
frag_items	Current number of items in IPV4 fragment list
session_items	Current number of items in session list

Table 23: Column description for global counters

Example

```
itf                         0
eth_too_long                0
eth_too_short               0
eth_not_ip                  0
packets_rcv          29334331
bytes_rcv          6721751970
average_size              229
packets_fwd                 0
frag_decision               0

frag_coll                   0
frag_ovl                    0
session_coll                0
session_ovl                 0
evt_ref_id                  0
evt_snd_ref_id              0
stack_used                314
frag_items                  0
session_items               0
max_ttl                     0
```

6.3.2 rc_cnt

This proc file provides return code counters from PMIA module.

NOTE: A return code (RC) is equal to a traffic classification identifier (TC). Normal value from traffic classification are 0 to 249. Values over 249 are reserved. (255: PMIA error, 254: forward, 253: internal storage, 250: trash)

Command:

```
cat /proc/driver/pmia/rc_cnt
```

Following columns are displayed on execution:

itf	Interface identifier (0-7)
id	Identifier of the return code (0 to 255). Display only if at least one of the counters is not null.
packets	Number of packets sent to the current RC
bytes	Number of bytes sent to the current RC

Table 24: Column description for rc_cnt

Example

```
itf  id              packets              bytes
  0   0             42975375         9847500028
```

6.3.3 encap_lvl_cnt

This proc file provides IPV4 fragment counters from PMIA module.

Command:

```
cat /proc/driver/pmia/encap_lvl_cnt
```

Following columns are displayed on execution:

lvl	Encapsulation level
frag_0	Number of IPV4 fragmented packets with fragment offset 0
frag_X	Number of IPV4 fragmented packets with fragment offset > 0

Table 25: Column description for encap_lvl_cnt

Example

```
lvl              frag_0              frag_X
  1              119878              123445
```

6.3.4 error_cnt

This proc file provides error counters from PMIA module.

Command:

```
cat /proc/driver/pmia/error_cnt
```

Following columns are displayed on execution:

itf	Interface identifier (0-7)
id	Error code type (see error code list)
value	Number of packets

Table 26: Column description for error_cnt

Error code list

Error code	Description
1	System error (environment, stack, memory allocation)
2	Looping packet on PacketTreatment function, TTL reach zero
3	Data too short, frame length smaller than IP header length
4	IP version error
5	Packet too small, IP total length too short
6	Packet too large, IP frame length greater than MTU 9000
7	Looping packet on main Filter function on entry, TTL reach zero
8	RC list error, RcList limitation reached
9	Looping packet on main Filter function into loop on filter lines, TTL2 reach zero
10	Filter position error, exceed the max number of filter lines
11	Filter length error
12	Push list error, max number of context is reached or Var[1] error (Var[1] exceed the max number of filter lines)
13	None
14	Push list error, on action FILTER_ACTION_CODE_POP_CTX
15	Push list error, on action FILTER_ACTION_CODE_END
16	Action code error (Action not exist)
17	Looping packet on Tag function, TTL reach zero
18	Looping packet on ShiftPacketLeft function, TTL reach zero
19	Looping packet on Fragment0 function, TTL reach zero
20	Looping packet on OtherFragment function, TTL reach zero
21	None
22	Data.FilterActions error in Fragment0 function
23	Data.FilterActions error in OtherFragment function
24	SetOffset error, action FILTER_ACTION_CODE_SET_OFFSET

Table 27: Error code list for error_cnt counter

Example

```
itf  id                value
```

6.3.5 filter_cnt

This proc file provides filter line counters from PMIA module.

Command:

```
cat /proc/driver/pmia/filter_cnt
```

Following columns are displayed on execution:

itf	Interface identifier (0-7)
id	Filter line identifier with packets
packets	Number of time the filter line is used

Table 28: Column description for filter_cnt

Example

```
itf   id              packets
  0    0             54026658
  0    1             54026658
```

6.3.6 max_line

This proc file provides the maximum number of lines from PMIA module.

Command:

```
cat /proc/driver/pmia/max_line
```

Example

```
2048
```

6.4 Af_Pmia module proc files

These files provide AF_PMIA driver module counters. This module creates links between Ethernet ports and AF_PMIA traffic classification sockets.

Proc files location: /proc/net

6.4.1 pmia_dev

This proc file provides counters per interface from AF_PMIA module.

Command:

```
cat /proc/net/pmia_dev
```

Following columns are displayed on execution:

dev	interface name
pmiaid	pmia identifier
sysid	system identifier
refcnt	reference counter
rcvtot	packet received
rcvrt	packet routed
rcvdrp	packet dropped

type	interface type

Table 29: Column description for pmia_dev

Example

```
dev         pmiaid      sysid       refcnt      rcvtot      rcvrt       rcvdrp      type
eth41       0           11          1           128510389   128510389   0           PMIA
```

6.4.2 pmia_s k

This proc file provides socket counters from AF_PMIA module.

Command:

```
cat /proc/net/pmia_sk
```

Following columns are displayed on execution:

sk	socket pointer
filter	traffic classification identifier (-1 equal control socket)
rxcnt	packet received
rxB	bytes received
drpcnt	packet dropped
drpB	bytes dropped

Table 30: Column description for pmia_sk

Example

```
sk          filter      rxcnt       rxB         drpcnt      drpB
f73b5000    0           65548023    2190986922  56920701    0
f7220800    -1          0           0           0           0
f73af600    254         0           0           0           0
f73ada00    -1          0           0           0           0
```

6.4.3 pmia_kthreads

This proc file provides packets counters from AF_PMIA module for each processor. This command is only for PMF.

Command:

```
cat /proc/net/pmia_kthreads
```

Following columns are displayed on execution:

cpu	processor identifier

frames	number of frames processed by this processor
bytes	number of bytes processed by this processor

Table 31: Column description for pmia_kthreads

Example

cpu	frames	bytes
0	130907892	29996608760
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0

7.0 Solving specific problems

This chapter contains list of most common problems found on xMF and notes how to solve them.

7.1 IDB Database Access Utilities

COMCOL IDB is the in-memory database developed by Tekelec. IDB (in memory database) is well suited to telecommunications applications with a variety of interfaces including C++ native interfaces, perl, and SQL. The integration of the IDB and MYSQL makes GUI applications easier to develop since MySQL provides well defined utilities. This section provides various commands that can be used to monitor, update and analyze the IDB :-

Some common commands used for IDB table access, and manipulation. Please refer to man pages on XMF server or to comcol WIKI pages to have more information about each command:

- *itable*
- *ifield*
- ikey
- imkload
- ivi
- ifrag
- iqt
- *igrep*
- iaudit
- ...

If a problem is detected and any COMCOL mechanisms are suspected, use `cm.sysdiag -Z` command. This command creates a complete picture of comcol logs and debugging commands. Comcol team uses this input for all investigation.

For correcting minor problems following commands can be used :

- `iaudit -cf`
- `idbpartinit can rebuild a part based on some good files (for more info please see it's man page)`

For taking the back-up of the IDB following commands can be used: see comcol WIKI pages.

7.2 MSU/Event Access Utilities

For the MSU/Event access utilities we have following set of commands

```
dataRd [-b begin] [-e end] [-d data type {M|E}] [-u(nsort] [-a(udit)] [-c(ount
only)] [-t(ail)] [{-o outfile|-v(erbose)|-V(erbose)] [-T(otal MSU generated)] [-
s(tatistics)] [-f(ilter on SIO) <sio>] {Link...|all}
```

dataRd reads MSU/Event from IDB table and dump the contents. –v shows the table number, timestamp, type, size and direction. –V also dump the MSU data in hex, Ascii characters and decimal values. The <begin>/<end> are timestamp with the format of "mm/dd/yy hh:mm:ss" with double quotes.

```
dataRd -i infile [-c(ount only)] [-v(erbose)|-V(erbose)] [-s(tatistics)] [-f(ilter
on SIO) <sio>]
```

Similar previous but reading to the Message Buffer File instead of IDB tables.

```
dataWr [-o based on Original time ] [-r Realtime insertion, imply -o] [-e Event gen
percentage ] [-s  Statistics] [-S Statistics only] [-v Verify only] [-V
Verify/detail] [-c  repeat Count, -1 for infinite] –i <msgbufferFile>
```

write to MSU/Event IDB tables from a Message Buffer File

```
dataAccess
```

Interactively access the MSU/Event from an IDB table. The commands include:

[O]pen linkId, [B]lock [cnt], [F]irst [time], [N]ext [cnt], [L]ast [time], [P]rev [cnt], [T]ail [cnt], [D]etail toggle [Q]uit

Note: the <time> is "mm/dd/yy hh:mm:ss" with double quotes

7.3 Process Management Utilities

Process management features including starting processes, restarting processes, abnormal termination handling, scheduling processes, and leak detection.
Following are the different popular commands used for process management. Please refer to man pages on XMF server or to comcol WIKI pages to get more information about each command:

- `prod.state`
- `prod.start`
- `prod.stop`
- `prod.dbup`
- `prod.dbdown`
- `prod.clobber`
- `prod.recover`
- `pm.getprocs`
- `procshm`
- `pm.kill`
- `pm.set`
- `procstat`
- `ProcWatch`

7.4 COMCOL Tracing Utilities

COMCOL (Communications Management Core Object Library) is an infrastructure for building telecommunications (and other) applications on UNIX and Linux operating systems; COMCOL consists of a collection of C++ objects and subsystems. One of the most important features of COMCOL is program tracing feature.
Program trace feature permits high efficiency tracing, run-time controls, memory mapped "circular file" output. Trace is (almost) never conditionally compiled making trace available when needed.

Please refer to man pages on XMF server or to comcol WIKI pages to get more information about each command:

- `tr.show`
- `tr.setmask`
- `tr.setsize`
- `tr.setdest`
- `tr.trunc`
- `tr.put`
- `tr.cat`
- `tr.tail`
- `tr.name`
- `pm.chgenv variable_name new_value`

7.5 Platform utilities

It helps in knowing the present state of our hardware and to check if our hardware is in correct state or not.

Following are the commands which can be used for the knowing the information about the Platform (Please refer to man pages on XMF server):

- `getPlatRev`
- `pstack pid`
- `ltrace`
- `strace`
- `iostat`
- `gprof`

7.6 Solving lockup or high CPU usage issues

In case a process seems to be locked up, it is very useful to capture its COMCOL traces, repeatedly and run the *pstack* command to capture process stack.

If a process seems to be taking too much CPU or simply slow, linux commands *ltrace* and *strace* provide information about library (or system) calls made by a process. When run with appropriate parameters (e.g., strace –r), they show relative time spend by the process in system (library) call.

7.7 Automatic Failover

This feature provides automatic failover of a failed IMF server in a subsystem of IMFs. For automatic failover spare server(s) must be available. Here no manual intervention is required. Configuration of the failed IMF server is automatically transferred to the spare server. It considers a total number of ‘N+m’ IMF servers in a subsystem where ‘N’ servers are monitoring links and are considered active while ‘m’ servers are not assigned any links and are considered spare. The determination of active and spare is automatic. The spare and non-spare servers belong to the same subsystem and complete functionality of one failing server is transferred to one spare server. Transferring the functionality of one failing server partially to more than one spare servers or any other combination is not supported.

Current implementation is that fail over happens when the whole box is gone, including power failure, lost entire network, on fire or anything you can think of total lost.

7.7.1 Command to configure/debug Failover

- `setSSVIP` – force to configure the VIP on Master server. It may be use when a clobber has been done on a entire IMF sub-system
- `foStat` – show the current status of failover (direct idb version)
- `iFoStat` – show the current status of failover (mysql version)
  Usage: iFoStat [{VIP of the site|name of the site}]
- `foAlarm` – dump failover alarms from the primary, -m can be used from A-node to get system view
- `failOver` – move load from one host to a spare host (work on IMF only)
  Usage: /opt/TKLCmf/bin/failOver [-f(orce)] <from> [<to>], w/o <to>, it will find an available spare

Few important tips and information to keep in mind before using above mentioned steps:-

- VIP is used to represent an xMF subsystem; it is a Virtual IP for the current Active Master. The Active Master server can be changed physically but not the VIP. The VIP has to be in the cust network and not be physically used by any host.
- There are two parts in failover, one is the network role (active master, standby master and non-master) and the other is for IMF only, the active and spare server based on the group assignment to indicate if spare servers are to be present. Network role and IMF group assignment are independent configurations no dependency between them (active master can be a spare IMF server). Failover on IMF will do both network role change and host/group assignment if need. Failover on PMF will do network role change only including the VIP for site access (subsystem sync, IXP DTS connection).
- use iFoStat to check the VIP connectivity, 'iFoStat <vip>' from other site or just iFoStat to run from the host in the same site
- use foStat/iFoStat to find out the current primary
- use failOver to change IMF host responsibility
- use prod.dbup/prod.start on one server (shutdown/start xMF service) to see the failover action (failover is implemented using heartbeats mechanism)
- trace daqManager on primary server to see the failover activities.

7.8 End-To-End Flow Verification Procedure

When any issue is observed on the PIC system, it is required to identify the correct entity which is causing the problem or operating in un-desirable mode. Once the sub-system causing the erroneous behavior is identified it is required to further identify which component is causing the problem. For this purpose end-to-end verification is required for the sub-system. Following things can be verified for end to end verification of the xMF

- use linkDisp for detailed statistics for the links assigned to Message Feeder (for more information refer 6.1.5).
- use routeDisp for all destinations for the particular server (for more information refer 6.1.2)
- use pm.getprocs to see no process is getting restarted (for more information refer 8.3)
- use prod.state to see the state of the system (for more information refer 8.3)
- verify if the DataFlow and monitoring groups are configured correctly.

- in case of any erroneous behavior, look at the logs of the different processes which may be leading to un-desirable behavior. This kind of analysis may require not only complex verification of the logs/traces but also checking of various configuration, counters and data which are stored in the database and are used during the operation.
- verify the IDB data base, if the data is being stored correctly or not (for more information refer 8.1).

7.9 Alarm Management Tips

Following are the Alarm management tips:-

- Restarting ProAlarm's JMX Agent after IMF upgrade/re-install:-
  After the IMF software is upgraded or re-installed on the IMF machines, if there are no alarms reaching the ProAlarm server, the JMX Agent service on the ProAlarm server needs to be restarted using the Windows Services GUI. (Wait for about 30 seconds after the JMX Agent service is stopped. Then restart it.) Sometimes, it is also necessary to restart the JMX Agents on the IMF servers.

- Restarting the JMX Agent on IMF after it is deleted and re-added into ProAlarm:-
  After the JMX Agent of an IMF server is deleted and re-added from the ProAlarm Configuration tool, the JMX Agent of the IMF needs to be restarted by using "`pm.set off jmxAgent`" and "`pm.set on jmxAgent`" commands before ProAlarm can accept the alarms generated from that IMF.

- Turning off WatchDog warnings:-
  In order to turn off WatchDog warnings in the ProAlarm Viewer, the ProAlarm users can change the severity of the "WatchDog set no defect" event from Warning to Clear using the ProAlarm Configuration tool.

- Re-discovering Signaling Links after Link Name change
  After the name of a particular link is changed, it is necessary to perform a re-discovery of the Signaling Links from the ProAlarm Configuration. Then, exit the ProAlarm Configuration and open it again to see the changed name. (Refreshing the link list alone will not allow the user to see the new link name).

- Finding the real version number of JMX Agent
  The real version number of the JMX Agent is in the “manifest.mf” file of com.steleus.jmx.jar. There is an "Implementation-Version" attribute recording the version number. That is also the number which will be returned when the JMX Agent is pinged by the ProAlarm Configuration/Viewer.

- ProAlarm JMX IP Ports
  Following TCP ports are used by JMX Agent 2.0.5.3 and ProAlarm 1.8.3.1:
  1099 (RMI Registry Port)
  3333 (RMI Adaptor Port)
  3334 (Proxy Factory Port)
  3335 (NotificationHandler Port)
  6969 (HTTP Adaptor Port)
  The alarms are sent from the JMX Agent to port 3335 of ProAlarm. RMI requests generated from ProAlarm are coming to port 3333 of JMX Agent

- Things to check when no alarms are displayed in ProAlarm Viewer
  Check if ProAlarm IP address is in the /etc/hosts file on IMF-1a machine.
  Check if you can ping the ProAlarm server from all the IMF machines.
  Check if you can discover all the JMX Agents from ProAlarm Configuration tool.
  Check if alarms are logged in IMF's AppEventLog table.

Check if the jmxAgent is running using the command "`pm.getprocs`".
(Sometimes, when the ProAlarm database grows too big, it is faster to see the new alarms by exiting the ProAlarm Viewer and restarting it.)

- Enabling ProAlarm to allow Second Level Discovery for IMF
  ProAlarm has two levels of discovery. The first level discovery is mainly to discover the MBeans and Alarm Definitions. The second level discovery is for Signaling Points, Signaling Link Sets and Signaling Links. Current ProAlarm does not support IMF in the discovery of Signaling Points, Signaling Link Sets and Signaling Links, which is the required information for Topology Map.

  Workaround for this problem is, in the Configuration.xml file; add the string "(msw)" at the end of the name field as follows:

  ```
  name=":type=IMFMBean,name=__HOSTNAME___(msw),manageable=yes,nsm=yes"
  ```

  .
- JMX Agent in Eclipse
  For running the JMX Agent after setting up Eclipse, following steps are to be followed in order to configure the run time environment properly for JMX Agent. (The following steps are only necessary in the XP development area. They are not required in the production environment.)
  - Put com.steleus.jmx.jar at the top of the class path list: In the property window of the Eclipse project you have, move the com.steleus.jmx.jar to the top of the list. If you do not do it, you might get an error complaining com/tivoli/jmx/JMX Security class not found.
  - Copy the Configuration.dev.xml file to Configuration.xml. In your jmx_agent/in directory, copy the Cofiguration.dev.xml to Configuration.xml. It has the hardcoded IMFMBean name and it has the proper codebase for development environment.
  - Set up ProAlarm in /etc/hosts. If you want to send alarms to ProAlarm Server, you need to add the ip address of the ProAlarm into /etc/hosts file.

- ATM Event Mapping Information
  The following ATM link events are mostly mapped to the LSL events and processed by the IMF servers to generate related SS7 link alarms: (Note that IMF is supporting the following alarms related to ATM links, but not the Q752 counts for the ATM links.)
  - Out of Service (mapped to SIOS, causing Link Failure alarm)
  - Processor Outage (mapped to SIPO, causing Processor Outage alarm)
  - In Service (no alarm)
  - Normal (mapped to SIN, causing Link Failure alarm)
  - Emergency (mapped to SIE, causing Link Failure alarm)
  - Alignment not successful (mapped to SIO, causing Link Failure alarm)
  - Mgmt Initiated (no available mapping, traceable)
  - Protocol Error (no available mapping, traceable)
  - Proving not successful (no available mapping, traceable)

8.0 Eagle monitoring interfaces (IMF)

8.1 Eagle connectivity

The IMF group provides DHCP service to Eagle STP's Sentinel Transport Cards (STC) to provision IP address information on the "yellow" and "blue" networks. Eagle LIM cards have static IP addresses within the private virtual network of the Eagle STP. These addresses are derived from the frame/shelf/slot position of the LIM card within the Eagle STP. IMFs and LIM cards communicate using the STC card as a router. Tekelec Routing Protocol (TRP) is used between Eagle STP and IMF to dynamically discover the active route in a redundant network between Eagle STP and IMF. IMF also provides NTP services to Eagle STP for the purpose of time-stamping MSUs at Eagle LIMs before they are sent to IMF.

While all IMFs participate in TRP conversations, the IMF designated as "1A" is responsible for providing DHCP and NTP on the yellow network while "1B" provides these services for the blue network.

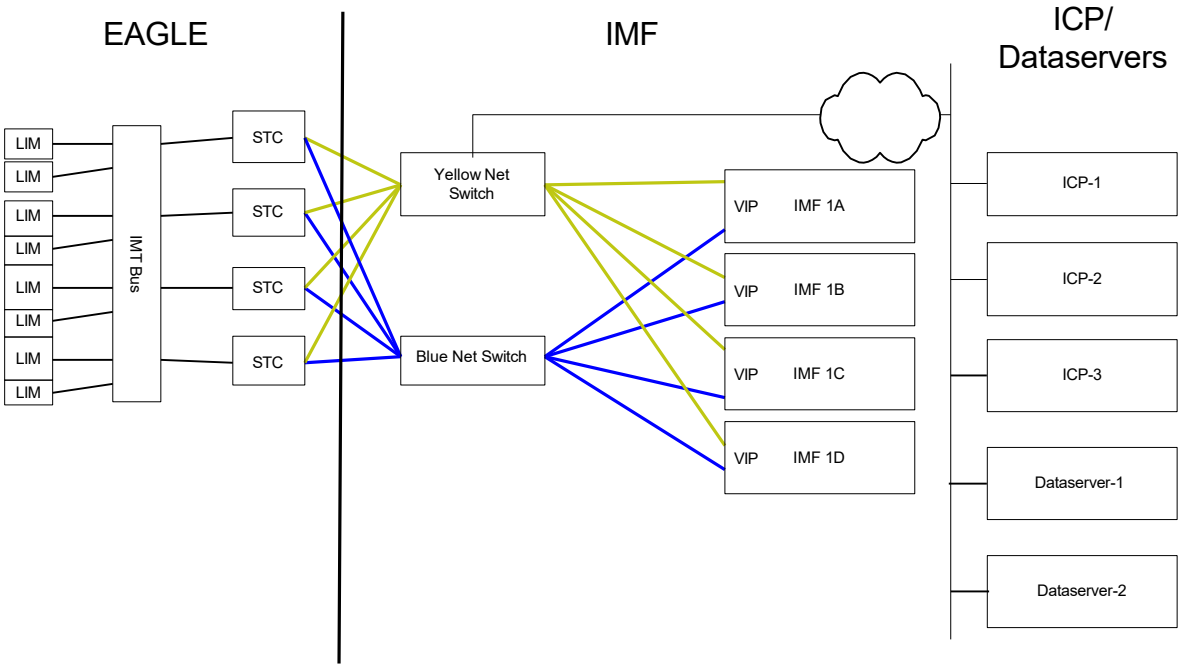


Figure 6: IMF-Eagle-ICP/IXP Interface

Warning: in case of switch reconfiguration, the communication to IMF server may be lost (for more details, see information provided in Installation document).

8.2 DHCP

DHCP for STC Cards is provided for the Yellow and Blue IP Networks from the IMF-1a and IMF-1b servers. Each server manages a different range of these subnets to avoid any issues. The STC Card uses the IP address from the first dhcpd service to respond. IPSG Cards have Static IP Addresses and do not use DHCP.

8.3 NTP Broadcast

NTP is provided to the STC Cards on the Yellow and Blue IP Networks from the IMF-1a and IMF-1b servers. The IMF-1a server provides NTP on the Yellow IP network.

```
IMF-1A:/export/home/cfguser  netstat -r

Kernel IP routing table

Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface

…

224.0.0.0       *               240.0.0.0       U         0 0          0
bond0.100

…
```

The IMF-1b server provides NTP on the Blue IP network.

```
IMF-1B:/export/home/cfguser netstat -r

Kernel IP routing table

Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface

…

224.0.0.0       *               240.0.0.0       U         0 0          0
bond0.101

…
```

8.4 NTP Usage

The STC Card expects a NTP broadcast every 64 seconds on the subnet that it is listening to. If an NTP broadcast is not received on time then the STC Cards changes which subnet it is listening to for its NTP source. NTP is only used by the Eagle STP for time-stamping MSU's for monitoring. The Eagle STP uses the NTP time sync and its clock source (TSC sync) for time-stamping. If there is a clock source issue then there will be a timestamp issue.

There is a 9 hour NTP history on each STC Card that can be gathered by Eagle TAC for analysis. This is only needed in rare troubleshooting cases.

8.5 STC Copy

STC Copy can be used with any STC Card and any TekServer (T1000 / T1100 / T1200) and HP IMF.

STC Copy for Sigtran Links should use E5-STC Cards on the same shelf as the link(s) being monitored and with any TekServer (T1000 / T1100 / T1200) and HP IMF.

MSU's are copied using a socket connection from the Eagle's LIM Card to the IMF. This connection is routed through an STC Card and an IMF Switch.

Copy coordination messages like Service Request Messages are received through an STC Card IP route.

8.6 Fast Copy

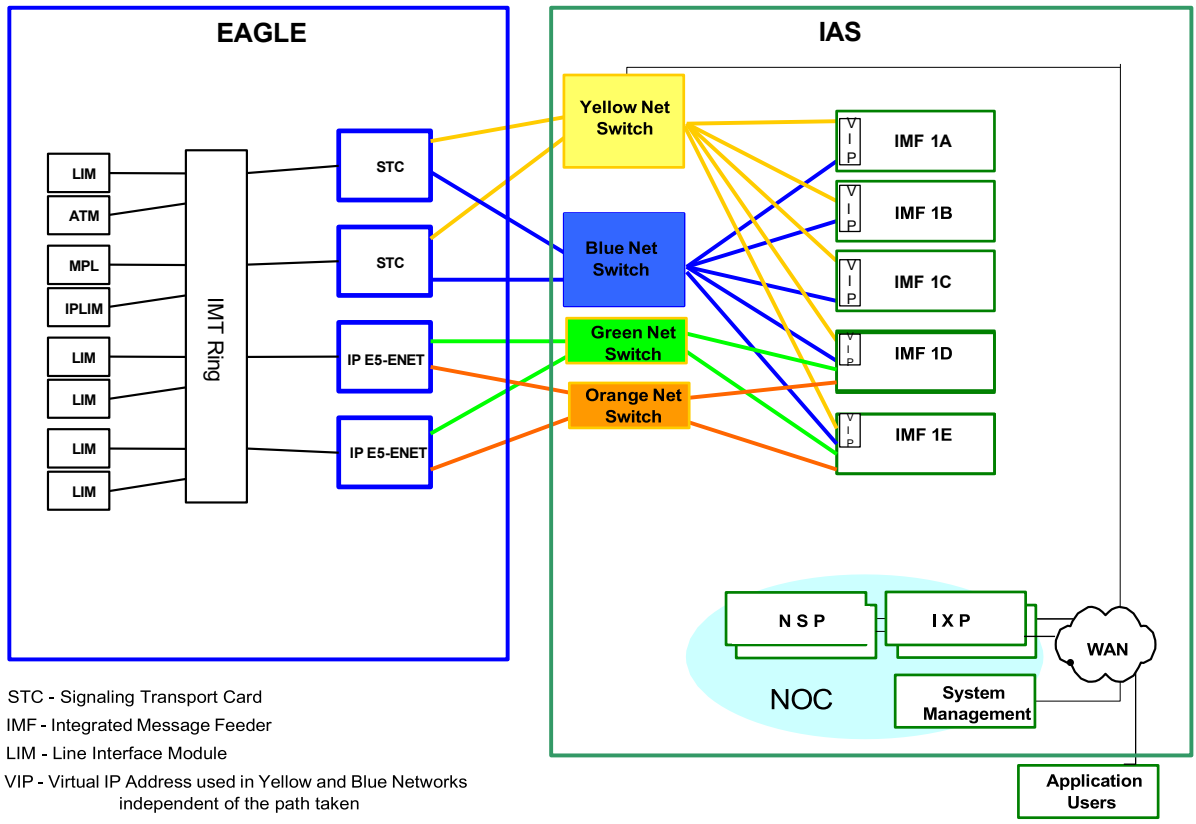


Figure 7 : EAGLE - PIC interface with Fast Copy feature on

Earlier design of the Integrated Monitoring for SIGTRAN interfaces required significant IPLIM and IPGW CPU processing and therefore decreased the available transaction units (TU) that could be used for other functions such as processing signaling messages, etc. In addition, the earlier design had an additional load on the IMT bus and required additional STC cards to be equipped to handle the copied traffic from the monitored SIGTRAN interfaces.

For this reason architectural changes were performed to provide a direct Fast Copy interface. Fast Copy is a direct IP connection between the IPSG (LIM) Card and the IMF with only the IMF Switches in between. Sigtran Packets sent or received on the IP SS7 Network by an IPSG Card are forwarded (routed) to the IMF by changing the MAC address to the assigned IMF's MAC address and sent out on the Green or Orange IP Network thus completely bypassing the existing Lim->IMT->STC->IMF path. Copy coordination messages like Service Request Messages are received through an STC Card IP route only. Fast Copy can only be used with E5-ENET Cards (ex. IPSG Card) and TekServer 2 & 3 (T1100 / T1200) and HP IMF's.

8.7 IP Troubleshooting

Use ping to test IP activity to and from STC, LIM, or IPSG cards and IMF servers. Use traceroute to test IP routes from the IMF servers to STC, LIM, and IPSG cards. Verify that the STC or IPSG MAC address for the correct port shows on the switches. The "show mac addr" command shows MAC addressed only when there is IP network activity

8.8 Eagle IMF interface Troubleshooting

This section shall elaborate on the various scripts that can be used for the diagonsis of the issue(s) between Eagle & IMF interface. The scripts can be helpful in the following scenarios:

1) TRP messages not getting exchanged between Eagle & IMF servers
2) To identify which STC routes are currently active
3) To identify what is the status of the Eagle link card and what is the current route used by the card.
4) To identify if the Eagle Link card is down.
5) To identify which cards are not being monitored.
6) To capture the packets flowing on yellow & blue Vlans for cards, links & associations.
7) To identify which cards supports Fast Copy & which supports STC copy.
8) To identify the hardware address of the route
9) To identify the links of the given cards and to see the bandwidth counters of the given links
10) To see the bandwidth of the given route.

Following utilities can be used to get information about EAGLE cards and also the bandwidth counters.

showCardInfo

This script can be used to display the card related information with the following options:

- default option shall display all information related to card
- -f for fastcopy Ip address
- -r for route Ip address for the card

Usage: showCardInfo [-r] [-f]

The options can not be combined together

showCardInfo

Following columns are displayed when it is executed with no option

cardId	Name of card
cardIp	Ip Address of the card
cardLoc	Location of card
cardApp	Card type
monType	Type of monitoring. It is deduced by card application
status	Status based on ping message. The ping is done on the card Ip to get the current status
modified_at	Last update time for the card.

Sample Output:

```
[cfguser@imf0501-1a ~]$ showCardInfo
             cardId            cardIp         cardLoc         cardApp         monType          status     modified_at
   stpb9070801-1207      172.20.48.7            1207       UNDEFINED             STC              OK   12/31/1969 19:00:00
   stpb9070801-1106     172.20.48.246            1106       UNDEFINED             STC              OK   12/31/1969 19:00:00
   stpb9070801-1107     172.20.48.247            1107       UNDEFINED             STC              OK   12/31/1969 19:00:00
   stpb9070801-1105     172.20.48.245            1105       UNDEFINED             STC              OK   12/31/1969 19:00:00
   stpb9070801-1216      172.20.48.14            1216       UNDEFINED             STC              OK   12/31/1969 19:00:00
   stpb9070801-1208       172.20.48.8            1208       UNDEFINED             STC              OK   12/31/1969 19:00:00
   stpb9070801-1205       172.20.48.5            1205         CCS7ITU             STC              OK   10/17/2013 05:56:47
   stpb9070801-1203       172.20.48.3            1203         CCS7ITU             STC              OK   10/17/2013 05:56:47
   stpb9070801-1317      172.20.48.31            1317         CCS7ITU             STC              OK   10/17/2013 05:56:47
   stpb9070801-1202       172.20.48.2            1202         SS7ANSI             STC              OK   10/17/2013 05:56:47
   stpb9070801-1304      172.20.48.20            1304         SS7ANSI             STC              OK   10/17/2013 05:56:47
   stpb9070801-1111     172.20.48.249            1111            IPSG        FASTCOPY              OK   10/17/2013 05:56:47
   stpb9070801-1204       172.20.48.4            1204         CCS7ITU             STC              OK   10/17/2013 05:56:47
   stpb9070801-1305      172.20.48.21            1305         SS7ANSI             STC              OK   10/17/2013 05:56:47
   stpb9070801-1103     172.20.48.243            1103            IPSG        FASTCOPY              OK   10/17/2013 05:56:47
   stpb9070801-1201       172.20.48.1            1201         SS7ANSI             STC              OK   10/17/2013 05:56:48
   stpb9070801-1302      172.20.48.18            1302         SS7ANSI             STC              OK   10/17/2013 05:56:48
   stpb9070801-1303      172.20.48.19            1303         SS7ANSI             STC              OK   10/17/2013 05:56:48
   stpb9070801-1104     172.20.48.244            1104            IPSG        FASTCOPY              OK   10/17/2013 05:56:48
   stpb9070801-1315      172.20.48.29            1315            IPSG        FASTCOPY              OK   10/17/2013 05:56:49
   stpb9070801-1101     172.20.48.241            1101            IPSG        FASTCOPY             NOK   10/17/2013 05:56:50
   stpb9070801-1307      172.20.48.23            1307         SS7ANSI             STC              OK   10/17/2013 05:56:50
```

showCardInfo –r

This option displays route's ip address for the card along with the other information.

Following columns are displayed when it is executed with -r option

cardId	Name of card
cardIp	Ip Address of the card
route	Route being used by the card
status	Status based on ping message

Sample Output:

```
[cfguser@imf0501-1a ~]$ showCardInfo -r
             cardId            cardIp           route          status
   stpb9070801-1207       172.20.48.7   172.22.49.253              OK
   stpb9070801-1106     172.20.48.246   172.21.49.125              OK
   stpb9070801-1107     172.20.48.247   172.22.49.120              OK
   stpb9070801-1105     172.20.48.245   172.22.49.249              OK
   stpb9070801-1216      172.20.48.14   172.21.49.126              OK
   stpb9070801-1208       172.20.48.8   172.22.49.124              OK
   stpb9070801-1205       172.20.48.5   172.22.49.249             NOK
   stpb9070801-1203       172.20.48.3   172.21.49.124             NOK
   stpb9070801-1317      172.20.48.31               -              OK
   stpb9070801-1202       172.20.48.2   172.22.49.247              OK
   stpb9070801-1304      172.20.48.20               -              OK
   stpb9070801-1111     172.20.48.249   172.21.49.124              OK
   stpb9070801-1204       172.20.48.4   172.21.49.126              OK
   stpb9070801-1305      172.20.48.21               -              OK
   stpb9070801-1103     172.20.48.243   172.21.49.248              OK
   stpb9070801-1201       172.20.48.1   172.22.49.253              OK
   stpb9070801-1302      172.20.48.18   172.22.49.124              OK
   stpb9070801-1303      172.20.48.19               -              OK
   stpb9070801-1104     172.20.48.244   172.21.49.121              OK
   stpb9070801-1315      172.20.48.29               -              OK
   stpb9070801-1101     172.20.48.241   172.22.49.247             NOK
   stpb9070801-1307      172.20.48.23               -              OK
   stpb9070801-1312      172.20.48.26               -             NOK
   stpb9070801-1301      172.20.48.17   172.21.49.122              OK
   stpb9070801-1313      172.20.48.27               -              OK
   stpb9070801-1112     172.20.48.250               -              OK
   stpb9070801-1306      172.20.48.22               -              OK
   stpb9070801-1318      172.20.48.32               -              OK
   stpb9070801-1308      172.20.48.24               -              OK
```

When IPADDR showing blank in the output of below command on Eagle-STP and “showCardInfo -r” on IMF showing blank output.
rept-stat-card:mode=full:loc=<stc/eroute card>

Then we can follow below steps on Eagle-STP.
Init-card:loc=<stc/eroute card>
chg-eisopts:fcmode=off:eiscopy=off
Chg-eisopts:fcmode=fcopy:eiscopy=on
Init-card:loc=<stc/eroute card>

showCardInfo –f

This option displays fastcopy ip address of the card along with the other information.

Following columns are displayed when it is executed with -f option

cardId	Name of card
cardIp	IP Address of the card
greenIp	IP address for green interface
orangeIp	IP address for Orange interface
status	Status based on ping message. Status will be OK for a fastcopy card when all the interfaces (CardIP, greenIp & orangeIP) are OK. For STC cards the status will be OK if the CardIP is OK.

Sample Output:

```
[cfguser@imf0501-1a ~]$ showCardInfo -f
            cardId                  cardIp               greenIp              orangeIp              status
   stpb9070801-1207           172.20.48.7                    -                     -                  OK
   stpb9070801-1106          172.20.48.246                   -                     -                  OK
   stpb9070801-1107          172.20.48.247                   -                     -                  OK
   stpb9070801-1105          172.20.48.245                   -                     -                  OK
   stpb9070801-1216           172.20.48.14                   -                     -                  OK
   stpb9070801-1208            172.20.48.8                   -                     -                  OK
   stpb9070801-1205            172.20.48.5                   -                     -                  OK
   stpb9070801-1203            172.20.48.3                   -                     -                  OK
   stpb9070801-1317           172.20.48.31                   -                     -                  OK
   stpb9070801-1202            172.20.48.2                   -                     -                  OK
   stpb9070801-1304           172.20.48.20                   -                     -                  OK
   stpb9070801-1111          172.20.48.249          172.21.48.249         172.22.48.249              OK
   stpb9070801-1204            172.20.48.4                   -                     -                  OK
   stpb9070801-1305           172.20.48.21                   -                     -                  OK
   stpb9070801-1103          172.20.48.243          172.21.48.243         172.22.48.243              OK
   stpb9070801-1201            172.20.48.1                   -                     -                  OK
   stpb9070801-1302           172.20.48.18                   -                     -                  OK
   stpb9070801-1303           172.20.48.19                   -                     -                  OK
   stpb9070801-1104          172.20.48.244          172.21.48.244         172.22.48.244              OK
   stpb9070801-1315           172.20.48.29          172.21.48.249         172.22.48.249              OK
```

showRouteInfo

This script can be used to get the following information:

- List of available routes – Routes with status OK and NOK collectively represent the list of available routes.
- List of IP address for the route
- MAC address of route
- List of active routes for an IMF server - Routes with "OK" status are active.
- List of used routes for an IMF server - Routes with "NOK" status are inactive.

Following columns are displayed when it is executed

networkId	Network id
stcRoute	Ip Address of the route
status	Status of route
hwAddress	MAC address of the route (if available in "arp –a" command)
modified_at	The last time when the status of the route was changed.

Sample Output:

```
[cfguser@imf0501-1a ~]$ showRouteInfo
        networkId                stcRoute                status              hwAddress            modified_at
      172.20.48.0          172.21.49.121                    OK       00:00:17:0D:B3:9E       10/24/2013 13:06:15
      172.20.48.0          172.21.49.122                    OK       00:00:17:0C:E8:6E       10/24/2013 13:06:02
      172.20.48.0          172.21.49.124                    OK       00:00:17:0C:E8:B8       10/24/2013 10:31:13
      172.20.48.0          172.21.49.125                    OK       00:00:17:0C:27:FE       10/24/2013 13:06:20
      172.20.48.0          172.21.49.126                    OK       00:00:17:0C:E8:DA       10/24/2013 10:31:13
      172.20.48.0          172.21.49.248                    OK       00:00:17:0C:99:77       10/24/2013 13:06:16
      172.20.48.0          172.22.49.120                    OK       00:00:17:0D:C3:8C       10/24/2013 13:06:02
      172.20.48.0          172.22.49.124                    OK       00:00:17:0C:E8:B9       10/24/2013 13:06:15
      172.20.48.0          172.22.49.247                    OK       00:00:17:0C:99:78       10/24/2013 13:06:19
      172.20.48.0          172.22.49.249                    OK       00:00:17:0C:E8:6F       10/24/2013 13:06:15
      172.20.48.0          172.22.49.250                   NOK                               10/24/2013 00:44:55
      172.20.48.0          172.22.49.251                    OK       00:00:17:0C:27:FF       10/24/2013 10:31:07
      172.20.48.0          172.22.49.253                    OK       00:00:17:0C:E8:DB       10/24/2013 13:06:15
```

NOK Status: The NOK status for the route denotes that route is currently not accessible. The route with NOK status may be temprorary unavailable or the route IP address has been changed on the STC card. This means that NOK status shall be set for the STC route, when the STC card does not exchange TRP messages with the IMF TRP server, then restart TRP server using below command:-
pm.set off trpd
pm.set on trpd

TRP messages are exchanged but the TRP message contain different IP address than previously used for the STC card i.e the IP address of the STC card has been changed.

In other words, The "Tekelec Routing Protocol" indicates only all available routes to communicate with eagle cards. If no more message is received from an IP address, then it means this IP address is no more available but the reason is not known (normal case because the IP address of stc card has been changed or issue with one of the available stc card). To know the status of STC cards, it is better to use the showCardInfo. If all routes are defined in the system then the number of route in OK state should be equal to the double of the number of STC cards.

showRouteBandwidth.sh

This script displays bandwidth of the links for the given cards. It can be used to get number of packets & number of bytes for each eagle STC route.

Usage: showRouteBandwidth.sh <route>

Following columns are displayed when the script is executed

route	Ip address of the route
msuRx	MSU received per second
msuTx	MSU transmitted per second
bytesRx	Number of bytes received per second
bytesTx	Number of bytes transmitted per second
bandwidthRx	Mbps received
bandwidthTx	Mbps transmitted

Sample Output:

```
Every 1.0s: perl /opt/TKLCmf/bin/showRouteBandwidth 172.22.49.124

        route           msuRx           msuTx         bytesRx         bytesTx     bandwidthRx     bandwidthTx
172.22.49.124            1226             979          138299          152802            1.06            1.17
```

Note: The showRouteBandwidth.sh <route> tool will not be available in release 10.0, as it depends on the iproute module and there is a known issue "https://bugzilla.redhat.com/show_bug.cgi?id=1045200" in iproute RPM. The child PR#238765 has been opened on open_platform and fix of the issue is not expected to be available in release 10.0 time frame.

CaptureCardLinkPkt.sh

This script can be used to capture the PDUs for cards, links and associations. The script shall be provided the link name or association name or card name, the script shall internally translate the link name/assoc name or card name to the IP@ filter expression. The filter expression should be then passed to the tcpdump utility to perform the capturing of the PDUs.

The capturing script provides the capability to generate the signal to interrupt the tcpdump command to stop the capture. This can be done by monitoring the size of the total capture or the time duration of the capture.

The capturing script provides the following options:

- –l <linkname(s) separated by space as delimeter >: capture the link PDUs for all the links mentioned. This is a mandaotry parameter
- –c <cardname>: capture the link PDUs for all the links on this card. This is a mandatory parameter
- –t: Time duration of the capture session in sec. This is an optional parameter, if not provided the script provides the default value for this parameter.

The options "-c", "-a" & "-l" cannot be specified together. The script shall be run only with root privileges. The sudo option shall be provided with the script to gain the root privileges.

The script merges the captured files on VLAN 100 (yellow) & VLAN 101 (blue) into one single file.

Usage:

To run the script, login as **cfguser** and use *sudo* to gain root privileges

```
sudo CaptureCardLinkPkt.sh  -t<capture duration>  -l <linkname1, linkname2…n>
```

The default parameters can be omitted:

```
sudo CaptureCardLinkPkt.sh -t 15 -l <linkname1, linkname2…n>
```

Example:

- `sudo CaptureCardLinkPkt.sh –t 60 –l linkname1, linkname2, linkname3 …`
- `sudo CaptureCardLinkPkt.sh  –t 60 –c card-name1, card-name2  …`
- `sudo CaptureCardLinkPkt.sh  –t 60 –a assoc1, assoc2…`
- `sudo CaptureCardLinkPkt.sh`
  The last example shall capture all the packets on Vlan 100 & Vlan 101 without any filter for the default duration of 15 sec.

Note: The script shall kill the tcpdump running on the servers after the capture, so this may impact the other tcpdump instances running on the server at the time of capturing.

The captured files can be opened using wireshark tool (PCAP format). However it is necessary to install an additional plugin to decode EMP header (TR005008 messages). The wireshark and EMP plugin are available on XMF server. Run "`wireshark /tekelec/TcpdumpCapture/Global.pcap`" on SSH console in root (X11 forwarding must be enabled on your SSH console and X11 server must be available on your computer).

Note: If you prefer to use local wireshark program (on Windows), then contact Design Support to get the supported wireshark version and Windows EMP plugin.

8.9 E5-APP-B Disk Replacement

Refer to document **910-6533-001** (EAGLE Application Processor, Alarms and Maintenance on the EAGLE Application B Card), chapter "Removing and Replacing a Drive Module Assembly" to perform SSD Disk replacement

9.0 SS7 Probed Message feeder (PMF)

9.1 Solving "Cannot assign more than 50 SS7 links to Mistral Cards in a PMF (Bug 21856562)" limitation

By default, the number of SS7 links that can be assigned to Mistral cards on a SS7 PMF cannot exceed 50. Apply the workaround below to increase the number of SS7 links to 512 if more than 50 SS7 links need to be assigned:

- Edit table "PartDef" `(ivi PartDef).`

- Modify line `" 34|MsuPart|Log|ABC|50|-1|212|12 ... "` to
`" 34|MsuPart|Log|ABC|512|-1|212|12 ... ".`

- Save the table `(:x!) and enter "yes"` when prompted to reload the table.

- Stop probeMonitor `(pm.set off probeMonitor).`

- Start probeMonitor `(pm.set on probeMonitor).`

10.0 Integrated OCDSR monitoring procedures

10.1 Bandwidth limitation customization

By default, bandwidth limitation of each traffic flow is configured to 125 Mbps. If it is necessary to modify this default limitation, use following command (in cfguser):

```
iset -f maxBw=100 -f lastTime=now DsrMonitorTrafficFlow where
"name='DIA_S6a_SESS_ID'"
```

By changing:

- bandwidth limitation value (set to 100 Mbps in example)
- traffic flow name (set to DIA_S6a_SESS_ID in example)

Remark: bandwidth limitation is configured for all load shared traffic flows. For example if you have traffic flows DIA_S6a_SESS_ID and DIA_S6a_SESS_ID_LS1, you must use the name DIA_S6a_SESS_ID in previous command to modify the bandwidth limitation and this limitation is applied to all DIA_S6a_SESS_ID* traffic flows.

10.2 Load sharing customization

By default, load sharing criteria is automatically updated by taking into account traffic bandwidth and bandwidth limitation. It may be necessary to modify the load sharing criteria in following cases:

- prepare IXP Data Flow Processing configuration for highest traffic bandwidth context
- reduce number of traffic flows used to process input traffic

Use following command (in cfguser):

```
iset -f loadBalancing=8 -f lastTime=now DsrMonitorTrafficFlow where
"name='DIA_S6a_SESS_ID'"
```

By changing:

- load balancing value (set to 8 traffic flow in example)
- traffic flow name (set to DIA_S6a_SESS_ID in example)

10.3 Application Identifier customization

If an application identifier is received on traffic flow DIA_SESS_ID and its value is enabled in the table DsrMonitorDiaAppId, then a specific traffic flow is created to receive this traffic. Its name is DIA_{name in DsrMonitorDiaAppId table}_SESS_ID.

It is possible to enable this mechanism for each application identifier by adding an entry from DsrMonitorDiaAppId table, using following command (in cfguser):

```
iset -fmonitor=Yes DsrMonitorDiaAppId where "name='S6a'"
```

By changing:

- Application Identifier name (set to “S6a” in example)

It is possible to disable this mechanism for each application identifier by removing the entry from DsrMonitorDiaAppId table, using following command (in cfguser):

```
iset -fmonitor=No DsrMonitorDiaAppId where "name='S6a'"
```

By changing:

- Application Identifier name (set to “S6a” in example)

10.4 Change period of OCDSR configuration synchronization

It is possible to change the synchronization period with OCDSR, using following command (in cfguser):

```
iset -f checkPeriod=60 DsrMonitorCfg where "id=0"
```

By changing:

- Period value in second (set to “60” in example)

In order to restore default value, use following command (in cfguser):

```
iset -f checkPeriod=0 DsrMonitorCfg where "id=0"
```

10.5 Change OCDSR soap log

It is possible to enable soap log in OCDSR, by using following command (in cfguser):

```
iset -f value=1 LongParam where "name='DsrSoapLog'"
```

The logs will be created in following files:

- /var/TKLC/log/xMF/SENTlog

- /var/TKLC/log/xMF/RECV.log

To disable soap log in OCDSR, run the above command with `value=0`.

Note: by default, logs are disabled, as it involves certain overhead, in terms of space and performance.

11.0 Web service interface procedures

11.1 Change SSL encryption at web service interface

It is possible to disable SSL encryption at web service interface, by using following command (in cfguser):

```
iset -f value=0 LongParam where "name='Encrypt'"
```

And restart the processes

- Restart the webServerApp
  - `pm.set off webServerApp`
  - `pm.set on webServerApp`

- Restart the dsrMonitor

  - `pm.set off dsrMonitor`
  - `pm.set on dsrMonitor`

Note: by default, SSL encryption is enabled at webserver interface. To enable SSL encryption, use the above command with `value=1`, and restart the processes webServerApp and dsrMonitor.

12.0 CIF Interface

The CIF interface is available when an "Acquisition DataFeed Export" stream is defined at the output of the Acquisition server. This configuration establishes a MFP connection with a third party server on a given port, named CIF port (default is 9090).

To enable CIF interface, use following command (in cfguser):

```
iset -f value=9090 LongParam where "name='UnsortedFeedPort'"
```

By changing:

- CIF port value (set to "9090" in example)

The CIF interface change also the "link name" information provided for each packet at Acquisition server output. This interface modification concerns Integrated Acquisition mode. This "link name" field is consisting of 8 characters and should have following structure: eeccccpp

Where:

ee	is the eagle id
cccc	is the eagle card id (e.g., '1201')
pp	is the eagle port id (e.g., '00')

Appendix A. Review Checklist

Use this checklist to verify that all required subject areas were documented.

Item #	Section #	Subject	Required/ Optional	Complete (Yes/No/NA)	Check
1.					
2.					
3.					
4.					
5.					