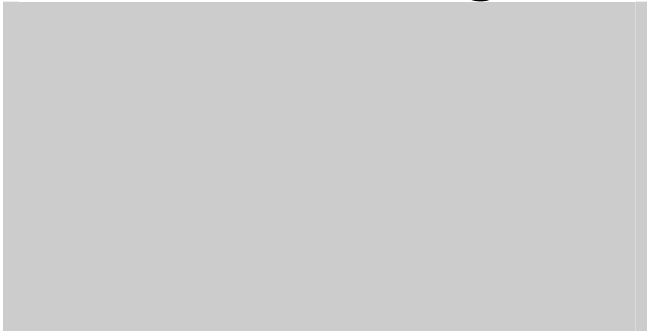
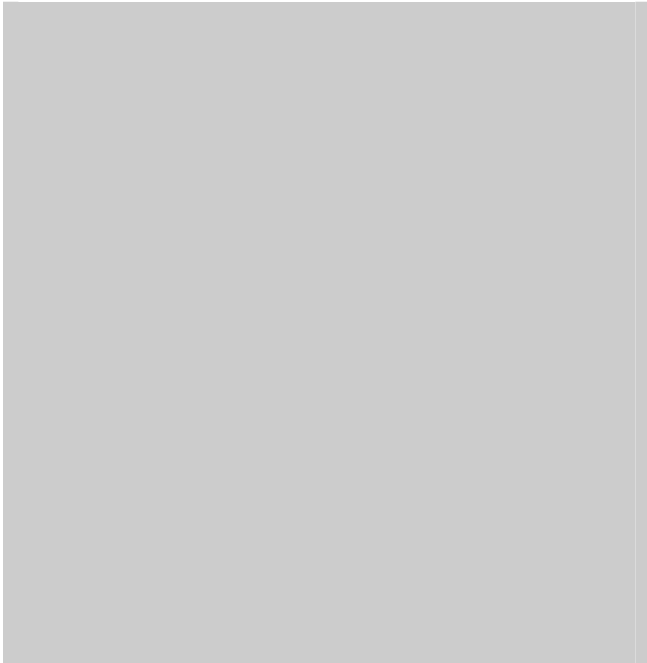




TSP Node Management



STUDENT BOOK
03802 - LZU 102 665 R1B



DISCLAIMER

This book is a training document and contains simplifications. Therefore, it must not be considered as a specification of the system.

The contents of this document are subject to revision without notice due to ongoing progress in methodology, design and manufacturing.

Ericsson assumes no legal responsibility for any error or damage resulting from the usage of this document.

This document is not intended to replace the technical documentation that was shipped with your system. Always refer to that technical documentation during operation and maintenance.

Copyright © 2003 by Ericsson AB

This document was produced by Ericsson AB.

- It is used for training purposes only and may not be copied or reproduced in any manner without the express written consent of Ericsson.

This Student Book, 03802 - LZU 102 665, R1B supports course number LZU 102 665 .

Contents

TSP NODE MANAGEMENT	1
1 TSP PLATFORM.....	5
2 TSP OPERATION AND MAINTENANCE	35
3 FAULT MANAGEMENT.....	63
4 CONFIGURATION MANAGEMENT	73
5 BACKUP AND RESTORATION	83
6 SYSTEM UPGRADE.....	103
7 NETWORK REDUNDANCY	127
8 EXPANSION, REPAIR AND REPLACEMENT OF HARDWARE... 	153
9 DISK LOG	167
10 SS7 SIGNALING MANAGEMENT	175
11 VIRTUAL IP.....	203
12 PERFORMANCE MANAGEMENT	227

Intentionally Blank

1 TSP Platform

CHAPTER OBJECTIVES

After completing this chapter you will be able to:

- Describe the TSP Platform hardware
- Describe the TSP Platform software architecture
- Explain the concepts of processes, database objects and the distributed software execution environment.

INTRODUCTION

The Ericsson Telecom Server Platform (TSP) is a carrier-class server technology designed for server solutions where high availability is required.

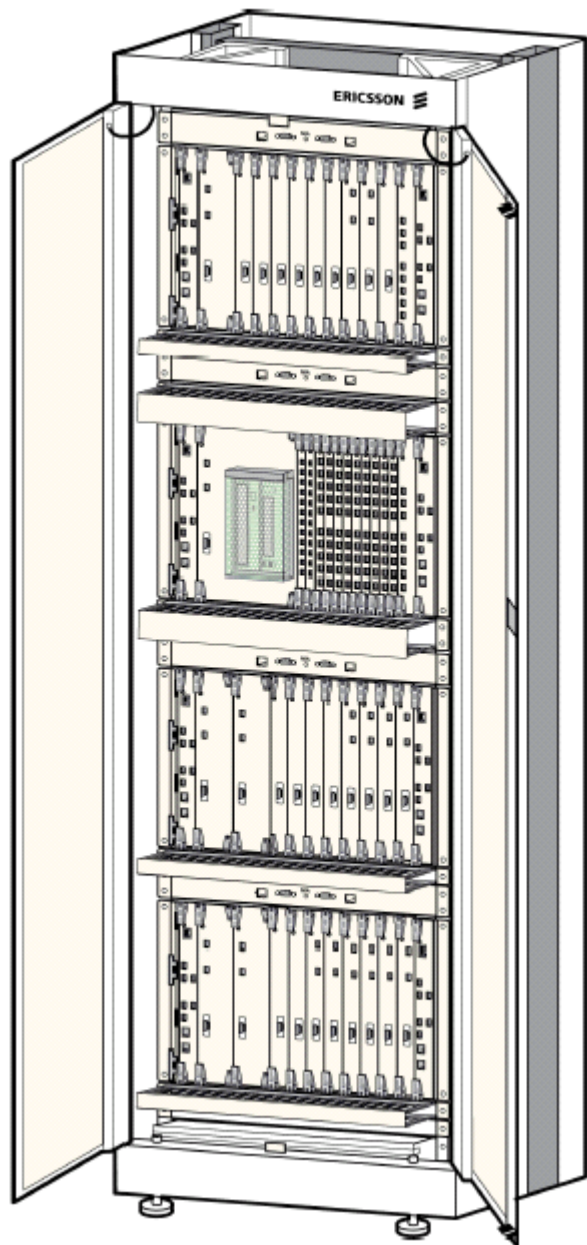


Figure 1-1 The Ericsson TSP Hardware

TSP is a platform for building reliable and scalable, telecom and datacom, network products. In general the platform provides application builders with the following support:

- Distributed Execution
- SS7 signaling
- IP based signaling
- Operation and Maintenance
- Security

Fundamentally, the TSP platform is a number of general purpose processors, loosely interconnected over a network, forming a *cluster* of processors where the distributed execution software is run along with the applications.

The TSP 4.0 system can be divided into four subsystems:

- Hardware
- Operating System (OS) & Clusterware (CW)
- Signaling (SS7 & IP)
- Node Management

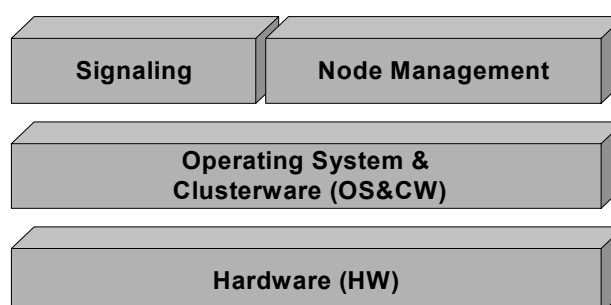


Figure 1-2 TSP Internal Architecture

The purpose of this course is to explain the Node Management techniques performed on the TSP platform. In order to achieve this, this first chapter will guide you through an overview of the lower subsystems of the TSP Platform. This includes the TSP hardware, clusterware and operating systems.

TSP HARDWARE PLATFORM

The Ericsson TSP architecture provides the TSP 4.0 hardware platform, based on commercially available hardware with high capacity and dependable node performance. Use of standard hardware means low costs for installation, operation and maintenance. The inherent modular design of TSP also allows for easy upgrade.

The TSP platform is intended for server and control node applications requiring scalable capacity and high availability. The hardware consists of open industry components and a robust and flexible infrastructure with telecom characteristics.

A TSP platform consists on the hardware level of a cluster of Processor Modules within a GEM magazine in a BYB 501 cabinet. All Processor Modules (PMs) are interconnected by duplicated internal Ethernet connections running through the backplane of the magazine.

The power dissipation from the Processor Modules requires forced cooling by means of fans. Hence each GEM has a fan mounted above it. A serial cooling principle is utilized with an air flow going from the bottom to the top of the cabinet.

The PM's are intended for application processing such as traffic execution. A number of PMs are equipped with interface cards for external communication. The external interfaces are E1/T1/J1 for SS7 signaling and Ethernet for IP-based communication.

Some PMs are hard disk equipped used for operation and maintenance and other support applications. They are also used for the IO system and can contain additional IO units such as DVD and tape drives. The hard disk equipped PMs have Ethernet interfaces for node external O&M communication and RS-232 serial interface for remote access.

A Processor Module may run either a Dicos or Linux operating system. The selection is made during installation in a configuration file. The hard disks are supported only if the processor module runs the Linux operating system.

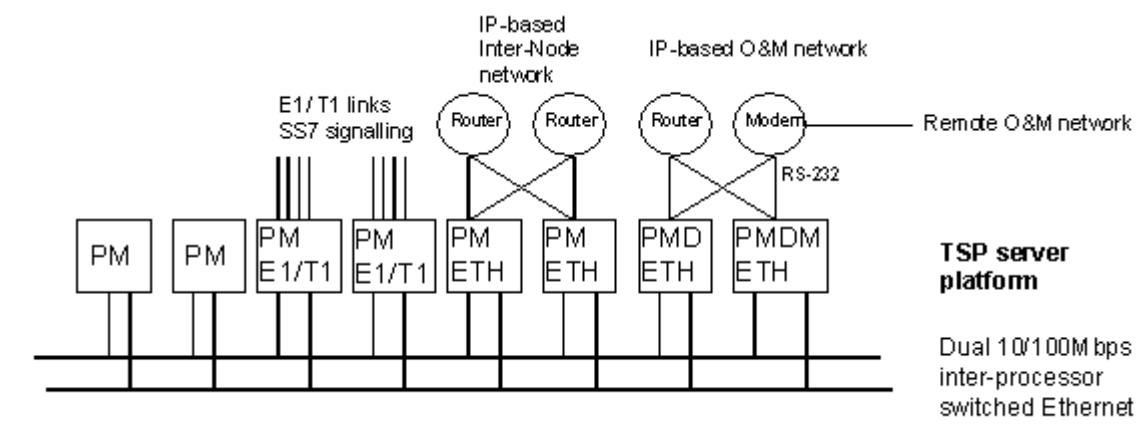


Figure 1-3 TSP Hardware Platform

The hardware is essentially a “networked computer”, linking processing modules into a cluster:

- TelORB or Linux traffic Processor Modules (PM), also called Traffic Processors (TP). These may have special interfaces to take incoming traffic in the form of IP using Ethernet (PM-ETH) or SS7 signaling using E1/T1 ports (PM-E1/T1)
- IO processors - Processor Modules with hard disks (PMD-ETH) and may also have DVD and tape streamer (PMDM-ETH), the latter being used for initial loading purposes and backups.

The PMs are all based on a sourced Pentium-based cPCI processor board. The PMs as previously mentioned, can be equipped with hard disks, DVD reader, tape streamer and external interface modules such as 100BaseTX Ethernet (PM-ETH) or E1/T1 (PM-E1/T1). An Inter Processor Network (IPN) is implemented as a duplicated switched 100 Mbps Ethernet, integrated in the GEM subracks.

ETHERNET SWITCHES

There are two physical levels of Ethernet switches, used to interconnect the PMs. The hardware involved is

- SCB-RP – Support and Connection Board and,
- GESB - Gigabit Ethernet Switch Board.

The Support and Connection Board (SCB-RP) is the interface board for power (-48V) and Ethernet connections to the plug-in units. It is also the maintenance controller for the GEM subrack. The SCB collects alarms from the fans in the cabinet and the supply voltage supervision and transports the alarms via Ethernet to the TSP Node Management using SNMP.

Each GEM subrack includes two SCB-RP boards for redundancy.

The SCB-RP boards boots from one of the Processor Modules with a hard disk, which makes it possible to upgrade the software on-site.

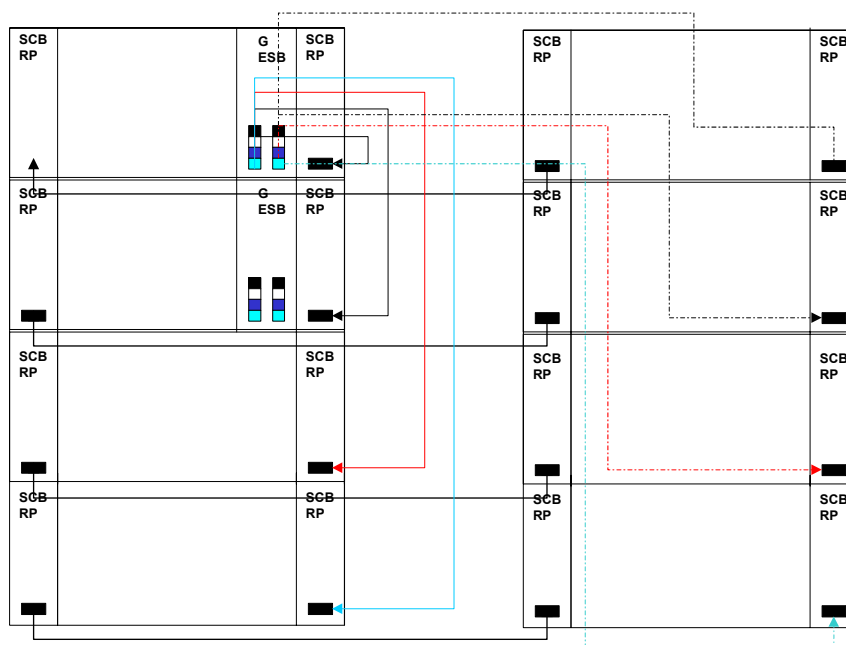


Figure 1-4 GEM subrack interconnection using 2 GESB boards

The interconnection of GEM subracks is performed using the Gigabit Ethernet Switch Board (GESB). A GESB board has eight 1000Base-T Ethernet ports at the front, therefore allowing up to 8 GEM sub-racks to be interconnected, according to the diagram. Two GESBs per 8 subracks is required for redundancy. Note that the two GESBs are housed within the same cabinet but in different subracks.

The GESB boards boots from one of the Processor Modules with a hard disk (also called IO), which makes it possible to upgrade the software at site. The Ethernet switch is managed by means of the SNMP protocol.

Patch Panel

To facilitate requirements on protection, filtering and security (physical disabling of external connections), a patch panel is included in the cabinet.

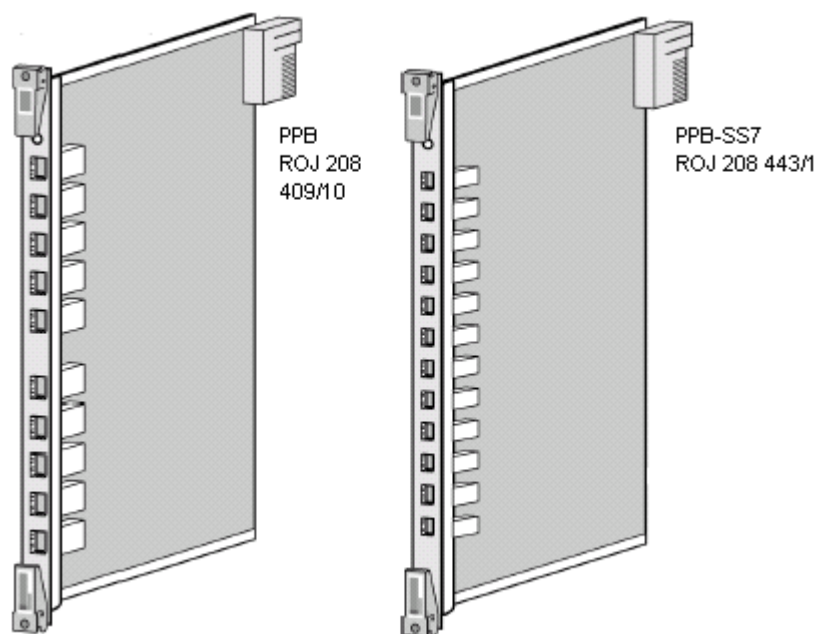


Figure 1-5 Patch Panel Boards

The patch panel consists of a number of Patch Panel Boards (PPBs). There are two types of patch panel boards. One type is used for SS7 connections and the other for connection of Ethernet links, RS-232 and external alarms cables.

External Equipment

Alarm Panel

A stand-alone alarm panel is optional equipment supported in the TSP 4.0 platform. It indicates the number of critical, major and minor active alarms. The panel is used on site to notify the operator via light indicators and acoustic signal that an alarm has been raised. The panel is intended for wall mounting.

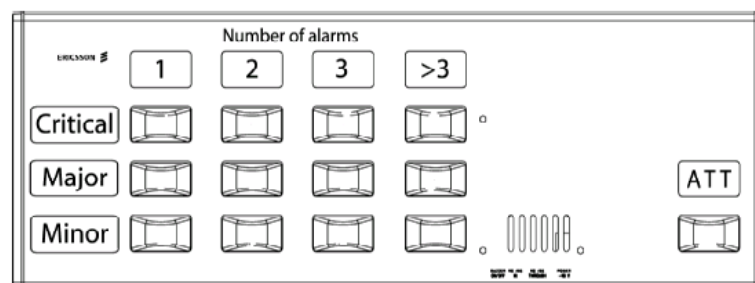


Figure 1-6 Alarm Panel

Mass Storage Unit

For persistent storage of large data volumes a mass storage unit is supported (but not included) by the TSP platform as an option. The recommended mass storage unit is a CLARiiON IP4700 from the vendor EMC 2 .

If attached, all Processor Modules can access the Mass Storage a NAS (Network Attached Storage) utilizing of the NFS protocol. The storage capacity of the mass storage is in the Tbyte range. The unit provides reliable storage by means of RAID disks.

The unit is a stand-alone self-contained cabinet. It is connected to the inter-processor network via the GESB boards.

Node Design

Since TSP is a scalable product the hardware can be built as several standard configurations from micro (10 PMs) to maxi configurations (42 PMs).

TSP 4.0
configurations:
Micro - 10 Processors
Mini - 21 Processors
Midi - 31 Processors
Maxi - 42 Processors

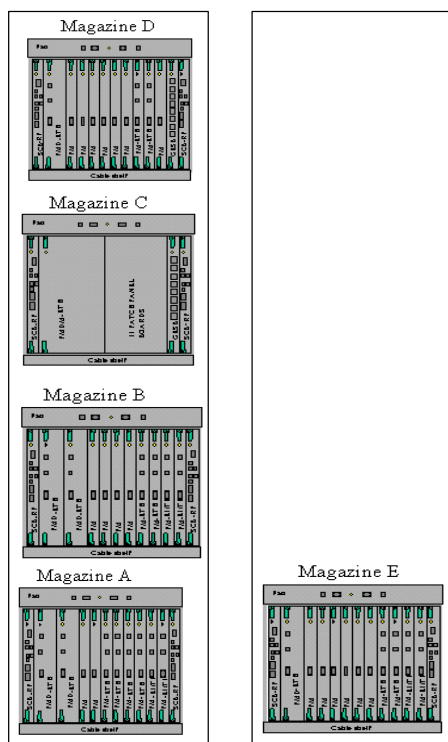


Figure 1-7 TSP Physical Configurations

TSP OPERATING SYSTEMS AND CLUSTERWARE ARCHITECTURE

The layer above the hardware includes the operating systems and clusterware. As mentioned in the first pages of this chapter, each Processor Module (PM) in the platform can be configured to run Dicos or Linux operating systems. The operating systems provide execution support for processes, threads, memory management, scheduling, priorities and so on. Together with the operating system, clusterware software, enables the processors to behave like one single system.

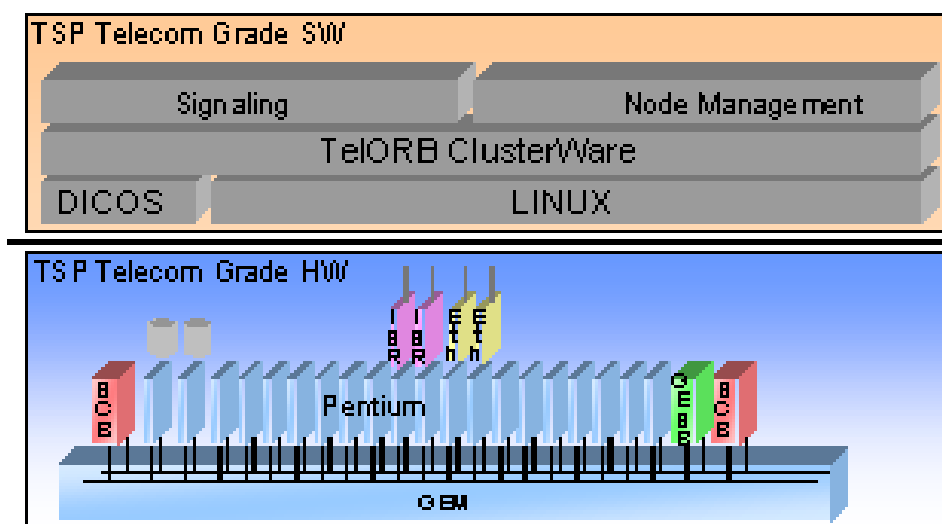


Figure 1-8 TSP Hardware and software

The applications are not considered part of the TSP platform, but sit on top of the node.

TSP uses commercially available components within a framework of open interfaces. This allows the same software to be run on hardware from different suppliers. The component that facilitates this is TelORB. TelORB is software capable of interacting with different processing architectures and allows TSP to be built as a distributed processing environment where software (application software and much of the operating system software) is replicated so that it can be executed by different processes.

TSP OPERATING SYSTEMS

As previously mentioned, each processor in the platform can be configured to run Dicos or Linux. The processors can be flexibly configured between Dicos and Linux during operation with no disturbance to service.

A standard distribution of Linux, Red Hat, is certified to run on Intel processors, to provide Unix-type performance.

Dicos provides a dynamic execution model with very fast process start and isolation, which is ideal where one process maps to one resource (a resource being a call, a database request or whatever is applicable for the application). Since a process and/or thread start is a performance consuming operation in Linux, a static execution model is supported. Thus this execution model is built from more long-lived processes that might use thread pools to execute a set of transactions. In terms of availability, the dynamic execution model is preferred since the smallest recovery and replacement domain is a process. In terms of capacity and real-time, the two models are comparable.

TELORB CLUSTERWARE FEATURES

TelORB itself is a distributed object-oriented operating system, which allows applications run in processes to be distributed across a number of processors. This distribution is transparent to the applications.

Not only is the operating system itself object oriented, it is designed to run object oriented programs. TelORB includes an Object Request Broker (ORB), which can communicate information around the system. Using CORBA (the Common Object Request Broker Architecture), TelORB can distribute information to other ORB's running on different (and non-TelORB) machines and has the following features:

- Distributed object oriented database management system. TelORB's database management system, TelORB database, is responsible for data storage, data backup and recovery and data configuration.
- Support in software for exceptionally high availability.
- Support for configuration management and maintenance.

- Support for online software and hardware upgrades.
- Scalable architecture.
TelORB is linearly scalable i.e. add processors as the demand increases will directly increase the capacity.
- Standard object request broker according to CORBA 2.
- Runs on hardware platforms based on both Sparc and Intel.
- Includes a hotspot Java virtual machine.
- Provides support for a geographically separated redundant system.
- The development environment and simulated environment is based on Sun/Solaris or Linux.
- TelORB is designed to keep applications running whatever happens. A processor may go down, an Ethernet switch may fail, the hard disk may go wrong. Throughout all of these potentially disruptive events, an application running on TelORB will carry on doing whatever it was designed for. This is achieved in software through duplication.

The different sections of the TelORB Clusterware are discussed on the following pages. The following diagram, illustrates the software architecture of the TSP platform in more detail, showing the components of the TelORB Clusterware.

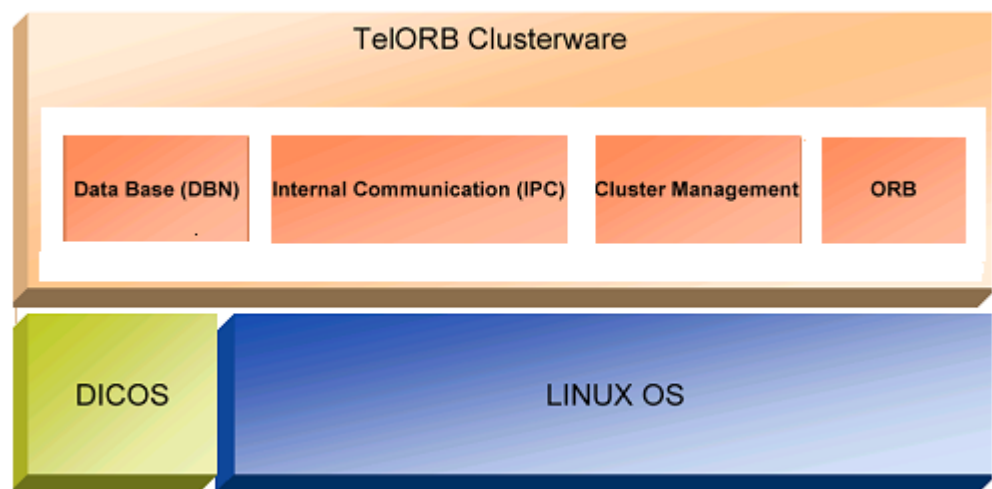


Figure 1-9 TSP Operating Systems and Clusterware Architecture

Communication Layer – T-IPC

The communication protocol of the TelORB Clusterware connects the different processors to each other thus enabling inter processor communication (IPC) or TelORB IPC (T-IPC).

The reason for using the optimized protocol IPC internally in the cluster is to provide high performing communication and high level of robustness when managing the duplicated Ethernet backbones. When communicating to the external world TCP/UDP/IP, version, IPv4, is used.

Database Layer - TELORB DATABASE

The built-in object oriented database of the TelORB Clusterware (TelORB database) provides persistent storage in RAM, which allows for major performance advantages over disk-based storage. This means that the database is always held in primary memory and distributed over the processors. All data is replicated and stored on more than one processor. In the event of a software or processor failure, the entire database is still available to the software.

In this release the database stored in RAM resides on the processors executing DICOS. A client interface on the Linux part of the cluster provides access to the database.

Cluster Management Layer

Cluster Management consists of two components:-

- Software Management Layer

The software management layer automatically configures the executing software to ensure that it runs efficiently on any of the processors available to the TelORB cluster. It provides support for software upgrade, monitoring and management of software components thus achieving high availability. This is also true for binaries from third party vendors.

- Node Management Layer

The Node Management includes an element manager, which is used to configure and manage all the managed objects of the Clusterware and OS.

An example of these cluster management processes include:-

- Configuration Manager processes
- Distribution Unit Group processes

These concepts are described in further detail in the following pages.

In order to understand some of the principles involved in TelORB and how the software architecture is built, it is necessary to explain some common terminology to facilitate understanding. This includes the terminology surrounding processes and the TelORB databases, since TelORB, which is discussed in detail later, is a distributed operating system that incorporates database functionality.

DISTRIBUTED SOFTWARE ENVIRONMENT

PROCESSES

Processes are the fundamental executing objects. A process is an operating system resource that executes program code. A process is specified in Delos. Delos is Ericsson's TelORB specification language and is used to specify processes and objects in the system. A process is implemented in C++ or Java. The code for a process is found in one of the load modules loaded on the processor.

All processes have a 'type' that determines the processes' function, i.e. what program a process executes. There can be several instances of the same process type that do the same things, but on different data.

Processes communicate by means of *dialogues*. Dialogues are software entities specified in Delos, used to handle communication between processes.

Programs executed by different processes interact through message-based communication only, using an inter-process communication (IPC) service provided by the operating system. In the case of TSP, this IPC is provided by TelORB and is hence called TelORB IPC or T-IPC.

At a lower level Dialogues are based on T-IPC (TelORB Inter Process Communication) which takes care of packing, sending, receiving and unpacking messages between processes. Messages can be sent to processes executing on the same processor or any other processor in the TelORB cluster.

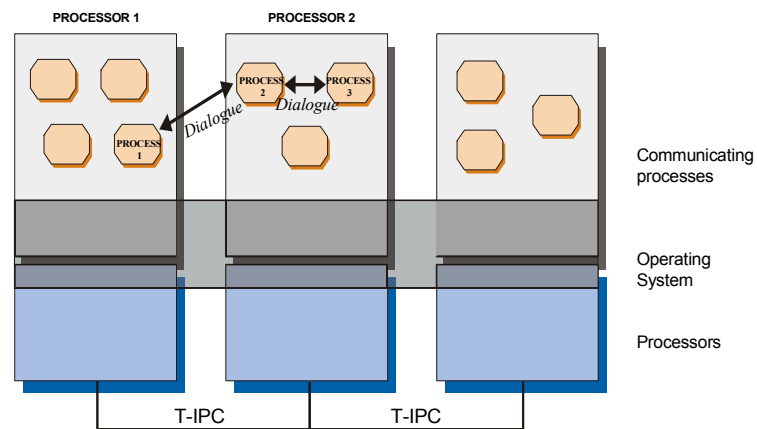


Figure 1-10 Processing Environment

TELORB DATABASE

The distributed database is a central, logical resource that is distributed physically over several processors within the same network element.

It performs the following functions:

- Storage management, which actually stores the data in the primary memory of a processor board. The storage management functionality is integrated with the Dicos kernel.
- Allows access to data via transaction and lock management, making sure that ACID (Atomicity, Consistency, Isolation, Durability) transaction properties are maintained.
- Backup and recovery of processor and network element information
- Configuration of data storage
- Functions to support schema upgrades
- Optional disk logging of transactions
- Optional support for geographical network redundancy

The TelORB database, stores the information that requires persistency such as:

- Long-lived data
- Configuration data
- Data that must be stored safely (such as charging data)
- Data that must be consistent
- Data to be shared by several processes

This information is stored in primary memory of the processors.

The TelORB database does not hold all the information for the network element. Data that is local to the process, temporary, or which does not require the integrity offered by the TelORB database, is stored within the process.

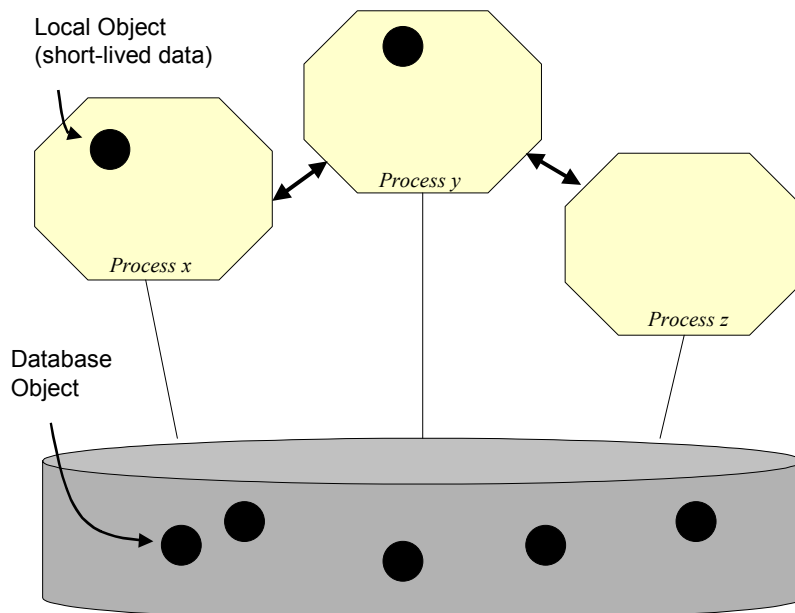


Figure 1-11 Use of the database

Database Objects

The database objects stored in the database are called Persistent Object Types or POTs. POTs are instances in the TelORB database and are used for data that need to survive crashes. It is possible to create, delete, read and modify the POTs. The POT is composed of several attributes (the actual data stored) and can have several methods that implement operations on the attributes. The database concept is discussed on the following pages.

The database is kept consistent at all times. To ensure that different processes have access to consistent information, database changes made by a process, are packaged as transactions.

Database Transactions

Transactions are used to group together operations on database objects so that updates are synchronized with other objects. The changes made by a process are not seen by other processes until the transaction is done, and then all the changes are visible at once.

TelORB prevents several processes from simultaneously changing the same database data by maintaining locks on the objects.

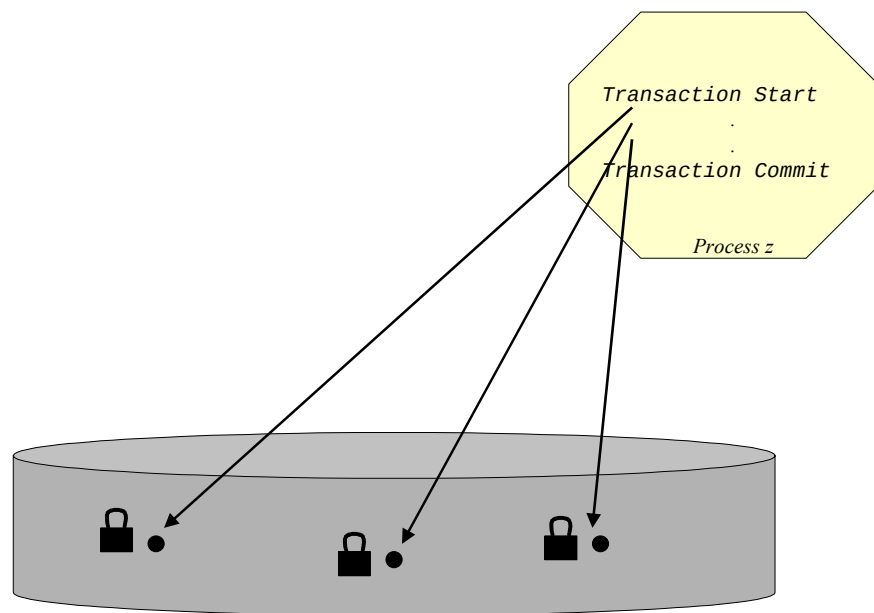


Figure 1-12 Database Transactions

When using transactions, a lock will be put on the accessed instance. The following locks are supported:

- Update Lock: if the instance is updated it will be an exclusive lock allowing no other transactions to open the object.
- Read Lock: this lock allows shared access to an object, is used if the instance is just read.

Application software introduces its own integrity checks by implementing trigger functions that are called during certain operations.

Database Look-up

Objects can be looked up in the database by two means, *primary keys* and *references*. Some POTs distinguish one of its attributes as the primary key for the POT. This can then be used for directly finding a particular object, given its type and key. Every object is assigned a unique identifier at creation that constitutes the basis for look-up by reference. The reference has first to be found as an attribute of some other object.

Keyed Objects

When specifying a database object in Delos, it is possible to give the object a primary key. For example a Subscriber object could have subscriber number as the primary key. An instance can be then be addressed by using the key.

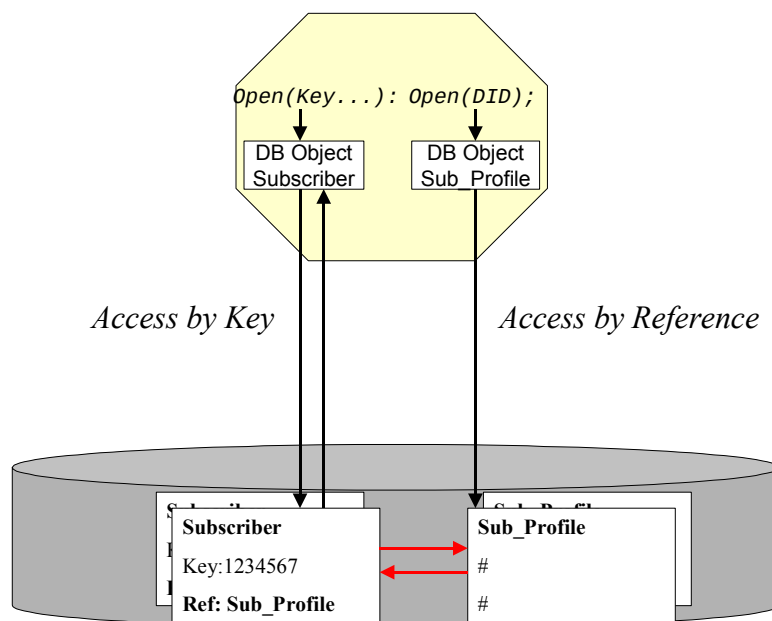


Figure 1-13 Database Object Access

Unkeyed Objects

Not all database objects are specified to have a primary key. For example the Sub_Profile object of the diagram above, has no primary key. But the keyed Subscriber object includes a reference to a Sub_Profile object.

This reference attribute holds a database reference called the Database Identifier (DID). The DID is unique for this database instance within the TelORB system. By first reading this attribute, the Sub_Profile instance can be addressed through the DID.

Data Replication and Recovery

The TelORB database is kept in primary memory. This provides database functionality with very little execution time penalty.

In order to survive crashes, data is replicated on at least two processors and an extended, optimized two-phase commit protocol is employed to replicate changes quickly and safely.

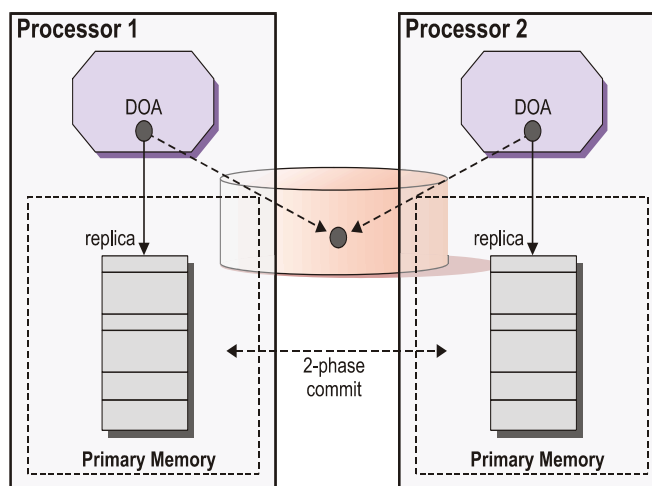


Figure 1-14 Data Replication

The database provides mechanisms for automatic recovery whenever failures occur. The recovery procedures provided by database are as follows:

Crash of application processes do not effect the database. This is also the case when a processor is restarted

Processor restart with database reload which can be due to a kernel error or the crash of a process in the database. When the database on a certain processor is lost it will be recovered from the corresponding replicas. Transaction protected data is recovered to the last committed value. Non-transaction protected data is lost.

Restart of network element with database archive reload. The database contents are recovered from archive backup media. The state of the database will be consistent, but will be older than before the crash.

Both database objects (POTs) and processes must be distributed in a way that evens the load over the processors, but at the same time minimizes communication between the processors. The concept of distributing POTs is discussed in the next section

SOFTWARE DISTRIBUTION

Configuration Manager

The distribution of processes and POTs over the cluster is controlled by the CM executing on Processor 1 or 2 (the IOs). All processors have CM Agents (CMA) through which the CM can control the processes and data of the individual processors.

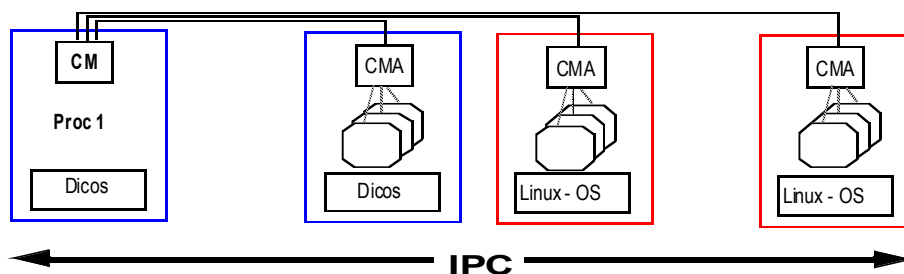


Figure 1-15 Configuration Manager

The grouping of POTs and process types is done by associating a Distribution Unit Type (DUT) specification with the specification of the Process Type or Persistent Object Type.

Distribution Unit Types (DUTs) and Distribution Pools

The DUT is a container for Process Types and POTs. The DUT is specified in Delos, and includes a *distribution limit* for the DUT, indicating the number of so-called distribution unit individuals. A high distribution limit implies more even distribution. The distribution limit tells the software execution environment that the database objects and process instances belonging to this DUT are to be partitioned into that many distribution units.

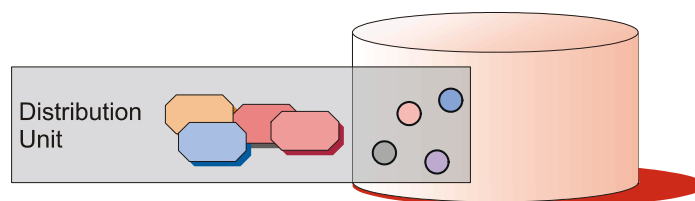


Figure 1-16 Distribution Units

The DUT is allocated to a processor pool. A processor or distribution pool is a virtual set of processors that restricts where database data should be stored and where processes are to execute.

The “*epct* language” or *epct* file, is used for specifying which distribution pools are associated with which physical processors. Each processor can be part of several processor pools.

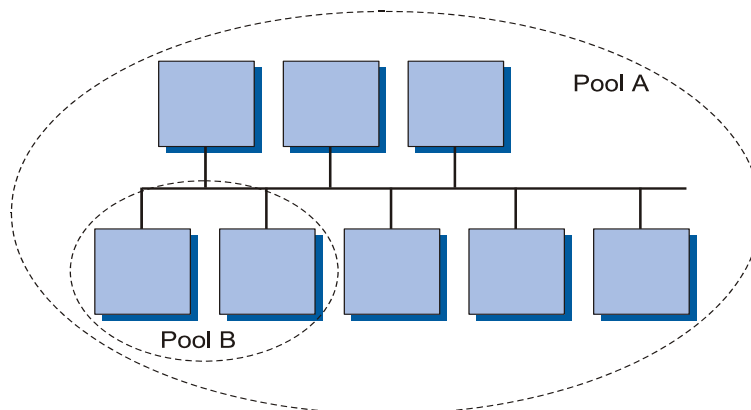


Figure 1-17 Processor Pools

On a 'live' site the following configuration file specifies what pool a processor belongs to and whether or not the processor is a loader:

`/opt/telorb/system/processor.epct`

Physical processors are later associated with the distribution pools on a per site basis to allow for flexible dimensioning.

The information concerning what pools there should be and what DUT's should be allocated to the different pools, is part of the Internal Delivery Package, IDP.

The allocation of DUT's into processor pools is found in the *config.epct* files in the following directory:

`/opt/telorb/system/TelORB_OS/TelORB-ADxx/system/runtime/IDP-xx/`

TelORB-ADxx and IDP-xx are the TelORB version and IDP name respectively.

Process Distribution

As previously mentioned, every *Process Type* is associated to a DUT, which in turn is associated to a processor pool. The DUT is specified in Delos, while the processor pool is specified in a special configuration file. As this is the designers responsibility many DUTs can be “allocated to” the same processor pool as long as they can be distributed in the same way.

The *epct* file will list which pools each processor participates in, this initial configuration can be changed at run-time.

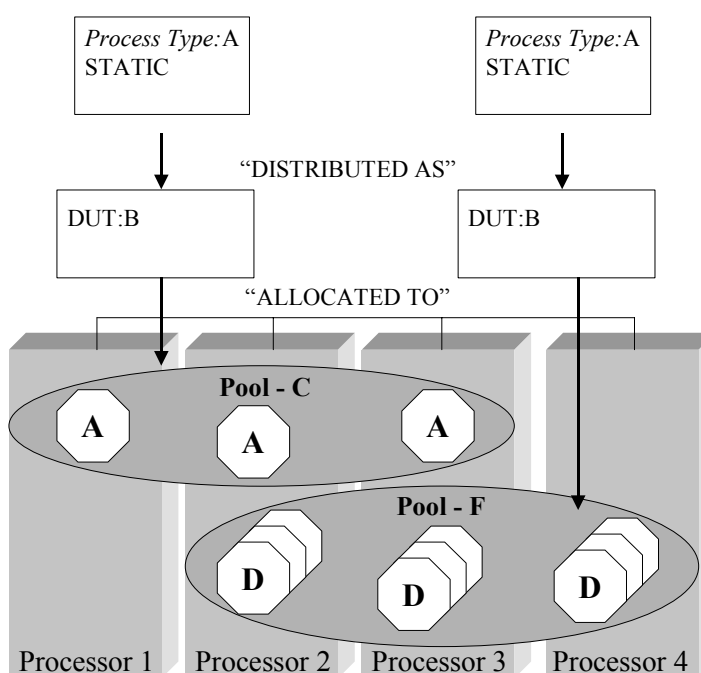


Figure 1-18 Process Distribution

POT Distribution

The Delos specification of a POT includes a reference to a DUT.

It includes the distribution limit for the DUT indicating the number of so-called distribution unit individuals. A high distribution limit implies more even distribution.

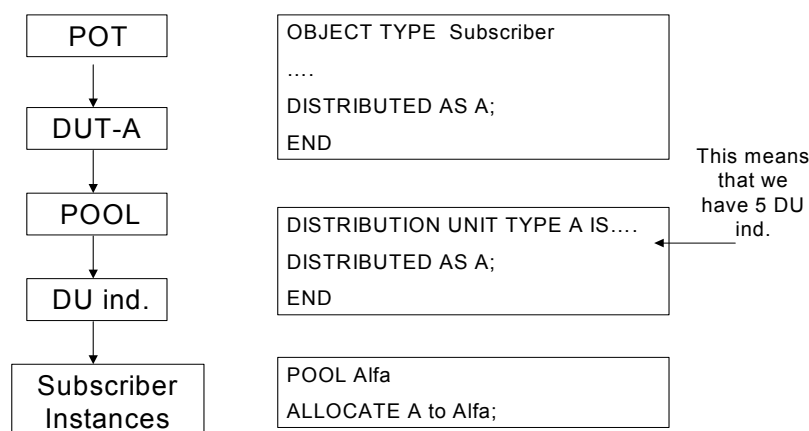


Figure 1-19 POT specification in Delos

Through a specification in a configuration file the DUT is allocated to a processor pool. Typically an application has 1-3 processor pools.

A plant engineer will prepare a site-specific configuration file. This file will, among other things, associate the physical processors to the logical processor pools (see following figure). Note that one processor can be a part of several processor pools. The TelORB operating system is contained in 6 pools.

In the following example the distribution limit is 12, i.e. 12 DU individuals, the pool to which the DUT is allocated is distributed over 3 processors, according to the site configuration (*epct*) file.

TelORB will automatically allocate the 12 DU individuals to the 3 processors, this means 4 DU for each processor.

Processor 1	Processor 2	Processor 3
4-5	0-1	2-3
10-11	6-7	8-9

Table 1 Distribution of 12 DUs on three processors

Note that the algorithm for allocating DUTs to processors is not deterministic, two DUTs with the same key and distribution limit will not necessarily be distributed in the same way.

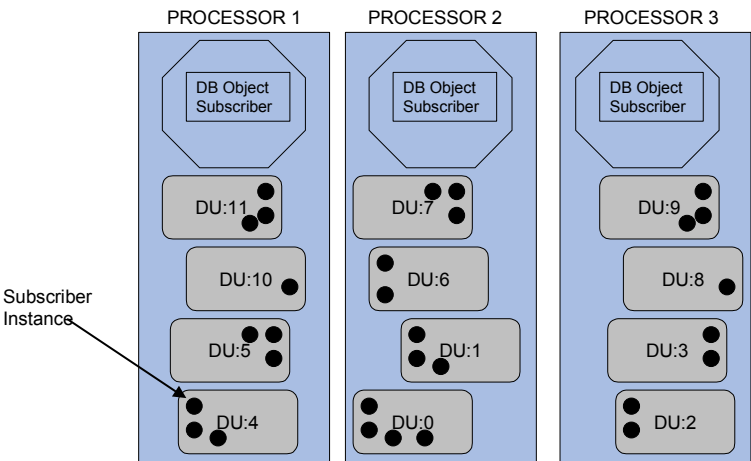


Figure 1-20 A DUT with a distribution limit of 12 allocated to a Pool of three processors

To achieve a fault tolerant database each piece of data is stored in more than one processor. This means that each DU individual exists in two copies, one primary replica and one redundancy replica.

	Proc 1	Proc 2	Proc 3
Primary Replicas of DUs	4-5	0-1	2-3
	10-11	6-7	8-9
Redundanc y Replicas of DUs	2-3	4-5	0-1
	6-7	8-9	10-11

Table 2 Distribution of 12 DUs on three processors

Using this distribution it's clear that even if a processor crashes, we have all the DU's stored in the remaining two processors.

SUMMARY

- The Ericsson Telecom Server Platform (TSP) is a carrier-class server technology designed for server solutions where high availability is required.
- The TSP 4 Platform can be divided into four subsystems – Hardware, Operating System & Clusterware (OS&CW), Signaling (SS7 & IP) and Node Management
- The TSP Platform hardware includes traffic processors, support processors with their peripherals, signaling processors, Ethernet switches, power supplies and fans. All processors are off-the-shelf, single board cPCI standard computers sourced from commercial suppliers.
- The TSP platform is housed in an Ericsson cabinet (BYB 501). An expansion cabinet is available if additional processor modules are needed to give a maximum configuration of 42 PMs.
- The TSP is based on a distributed software execution environment where a number of general purpose processors are loosely interconnected over a network, forming a *cluster* of processors where the execution software is run along with the applications.

Intentionally Blank

2 TSP Operation and Maintenance

CHAPTER OBJECTIVES

After this chapter the student will be able to:

- Describe the TSP Operation & Maintenance architecture
- Outline the Operation & Maintenance functional areas on the TSP Platform
- Navigate the embedded Element Managers – *TSP Node Management Toolbox*
- Use the on-line Documentation.

INTRODUCTION

This module outlines the typical tasks an operator can perform on the TSP platform. It describes the Operation and Maintenance (O&M) architecture. It introduces the interfaces and tools available to interact with the TSP and the on-line user documentation. Using the on-line user documentation, you can become familiar with the available interfaces and tools.

The module explains how user access is managed in a TSP platform. It describes how O&M automatic redundancy is achieved in a TSP platform.

The Operation and Maintenance in the TSP platform allows various network management operations, such as access to subscriber data, network data, access privileges, event recording, performance management and alarm handling, while isolating clients from the database.

OPERATOR TASKS

Operator node management tasks are categorized as Operation and Maintenance.

Operation

Co-ordination and operation of a network on a day-to-day basis, including tasks such as collection of data, monitoring of alarms etc.

Maintenance

Keeping the network working correctly, including tasks such as troubleshooting, testing, repairing, reporting, etc.

Examples of typical O&M tasks are:

- Administration of log files.
- Administration of O&M related processes
- Checking the configuration and state of the system

- Performing backups and reloads
- Upgrading software, when software with additional or changed functionality is added to the system
- Replacing, adding and removing hardware
- Loading initial data into the database
- Recognizing notifications
- Responding to alarms.

OVERVIEW OF TSP O&M

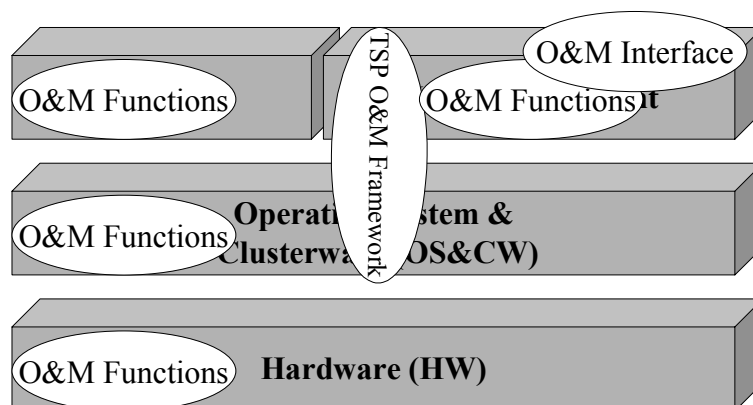


Figure 2-1 TSP Architecture

The above diagram shows the four subsystems in the TSP Platform, namely Hardware, OS & Clusterware, Signaling and Node Management. The O&M functionality is distributed over all subsystems.

The O&M functionality can be divided in three different areas:

- The **O&M Framework**, which provides the platform O&M support for the applications as well as for the platform software.
- The **TSP Platform O&M functions**, which uses the O&M framework to provide O&M data such as alarms, counters and configurable parameters.
- The **O&M interface**, which will interface to the external systems and users.

O&M FRAMEWORK

The O&M framework provides the interfaces for O&M that are used by the applications. It is implemented by the operating system and clusterware and node management subsystems, with the low-level interfaces often implemented at operating system and clusterware level.

The O&M framework is built based on the Telecommunications Management Network (TMN) model (as recommended by the ITU-T) and corresponding protocols pertaining to network management.

All subsystems provide O&M functions e.g. by generating alarms or counters for performance management.

TSP O&M PLATFORM

The TSP O&M Platform offers the following features:

- **Development Environment.** The TSP O&M framework offers software classes or packages that can be extended by an application to provide functionality to their end-user.
- **End-User O&M applications** The TSP O&M also provides O&M applications used by the end-user - for example, the Configuration Management is an example of a stand-in O&M application used directly by the end-users (for example, operator).
- **High Availability.** If, for example, the O&M connection with the TSP platform is lost, service users must reconnect to the system and the current transactions during the failure must be resubmitted. In the case of fail-over due to problems from traffic perspectives, provisioning, events and exporting still work. The current transaction during the failure must be resubmitted.
- **User Control.** Both user authentication (at login time and before certain other sensitive operations) and authorization are offered. All application user names and passwords, access groups and access lists are administered through the TSP O&M interfaces.

- **Open System.** The TSP O&M is an open system; it has standard interfaces for Configuration, Fault and Performance Management (for example, SNMP compliant for Configuration Management, 3GPP and SNMP compliant for Fault Management, and compliant to the 3GPP IRP Performance Management XML output file format). In this way TSP 4.0 can be easily integrated with external O&M products using the standard mentioned interfaces.

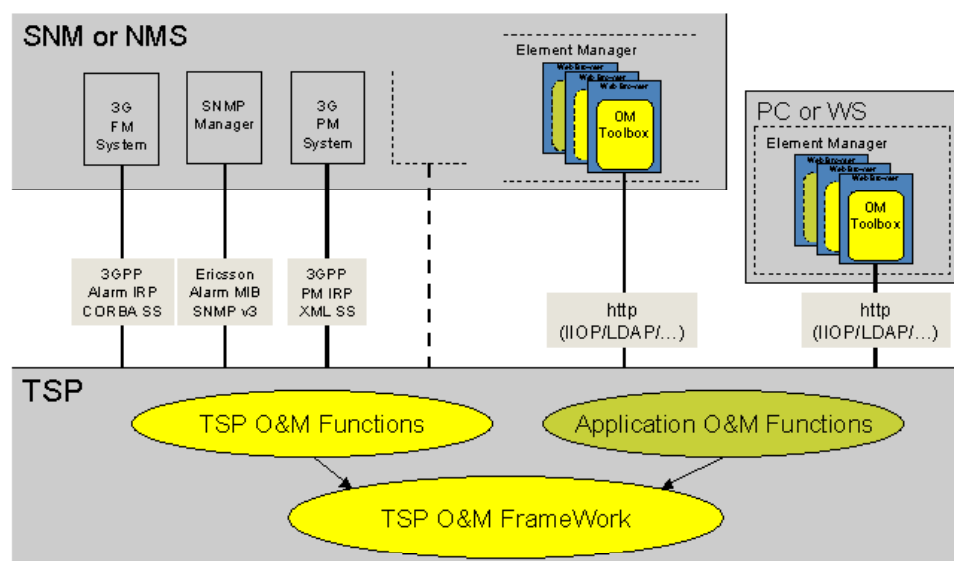


Figure 2-2 Overview of TSP O&M Support

O&M INTERFACES

The TSP platform provides two types of external interfaces for management:

- An embedded Element Manager (EM)
- A number of open standard protocols for applications executing in Sub-Network Managers (SNM) (like GSM OSS,CN-OSS) or in Network Management Systems (NMS).

The Embedded Element Manager

The Embedded Element Manager (EM) is a set of GUI (Graphical User Interface) applications for thin clients (i.e. web browsers) that covers all tasks required to manage the platform part of a TSP based node. It can also be extended to include GUIs for management of application specific areas. It can be used as an element manager running on a PC or workstation and can alternatively be used as an element management application (typically for configuration management) in an SNM or NMS environment.

Therefore, the web browser is used to download a GUI client application, which then provides access to the management capabilities of TSP, so no special-purpose local terminal is required. Node Management can therefore be executed locally or remotely. The EM is also known as the *NM Toolbox* or *O&M Toolbox*.

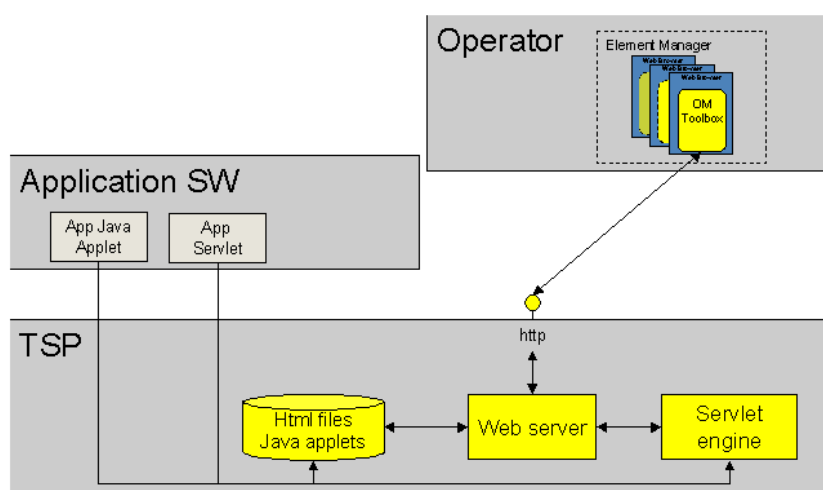


Figure 2-3 The Node Management Toolbox

The Toolbox is the main interface to manage the nodes as seen in the figure above. It provides the interfaces/GUIs to perform logging, configuration management, fault management and performance management. The center of the Toolbox is a web server from which all applications are launched. The Toolbox also provides authentication and authorization of users. Applications can extend the Toolbox with html pages, applets or servlets. The Toolbox provides functionality to incorporate:

- Software upgrades
- Backups

- Processors
- Software inventory
- Hardware inventory
- Domain Name Server (DNS)/Network File System (NFS) configuration
- Data and time management
- Internet Service Provider (ISP) History
- Network Redundancy
- TelORB Database Disk Log
- Virtual IP configuration

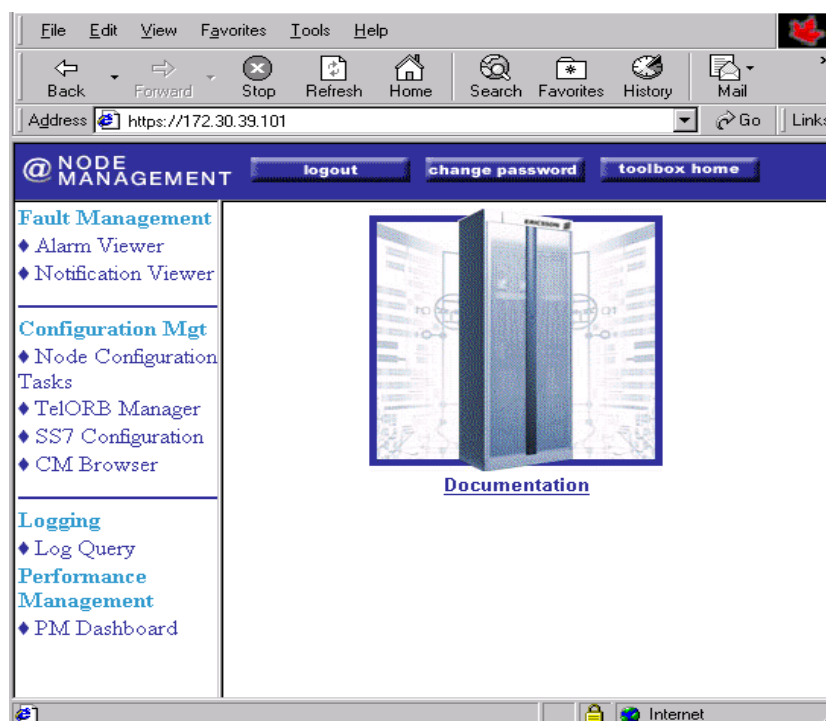


Figure 2-4 Node Management Toolbox

The Toolbox contains the following components:

- User Login GUI –for user authentication and authorization purposes.
- Live Alarm Display (LAD) – for alarm presentation
- Live Notification Display (LND) – for presentation of notifications
- TelORB Manager – for configuration of TelORB

Operating System and Clusterware. This includes task-oriented GUIs for software upgrades, backups, processors, software inventory and hardware inventory.

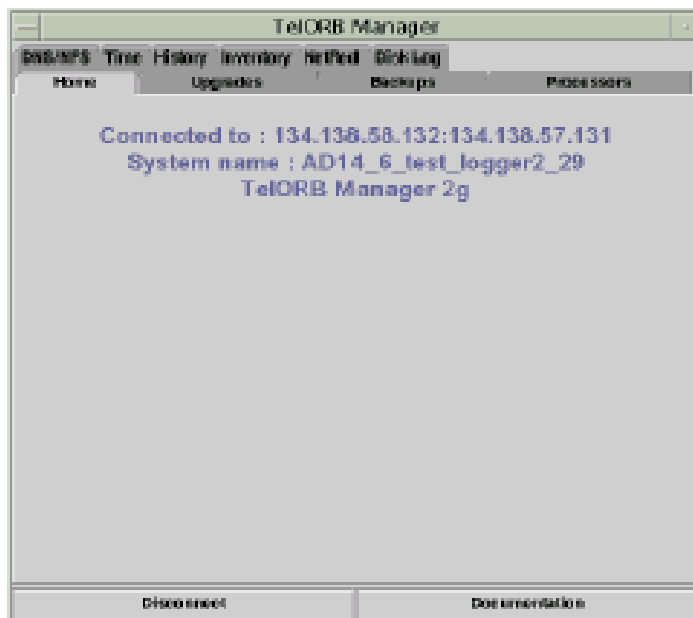


Figure 2-5 TelORB Manager

- Ericsson Signaling Stack (EIN) GUI –for configuration of the SS7 stack
- Query Tool – for querying logged information stored in the on-board relational database.
- Lightweight Directory Access Protocol (LDAP) browser – for platform and application configuration.
- Configuration management – a set of node configuration tasks to populate, extract and manage configuration data.
- Performance Management Dashboard (PMD) – for on-line inspection of performance management data.

A web browser for Customer Product Information (CPI) can also be launched from the Toolbox.

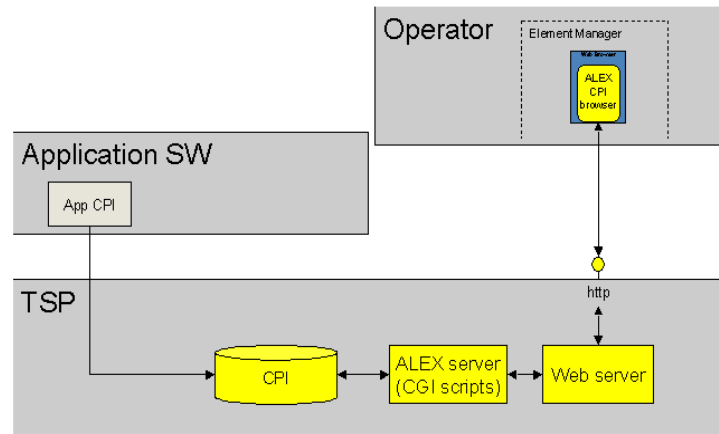


Figure 2-6 Alex server for CPI

Open Standard Protocols

The other types of external interfaces to allow node management, are the protocols for termination in some O&M application in the SNM or NMS. The purpose is to provide a uniform view of the node according to a set of standards like the 3GPP Integration Reference Points (IRPs).

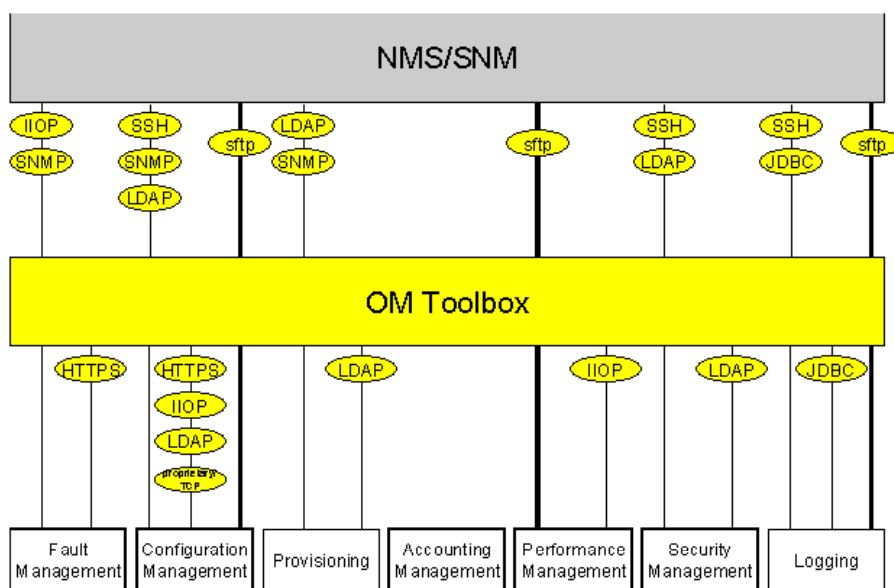


Figure 2-7 Application level protocols

The above figure shows the application level protocols that are provided towards Network Management Systems (NMS) Sub-Network Management Systems (SNM) as well as the protocols used between the node and the Embedded Element Manager. Note that the OM Toolbox can also be used as an application within an NMS or SNM.

NODE MANAGEMENT FEATURES

This section describes Node Management features (as shown in the previous diagram), which include Fault Management, Configuration Management, Provisioning, Performance Management, Security and Logging. These also fit in with the standard, traditional TMN FCAPS functional areas, namely:

Fault Management

Configuration Management

Accounting Management

Performance Management

Security Management

Currently, the platform does not provide any dedicated support for accounting management.

FAULT MANAGEMENT

Fault Management feature in TSP 4.0 is an O&M function that informs an operator about the health of the TSP platform. The TSP platform may be integrated into customer fault management systems using either of the following interfaces:

- SNMP v3 interface
- 3GPP IRP Fault Management interface

The TSP platform also offers its own Live Alarm Display (LAD) and Live Notification Display (LND) GUIs.

Alarms are discussed in detail in the following chapter.

Fault management provides a set of services to notify an end user (Network Management Systems, NMS) that a problem, requiring human intervention, has been detected. Of all interfaces that the platform provides, only the alarm interface will be monitored 24 hours a day.

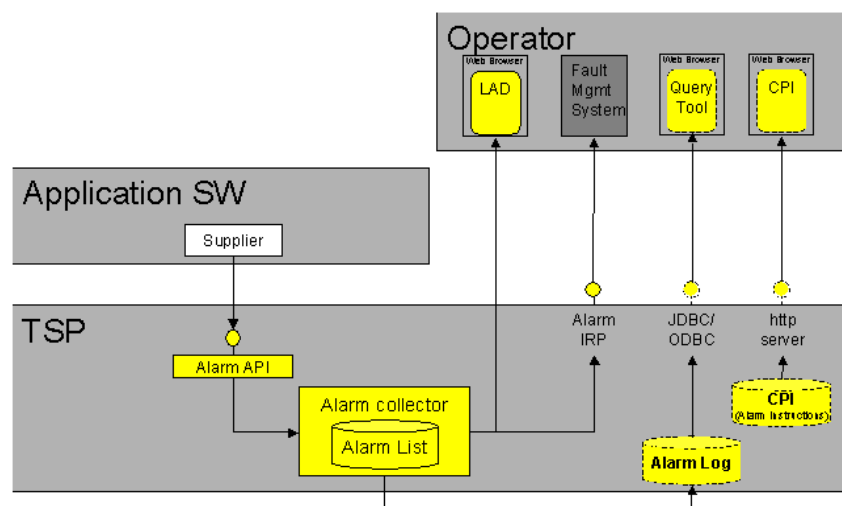


Figure 2-8 Overview of Fault Management

CONFIGURATION MANAGEMENT

The TSP platform may be integrated into customer centralized configuration management systems through the SNMP v3 interface. Commercially available SNMP v3 management products (such as “Hewlett Packard’s OpenView”) can connect and manage platform and application-managed objects within the TelORB database. The configuration management interfaces that an operator can use are:

- LDAP generic browser

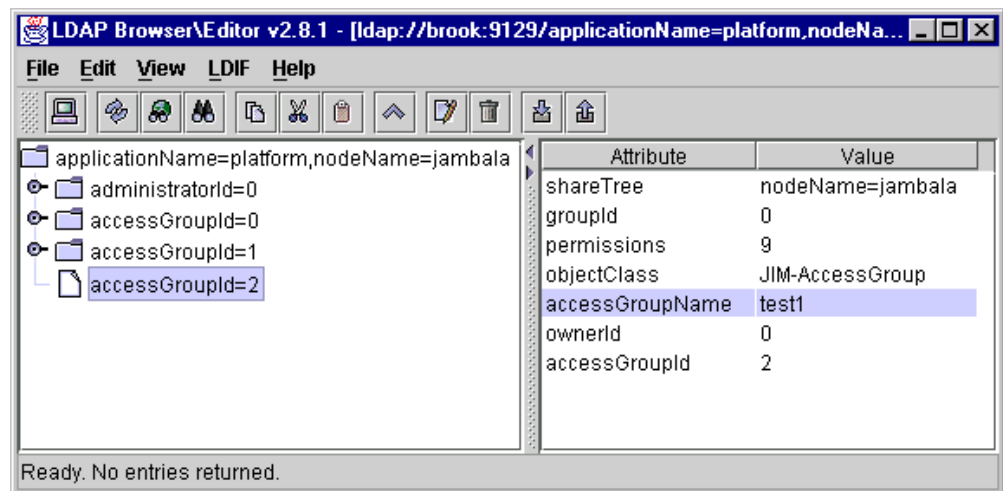


Figure 2-9 LDAP Browser

- Specialized GUIs

The specialized GUIs and the LDAP browser are initiated from the Node Management (or O&M) Toolbox.

Provisioning

Provisioning refers to the loading or retrieving of subscriber related data.

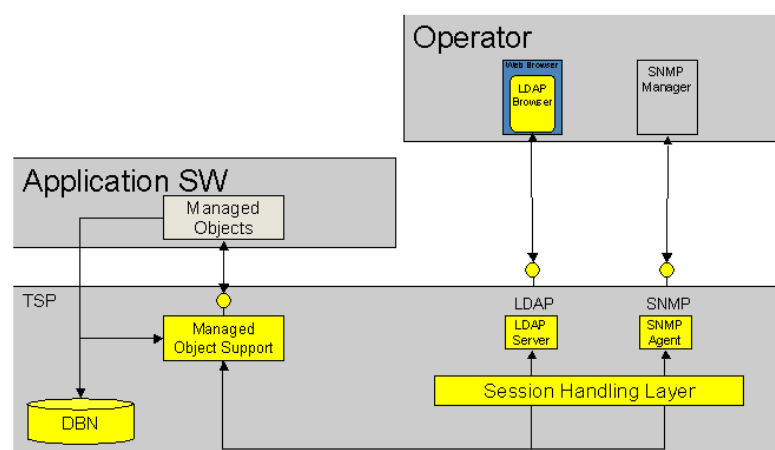


Figure 2-10 Overview of provisioning

The provisioning support can be seen as a layered structure consisting of:

- An access layer that provides scalable access for IP traffic. This layer enables having a single system view (single IP address) to the distributed system. The access layer is used also for purposes other than O&M.
- A session handling layer that provides authentication and management of sessions between clients and the server.
- Management of provisioning data (or configuration data) is provided as a managed object model. The application implements managed objects and the relations between the managed objects.
- Finally, the common point for persistent storage is the distributed database.

A generic provisioning framework called JAMBALA Information Manager (JIM) is offered to TSP applications wishing to take advantage of common provisioning services like, security management, authentication, authorization and logging.

JIM

JIM provides a set of APIs that allows applications running on TSP management of objects in TelORB's database. JIM offers a robust, generic way of getting data into the database. JIM clients can perform the following operations necessary to maintain the database:

- Create
- Retrieve
- Update
- Delete
- Action Invocation
- Attribute Compare Operations

JIM is designed to support the type of data model that can be accessed by protocols like LDAP, CMISE and CAI, but is not tailored to any specific protocol.

PERFORMANCE MANAGEMENT

Performance Management provides the functions needed for reporting statistical measurements for analysis and tuning of the network element. It has functions for collecting performance data and a management interface to the external operators.

An application (as well as platform components) that is to provide statistical measurements reports their values via the reporting interface to the PMF (Performance Management Framework) of the TSP platform. TSP provides both C++ and Java interfaces on both Dicos and Linux. PMF collects the values and stores them in the TelORB Database.

The supported measurement types are:

- counters
- gauges
- status inspection

At each reporting occasion PMF will collect all values from the TelORB database and create a snapshot of the measurement data and store this in a measurement file in XML format.

The TSP platform can be configured to automatically generate threshold-based alarms. If a measurement value exceeds a configurable threshold value, then a (configurable) alarm will be raised, and if, for a subsequent measurement, the value goes below another threshold value, then the alarm will be cleared.

The TSP platform provides measurements for:

- Signaling statistics (e.g. SS7 throughput, components sent and received at TCAP level)
- Platform statistics (e.g. CPU load, memory usage, processor uptime).
- O&M statistics (e.g. provisioning throughput in TSP).

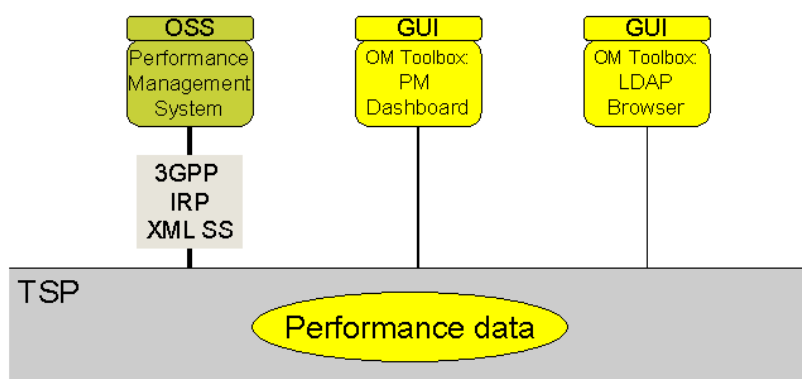


Figure 2-11 External interface and protocols for PM

The platform provides PM files in XML file format according to:

- The 3GPP PM IRP XML SS (3GPP TS 32.104 v3.4.0)

These files can be fetched or pushed from the node to the support systems using secure ftp. The configuration of measurements is done by means of the LDAP browser through the JIM interface described previously.

For element management purposes the platform provides the Performance Management Dashboard that is an application of the NM Toolbox. The PM Dashboard (PMD) can be used to view subsets of the counters, gauges and status inspections that the platform provides.

LOGGING SUPPORT

The logging support enables applications to store information on disk for later analysis. The platform provides a best-effort logging function with minimal impact on system performance.

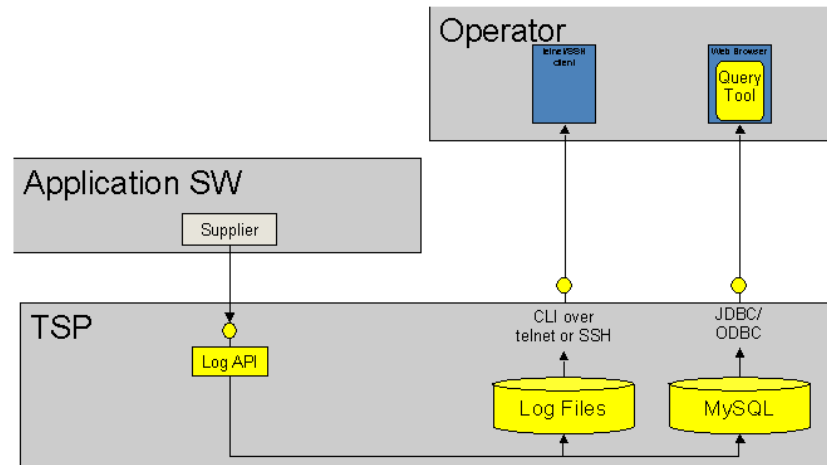


Figure 2-12 Overview of logging support functions

The applications can provide data for logging through APIs provided by the platform. TSP provides C++ and Java interfaces on the Dicos and Linux as well as an IDL interface. The logs are stored either as flat files, or in a relational database.

The platform is responsible for storing the log records on files, eventually inserting them in the relational database, and for providing all the external interfaces to the operator/trouble-shooter. The logging service can be configured to collect log records generated on all processors by a specific application in a separate file thus providing a system wide log for that application. The logging service does not guarantee delivery of every log record, since the support has been optimized for performance rather than reliability.

The platform provides a set of default logs. Among this is the TelORB system log, which logs important events related to the TelORB operating system, the Alarm Log, which provides history information on raised, updated and cleared alarms, and the provisioning log which logs all provisioning commands.

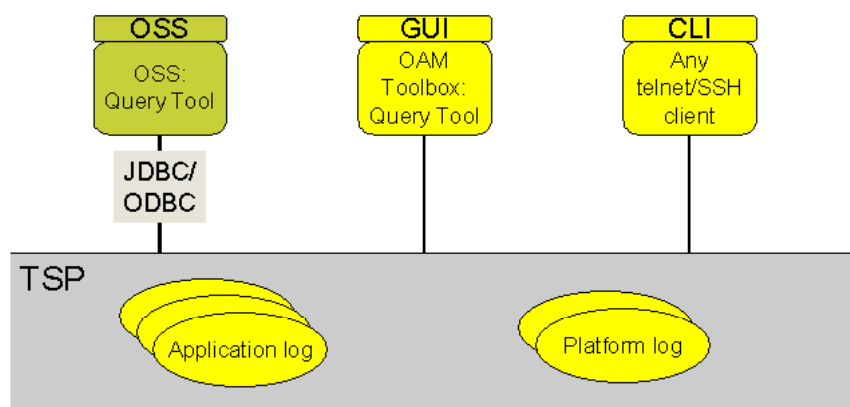


Figure 2-13 External interfaces and protocols for logging

The logging support in TSP provides three external interfaces: a Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) interface to the logs stored in the relational database, a query tool that is included in the O&M Toolbox or by means of logging in on the IO processors and using the standard Unix command line interface to view the simple log files.

ACCESSING A TSP NODE

From a physical point of view, the TSP platform is connected to the O&M network through two kinds of links:

- Connection through IOs
- Virtual IP

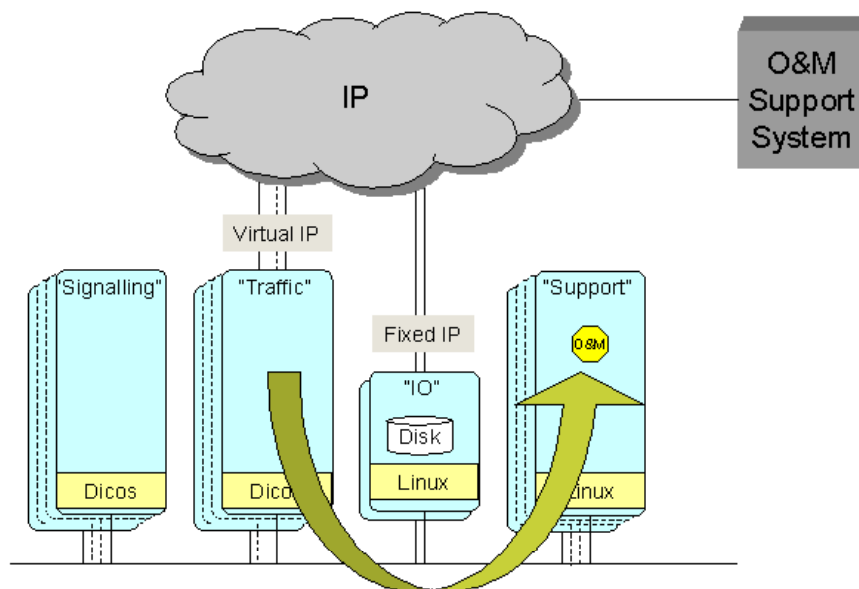


Figure 2-14 Physical connection to O&M network

The two IO processors are connected to the IP network and are accessible by means of two fixed IP addresses, one for each processor. The IO machines are only used at initial installation and during emergency repair, when the cluster is not working properly.

For all ordinary O&M, the connection from the O&M Support Systems to the node is made through Virtual IP, which is described in detail in Chapter 11 of this course. The physical connection goes from a router on site to several of the traffic processors (PM-ETH) running the Dicos operating system. An incoming IP message is routed internally in the system to the receiver, which can be running on e.g. a support processor.

VIRTUAL IP

For normal, ordinary Node Management, the connection from the Node Management Support Systems to the node is made through Virtual IP. The use of Virtual IP for Node Management in TSP implies that the configuration of the O&M client in the support systems will be simple, since the connection is based on a single (Virtual) IP address for each function and not on the physical location of the software. The operator does not need to know which processor handles a specific service. It also implies improved capacity and scalability for Node Management functions since important software components can be distributed over several processors while still being accessible over a single IP address.

CORBA Interface

TSP supports the open Common Object Request Broker Architecture (CORBA) interface. This means that operators can communicate with the TSP platform using any type of interface that is CORBA-compliant.

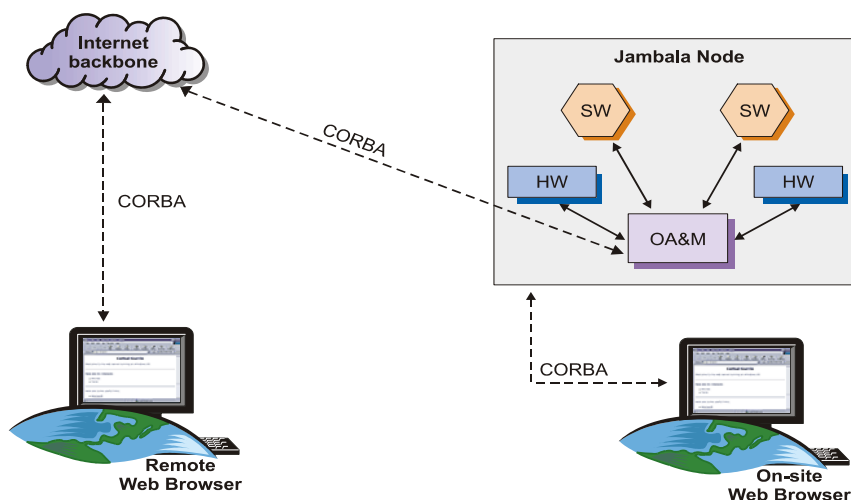


Figure 2-15 Remote and on-site O&M using a CORBA interface

CORBA is an open, distributed object, computing infrastructure standardized by the Object Management Group (OMG).

The TSP platform supports an external CORBA interface over an Ethernet connection to the Network Management Systems (NMS). The operations used adhere to CORBA standards for customer data administration.

CORBA makes it possible to use the Internet/Intranet as a backbone through which Network Management Systems can bridge to perform on-line, on-site and remote O&M operations. This provides operators with a web-based O&M solution and the ability to develop and use new O&M applications.

OPERATOR TOOLS

Operators can communicate with the TSP using a number of software tools. These tools run on the IO processor and display information on the operators' workstations. To access these tools you need to connect to the IO using *telnet* or *rlogin* and set display to your machine. For example:

- `rlogin (IO processor IP address) -l telorb`
- `password:`
- `setenv DISPLAY (your IP address):0.0`

COMMAND LINE INTERACTION

Many of the interactions with the TSP are by means of a GUI, however there are a number of operator tasks, particularly those involving the IO processors, where the operator uses commands. The IO processors can be Linux processors and therefore Linux commands are used when working with them. In addition, the operator may be required to invoke Ericsson tools running on the IO processors by command. These tools are outlined in the following section. The on-line user documentation describes the commands used with these tools. The names of the files and directories used in conjunction with these commands will vary from site to site.

PROCESSOR LOAD VIEWER

The processor load viewer displays the load on each TelORB processor. Its appearance is similar to a bar chart, with a bar representing each TelORB processor. Since the load is fluctuating continuously, the processor load viewer is updated constantly. The load on a processor is the number of tasks executing and the priority of those tasks. Four different colors are used to represent the priority of the tasks being executed:

1. Red is for a high priority task, such as servicing hardware or operating system tasks such as database transaction recovery.
2. Green is for a normal priority task, such as traffic processing
3. Yellow is a low priority task, such as a backup

Command input:

```
io1>cd /opt/telorb/
```

```
io1>ls
```

```
io1>tssh
```

```
io1>source EnvSet
```

```
io1>win -n -i&
```

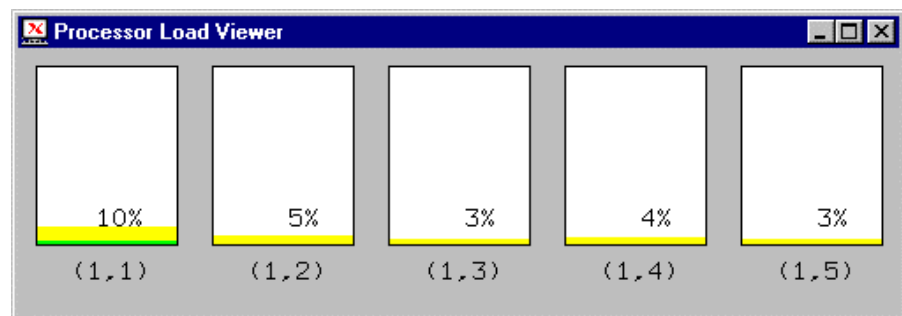


Figure 2-16 Processor Load Viewer

TELORB CONSOLE WINDOWS

The TelORB console windows can be used to see start-up messages and CORBA messages, received by each processor. There is a console window for each processor. They can be used during initial installation and for trouble-shooting. There can only be one instance of these windows displayed at any one time.

```

0054656c6f726250726f633200c1da0000140000000000002000f469f000000000000000000000
00000000030000000010100000c00000054656c6f726250726f633100c1da0000140000000000
02000f469f00000000000000000000000000000000
-----> Starting central process(es) in DU(s) with id=1001095
DEBUG INFO: DicosDnsMQ_Impl: IOR:010000001c00000049444c3a4469636f73446e732f4469
36f73446e734d4f3a312e300002000000000000030000000010100000c000000054656c6f726250
726f633200c1da00001400000000000002000f468700000000000000000000000000000000300000
0010100000c00000054656c6f726250726f633100c1da00001400000000000002000f4687000000
0000000000000000000000000000000000000000
-----> Starting central process(es) in DU(s) with id=1001131
DEBUG INFO: DicosCmInfo_Impl: IOR:010000002200000049444c3a4469636f73436d496e666
2f4469636f73436d496e666f4d4f3a312e3000000002000000000000030000000010100000c000
0054656c6f726250726f633200c1da00001400000000000002000f46ab0000000000000000000000
00000000030000000010100000c00000054656c6f726250726f633100c1da0000140000000000
02000f46ab00000000000000000000000000000000
-----> Starting central process(es) in DU(s) with id=1002229
DEBUG INFO: void SystemTraceProcess_Impl::systemStarted ()
-----> Starting installed process(es) in DU(s) with id=998031
-----> Starting central process(es) in DU(s) with id=1000698

```

Figure 2-17 TelORB Console Window

It is useful to watch the TelORB console windows during a zone reload. The message *Total Functionality Started* in a TelORB console window indicates that the zone reload is complete.

Command input:

```
io1>cd /opt/telorb/
```

```
io1>ls
```

```
io1>STS_console {1 2 3.....or all}
```

ON-LINE DOCUMENTATION

On-line documentation provides information related to objects, alarms, notifications and job procedures/OA&M tasks. It also has a search feature.

Examples of the documentation include:

- Backup and restore
This document gives general information about the tasks involved in backing up a copy of TelORB either to the disk on the IO processor or to an archive tape, and how to restore the system from such a backup or archive. This area includes the following procedures:
 - Backup TelORB
 - Backup IO processor
 - Restore IO processor
 - Restore TelORB
- TelORB clock procedures
These documents describe the functions provided by the TelORB clock and how to modify clock information.
- Dealing with error dumps
- Upgrading software resident on the IO processor
This document takes you through the process of upgrading any of the software parts of the TelORB system that are resident on the IO processor.
- Upgrading TelORB software
This document takes you through the process of upgrading any of the software parts of the TelORB system.

SUMMARY

- There are a number of interfaces and tools for the TSP, the main interface being the Node Management Toolbox, from which GUIs for performance, configuration etc are derived.
- On-line user documentation is available on the TSP platform containing the commands required to operate these interfaces and tools.
- The O&M can support different access privileges to represent operators. It also supports transaction logging to trace a series of events.

Intentionally Blank

3 Fault Management

CHAPTER OBJECTIVES

After this chapter the student will be able to:

- Use the user interface for receiving alarms and notifications
- Find the relevant alarm information in the on-line documentation

INTRODUCTION

Fault Management (FM) is the reporting system that informs an operator of the health of a TSP platform. The TSP platform may be integrated into customer fault management systems. In this respect, the platform provides two interfaces for OSS applications:

- The 3GPP Alarm IRP CORBA SS (3GPP TS 32.111-3 v 3.3.0)
- Ericsson Alarm IRP MIB over SNMP v3

The TSP 4.0 O&M includes two default viewers:

- Live Alarm Display (LAD) or Alarm Viewer
- Live Notification Display (LND) or Notification Viewer

These display the list of active alarms and notifications. The displays can be viewed in any simple browser that supports HTTP1.1 protocol. The displays or viewers support visual filtering of events based on event attributes. This filtering is for display purposed only. It has no effect on the performance of other interfaces like 3GPP IRP or SNMP.

ALARMS

An alarm is an event that may or does affect normal operations of the system. It has to be cleared in order to disappear. Alarm report faults in the system, which usually involve human intervention for correction, for example, disk storage shortage. Alarms are a special event, in that they have a simple lifecycle or state:

1. Active
2. Acknowledged
3. Cleared

Alarms also have severity's that are identified as follows:

- Critical
- Major
- Minor
- Warning

This helps the user prioritize their analysis and corrective actions. Only the reporter of an alarm may change its state from Active to Cleared, since only the reporter can figure out when the hazardous condition is gone.

NOTIFICATIONS

In addition to alarms, notifications are sent to the user, to indicate that something has happened. These are asynchronous messages telling a user that something has occurred. They are similar to alarms but notifications do not indicate an error situation, they do not have states as alarms do and they do not require any human intervention. Examples of notifications are messages telling the user that a backup has been started, or that a system upgrade has finished successfully. Although the functionality is similar, the TSP notifications are not to be mixed up with OMF's CORBA Notification Service or the 3GPP Notification IRP.

All alarm and notifications are characterized by name and value pairs. These pairs can include information such as the source of the event, their time stamp, the reason for it arising and other pertinent information.

FAULT MANAGEMENT

Fault management provides a set of services to notify an end user (Network Management Systems, NMS) that a problem, requiring human intervention, has been detected. Of all interfaces that the platform provides, only the alarm interface will be monitored 24 hours a day.

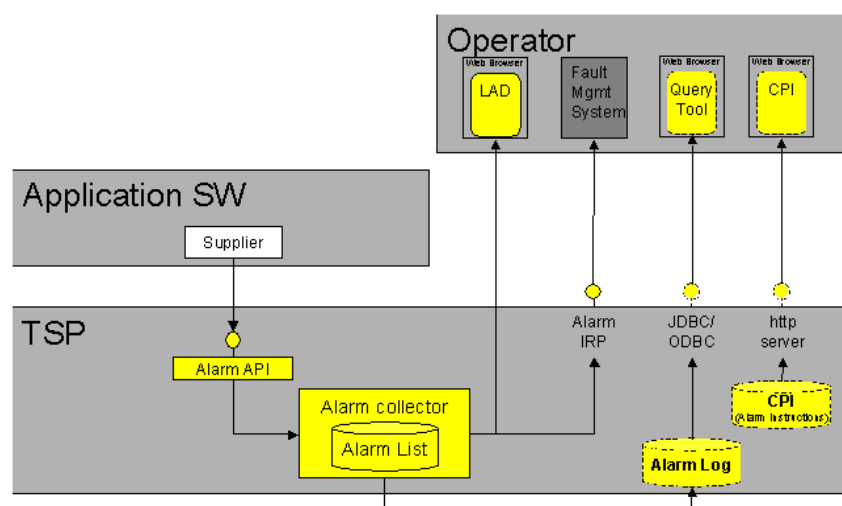


Figure 3-1 Overview of Fault Management

The *Application Software* is responsible for raising, updating and clearing alarms through Application Programming Interfaces (APIs) provided by the TSP platform. The TSP platform takes care of the internal storage, distribution and logging of the alarms – *Alarm Collector* and *Alarm Log* - as well making them available to the operator through a number of external interfaces. A simple alarm filtering mechanism is implemented to protect the OSS system from alarm flooding due to ill behaved alarm providers.

The applications (and the TSP platform) are also required to provide instructions that describe the necessary actions to be taken in order to remove the problem associated with the alarm. These instructions are included in the *Customer Product Information (CPI)*, which is stored on the node and which is made available for browsing through a web interface.

The TSP platform provides a set of alarms for platform-associated errors. The alarm history (such as when the alarm was raised, when it was acknowledged and by whom, and when the alarm was cleared) is stored in a log file, *Alarm Log*.

In addition it is possible to configure the system to raise alarms when a performance measurement value passes through a threshold. See the section on Performance Management for more information on threshold based alarms.

The operator may also connect external sensors (e.g. intrusion detection, fire detection) to the cabinet through the SCB boards, and have these events sent as alarms over the same interfaces as the alarms raised in the cabinet.

For element management an alarm viewer (Live Alarm Display or LAD for short) is included in the O&M Toolbox. In addition, TSP 4.0 will also provide an optional wall mounted alarm panel that can be used on site to give an overview of the alarm situation. The HW is identical to the alarm panel of AXE 10, but with different presentation layout.

FAULT MANAGEMENT CORE COMPONENTS

The Event Collector collects all alarms and notification events in the system. It provides an interface for registering the list of alarms that can be raised by a given source.

The TSP 4.0 alarm and notification collectors collect all alarms and notifications generated by a component of the TSP 4.0 cabinet – including TelORB alarms and notifications, SS7 stack and notifications and Unix environment alarms.

The collectors use two CORBA event channels:

- the first one is a bus for all alarms and notifications generated by reporters and,
- the second one is a bus for all events after handling by the collectors – these are shown by the event viewers.

All events, both alarms and notifications, are characterized by name/value pairs. These pairs can include (but are not limited to) the source of the event, time, reason and so on.

Since they have states, alarm information has to be persistent, which is achieved through the TelORB database.

Collected alarms are logged using the TSP 4.0 Logging module.

Live Alarm Display and Live Notification Display

The LAD and LND, as previously mentioned, are two default viewers in the Fault Management feature of the TSP 4.0, that provides viewing of alarms and notifications in the system. Furthermore, the LAD also supports alarm filtering.

Severity	ID	Time	Type	NOClass	NOInstance	Cause
Warning	Processor 11 Proc_m0_s17 11:23	20020610 22:52:59:000	Equipment	Processor	Proc_m0_s17	Transceiver or Cable fault
Warning	Processor 11 Proc_m0_s19 11:23	20020610 22:52:59:000	Equipment	Processor	Proc_m0_s19	Transceiver or Cable fault
Warning	Processor 11 Proc_m0_s15 11:23	20020610 22:52:59:000	Equipment	Processor	Proc_m0_s15	Transceiver or Cable fault
Warning	Processor 11 Proc_m0_s17 11:22	20020610 22:52:59:000	Equipment	Processor	Proc_m0_s17	Transceiver or Cable fault
Warning	Processor 11 Proc_m0_s19 11:22	20020610 22:52:59:000	Equipment	Processor	Proc_m0_s19	Transceiver or Cable fault
Warning	Processor 11 Proc_m0_s15 11:22	20020610 22:52:59:000	Equipment	Processor	Proc_m0_s15	Transceiver or Cable fault
Warning	Processor 11 Proc_m0_s17 11:23	20020610 22:52:59:000	Equipment	Processor	Proc_m0_s17	Transceiver or Cable fault

Figure 3-2 Live Alarm Display

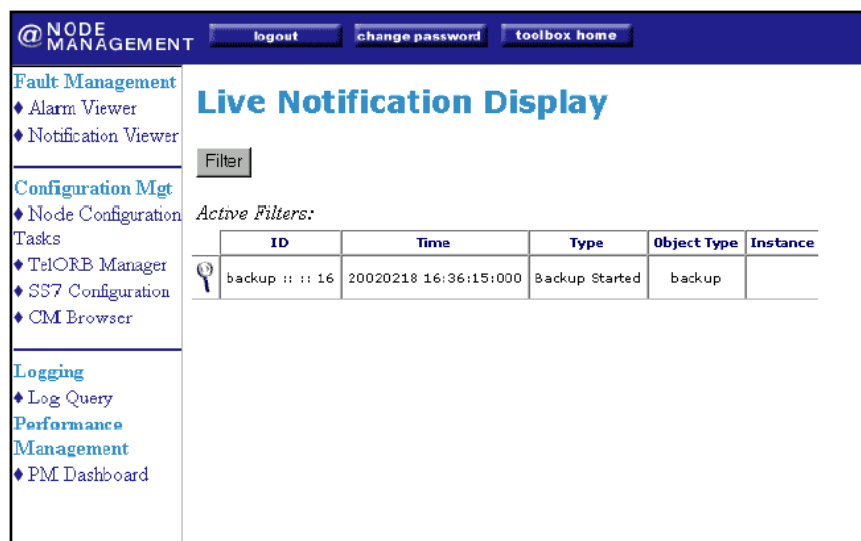


Figure 3-3 Live Notification Display

The alarms attributes can be filtered according to the following:

- Severity
- Managed Object Class
- Managed Object Instance
- Alarm Type
- Probable cause

The LND or Notification Viewer supports notification filtering to enable a user to examine a specific subset of the displayed notifications. The attributes that can be filtered are as follows:

- Managed Object Class
- Managed Object Instance

EXTERNAL INTERFACES

TSP Fault Management provides four different external interfaces with different protocols and characteristics. These interfaces interact with the TSP alarm list:

- The 3GPP Alarm Integration Reference Point (IRP) CORBA Solution Set
- The Ericsson IRP SNMPv2 Solution Set
- The Node Management Toolbox Alarm Viewer
- The Node Management Toolbox Notification Viewer

It also includes two other external Interfaces:

- Alarm Log interfaces
- User documentation accessible on the Alex Server

3GPP IRP for Alarms and Notifications

The 3GPP IRP Fault management CORBA Gateway is a 3GPP complaint alarm service. The 3GPP IRP interface is implemented over an open IT standard called CORBA. This allows simplified interoperability at a software level between clients and servers. The 3GPP IRP standard defines the operation that a client can perform on the active alarm list, extending in some cases to the content of those alarms.

The TSP implementation of the 3GPP IRP is called Event Channel Network (ECN), and requires clients to subscribe to its service. This TSP implementation also supports features, such as event filtering. Once a client is subscribed, the interface supports the basic operations as follows:

- Get the full list of active events
- Get the information for a specific event
- Acknowledge an event

In the case of TSP, the 3GPP IRP server resides on the TSP platform. The client can be an OSS machine, a web infrastructure, or even a local client running within the same or a distinct node.

SNMP Traps

Before 3GPP IRPs were defined for the fault management domain, the Simple Network Management Protocol (SNMP) standard emerged in the IT world. In the TSP platform, SNMP is used to send events to listeners through the User Datagram Protocol over IP (UDP/IP). In SNMP terminology, the listener is called the **SNMP Manager** when active operations are performed on the alarm list. In the case where its role is purely passive, it is called the **SNMP Viewer**. The server running on the TSP platform is called the SNMP agent and the event is called a "trap". The SNMPv2 plug-in translates collected events into SNMPv2 traps.

In SNMP, the data transmitted between managers and agents must fit the format specification of the MIBs. MIBs can be basically defined as a standard hierarchy of object classes and their attribute definitions. SNMP managers must have the MIB definition to be able to interpret the messages it receives from the agent.

For an SNMP manager to correctly see alarm data from the TSP platform, it must have the Ericsson alarm MIBs in place.

SUMMARY

- Fault Management is used to show if an alarm or event has occurred.
- Two default viewers are used which can be accessed using HTTP: Alarm Viewer or Live Alarm Display and Notification Viewer or Live Notification Display.
- Alarms and notifications can be filtered to attributes such as Managed Object Class and Managed Object Instance.
- External interfaces are provided which allows Fault Management to be implemented over open IT standards.

4 Configuration Management

CHAPTER OBJECTIVES

After this chapter the student will be able to:

- Describe the node configuration management functions on the TSP platform
- Describe provisioning, software management and hardware management
- Outline the concept of the Business Object Layer, and the Jambala Information Manager

INTRODUCTION

Configuration Management (CM) consists of functions to read and change parameters that define the behavior and characteristics of the deployed node. It also includes functions to manage subscriber related data stored in the node.

In this chapter, the configuration management support is divided into two domains:

- Node configuration management – this provides the functions needed for configuration of the platform hardware and software and functions for the configuration of the application software. The main user of this support is the operator's O&M organization.
- Provisioning provides the functions needed for configuration of the applications with data that is related to the subscriber supported by the node. A typical example is the customer data in the HLR. The main user of this support is the operator's customer support organization.

The CM area differs from the other areas in O&M in that it requires the application to do more than just using a simple API. Rather it requires the application to follow one out of several implementation models.

CONFIGURATION MANAGEMENT OVERVIEW

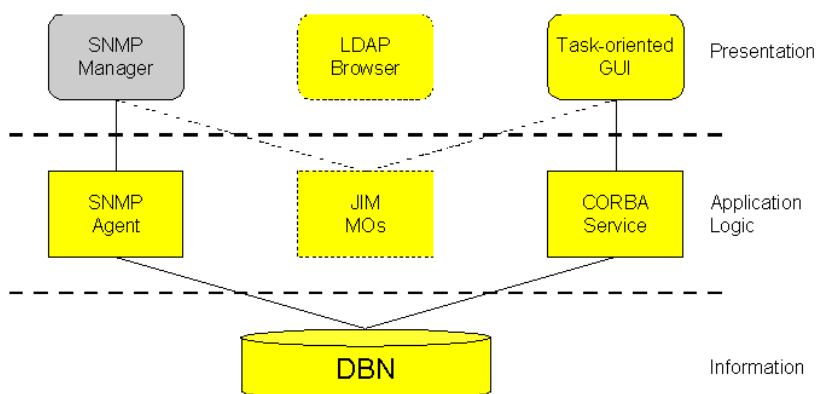


Figure 4-1 Alternatives for Node CM

Node Configuration Management provides the functions needed for configuration of the platform hardware and software and for the configuration of the application software.

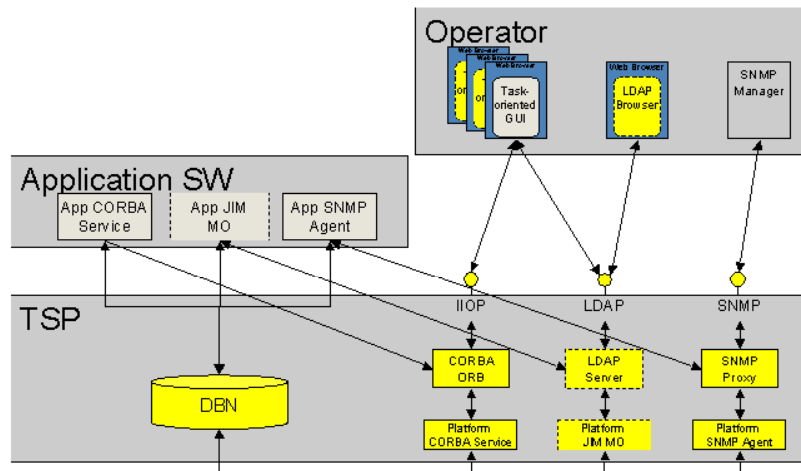


Figure 4-2 Overview of Node Configuration Management

The platform provides support for three ways of implementing node configuration management. Common for all approaches is that configuration parameters are stored as Persistent Object Types (POTs) in the TelORB database. External access to these parameters is then achieved through different mechanisms: by implementing an SNMP agent, by using the JIM framework or by implementing a CORBA service.

The basic support for configuration management is the TelORB database. Storing configuration data in the database ensures availability and consistency at upgrades and system reloads. If the database is not used, then it is up to the component to ensure consistency and rollbacks. An example of such a solution is where third party products, which typically stores its configuration in a file, is supported by a proxy that stores the configurable data in the database and that keeps the configuration and the database data synchronized.

Applications that chose to implement CORBA services as the CM interface are supported with an ORB and a naming service. In addition to the CORBA service, the application will also need to provide a GUI application, which can be integrated into the platform's embedded element manager, called NM Toolbox.

An SNMP proxy supports applications that chose to implement an SNMP agent. The platform does not provide an SNMP manager as part of the NM Toolbox.

Applications that chose to use the provisioning framework (JIM) will be supported by an LDAP server and a generic browser.

Three important sub areas of node configuration management are described in the subsequent sections. These are:

- software management,
- hardware management and,
- platform configuration

NODE CONFIGURATION MANAGEMENT

SOFTWARE MANAGEMENT

Software management is a set of functions that handles initial loading, initial configuration, software upgrades, and software inventory, as well as backups of the database. This functionality is provided by the distributed execution environment.

A software inventory is one part of software management. The inventory function includes all software of the cluster, i.e. all software executing on processors running Dicos, as well as on the Linux processors that are under cluster support (with the IO processors being the exception). Even third party products are included in the inventory. The interface to software management is the TelORB Manager that is part of the NM Toolbox.

Related to software management is the support for exporting the content of the database backup to support systems.

HARDWARE MANAGEMENT

Hardware management is a set of functions that handles installation, configuration and removal of hardware, as well as a hardware inventory. This functionality is provided by the distributed execution environment. A hardware inventory is provided by means of the TelORB manager. It is possible to list all installed hardware in the node in a GUI, and to generate an XML-formatted file with hardware inventory information that can be exported to other systems.

PLATFORM CONFIGURATION

The platform provides most of the configuration support to the operators by means of CORBA services and task-oriented GUIs. The support given by the TelORB manager of the OM Toolbox is implemented in this way. The TelORB manager is described in Chapter 2.

The provided SNMP MIBs are few and only covers a minor part of the platform configuration:

- Ericsson Apache MIB for configuration of the web server
- Ericsson Logging MIB for configuration of the logging support
- Ericsson Open LDAP MIB for configuration of the LDAP server

PROVISIONING

Provisioning provides the functions needed for configuration of the applications with data that is related to the subscribers supported by the node. The provisioning support is designed to provide scalability by utilizing the distribution mechanisms of the platform in order to match the performance requirements of the applications.

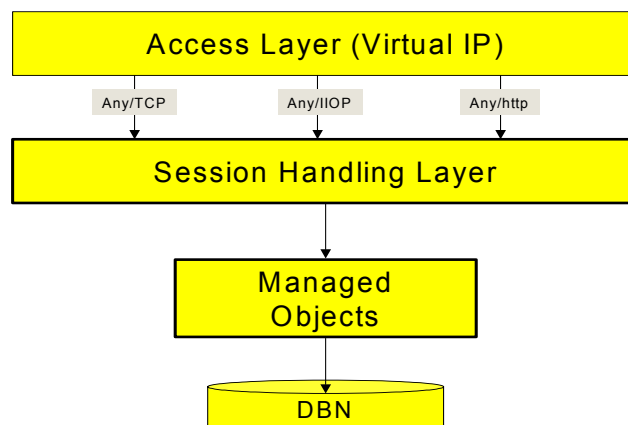


Figure 4-3 Overview of support for provisioning

The provisioning support can be seen as a layered structure consisting of:

- An access layer that provides scalable access for IP traffic. This layer enables having a single system view (single IP address) to the distributed system. The access layer is used also for other purposes than O&M.
- A session handling layer that provides authentication and management of sessions between clients and the server.
- Management of provisioning data (or configuration data) is provided as a managed object model. The application implements managed objects and the relations between the managed objects.
- Finally, the common point for persistent storage is the distributed TelORB database.

When comparing this figure to the figure on CM alternatives in the sections on Node Configuration Management, then the provisioning support can be viewed as one way, with additional support for implementing application logic.

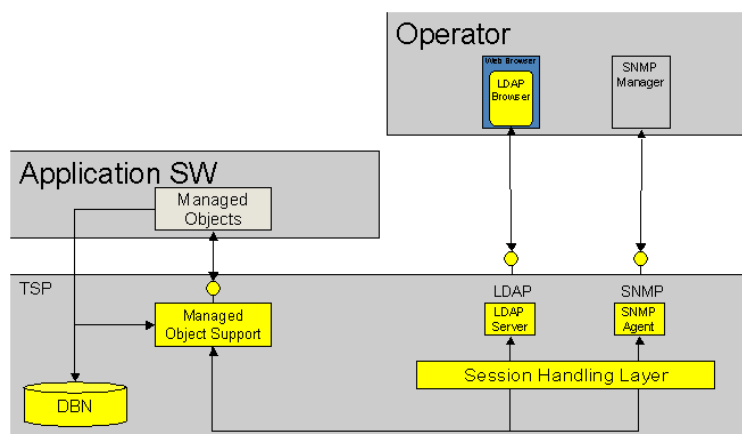


Figure 4-4 Run-time overview of provisioning

The provisioning support is a contract between the application and the platform. When the application implements their managed objects (e.g. subscribers) as POTs, according to the framework defined by JIM, then the platform will provide an external interface to these managed objects. The framework is called Jambala Information Manager or JIM for short. This is described in detail in the following pages.

The framework requires that the POTs defined as managed objects provide a set of attributes and methods that enables JIM to navigate the tree of managed objects. The application also needs to provide JIM with meta-data, the structure of the managed objects.

By using the managed objects approach the applications can rely on two application level protocols provided by the platform (SNMP and LDAP) and a generic browser. The platform provides an LDAP server as interface to JIM as well as an SNMP agent. There are certain restrictions that have to be applied if the managed objects are to be visible through the SNMP interface.

The platform provides an LDAP Browser as part of the ToolBox. JIM also takes care of authentication and authorization. The support for managed objects is limited to C++ implementations on Dicos.

The capacity and scalability of the provisioning framework is achieved by distributing the front-end LDAP server on several Linux processors while the back-end part of JIM is distributed on Dicos processors.

CONFIGURATION MANAGEMENT INTERFACES

The platform provides configuration support through task-oriented interfaces launched from the NM Toolbox. The four configuration interfaces launched from the toolbox's Configuration Management section are:

- CM Browser
- Node Configuration Tasks
- TelORB Manager
- Dynamic Configuration Manager (DCM)

In addition, TSP provides an SNMP v3 interface to parts of the platform. This interface presents the SNMP MIBs provided by the platform and applications.

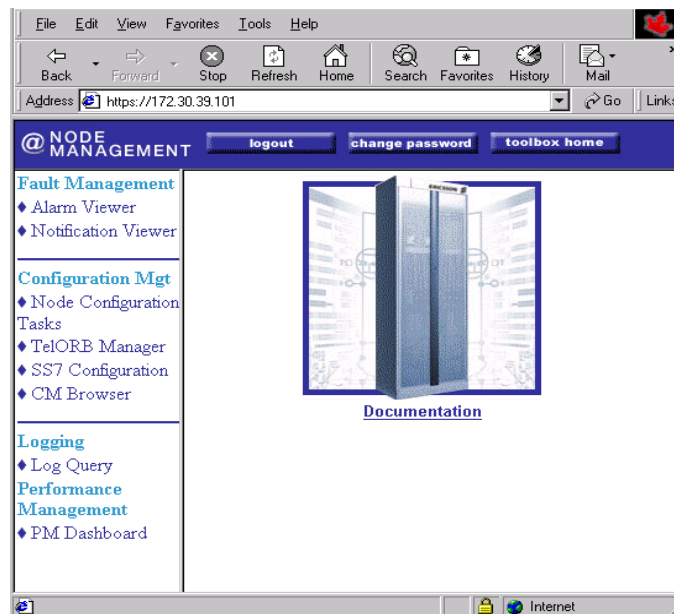


Figure 4-5 Configuration Management user interface

SUMMARY

- Configuration Management consists of functions to read and change parameters that define the behavior and characteristics of the TSP platform.
- Configuration Management supports Node Configuration Management (includes software, hardware and platform configuration) and Provisioning.
- Configuration interfaces are the CM Browser, Node Configuration tasks, TelORB manager and the Dynamic Configuration Manager (DCM).

5 Backup and Restoration

CHAPTER OBJECTIVES

After this chapter the student will be able to:

- Describe the principles of backup and restoration of the TSP Platform
- Backup and restore the TelORB Database
- Explain the IO Processor Backup and Restore procedures
- Describe the use of error dumps

INTRODUCTION

All persistent data is covered by different backup procedures. The data on the traffic processors is secured on disks, and selected contents on the disks is secured on a tape. The important contents of the IO processors and File Server processors is stored on tape to be able to restore all data when needed.

The Restore tasks are related to the backup tasks, and they are therefore described together, even though they are performed at completely different occasions.

When restoring, you will not be able to restore your system to exactly the same state it was in before the reason for the restore operation. If you are using your latest backup, all the data in that backup will be restored together with all present disk log data.

The software in the network element changes over time:

- Data will change as a result of operations by operators and applications
- System upgrades cause both the code and the data to change

The activities involved in performing backup and restoration described in this chapter:

- TelORB Database Backup and Restore
- TelORB Database Backup data on IO1 and IO2
- IO processor Backup and Restore
- File Server Backup and Restore

The description of the procedures for backup and restoration in this chapter are derived from the O&M documentation. You should refer to the most recent version of the documentation when performing these actions.

BACKUP AND RESTORE DOMAINS

The below figure contains three types of processors: traffic processors (TP), IO processors, and file server processors (FS). Some of the traffic processors can also be dedicated to handle SS7 traffic. IO1 and IO2 are used for TelORB specific data, and file server processors are ordinary Linux processors containing application data. There can be zero or more FSs configured in a system. See the file *telorb.conf* in */opt/telorb/etc* for processor specific details.

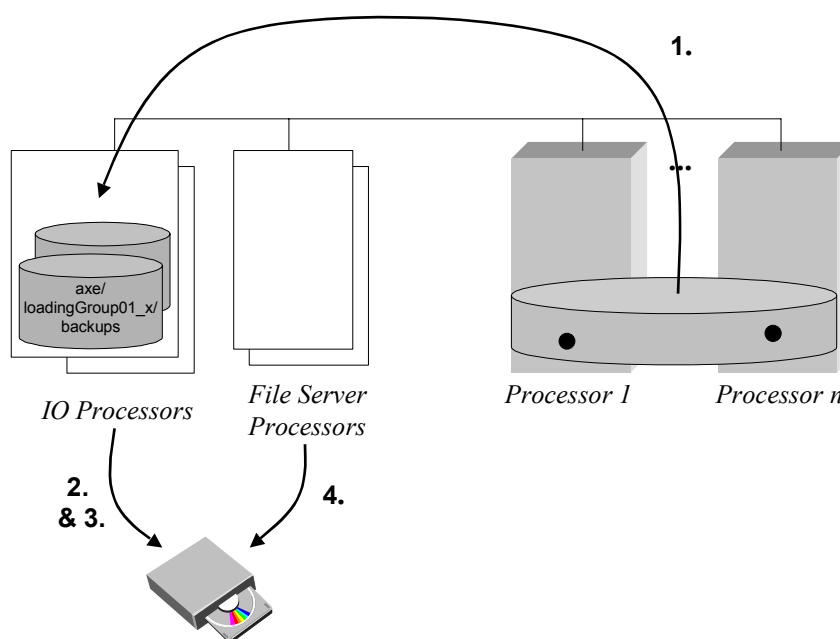


Figure 5-1 TSP Backup and Restore domains

1. Backup the TelORB Database to disk on IO1 and IO2

For IO1 and IO2:

2. Copy selected Database backups (from *opt/telorb/loadingGroup01-x/backups*) to tape
3. Copy the complete */home* and complete */opt/telorb* directories to tape.

For every File Server:

4. Copy the complete */data* directory to tape.

In normal operation, TelORB backups are simultaneously stored on disk on both IO1 and IO2. The tape drive and the CD-ROM are located on IO2. The hard disk is bigger than the capacity of a tape, therefore you may need up to two tapes to backup all relevant data.

The four domains for backup and restore are:

1. TelORB Database on traffic processors.
2. TelORB Database Backups area on IO1 and IO2 (i.e. the directory structure under `/opt/telorb/axe/loadingGroup01_x/backups/`)
3. Relevant data on each IO (i.e. the `/etc`, `/home`, and `/opt/telorb` file systems)
4. Relevant data on each FS (i.e. the `/data` file system).

TELORB DATABASE BACKUP AND RESTORE

Backing up TelORB means copying data from the TelORB processors to the disks attached to the IO processors.

The data to be backed up is:

- The TelORB database
- The `bootLoadTable.0` file for each TelORB processor

As long as there is one functioning IO processor, the data on the TelORB processors can be backed up.

The operator creates a backup object using the user interface. The object should have a meaningful name, so that it will be easier to distinguish when the backup was performed.

The operator receives a notification when the backup starts. After the backup, TelORB sends a notification indicating whether the backup was successful or not.

No code is included in the backup. The backup is stored under the backup directory of the IO processors. Several backups can be stored there, but only one is active. A backup can be initiated through the management interface or by the system itself. For example, before a software upgrade is executed, a backup is taken automatically. Backups are used in a zone reload.

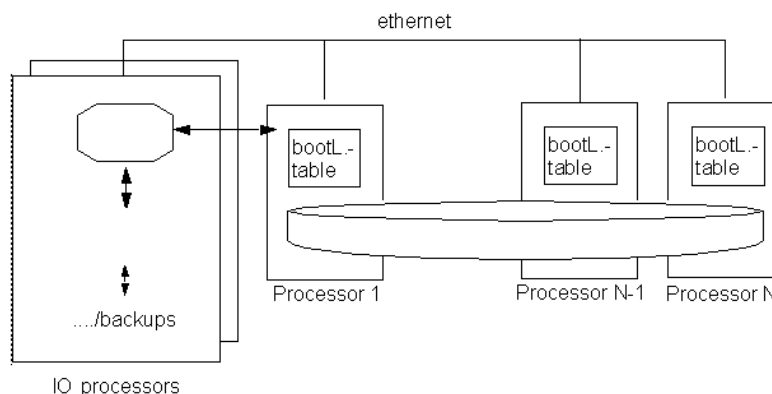


Figure 5-2 Backup procedure

When a backup has been performed, the backup files can be seen on the file structure of the IO processor:

`/opt/telorb/axe/loadingGroupxx_y/backups/Nameofyourbackup/`

The *Nameofyourbackup* directory will contain a directory for each processor present in the system. The `bootLoadTable.0` will be present in each processor directory.

THE TELORB DATABASE

The TelORB database backup consists of a large number of files/load modules. Each file contains a Master Data Partition (MDP) either as a whole, or as part of an MDP. The MDP is identified by a number calculated according to the formula:

$$\text{MDP} = \text{Classnr} \times 1024 + \text{DU}$$

The Classnr is the run-time (rtid) of the Persistent Object Type (POT). In most cases one MDP corresponds to one Distribution Unit (DU) in the database (the only exception is if you have two keyed POTs distributed by the same distribution unit type – these two POTs share the DU's but have their own separate MDPs).

The name of a backup file is in the format: 1m.997965.5.0.ddb where 997965 is the rtid/classnumber of the keyed POT; 5 is the DU individual; 0 is the just a sequential number to separate the parts of an MDP, if it is divided into several files.

The backup files will be found in the /opt/telorb/axe/loadingGroup01/backups/<backup_name>/database Backupdirectory. The MDP may also contain instances other POTs co-located with the keyed POT.

The “BackupFormatter” command can be used to format the database backup into a readable format.

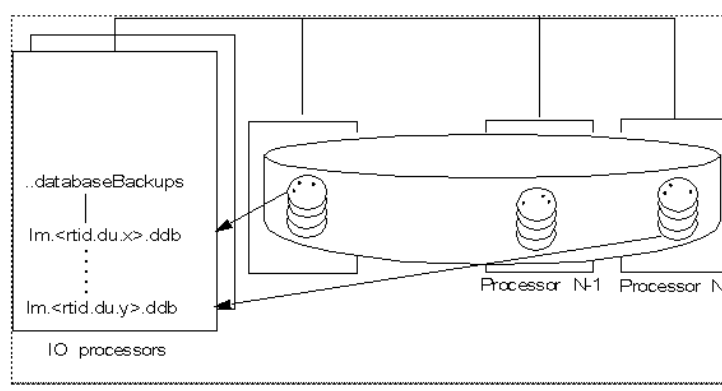


Figure 5-3 Database backup

BOOTLOAD TABLES

The bootload tables contain information about what load modules should be loaded on a particular processor. At initial start-up the bootload tables were produced by the *epct* command, but the initial configuration may have changed with new data being added to the system over time. Thus there is a need to backup the information regarding what is currently configured on each processor. The bootload tables are stored in the `axe/loadingGroup01/backups/<backup_name>/bootLoadTables` directory.

TELORB DATABASE RESTORE

To restore the traffic processors from the active backup, a **zone reload** needs to be performed. This should (ideally) never be ordered manually.

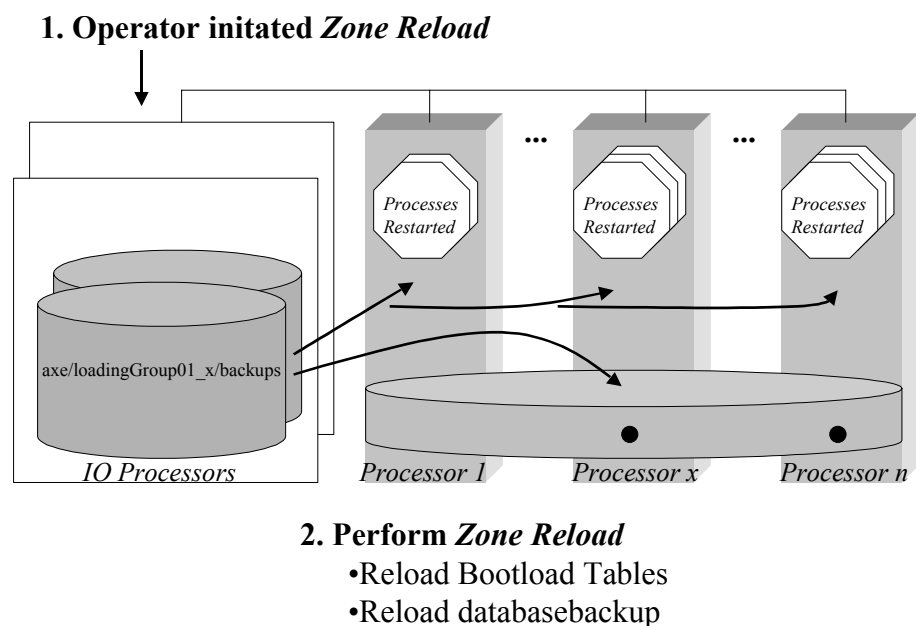


Figure 5-4 TelORB Database Restore

The active backup is restored to the TelORB processors by reloading the entire system.

Normally, the version that is being restored is the most recent, stable backup. However, the operator may wish to restore an older version; to do this the operator must first activate an older backup.

GENERATING BACKUPS

Backups can be initiated in two ways:

- Manual backups – A manual backup is directly ordered by the operator by performing an action on a managed object.
- Automatic backups – the operator generates a schedule of when backups should be made.

Backups can be initiated by operators or by other system functions such as system upgrade. There is a physical memory limit to the number of versions that can be stored on disk. Backup versions can be deleted, but should first have been copied to tape for long- term archive.

The operator restores software by loading the most recent, stable backup. If the operator wishes to load an earlier version, it is possible by activating an older backup and then ordering a zone reload.

Backup using the TelORB Manager

Within the TelORB Manager a tab called “Backups” as seen at the top of the following diagram, can be clicked. This screen is the area where the operator can control the backups. The window shows a list of all the backups currently available on the TelORB system to which the operator is connected.

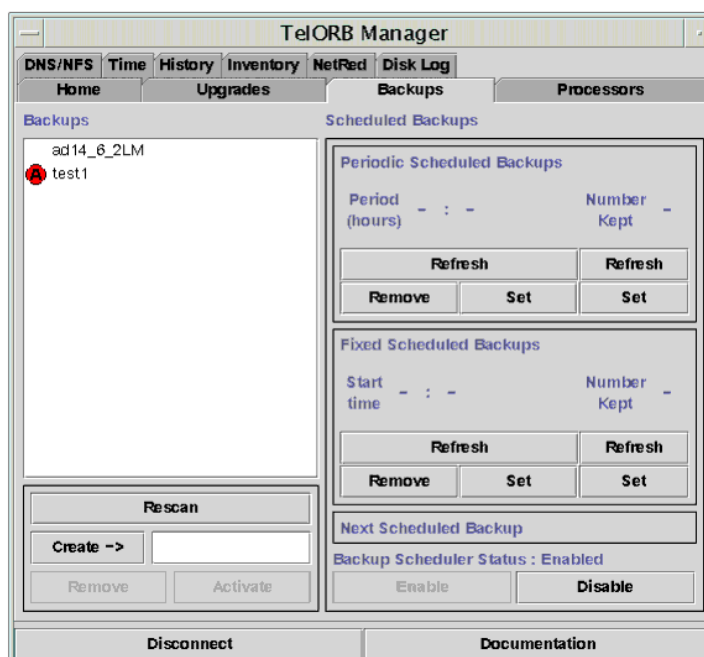


Figure 5-5 TelORB Manager Backup screen

One of these backups is marked with an “A” to show that it is the Active backup. A full description of backups can be found in the documentation icon.

The tasks that can be performed here are:

- Backup of TelORB
- Removal of TelORB backup

TelORB Manager Buttons

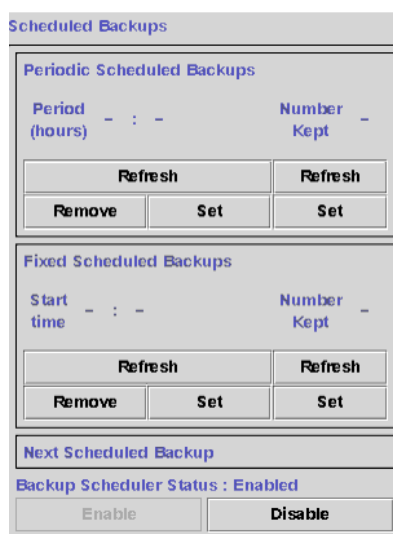
The following buttons under the Backup tab area have the following effect:

- Rescan – rescans the IO machines to see whether any additional backups can be found.
- Create – creates a new backup, the name of which is defined in the space next to the button.
- Remove – removes whatever backup that has been selected from the IO processors, except the Active one.

- Activate – makes the selected backup become the active backup.

Scheduled Backups

On the right of the backup screen, is the area for the scheduled backups as seen in the figure below.



The screenshot shows a window titled "Scheduled Backups". It contains three main sections:

- Periodic Scheduled Backups:** Includes input fields for "Period (hours)" and "Number Kept", each with a "Refresh" button. Below these are "Remove" and "Set" buttons.
- Fixed Scheduled Backups:** Includes input fields for "Start time" and "Number Kept", each with a "Refresh" button. Below these are "Remove" and "Set" buttons.
- Next Scheduled Backup:** A single input field.

At the bottom, it shows "Backup Scheduler Status : Enabled" with "Enable" and "Disable" buttons.

Figure 5-6 Scheduled backups screen

If the system is working with network redundancy, a scheduled backup also implies that a remote synchronization will be carried out at the same time, as activating network redundancy actually activates periodic backups. It is therefore possible to use the scheduled backup service to ensure that the two zones are regularly in-synch.

Scheduled backups can be of different types:

- Periodic Scheduled Backups – are given an interval and occur at that time and at the specified period (in hours) after that time.
- Fixed Scheduled Backups – are taken at the specified time each day.

Both types of scheduled backups are automatically named with the data and time they were created. As soon as a backup is completed successfully, it automatically becomes the active backup.

To prevent fixed and periodic backups coinciding, TelORB checks to see whether or not there is a fixed backup within 1 hour of the scheduled backup. If so, the scheduled backup will not be taken. The fixed backup will be used in its place. The following scheduled backup will be taken as normal.

The screen also displays the “Next scheduled backup”, irrespective of whether it is fixed or periodic. If the settings are altered for either the fixed or the periodic backups, such that a different backup becomes the “next” backup, the value will not change until “Refresh” is clicked.

TELORB DATABASE BACKUP DATA ON IO1 AND IO2

Both the backup procedure and the restore procedure are carried out by running the script *telorbBackups.ksh*.

The backup option in the script secures all selected TelORB database backups located in
/opt/telorb/axe/loadingGroup01_x/backups/ to tape.

The restore procedure is carried out by running exactly the same script as in the backup case. You just choose the restore option instead of the backup option. The script reads information about available backups on the tape, then you select one or more individual backups to be restored from the tape.

IO PROCESSOR BACKUP AND RESTORE

There is a limit to the amount of disk space on the IO processors and so for this reason it is necessary to copy backups from the IO processor disks on the external storage media such as a tape. In order to do this, the operator uses Linux commands to interact with the IO processor.

This backup procedure must be performed for a specific TelORB backup stored on an IO processor. The backup procedure takes a backup of the */etc*, */home*, and */opt/telorb* directory structures. It must be performed individually for each IO processor. Beware that, depending on the hardware, you may need two tapes. NSP4.0 has a 20 GB tape streamer, NSP4.1 has an 80 GB tape streamer.

For backup, the Linux `tar` command is used. The `tar` command archives and extracts files to/from a single file called a tar file. A tar file is a file of compressed data, usually stored on a magnetic tape.

It should be noted that you should refer to the CPI documentation *“IO Processor Backup and Restore”* for the specific details of these procedures.

Prerequisites

In order to backup an IO processor, the operator needs to:

- Know the names of the processors in the current configuration
- Have root user privileges
- Know the drive number for the tape device. The `tar` command needs to know the drive number. For example, in `/dev/rmt/0/` the drive number is zero. The drive number is a figure between 0 and 7. It is possible to have multiple drives connected.

The procedure is as follows:

1. Remotely login to the IO 2 as telorb user:

```
#ssh telorb@ <hostname> -l telorb
```

2. Change directory to `/opt/telorb`:

```
io2:telorb>cd /opt/telorb
```

3. List the contents of the tape:

```
io2:telorb>tar tvf /dev/rmt/0
```

4. Rewind the tape:

```
io2:telorb>mt rewind
```

5. Create a backup on tape by entering the following:

```
io2:telorb>tar cvf /dev/rmt/0  
axe/loadingGroup01_2/backups/Backup1_ddmmy  
yyy
```

NOTE: The following directories get sent to the backup tar file on the tape unit:

```
axe/loadingGroup01_2/backups/Backup1_ddmmyyyy/databaseBackup/  
axe/loadingGroup01_2/backups/Backup1_ddmmyyyy/Processor1  
axe/loadingGroup01_2/backups/Backup1_ddmmyyyy/Processor2  
axe/loadingGroup01_2/backups/Backup1_ddmmyyyy/Processor3  
axe/loadingGroup01_2/backups/Backup1_ddmmyyyy/Processor4  
axe/loadingGroup01_2/backups/Backup1_ddmmyyyy/Processor5  
axe/loadingGroup01_2/backups/Backup1_ddmmyyyy/Processor6  
axe/loadingGroup01_2/backups/Backup1_ddmmyyyy/Processor7
```

6. Verify that the backup to tape was successful:

```
io2:telorb>mt rewind
```

```
io2:telorb>tar tvf /dev/rmt/0
```

NOTE: The previously listed directories are contained in the backup tar file on the tape unit.

7. Remove the tape from the tape drive and write the date and name of the backup file, used on the tape label, then make sure that the tape is write-protected.

RESTORE IO PROCESSOR

Restoring the IO processor involves copying data from the tape back to the IO processors. IO processor 2 is connected to the tape drive, whereas 1 is not.

1. Remotely login to the IO 2 as telorb user:

```
#ssh telorb@ <hostname> -l telorb
```

2. Change directory to /opt/telorb:

```
io2:telorb>cd /opt/telorb
```

3. Rewind the tape:

```
io2:telorb>mt rewind
```

4. Restore the backup from the tape:

```
io2:telorb>cd axe/loadingGroup01_2/backups
```

```
io2:telorb>mv Backup1_ddmmyyyy  
Backup1_ddmmyyyy.old
```

```
io2:telorb>cd /opt/telorb
```

```
io2:telorb>tar xvpf /dev/rmt/0
```

```
io2:telorb>ls axe/loadingGroup01_2/backups
```

NOTE: The directory Backup1_ddmmyyyy has been restored.

5. Create a tar file containing the contents of loadingGroup01_2:

```
io2:telorb>cd axe
```

```
io2:telorb>tar cvf loadingGroup01_2.tar  
loadingGroup01_2
```

6. FTP the tar file to IO 1

```
io2:telorb>ftp <io1>
```

```
ftp> cd /opt/telorb/axe
```

```
ftp> bin
```

```
ftp> hash
```

```
ftp> put loadingGroup01_2.tar
```

```
ftp> quit
```

```
io2:telorb>rm loadingGroup01_2.tar
```

7. Remotely login to the IO 1 as telorb user:

```
#ssh telorb@ <hostname> -l telorb
```

8. Change directory to /opt/telorb:

```
io1:telorb>cd /opt/telorb/axe
```

9. Extract the tar files:

```
io1:telorb>tar -xvf loadingGroup01_2.tar
```

10. Change the directory name:

```
io1:telorb>mv loadingGroup01_1  
loadingGroup01_1.old
```

```
io1:telorb>mv loadingGroup01_2  
loadingGroup01_1
```

11. Remove the tar file:

```
io1:telorb>rm -rf loadingGroup01_1.old
```

```
io1:telorb>rm loadingGroup01_2.tar
```

```
io1:telorb>exit
```

FILE SERVER BACKUP AND RESTORE

The reasoning for the IO processors is also valid for FS processors. For FS processors, it is the */data* file structure that is copied to the tape. Here you may also need two tapes.

All File Server processors use both of the IOs as Kickstart servers. See the latest CPI documentation "*File Server Processor Backup and Restore*" for specific details of this procedure.

BACKUP RECOMMENDATIONS

Backup interval requirements vary between different domains and specific customer requirements. In general, the backups can be divided into three different time scopes.

- **Daily backups.** This includes taking backups of the TelORB Traffic Processors to disks on IO processors, either manually or using scheduled backups. This is most likely done several times a day using scheduled backups.
- **Weekly backups.** TelORB Backups stored on IO processors should be backed up to tape fairly often due to frequent changes on the disks. This also applies to the FS processors if the changes are fairly frequent.
- **Monthly backups.** These backups contain files (Linux configuration files, TelORB configuration files etc.) that are rarely (or never) changed. In some cases the "monthly backup" of both Traffic processors and File Server processors can be taken when needed, and it does not have to be dependent on a scheduled schema.

BACKUP FOR NETWORK REDUNDANCY

The Network Redundancy solution adopted in TelORB is based on having two nodes where one is a primary node and the other is a stand-by node. Any transactions that are committed in the primary node are propagated to the stand-by node with TCP/IP. This is done by configuring Persistent Object Types (POTs) as *netshared*.

This means that the primary node ever suffers a complete shutdown, due to circumstances such as total power failure, the stand-by node should take over within approximately one second.

When the broken node recovers, it becomes the stand-by for the other node, which is now primary.

To increase capacity and performance, and decrease latency, the Standby node is not updated synchronously with the Primary node: updates are sent unchecked to the Standby. It is therefore possible that updates get lost in certain error situations, and inconsistencies between the two nodes can thus arise.

To compensate for this, the backups from the Primary node are sent over to the Standby node.

The backup is used to reconcile the databases such that the stand-by node is once again a true copy of the primary node. The stand-by node, of course, takes its own backups as part of normal procedure. These backups are not sent to the primary node. Operation moves to the stand-by node when under the control of the application. The entire redundancy cannot be under the control of the TelORB because the traffic needs to be re-routed as well as the program execution, and the traffic handled only by the application.

NOTIFICATIONS

Backup Transfer - backups are transferred from the primary zone to the stand-by zone in a system with network redundancy to synchronize the database.

Transfer Started - there is almost no degradation in system performance as a result of a backup transfer starting, because it runs on the IO processors. There will be a slight increase in the load on the communications link between the two nodes.

Depending on the application and the amount of data in the database, a backup transfer can take up to two hours. There is not mechanism for checking the progress of a transfer.

Transfer Succeeded – this is an indication that the transfer of the backup has ended successfully. When this notification is received an external sync is expected to take place soon in the stand-by zone. The value of “soon” cannot be defined.

Transfer Timeout - if the backup transfer has exceeded the time limit set (by default, 2 hours) then this notification is generated. Any reasonable sized backup should have transferred within this time, so the notification indicates that you should look at the backup being transferred to ensure that there is nothing wrong with it, or with the link between the zones.

Transfer Failed - the most likely cause of a failure is a problem with the link between the zones, in which case you will most likely get a “Connection Lost to Peer Zone” alarm.

Alternatively there may, in very rare circumstances, have been a failure of the TelORB Zone of which the transfer manager is unaware. If this happens then there will have been a TelORB System Reload Alarm generated.

Transfer Ignored - this notification is sent when the transferred backup will not be used because, for example, the backup was not properly generated.

Alternatively there may be nothing wrong with the backup but there has been a communication failure between TelORB and the Transfer Manager so that the Transfer Manager cannot be informed that the backup generation completed.

ERROR DUMPS

There are various levels of errors that can occur and how the system recovers from these problems:

- A transmission problem, which occurs between two processes. This is handled by the application software and is not visible from an operation and maintenance perspective.
- Process restarts. Static processes are automatically restarted; dynamic processes are not. Processes can use the **Forlopp** mechanism to inform processes. The recovery mechanism is automatic and an error dump is generated which can be retrieved by operation and maintenance staff.
- Processor reload. If an entire processor fails, the processor reloads.

In an executing TelORB system, error dumps may be generated by:

- A processor crash
- A process crash (also called Capsule Abortion)

In both cases, an error dump is generated and stored on the IO processor. A processor crash generates a notification and an error dump. Every 5 minutes the *crashcollector* process checks the directory `/opt/telorb/dumps` for the error dumps and then packs the error dumps together with the *consolelogs*, the *syslog* and other information stored under the directory `/opt/telorb/crashcollector` in a gzipped tar file, called 'crashcollector<datetime>.tar.Z'. It is also possible to check for error dumps under `/opt/telorb/dumps`. It is recommended to check for new dumps every time a new backup of the TelORB system is created.

The error dumps are stored in the directory:

`/opt/telorb/dumps/`

The error dumps are names:

`DicosTextDump-n` for process dumps

`DicosProcessorTextDump-n` for processor dumps

The letter n is a number that is increased for every new error dump. The default value for the maximum number of error dumps is 1000. The maximum number exists because otherwise there is a risk that a faulty process may generate dumps all the time using up all the existing disk space.

An error dump is sometimes referred to as a capsule abortion. The dump contains call and register values that can be used to analyze the cause of the fault.

Check for New Dumps

The output in the directory can be sorted by issuing the command:

```
.>>ls -lrt
```

where ls lists the contents of the directory. The -l switch gives the listing in long format (including date and time). The -t switch sorts the output by time stamp and the -r switch reverses the output giving the newest files at the bottom of the listing.

SUMMARY

- Backup and restoration procedures allows system data to be recorded at one time and then restored at a later time. The procedures involve operator interactions using the GUI and using Linux commands.
- TelORB Database backups copy the following onto the disks on the IO processors:
 - The TelORB database
 - The bootLoadTable.0 file for each TelORB processor
- TelORB Database backups can be perform manually or automatically.
- IO Processor Backup procedure takes a backup of the **/etc**, **/home** and **/opt/telorb** directory structures.
- Error dumps are generated by the TelORB system at a *Processor* or *Process* crash. Operators may send copies of error dumps when reporting faults but do not need to interpret these dumps.

6 System Upgrade

CHAPTER OBJECTIVES

After completing this chapter you will be able to:

- Explain the Load and Start functions on the TSP Platform
- Describe the various types of system upgrade that can be performed on the TSP Platform.
- Create and perform a system upgrade on the TSP simulated environment.

INTRODUCTION

A system upgrade is the process by which an operator replaces software with new and/or improved software. This function can be performed on a live system. Many of the principles concerned with a system upgrade (in particular to do with the file structures) are also true of initial start.

This module describes the system actions during the TSP initial load and start, a TSP system upgrade and explains the management of the system upgrade function via the user management interface.

TSP INITIAL LOAD AND START

The TSP Platform load and start functions offer services for loading and starting software when a network element or processors within a network element are initially started or restarted, and when there is a software upgrade.

All of the processors on the TSP must be brought to a state where the telecommunications applications are capable of running. The load and start functions are defined as follows:

Load - the moving of code and data into processor memory.

Start - the beginning of execution of software on processors.

Several factors determine the approach to load and start which has been adopted:

- Flexibility - the mechanism should be capable of coping with a variety of different processors, storage devices, programs and data.
- Adaptability - a similar approach should be applicable to the various occasions when it is necessary to load and start the entire system or part of it.
- Speed - the issue of speed is of particular importance when reloading the system in the event of a major failure. Time spent reloading is time when the system is not able to offer telecommunications services.

In order to achieve these objectives, programs and data must be loaded onto the processors and processes must be started in a controlled way. TelORB is designed to ensure that this occurs by dividing load and start functions into defined phases.

When Load and Start occur

There are several different circumstances when the TSP load and start functions are called into operation:

- **Initial Load and Start** of the all of the processors. In this case, the system will be taken from a state where the processors are powered off and do not contain software to a state where the system is operational and applications are running.
- **Zone Reload.** In this case, all the processors in the system are reloaded with code and data and then restarted. This may be done when, for example, the TSP error-handling mechanisms decide that previous measures to recover from an error were inadequate
- **Processor Restart.** In this case, a single processor in the system is reloaded and restarted.
- **Adding a Processor** to the system
- **Process Restart.** This is done when a static process has been aborted
- **System Upgrade and Reconfiguration.** System Upgrade is the process of adding and replacing software to a network element that is already in operation. Load and Start functions support system upgrade in moving software onto processors and controlling the start of processes.

Reloads and restarts are often done as a consequence of errors, but can also be the result of system upgrades or reconfiguration (where the software is redistributed between the processors in the system).

These actions may be either *Manually* initiated, i.e. by the operator or *Automatically* initiated by the TSP Platform functions.

TSP INITIAL LOAD AND START PROCEDURE

The TSP is designed to provide mechanisms for loading and starting a system irrespective of the type of processor hardware involved.

In order to provide such a general mechanism, the equipment and software for loading software and for running the loaded software are categorized as follows:

- IO Processors - contain equipment that stores and manages the software that is to be loaded. The IOs *contain File storage devices* and *File Servers* to provide this function.
- Traffic Processors – run the loaded software.

IO Processors

Before the software can be loaded onto the TSP traffic processors, the IO processors must be configured and running. The IO processors provide the following equipment:

- File storage devices

The IO file system must contain the required software for both TelORB and the telecommunications application. Both programs and data must be stored so that they can be loaded into the memory associated with traffic processors in the system. The file storage devices are the disks belonging to IO processors on the TSP.

- File server

The IO processors act as file servers for the TSP Platform. The tape streamer is used for loading the required software on to the file system and for archiving backups on tape. The file server software manages the files on the file storage devices.

Note: It is also possible to have disk-equipped processor boards (PMD-ETH) available to function uniquely as File Servers within the TSP configuration.

Contained in the IO processor Network File System (NFS) is the *load server*, which is responsible for sending the boot load table to the TelORB processors at initial start.

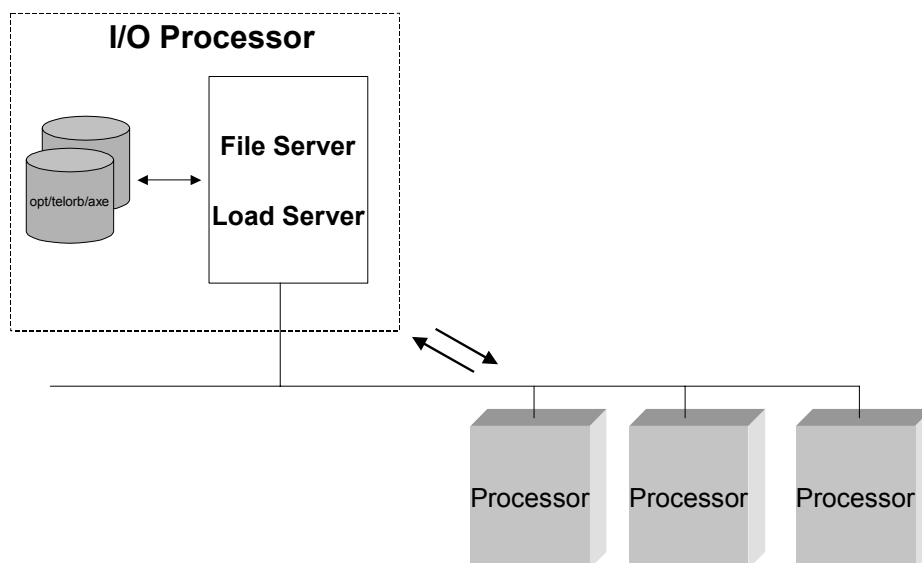


Figure 6-1 IO Processor and TelORB Load phases

The most important files on the IO processor as seen from the operator's point of view are the files under the directory */opt/telorb/axe*. This is called the TelORB file system.

Traffic Processors

The Traffic Processors run the loaded software. These can be described as:

- Loading processors

The Dicos processors load themselves with software and manage the loading of other Dicos processors defined as part of a Processor Loading Group. This is explained below.

- Other processors

In TSP 4 the Linux processors run software under the control of TelORB as an operating system. Linux processors load in an independent manner.

Processor Loading Group (PLG)

Dicos processors which are to be loaded are grouped together to form a Processor Loading Group. A processor can only belong to one PLG.

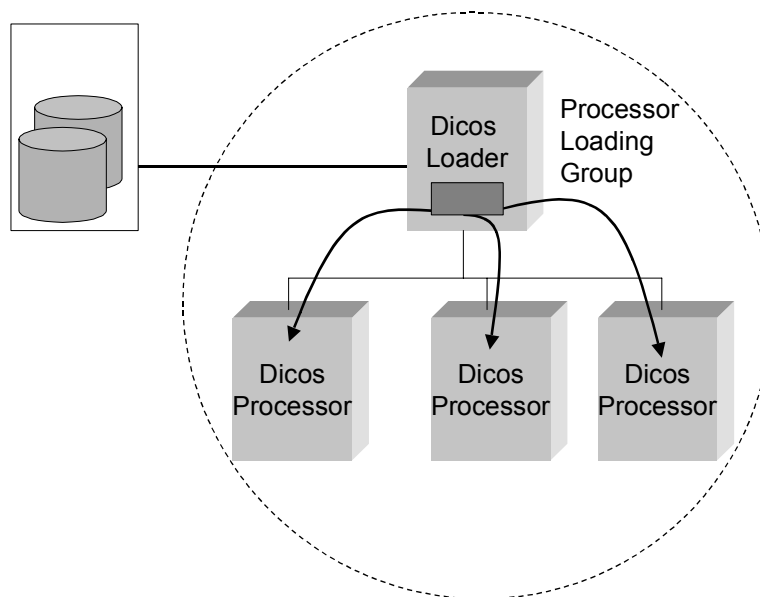


Figure 6-2 Processor Loading Group

PLG Loader

Within a PLG, two processors are designated as the PLG loaders. The loader both loads itself and all of the other processors in the group. During a load and start sequence, the one, of the two loading processors, that loads itself first begins the next phase, which is loading the remaining processors in the group.

The loader fetches loading information from the file storage device and distributes this information to the other processors in the loading group. Within a PLG it must be possible to establish communication connections between the loading processor and the processors which are to be loaded. The loader can load other processors in the PLG simultaneously using a point to multi-point connection.

TSP Initial Load and Start Sequence

TelORB load and start proceeds through a series of phases. The activities of one phase must be completed before the next phase can start. This means that TelORB itself must be fully working (with its distributed operating system, distributed database and inter-process communication) before applications software can be loaded and started. These phases must be coordinated between the processors to ensure that all processors are in step. The phases are identified under the following levels of functionality:

- PROM functionality/ OS loader
- Initial functionality
- Base functionality
- Total functionality

The TSP load and start sequence is described here:

1. Start IO Processors

- IO processors boot themselves from disk and start needed services such as DHCP¹, PXE², TFTP³ and NFS⁴ daemons.
- Any file servers on the IOs boot themselves from disk.
- All other processors start asking for information about their IP addresses over DHCP. They also start a PXE client.

2. Load and Start Traffic Processors

When at least one IO Processor has finished booting, all other processors can obtain their IP addresses via DHCP and start booting with the help of PXE and TFTP. The IP address determines whether to boot Dicos and Linux, according to configuration files on the IO processors.

- Dicos processors load and start a “PROM/OS loader” image.

¹ DHCP – Dynamic Host Configuration Protocol

² PXE – Pre-Boot Execution Environment

³ TFTP – Trivial File Transfer Protocol

⁴ NFS – Network File System

- Linux processors load and start a Linux kernel.

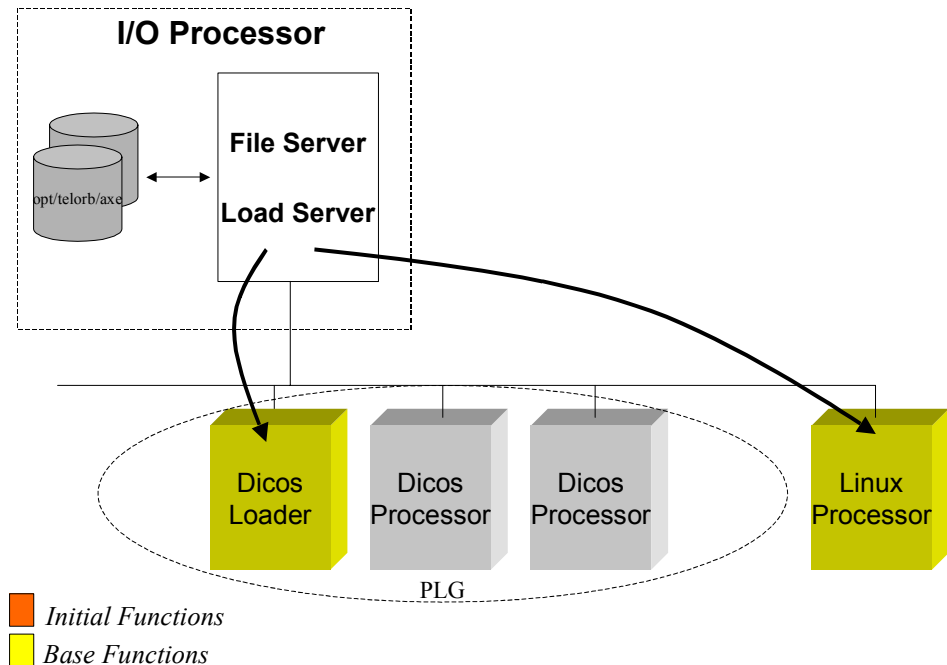


Figure 6-3 Initial & Base Functionality on the Loader

3. Initial and Base Functionality on Traffic Processors

- Dicos “loader” processors load and start a Dicos kernel and some drivers, called *Initial Functionality*, and continue to load and start *Base Functionality*.
- Linux processors load and unpack a RAM disk image containing *Base Functionality*.

Initial Functionality During this phase the loading processor acquires a boot load table from the file server on the IO, while all the other processors have to wait for their tables to be distributed by the loading processor. At this stage, the TelORB kernel has been loaded. The TelORB kernel starts a process that starts the next phase.

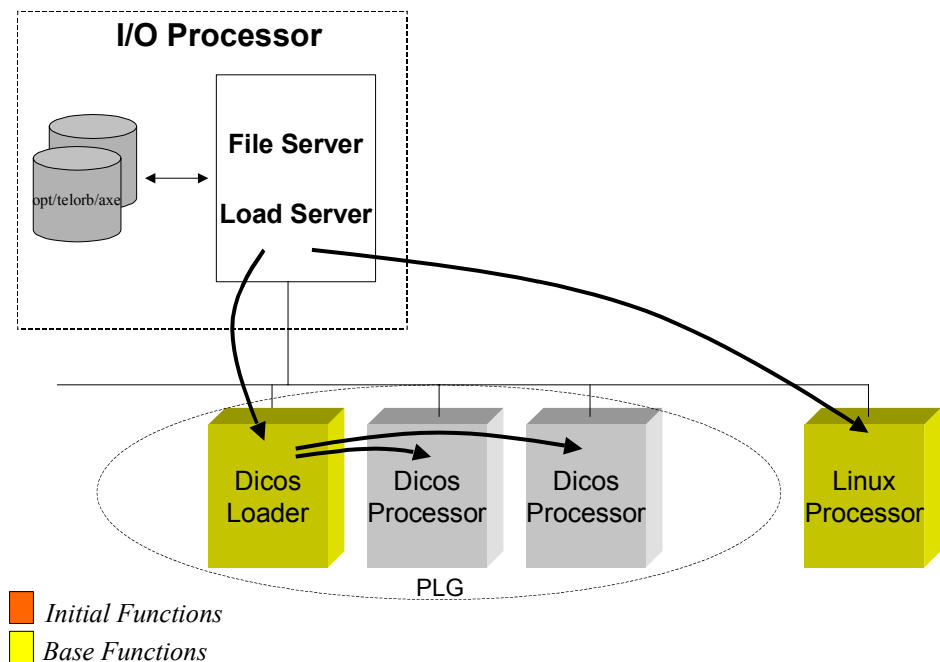


Figure 6-4 Multicast loading of Dicos processors

- One of the Dicos “loader” processors start to load the remaining Dicos processors with load modules, fetched from the bootstrap processors via NFS, over a proprietary multi-cast protocol. This concludes initial and base functionality.

Base Functionality TelORB functions are started locally within one processor. Among the functions running when base functionality has been established are:

- Supervision of static processes
- Local database
- Support for the installation of configuration data
- T-IPC communication within the processor.

The loading processor reads configuration information from the local database and decides what to load on each processor. A boot load table is created for each processor and then distributed. When this distribution is complete, the loading processor distributes the load modules of initial and base functionality to the rest of the processors using multicast (one load module to many receivers at a time).

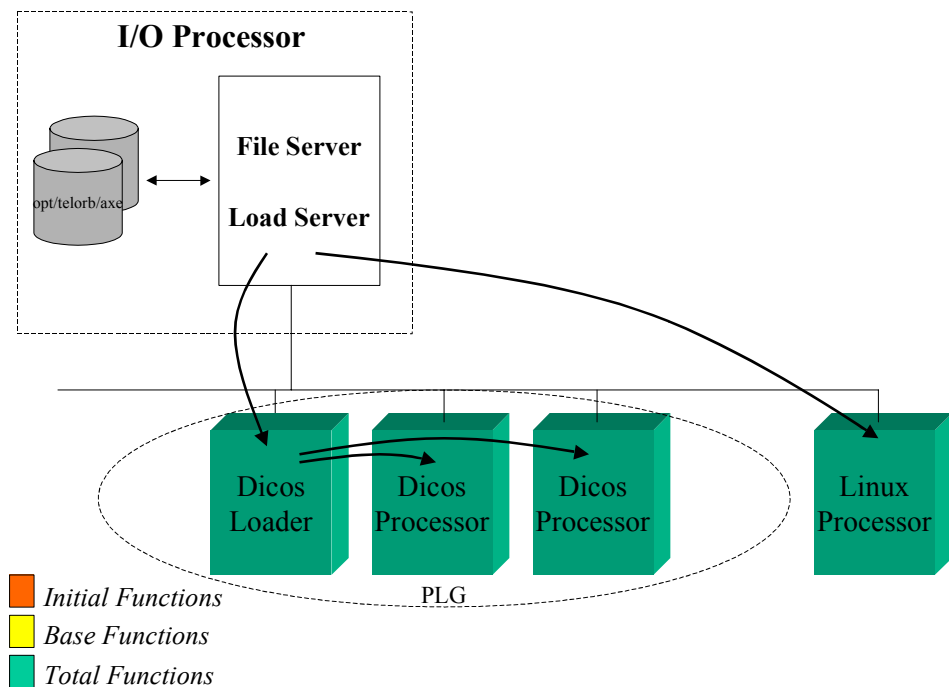


Figure 6-5 Total Functionality

4. Total Functionality on the Traffic Processors

- Dicos “loader” processors load themselves and the Dicos “non-loaders” with applications, *total functionality*, over the multi-cast protocol.
- Linux processors load total functionality fetched by themselves over NFS.

Total Functionality. When the base functions have been started on all the processors (synchronization point), both the distributed operation system (including IPC) and the distributed database (TelORB database) are available. The support for loading the telecommunication applications is now available.

A configuration file specifies which Load Modules are loaded in each phase.

The load module distributor continues to load all the processors, including the loading processor, with the load modules of total functionality. During this step, the remaining parts of the applications are loaded.

When the functions of total functionality have been loaded, the system is started. In other words, the TelORB Operating System and the application software is running.

SYSTEM UPGRADE

When introducing new software, patches, bug fixes and so on, this new software is loaded into a TSP platform by the system upgrade procedure.

There are several parts in a TSP that may be upgraded.

- TelORB code. New functions, enhanced functions, erroneous code that needs to be fixed.
- Application code. Application written on TelORB may suffer from the same errors as TelORB.
- IO processor software. This software runs on Linux, and cooperates with a TelORB system in cases like initial start, backup, upgrade, error dumps and so on.

The requirements on system upgrade are that:

- Upgrading the system should disrupt the running system as little as possible
- Generalized mechanisms for performing an upgrade should mean that an operator does not have to be aware of what software is being upgraded.

TelORB supports mechanisms for incremental system upgrade, and minimal disruption of ongoing services when taking the upgraded software into service.

The new software often performs similar functions to that implemented by software currently running in the system. System upgrade therefore can be a replacement, removal or an addition of software.

System Upgrade Methods

There are two methods of upgrading the TSP system:

Smooth upgrade

Redundancy upgrade

Reboot upgrade

Normally applications software should be upgraded using smooth upgrade.

The type of upgrade to be applied is specified in a configuration file delivered with upgrade package.

Delivery and Installation

System upgrade consists of delivery and installation phases.

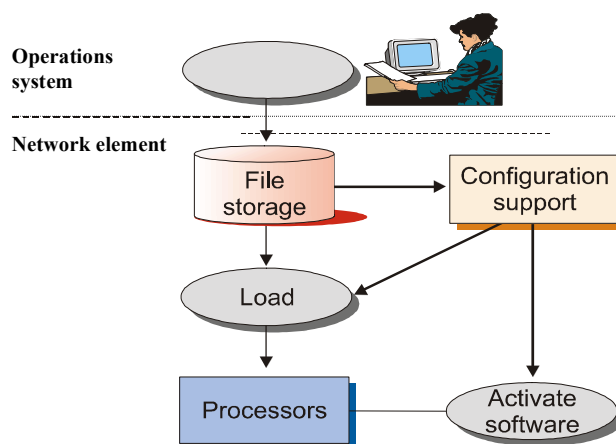


Figure 6-6 System Upgrade delivery and installation

Delivery

When software is being upgraded, code that contains corrections to repair faulty code, or code to implement new functions and configuration data for these new functions is generated.

Both the new code and the new configuration data is then sent to the file system as an upgrade package. This package may be copied from a tape into the file system or sent over data links to the file system. The load modules in this package also contain check files that are used in controlling the upgrade.

Installation

Installation is the procedure where upgrade load modules are loaded from the file system and into the memory of the processors that are being upgraded.

The system contains a managed object called Upgrade. The operator initiates an upgrade by performing a “start” action on the Upgrade MO in the graphical user interface.

Smooth Upgrade

With the smooth method, new versions of the software is loaded onto already running processors, and affected processes get to migrate over to the new versions. If the upgrade includes changes of persistent object types, commonly referred to as *schema upgrades*, all database objects of the changed types are migrated to the new schema before the processes are migrated. A smooth upgrade can include adding new process types and persistent object types, as well as removing process types and persistent object types.

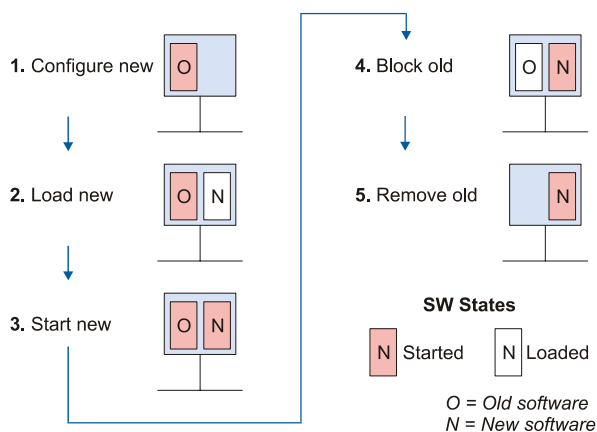


Figure 6-7 Smooth Upgrade

The steps followed on each affected processor during a smooth upgrade are:

1. The old software is running and the new software is configured off-line.
2. The new software is loaded. The old software continues executing.
3. The new software is started. The old software continues to handle existing activities. The new software handles any new activities.
4. As the number of activities using the old software diminishes, the old software is blocked.
5. The old software is removed. Only the new software is running.

The old software is only removed from service when all users of the service have ended their use of the service.

The smooth upgrade mechanism is ideal because external users of the system experience no disruption to the service. Smooth upgrade can be used for loading new versions of applications software. A disadvantage of this method is that it requires more processor memory, since both the old and the new software are loaded and executing at the same time.

State transfer

The new software may perform similar functions to that performed by the old software. Application designers can take this into account by having similar states and data that are understood by both the old and the new software. In this way, the control of a call is passed from the old software to the new software. The user of the service experiences no disruption and the old software can be removed more quickly, since no user is depending on the old software.

It is used for smooth upgrade of processes and is designed into the application itself.

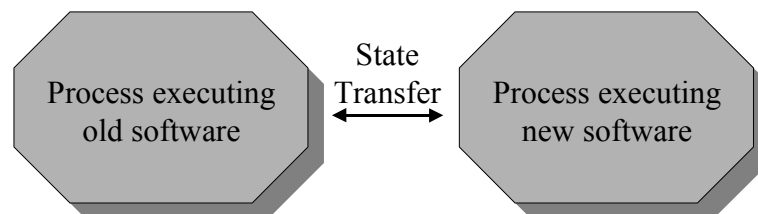


Figure 6-8 State Transfer

After the new software is started, TelORB's system upgrade function triggers the new process to begin a dialogue with its corresponding old process. Depending upon the solution chosen by the application designers, state transfer information can be transferred between old and new processes:

- Through the dialogue between the old and new processes
- Through the database, when the old process writes state information which can be read and understood by the new process.

Redundancy Upgrade

Redundancy Upgrade is a method where individual processors are upgraded by being rebooted. The old and the new software are never present at the same time on an individual processor. However, by moving processes from the processor that is to be rebooted, to another processor in the same processor pool, it is possible to maintain some continuity of service. Upgrade by redundancy may be used for upgrading the TelORB kernel.

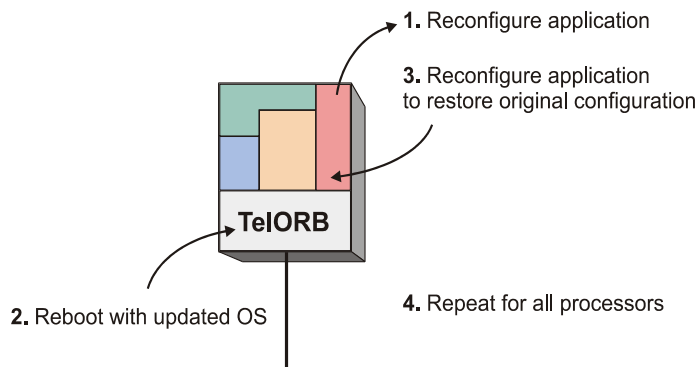


Figure 6-9 Redundancy Upgrade

Reboot Upgrade

This upgrade method which can be used causes a reload of the entire TelORB zone, which means that the TelORB system is unavailable for the period while the reload is being performed. As a result, this method should only be used when the interruption will not affect traffic, i.e. when upgrading a System that has Network Redundancy.

Reversion

The system upgrade mechanisms must also build in ways of handling system upgrades that fail. Typically, TelORB's error handling will detect errors and will request that the network element be returned to a previous state. The recovery action may be more drastic than the original upgrade, involving the rebooting and reloading of processors.

System Upgrade for Network Redundancy

The Primary and Standby Nodes are identical in a redundant TSP configuration, therefore any upgrades that are made to one node are also made to the other. Usually the two upgrades can be performed one after the other on the nodes and in any order.

However, if the system upgrade involves upgrading the database schema of the shared objects, then the upgrade cannot be performed while the link between the two nodes is active. This is because of the stream of committed database transactions that are sent from the Primary to the Standby Node must refer to the same database schema otherwise they can not be applied.

So when upgrading the database schema for shared objects between in a redundant TSP configuration the link between the two nodes is deactivated and each node is upgraded individually. The link is reactivated after the upgrades are completed.

SYSTEM UPGRADE - CONFIGURATION

The System Upgrade function has the following characteristics:

- Upgrades can be done during normal working hours
- The Upgrade Package delivered to a site includes all necessary support for introduction of application packages in the traffic system.

System Upgrade management is performed via the TelORB Manager user interface. Selecting the *Upgrades* tab on the TelORB Manager gives the following view:

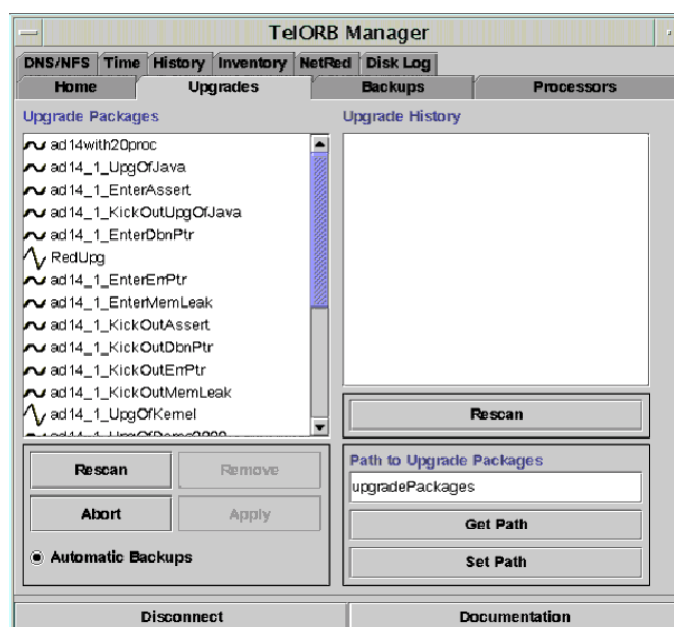


Figure 6-10 TelORB Manager Upgrades tab

UPGRADE PACKAGES

On the left you can see the *Upgrade Packages* field. This lists the current upgrades on the TSP platform. The names are preceded by a symbol, which tells you the upgrade type.

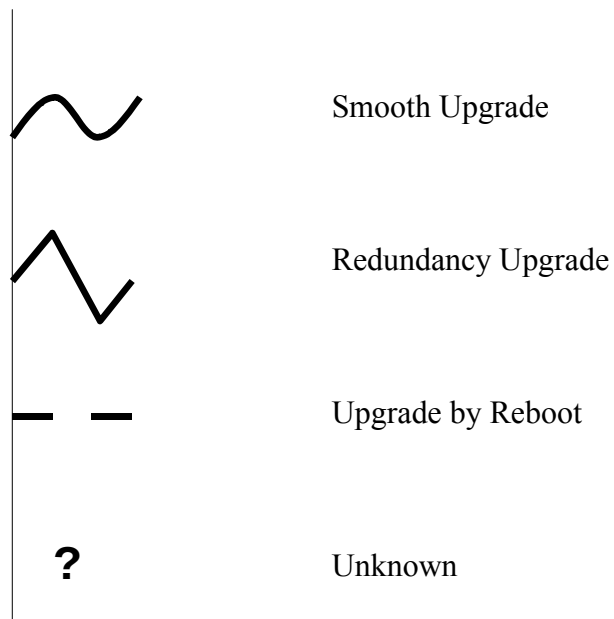


Figure 6-11 Upgrade Types in TelORB Manager

The following actions can be performed in this area:

Rescan

Rescans the IO machines to see whether any additional upgrade packages can be found.

Apply

Starts Applying whichever Upgrade package has been selected in the *Upgrades Packages* window.

Remove

Removes whichever Upgrade package has been selected in the *Upgrades Packages* window.

Abort

Aborts the current upgrade. A *System Upgrade Failed* notification is generated with the *Reason* field set to 3. A smooth upgrade will rollback if it is aborted early enough, whereas a redundancy upgrade will neither roll-forward nor roll-back - it will just stop.

Abort can leave the system in an unstable state and therefore should be used with extreme care.

Automatic Backups

It is also possible to automatically take a backup after an upgrade. If the function is enabled then a backup will be taken automatically if the system upgrade completes successfully. Note that this applies only to smooth and redundancy method upgrades, not to Upgrade by Reboot.

Upgrade History

On the right is the Upgrade History. This queries the system and produces a list of all the system upgrades that have successfully been applied since the system was first started. The names are preceded by a symbol, which tells you the upgrade type (same as in *Upgrade Packages*).

The list shows the symbol for the type of upgrade, the name of the upgrade package, and the date and time at which the system registered the upgrade as completing.

The list can be up to fifty entries long, after which the earliest entry will be removed from the list when a new upgrade package is applied successfully.

Path to Upgrade Packages

Upgrade Packages are normally stored on IO1, and it is there that the system scans for them by default. However it may happen that IO1 is not available when you want to store an upgrade or - for system administration reasons - the path has been changed on your system.

The *Get Path* button checks the system to find the current path to the directory where upgrade packages are stored. This defaults to the *upgradePackages* directory on IO1.

The *Set Path* button can be used to change TelORB's default setting for the path to the directory where the upgrade packages are stored.

SUMMARY

- TelORB Processors are loaded in a multiphase process under the control of a loader processor denoted as *OS Load*, *Initial*, *Base* and *Total Functionality*. Linux processors load in an independent manner.
- There are two official types of in-service System Upgrades - Smooth and Redundancy.
- Management of the System Upgrade function is performed via the TelORB Manager user interface.

Intentionally Blank

7 Network Redundancy

CHAPTER OBJECTIVES

After completing this chapter you will be able to:

- Describe the network configuration of the TSP Network Redundancy function.
- Identify the processes used to control and support consistent data between the Primary TSP node and the Standby TSP node.
- Configure and manage Network Redundancy on the TSP Platform.

INTRODUCTION

The TSP Platform achieves a very high level of availability at the cluster, (or node) level. However, this still does not provide a zero probability of cluster failure due to possibility of the following:

- serious software bugs causing multiple, simultaneous processor failures,
- complete power failure in TSP platform,
- natural disasters, for example, earthquake, fire, etc.

To increase the overall availability of the TSP platform, there is a need for an additional level of redundancy at the network level. This additional level of redundancy is referred to as a function on the TSP Platform called Network Redundancy or as Geographical Network Redundancy.

HOT STANDBY NETWORK REDUNDANCY

The TSP platform supports a Hot Standby type of Network Redundancy.

The essence of the Hot Standby Network Redundancy feature is to have two TSP network nodes, where one defined as the Primary node and the other as the Standby node. Then if the Primary node fails, the services provided by that node will be taken over by the other node, the Standby node.

Note: In this chapter when referring to the Primary node and the Standby node we often associate the Primary node with zone 1 and the Standby node with zone 2 as this reflects the default configuration. Zone 1 is configured to be the default Primary node or preferred Primary node which implies that when the system needs to decide which node should become the Primary in situations where both nodes could be Primary, zone 1 gets the preference.

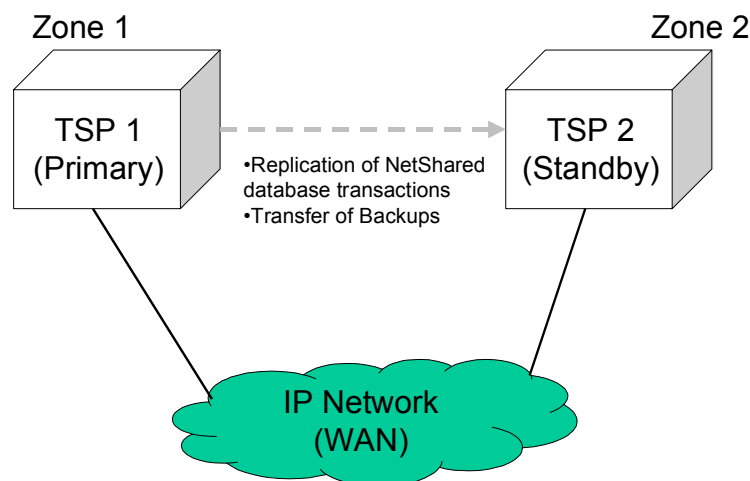


Figure 7-1 Hot Standby Geographical Network Redundancy

Usually these nodes are in geographically different places in order to minimize the risk of local disasters (e.g. earthquake, fire, floods, etc.).

In the Hot Standby Network Redundancy, the Primary TSP node is doing all the work, whereas the other, the Standby Node, is only there in case of failure of the Primary Node. From a data replication perspective the Primary Node forwards live transaction updates to the Hot Standby to keep its database replicated. The replication is done in a near-real time for all the transactions that are updating the database in order to keep the two nodes synchronized.

The two redundant nodes are, in terms of execution environment, separate zones. Network Redundancy requires that the two zones can communicate through a fully redundant *Update Channel*. The communication is IP based and for this purpose two dedicated Network Redundancy routers are used on each remote node. These routers are on the TSP side connected to the Platforms internal Ethernet infrastructure through the SCB-RP boards. On the Network side these routers are typically connected by WAN interfaces, e.g. a set of 2 MB E1 (or T1) circuits.

It should be noted that the critical *Update Channel* **only** uses these routers. Other routers, called VIP Gateway routers, would normally be found on a TSP site installation but are used for regular IP network traffic or the O&M IP service.

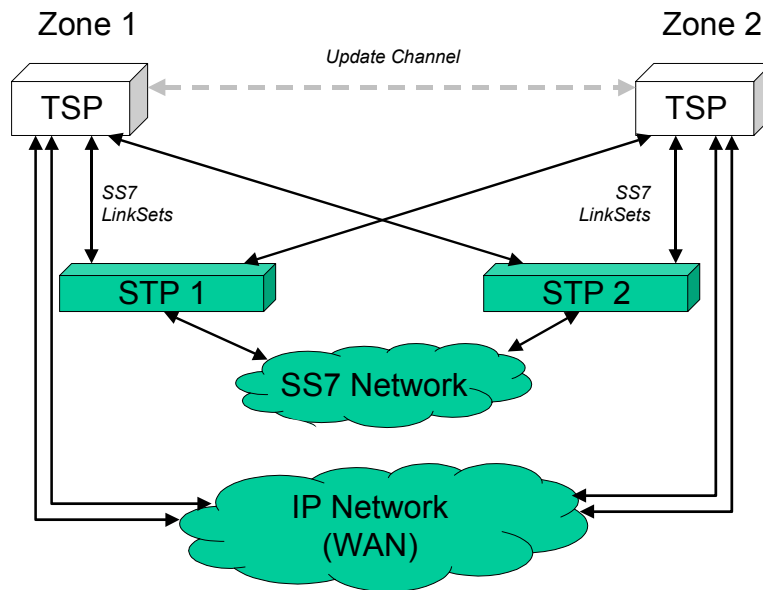


Figure 7-2 Network Redundancy Physical Configuration

In normal operation mode all SS7 traffic is routed by the STPs (SS7 Signaling Transfer Points) to the Primary TSP node. The Primary TSP node processes the incoming request, updates its database locally, forwards the response to the originator and then asynchronously updates the Standby TSP node using the same transaction that was used to update its local database. The Standby node does not process any SS7 traffic or provisioning updates, it simply keeps updating its database using the transactions that are forwarded by the Primary node.

PARTITIONED NETWORK REDUNDANCY

This new functionality has been introduced with TSP 4. The purpose with Partitioned Network Redundancy is to spread the load on both zones.

We can do that by letting each zone act as Primary for its half of the database. The reason why this will result in a more even load is that committing a mirroring transaction generally causes less processor load than it did creating the transaction.

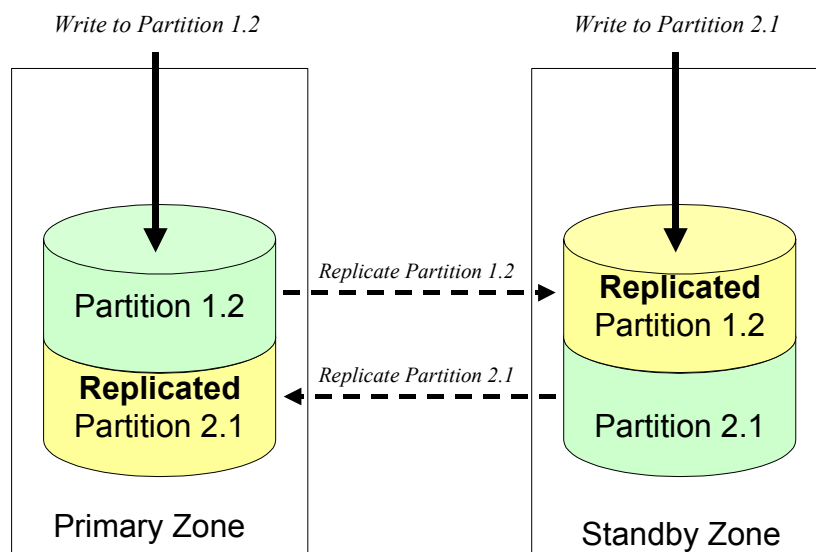


Figure 7-3 Partitioned Network Redundancy

As we can see from above figure, the database has been split up in two partitions one is called *Partition 1.2* and the other *Partition 2.1*. Zone 1 is updating the *1.x* part of the database, i.e. *Partition 1.2*. This means that this partition is replicated on Zone 2, identified as *Replicated Partition 1.2* in the figure.

NETWORK REDUNDANCY LOGICAL CONNECTIONS

To enable network redundancy communication, links between the two zones must be established. For robustness reasons, four logical links are defined, one for each Ethernet interface of the two processor's participating in the *DicosNetRedConnectPool*.

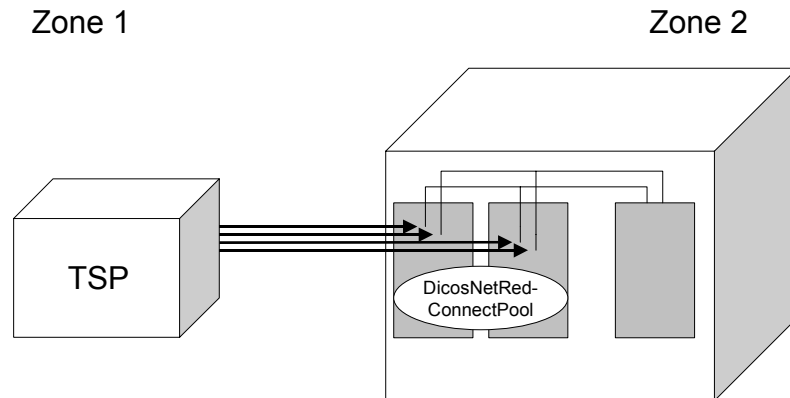


Figure 7-4 Logical links from Zone 1 to Zone 2

Four links are also setup in the opposite direction.

These links are only used to establish the connections actually used for replicating transactions. Each processor holding *netshared* database objects will have its own link for replicating database transactions to the other zone. This way of doing it avoids bottlenecks and is therefore highly scalable. Note that a processor might receive replicated transactions from more than one processor since the two zones do not necessarily have the same number of processors and/or pool configuration.

NETWORK REDUNDANCY PROCESSES

Processors on both sides involved in replication must participate in the *DicosNetRedPool*.

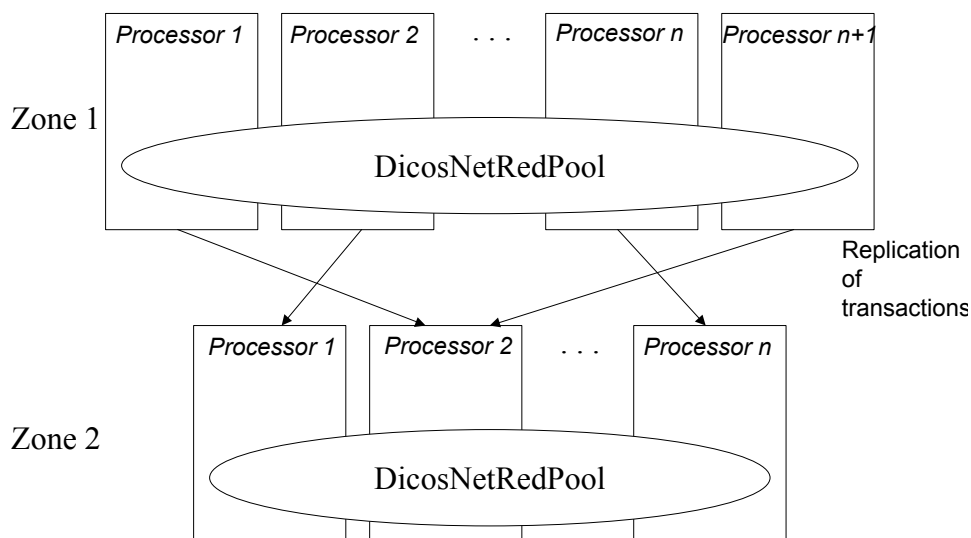


Figure 7-5 Replication of transactions between processors on the two zones

Replication of transactions between nodes requires a number of processes. The next figure shows how transactions that are created and committed on the Primary node are applied to the database on the Standby node. The following processes are involved:

PpClient

The *PpClient* is a static, replicated process, distributed on every processor of the node. Each *PpClient* process will consequently receive all the committed *netshared* transactions created on its own processor, package them into TCP/IP messages and send them on the link to the Standby node. To transport the transactions every *PpClient* process instance has its own dedicated TCP/IP connection to the Standby node. This ensures that the *PpClient* processes on all the processors can operate independently of each other.

PpDistributor

A *PpDistributor* process is present in the Standby node at the other end of each TCP/IP connection. The *PpDistributors* are dynamic keyed processes that are created when the TCP/IP connection is established. The main purpose of the *PpDistributor* is to *route an incoming transaction* to a suitable processor where the transaction will commit. When the *PpDistributor* has selected a suitable processor it just forwards the entire transaction to the *MirroringDistributor* on that processor.

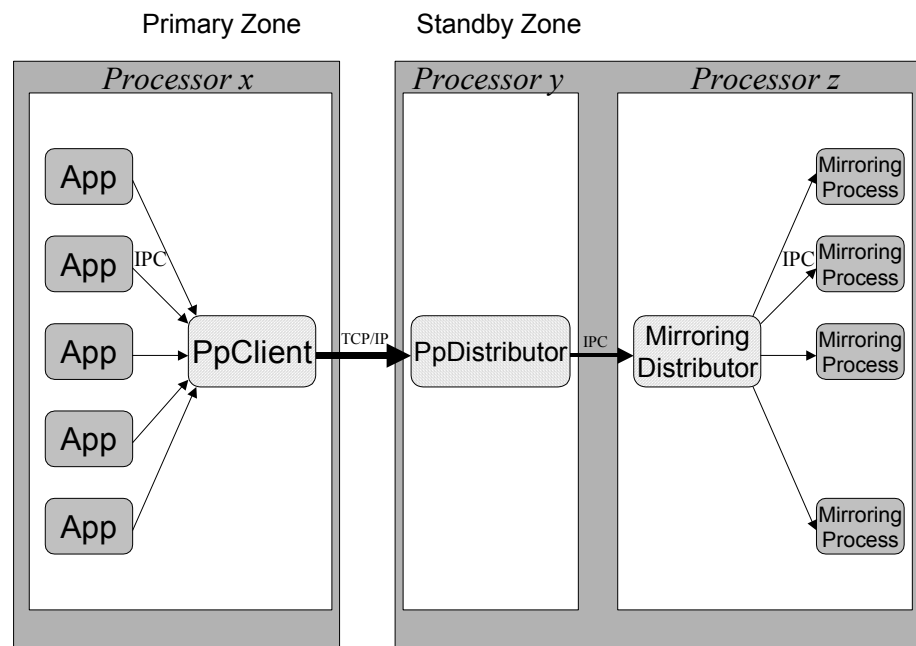


Figure 7-6 Network Redundancy Processes

The Applications (*App*) *netshared* transactions on the Primary node are transported by the *PpClient*, over the TCP/IP link to the *PpDistributor* via the *MirroringDistributor* to end up in a *MirroringProcess* on the Standby node. It is the mirroring processes that consume the transactions by applying the changes to the database on the Standby node.

Only transactions that contain changes to *netshared* data will be sent to the *PpClient*, e.g. a read-only transaction will not be sent to *PpClient* and consequently not be replicated to the Standby node.

NETWORK REDUNDANCY SOFTWARE ARCHITECTURE

The software involved in the control and support of the Network Redundancy function can be divided into two main groups:-

- Database Synchronization software
- Monitoring & Control software

Together they manage the upkeep of database replication and switchover management.

In each separate node database transaction consistency is always upheld.

DATABASE SYNCHRONISATION

From a replication perspective the Primary node forwards live transaction updates to the Standby node to keep its database replicated. However this cannot guarantee database completeness, say for example, if there is a communication failure between the two nodes some committed transactions on the Primary node might not be applied to the Standby node.

In order to keep the database at the Primary node synchronized with the database on Standby node, periodic database integrity checks are performed from the Primary database to the Standby database.

Remote Synch Operation

The periodic database integrity check is called a **Remote Synch Operation**. The Remote Sync operation consists of

- a **Quick Sync** Operation,
- Backup Transfer to the Standby zone, and
- an **External Sync** operation.

Remote Synch is ordered from the Primary node to synchronize the Standby. The simple logic of the remote synch is to first perform a Quick Synch and if that fails to then perform an External Synch.

This section describes the procedures of **Quick Synch** and **External Synch**.

Quick Synch

Within a Quick Synch the databases of the two zones are compared and a large compensating transaction will be prepared by the Primary node. This special transaction will be applied to the Standby node to repair for any lost update-, create- or delete operations.

There is a limit on the number of objects that the compensating transaction is allowed to contain, the current default value is 30000 objects. This can be changed in runtime by using the environment variable “*RSLimit*”. This also implies that the Quick Synch procedure is sufficient for updating the Standby node if the differences are within this number.

The Quick Synch does not require any IO disk access since there are no files to be transferred between nodes. A Quick Synch is initiated from the Primary node.

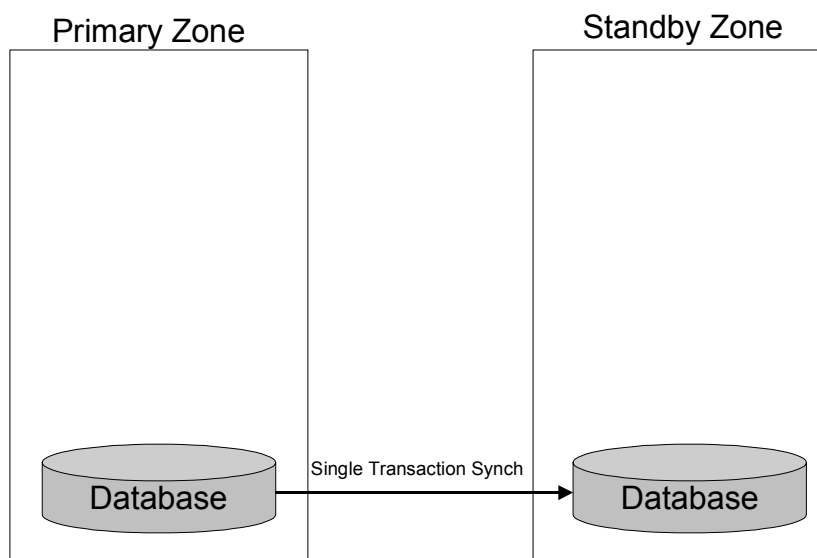


Figure 7-7 Quick Synch

If the Quick Synch operation is unsuccessful then the External Synch operation is started.

External Synch

In order to perform the External Synch operation a complete backup is transported from the Primary node to the Standby node where it is to be applied.

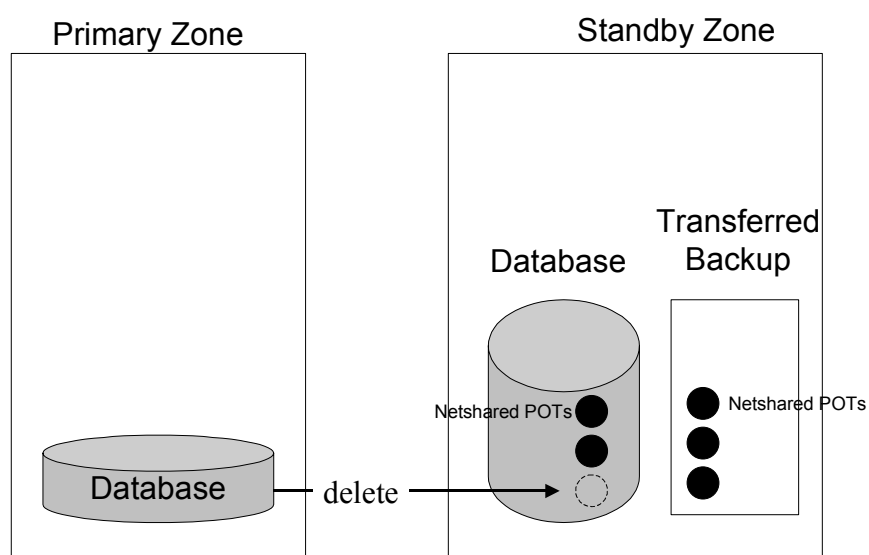


Figure 7-8 External Synch

Once the backup has been transferred an External Synch is carried out on the Standby node. It implies scanning through the backup and verifying that the *netshared* data of the backup is present in the Standby node. The replication of transactions goes on during the External Synch. The absence of a *netshared* instance deleted after the backup was taken will not be regarded as an inconsistency. Any “true” inconsistency is corrected by the External Synch. When the External Synch is completed a backup is taken on the Standby side. Traffic is not disturbed by this activity.

A timer in the TSP network redundancy software triggers the periodic database integrity check. Setting a parameter via the TelORB Manager controls the periodicity. The *netshared* database objects of the replica are not removed from the database, they are simply verified during the External Synch phase.

Reconciliation

Reconciliation can also be seen as a database synchronization function that is required after a hard takeover or when a TSP zone reloads. Reconciliation is automatically initiated when a TSP zone reloads.

During reconciliation the database replica of the Standby node gets regenerated from the database located in the Primary node. This task is accomplished with the collaboration of both zones. A backup is generated in the Primary node and used in the Standby node to regenerate the database replica using the **External Synch** mechanism.

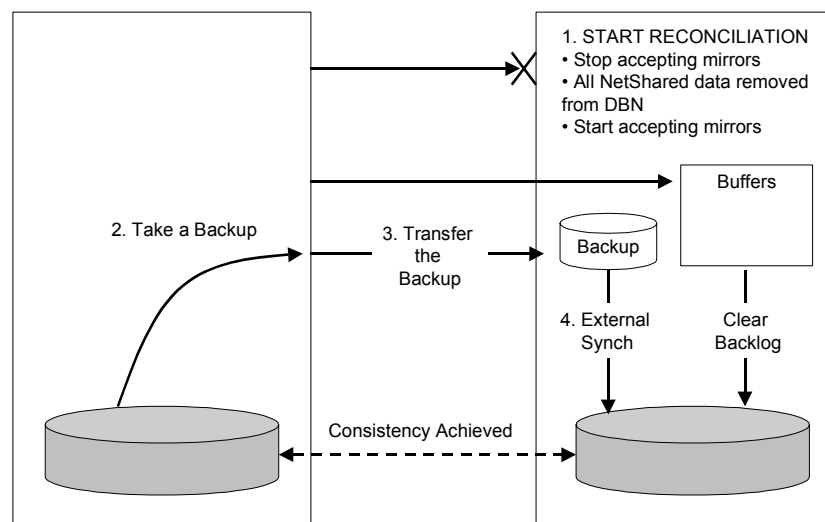


Figure 7-9 Reconciliation

The time line with all the events that are accomplished during the reconciliation process is as follows:

1. Start reconciliation on Standby node causes the following actions:
 - Standby node stops accepting transactions from the Primary node.

- The Reconciliation notifications, *Reconciliation Started as Standby in the Standby Zone* and *Reconciliation Started as Primary in the Primary Zone* are raised.
 - Standby node removes all *netshared* database objects from the database.
 - Standby node starts accepting transactions from the Primary node. The transactions are buffered in the Standby.
2. A backup is ordered in the Primary node. *Backup Started* and *Backup Finished* Notifications are raised.
 3. A backup starts to get transferred to the Standby node immediately. *Transfer Started Notification* raised. The backup transfer is an iterative process. *Transfer Complete* Notification raised.
 4. Backup transfer completed in Primary node. Start External Synch in the Standby node. This consists of reloading all the *netshared* POTs in the database from the data found in the newly transferred backup. *External Synch Started* and *External Synch Succeeded* Notifications raised in the Standby node when the External Synch is completed.
 5. A backup is created in the Standby node and the Reconciliation process is completed with the following notifications:

Reconciliation Succeeded as Standby Notification in the Standby node.

Reconciliation Succeeded as Primary Notification in the Primary node.

MONITOR & CONTROL SOFTWARE

The Monitoring and Control function within the Network Redundancy feature has the role of monitoring the status of the main components in each TSP node and exchanging that information with each node. In case of a failure in the Primary node, the control part takes decision for the Standby Node to take over the function of the Primary. The SS7 traffic is always routed to the current active Primary node where it gets processed.

The Monitoring and Control software runs in each TelORB cluster of each one of the TSP zones configured in a mated pair. The network redundancy software of each zone constantly monitors the following main components:

- TelORB State
- SS7 Subsystem state (SS7 links, SSNs, EIN stack BE and FE)
- Zone State.

Based on these three inputs, the software in the local zone and in the remote zone agree on which one of the two zones should be the acting Primary and which one should be the acting Standby. This decision mechanism is re-initiated every time any of the inputs involved in the decision changes in either of the two zones. This allows the system to be able to quickly react to failures and perform the proper re-configuration when required.

For example, if the software detects a failure in the SS7 stacks of the acting Primary zone then the software from both zones would trigger a failover and re-route the traffic to the other zone by deactivating links on the acting Primary zone. The Primary node switches zones.

Modes

The Monitoring and Control software sets the following Modes based on the condition of the TelORB database and the transaction update mechanism.

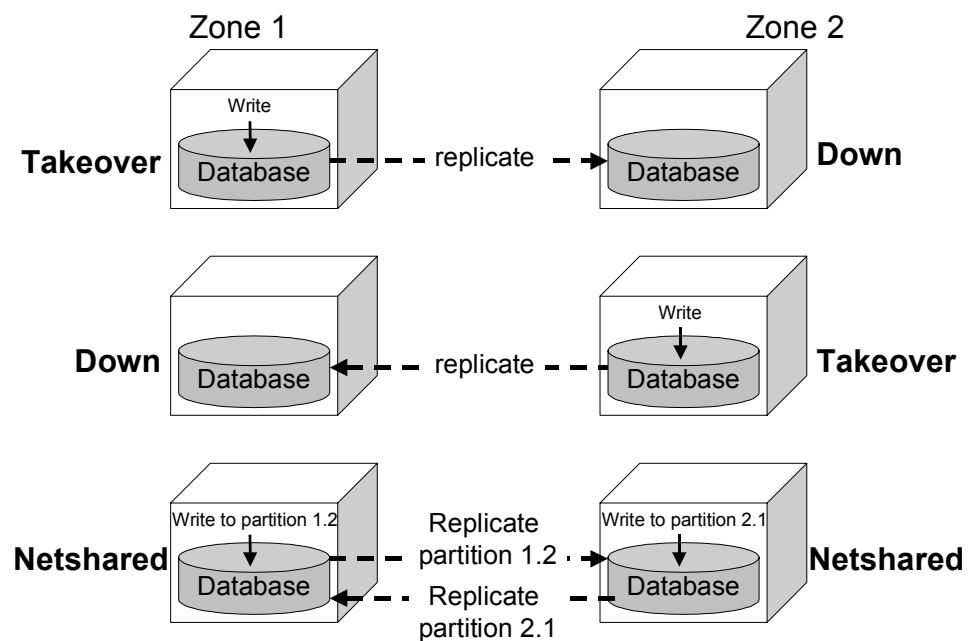


Figure 7-10 Network Redundancy Modes

The following is a list of all the existing modes and a description for each one of them:

Takeover. When the TelORB database is in Takeover its database is the database of the Primary node. The mode of TelORB database in the other zone is considered to be **Down**.

Down. When the TelORB database is in Down its database is the replica database. This means that the node is not allowed to order any updates of netshared data, it just passively receives updates from the other zone.

Netshared. The asynchronous transaction update mechanism is active and the replica, the database of the Standby node, is valid. The replica is most likely in synch with the database in the Primary node, however it could be behind.

NETWORK REDUNDANCY MANAGEMENT

The Network Redundancy area is an optional part of the TelORB Manager. In order to incorporate the Network Redundancy function onto the TSP Platform the Network Redundancy IDP must appear in the software list of the *epct* file:

IDP/D2, // NetRed_IDP

In addition, the identity of the zone needs to be specified with the following line:

ZONE 1;

And the two distribution pools, *DicosNetRedPool* and *DicosNetRedConnectPool* that have to be allocated to the processors in the zone.

The following is the view of TelORB Manager with Network Redundancy. From here you can control all aspects of Network Redundancy on the TSP Platform.

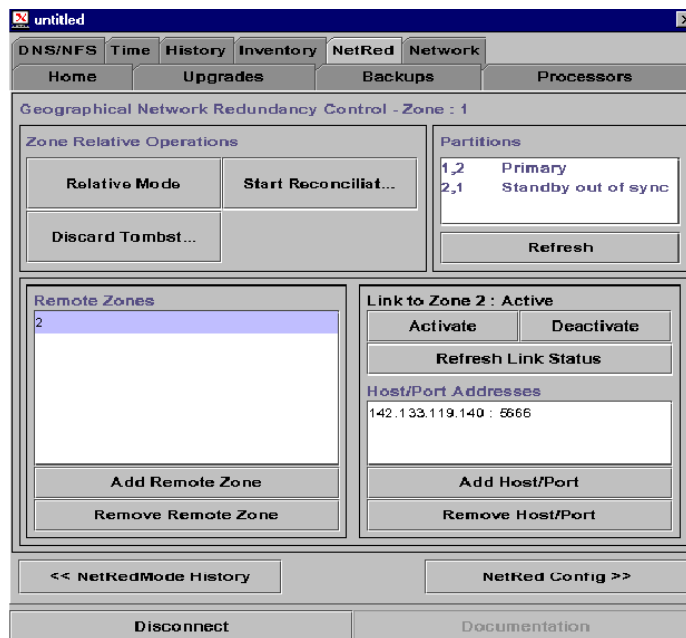


Figure 7-11 TelORB Manager Network Redundancy tab

The following functional areas are available to this function:

- NetRed Control

- NetRed Config
- Interzone IPC
- NetRedMode History

NETRED CONTROL

At the top left-hand side of the control screen is the area where you control the redundancy.

Relative Mode

The *Relative Mode* function displays the status of the zone relative to the zone specified. The usual status is



Figure 7-12 NetRed Mode

Other states would be “Takeover” (where only this zone can write to the database objects) or “Down” (where this zone has no write permissions on the database objects).

Start Reconciliation

The ***Start Reconciliation*** function allows you to start the reconciliation of the databases shared by the two zones.

Discard Tombstones

Delete operations, as well as updates and creates can be lost during transfer to the Standby node. A Tombstone is a marker to show that an object has been deleted on the primary. When the primary and the standby are synchronized, the tombstones (up to a certain timestamp) can be discarded because the deletions they were marking have now been applied on the standby.

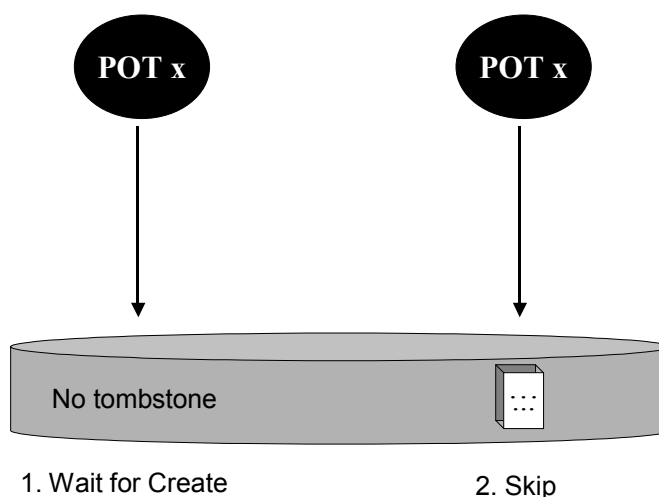


Figure 7-13 Database Update Scenarios

In rare error situations, it may become necessary to discard the tombstones. This should ***never*** be done without the approval of support. This function allows you to discard tombstones against the specified zone.

Partitions

Lists the partitions in the configuration and their current status. Clicking on “Refresh” will refresh both the available partitions and their status.

Remote Zones

The **Remote Zones** provides a list of all the remote zones to which this zone is connected. You can add a new zone or delete a selected zone here.

Selecting a zone highlights the zone information on the right-hand side of this screen.

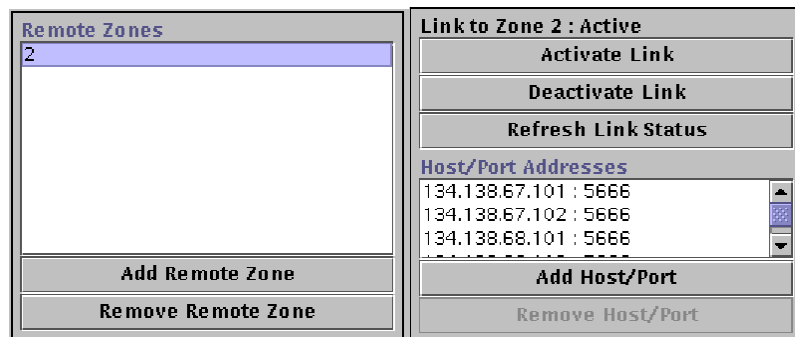


Figure 7-14 Remote Zones

Here you can see the status of the link. You have the option to Activate or Deactivate a link. The *Refresh Link Status* returns the latest link status values. Additionally you are able to add a new host address (with a port number).

NETRED CONFIG

The Network Redundancy configuration screen can be seen in the following figure.

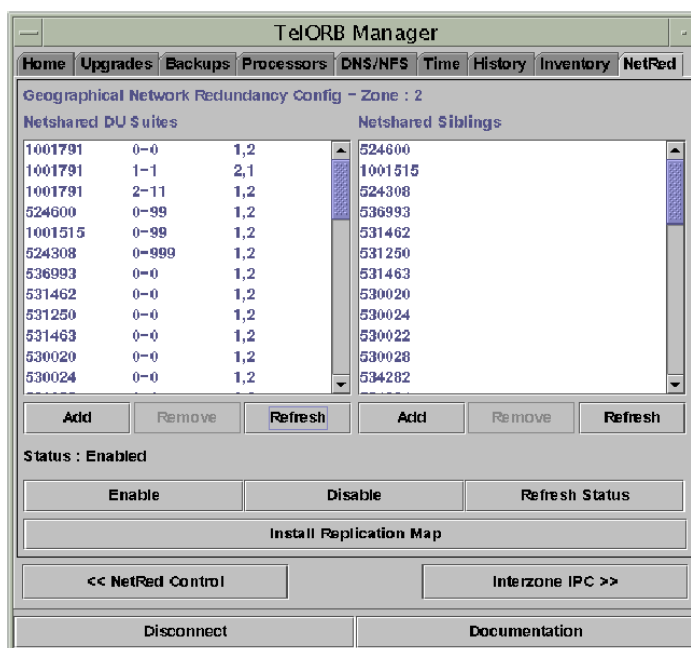


Figure 7-15 Network Redundancy Configuration

Netshared DU Suites

The Application POTs that need to be netshared are listed in the *Netshared DU Suites*. The Application POTs that need to be netshared should be configured as follows:

Class ID	524600
Low DU	0
High DU	99
Owner Zone	1
Backup Zone	2

Table 1 Netshared Application POTs

All DUTs that the application wants shared over the link should be configured in the same way. The owner zone should be 1 and the backup zone 2 for *all* DUs.

But in addition to the application DUTs, TelORB needs one of its own database objects to be *netshared*. That Persistent Object Type must be configured as follows:

Class ID	1001791	1001791	1001791
Low DU	0	2	1
High DU	1	1	11
Owner Zone	1	1	2
Backup Zone	2	2	1

Table 2 Netshared TelORB POTs

The reversal of the owner and backup zones for DU 1 is peculiar to this DUT, and is done to ensure that the redundancy is future-safe.

The list of *Netshared DU Suites* is identical, irrespective of which version of Network Redundancy your application is using.

Netshared Siblings

If you are going to be running Partitioned Netshared redundancy, you need to take some additional steps for which Zone 1 needs to be in “Takeover” mode. You will need to define the POTs that are to be partitioned netshared. This is done by adding these POTs as *Netshared Siblings*.

Removal of Siblings should, of course, be performed before the underlying DU Suites are removed. Both removal operations are done by highlighting the relevant Sibling or DU Suite, which then makes the “Remove” button available to you.

The Status display shows you the current status of the Network Redundancy, and also allows you to refresh the status display.

Replication Map

The **Replication Map** contains a list of all netshared POTs in the zone.

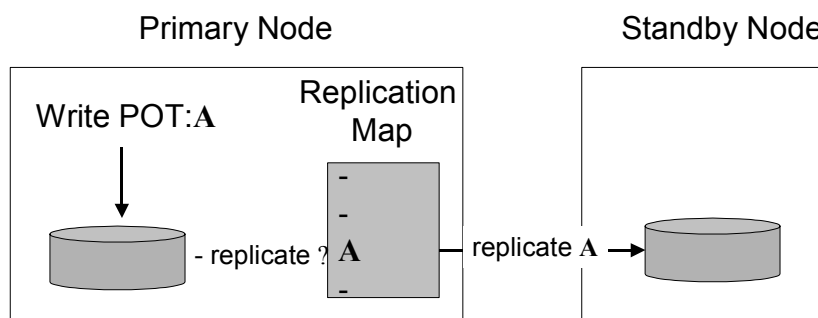


Figure 7-16 The Replication Map

This list is used by the operating system to know whether a database transaction should be replicated to the Standby zone or not.

This configuration does not become operational until you perform an installation of the replication map. This copies the configuration to the zone's local configuration data as one transaction.

Install Replication Map

The configuration of *netshared* objects is added to by adding the DU Suites, but this configuration does not become operational until you perform an installation of the replication map.

This procedure ensures the consistency of the *netshared* data between two zones as it allows the addition of several related DUTs to the basic configuration, while ensuring that the set is complete before the configuration becomes active. It also prevents a configuration change applied in one zone from having an immediate impact on the other zone.

To install the replication map, just click on the button *Install Replication Map*.

INTERZONE IPC

Clicking on the *Interzone IPC* button displays the following screen:

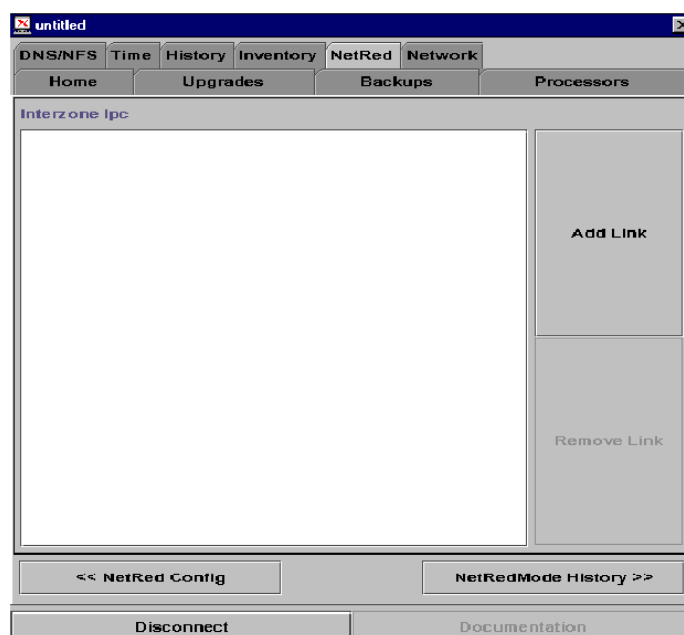
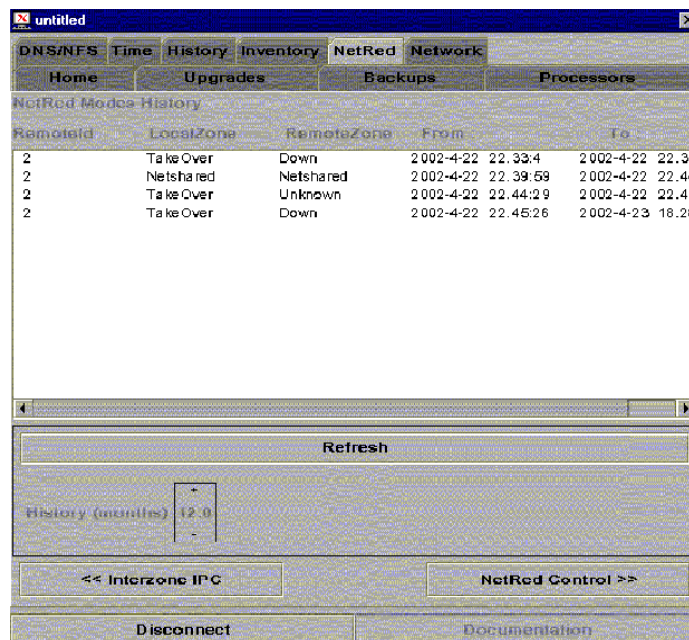


Figure 7-17 Interzone IPC

Here you can add or remove Interzone IPC links for T-IPC communication between zones.

NETREDMODE HISTORY

Clicking on the *NetRedMode History* button displays the mode history of the zones.



The screenshot shows a window titled 'untitled' with a menu bar (DNS/NFS, Time, History, Inventory, NetRed, Network) and a toolbar (Home, Upgrades, Backups, Processors). The 'History' menu is open, showing 'NetRed Modes History'. Below this is a table with the following data:

RemoteId	LocalZone	RemoteZone	From	To
2	Take Over	Down	2002-4-22 22:33:4	2002-4-22 22:35
2	Netshared	Netshared	2002-4-22 22:39:59	2002-4-22 22:44
2	Take Over	Unknown	2002-4-22 22:44:29	2002-4-22 22:45
2	Take Over	Down	2002-4-22 22:45:26	2002-4-23 18:28

Below the table is a 'Refresh' button and a 'History (months)' dropdown menu set to '12.0'. At the bottom are buttons for '<< Interzone IPC', 'NetRed Control >>', 'Disconnect', and 'Documentation'.

Figure 7-18 NetRedMode History

This screen displays the history of modes of both the local and the remote zones with the corresponding interval of time. Entries in the NetRed modes history panel describe the zone identity (ID) number of the remote zone, the local zone mode, the remote zone mode and the period of time (From - To) the two zones were in that mode. There are four possible values for a zone mode : Unknown, Down, TakeOver and Netshared. Zone mode is Unknown if the local zone couldn't get the remote zone mode when recording the modes.

Clicking on the refresh button will rechecks to see if there are new mode history records on the system. This is useful if you're performing some operations in a NetRed system - TakeOver, Reconciliation, Zone reload. Or if you have been connected to the TelORB system for a long time.

SUMMARY

- Network Redundancy is a feature of the TSP Platform to support higher availability due to the non-zero probability of cluster failure, for example, physical catastrophes like earthquakes or failure of the SS7 network.
- TSP supports Hot Standby Network Redundancy where the Primary Node handles the traffic and the Standby Node remains in a passive state. The Standby node will take over the traffic handling function if the Primary Node fails.
- In order to keep the database at the Primary Node synchronized with the database on Hot Stand-By Node, periodic database integrity checks are performed on the Primary database to the Stand-By database.
- The network redundancy software of each zone constantly monitors the following in order to determine which node should act as the Primary Node:
 - Tel ORB State
 - SS7 Subsystem state (SS7 links, SSNs, EIN stack BE and FE)
 - Zone State.
- The Network Redundancy function is configured and managed via the *TelORB Manager*.

Intentionally Blank

8 Expansion, Repair and Replacement of Hardware

CHAPTER OBJECTIVES

After this chapter the student will be able to:

- Describe the procedures required to add additional traffic processors
- Identify possible hardware faults
- Identify what corrective action to take against faults
- Isolate and correct hardware faults
- Replace faulty boards and cables

HARDWARE REPAIR HANDLING INTRODUCTION

The hardware of a TelORB based system is designed to cope with failure of individual hardware components. The replication of software between TelORB processors means that the system can survive the loss of processor boards.

The duplication of the Ethernet switches and IO processors means that the system can survive faults in these components as well. Generally the procedure for dealing with faulty hardware involves removing the faulty board, sending it away for repair and inserting a correctly functioning board. In some cases, it is also necessary to edit information in the system as the following descriptions indicate. These descriptions are based on the O&M documentation. Operators should always follow the most up-to-date version of the O&M documentation.

NOTE: When working with cables and boards, an Electrostatic Discharge (ESD) strap should be worn, to prevent damage to the components, and hence to the system.

REPLACING A TRAFFIC PROCESSOR

The traffic processor boards form the heart of the TelORB system. The boards are standard processor boards.

Replacing a traffic processor consists of changing the configuration, removing the old board, inserting the new board and reconfiguring the system.

A board must be replaced if it is faulty, or if it is to be upgraded. If a processor board has been in the system for 10 years, it should be replaced and sent to Ericsson. This is so that the lithium battery on the board can be replaced and returned to the operator.

PREREQUISITES

You must know the MAC addresses for all Ethernet interfaces on the new processor board.

CHANGING THE CONFIGURATION

Before you can remove a processor board you must alter the configuration of the system from within the User Interface.

Disable

If the processor is faulty, then the board should already have been disabled automatically, and the alarm *TelORB Processor Unavailable* is generated.

If you are replacing a functioning processor, you must do this manually.

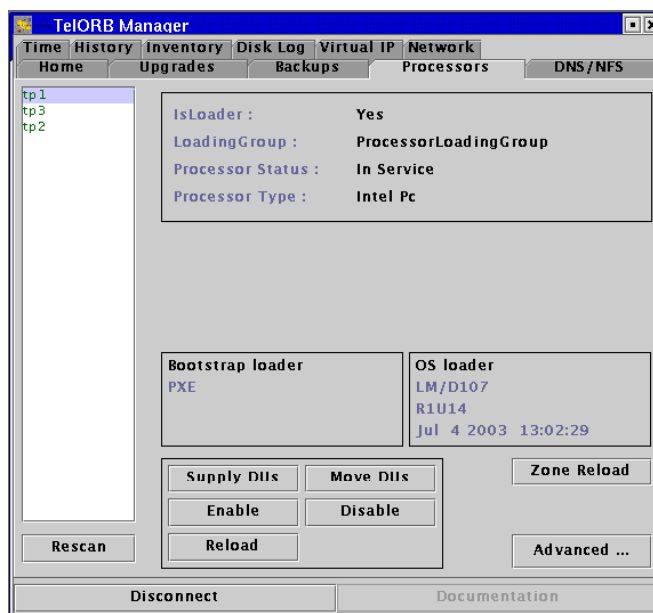


Figure 8-1 TelORB Manager Processors Tab

In the *Basic Information* screen of the *Processors* area of the TelORB Manager, the processor must be selected from the list and “Disable” button must be clicked.

The following notifications are generated

- *Disable Processor Started* followed by
- *Disable Processor Succeeded* or *Failed*.

If the disable fails, then the procedure should be tried again. If the disable is successful, a *TelORB Processor Unavailable* alarm is generated.

Backup

It is a good idea to backup TelORB, so that if there is a problem with another board, you will always have a system that can be restored.

REPLACING THE BOARD

The old board should be removed and replaced with a new one following the procedure outlined in the *Hardware replacement instructions* for the specific board to be replaced. This can be found on the CPI documentation under *Operation & Maintenance → HW Equipment Maintenance → Hardware replacement instructions*.

Replacement of a traffic processor

The traffic processor needs to be replaced if it is not working. The procedure is as follows:

1. Locate the faulty unit which may have the Manual Intervention (MIA) indicator lit.
2. Attach the ESD strap to the wrist and connect the free end of the strap cable to the ground socket in the upper part of the cabinet.
3. Make sure that all cables connected to the unit are properly labeled.
4. Disconnect all cables from the unit.
5. Loosen the screws holding the unit in the magazine, using a screwdriver of type Torx TX08.
6. Carefully slide the unit forward from the magazine and remove it.
7. Remove the new unit from its ESD bag and carefully slide the new unit into the magazine. Tighten the mounting screws.
8. Reconnect and fix the cables.
9. Remove the ESD wrist strap.
10. If required, use the packaging material from the replacement unit to repackage the suspected faulty unit, and return the unit to the service center

Steps 3, 4 and 8 only apply to faulty boards with external interfaces connected.

RECONFIGURE THE TELORB SERVICES ON THE IO PROCESSORS

The following steps are required in order to configure the MAC address of the new processor board in the TelORB system files.

The */opt/telorb/etc/telorb.conf* file and the */opt/telorb/axe/loadingGroup.../Networkaddresses* file need to be edited to reconfigure the TelORB services running on the IO machine.

Remote Login

Do a remote login from the controlling environment to an IO processor using telorb as the login name.

```
ssh telorb@hostname
```

telorb@hostname's password:

Type in the password for the user telorb.

Network Addresses File

Because you have changed processors, you have to modify the Network Addresses files. These can be found under *the /opt/telorb/axe/loadingGroup01_1* directory of IO processor 1, and *the /opt/telorb/axe/loadingGroup02_1* directory of IO processor 2. Change the entries corresponding to the replaced processor or processors.

Edit telorb.conf

Open the file */opt/telorb/etc/telorb.conf* in an editor of your choice, e.g. vi or emacs. Every TelORB processor has an entry for each of its MAC addresses, e.g.

```
dicosProc_m1_s17_0 eth_addr_for_eth0  
00:a0:a5:4d:00:9a
```

```
dicosProc_m1_s17_1 eth_addr_for_eth1  
00:a0:a5:4d:00:99
```

Change MAC address for the replaced processor.

Running the script

To let the changes take affect on the IO machine you have to reconfigure the TelORB services.

```
io1:etc> cd
/opt/telorb/codearchive/iom_scripts/

io1:etc> ./configureTelORB.sh -c

io1:etc> exit
```

Repeat this procedure, i.e. reconfigure TelORB services on the IO machine, for the second TelORB IO Processor.

RESET NEW PROCESSOR(S)

In order to introduce the new processor(s) into the system, it must first be reset, using the hardware reset on the front of the processor card. Press this reset switch once and then release it.

RECONFIGURE THE SYSTEM

Enable

In the *Basic Information* screen of the *Processors* area of the TelORB Manager, select the processor in the list and click on *Enable*.

The following notifications are generated

- *Enable Processor Started* followed by
- *Enable Processor Succeeded or Failed*.

If the Enable fails, then the procedure should be tried again. It may be that a processor reloaded while the Enable was running, which would cause the Enable to abort.

If the Enable is successful, the alarm *TelORB Processor Unavailable* with a severity of “cleared” will be generated. The procedure to replace the board is complete.

Backup TelORB

It is a good idea to perform another backup TelORB when you are notified that the Enable is successful. This means that you will have a complete system that can be restored.

EXPANDING THE TELORB SYSTEM

Expanding the TelORB system consists of creating an upgrade package which includes the new processor or processors and then applying that upgrade. This new configuration is saved in a backup and the new processor information is added to the files on the IO processors.

When these steps are complete, the new processors have to be enabled.

This procedure (especially the editing of the files on the IO processors) could damage the system if it is performed improperly. It should only be carried out by TSP personnel, or staff who have completed the appropriate training.

PREREQUISITES

There must be a number of vacant processor slots in the cabinet equal to or greater than the number of processors to be added. Some UNIX expertise is also needed to complete this procedure.

GENERATE THE UPGRADE PACKAGE

On the system where you are generating the package, you should edit the old *.epct* file by adding new entries for the processor or processors to be added.

Note that, when preparing an upgrade package for adding a new board, the package is absolutely site specific. Not only are the MAC addresses of the board to be added unique, but the System Generation ID must match that of the target system. If you have two systems with an identical number of boards and wish to expand both of them (both zones of a Network Redundancy system, for example), you must NOT prepare one upgrade package and send it to both systems.

Perform the steps in the document *Producing an Upgrade Package*.

ADD NEW BOARD

Fasten the new board (or boards) into the appropriate slot or slots.
Remember to label the board correctly.

Note: You must wear an ESD strap to prevent electrostatic discharge.

UPGRADE THE SYSTEM

Follow the steps in the document *Applying an Upgrade Package*.

If everything has succeeded, in the *Processors* area of the TelORB Manager the added processors should now appear in the list of processors.

If they do not, click on the *Rescan* button. If they *still* do not appear, there was a fault either in the editing of the files or the creation of the package.

RECONFIGURE THE TELORB SERVICES ON THE IO PROCESSORS

Network Addresses File

Because you have added new processors, you have to modify the network addresses files. These can be found under the */opt/telorb/axe/loadingGroup01_1* directory of IO processor 1, and the */opt/telorb/axe/loadingGroup01_2* directory of IO processor 2. When you carried out the upgrade in a previous step, a further *networkaddresses* file was created with an extension name of whatever name you entered in the *.epct* file (e.g. */opt/telorb/axe/upgradePackages/myUpackName/copy-data/networkaddresses.myUpackName*). This file has been automatically copied to the IO processors during the upgrade and located in the same directories as the network addresses files.

You should open this file in a text editor and copy *all* the lines in it, then paste them in to the bottom of the *networkaddresses* file.

Edit the Configuration File

Because you have added new processors, you also have to modify the TelORB configuration on the IO processor.

For each new processor you must specify the IP and MAC addresses, and whether or not the processor is to be a loader. For PM-ETH processors IP and MAC addresses for the extra interfaces must also be added.

Running the Script

To run the configuration script you must be in the right place and either logged in as the **telorb** user or at least member of the **telorb** group.

Enter the command:

```
prompt> cd  
/opt/telorb/codearchive/iom_scripts/  
prompt> ./configureTelORB.sh -c
```

Reset, Enable and Backup

Reset and enable the new processor(s), and perform a Backup as described on under *Replacing a Traffic Processor* at the beginning of this chapter.

THE ETHERNET INTERFACE

There are two identical Ethernet switches in the TelORB cabinet, each of which is connected to the outside world via the IO processors and connected to the TelORB system by a separate Ethernet connector on each TelORB processor. Each set of switch, cables, and connections, forms an Ethernet interface.

Ethernet switch 1 (the topmost of the Ethernet switches) is part of Ethernet Interface 0. Ethernet Switch 2 (the lower of the two) is part of Ethernet Interface 1.

The Ethernet cables are labeled at each end. Before you unplug a connector, please ensure that it is the correct one. The Ethernet switches are generally referred to as switch 1 and switch 2. In software, however the Ethernet interfaces are usually called A and B. Ethernet switch 1 corresponds to Ethernet Interface A.

DIAGNOSING THE ETHERNET INTERFACE

The alarm *Transceiver or Cable Fault* is generated for each processor and interface from which the fault is detected. The procedure for eliminating the fault is to replace suspected parts until the problem disappears.

Depending on which interface and how many boards are generating the alarm we can propose a list of likely failure points. There are three categories of failures listed below.

Many alarms from eth0 or eth1

If the alarms are from boards in the same magazine then they are probably related. If they are from boards in different magazines they are probably unrelated and should be treated as individual single alarms.

Multiple alarms are most probably caused by

- faulty SCB
- faulty Magazine backplane

Replace suspected units until the problem is resolved. Use the procedures in the *Hardware Replacement Instructions*.

Single alarm from eth0 or eth1

This alarm is most likely caused by

- faulty processor board
- faulty SCB
- faulty magazine backplane or connector

Replace suspected units until the problem is resolved. Use the procedures in the *Hardware Replacement Instructions*.

Alarm from eth2 or eth3

This alarm is most likely caused by

- faulty cables between processor board and external equipment
- faulty external equipment
- faulty processor board

1. Replace suspected units until the problem is resolved. Use the procedures in the *Hardware Replacement Instructions*.

SUMMARY

The operator may replace the following hardware items if faulty:

- Ethernet switches
- TelORB processors
- IO processors

9 Disk Log

CHAPTER OBJECTIVES

After completing this chapter you will be able to:

- Describe the Disk Log function on the TSP platform
- Configure and Manage the Disk Log function on the TSP platform

INTRODUCTION

The database within the TelORB clusterware (TelORB database), which already has extremely high availability, scalability and performance, also provides a higher degree of persistency. It is achieved by storing changes to selected persistent data to hard disks. This is a function called Disk Log.

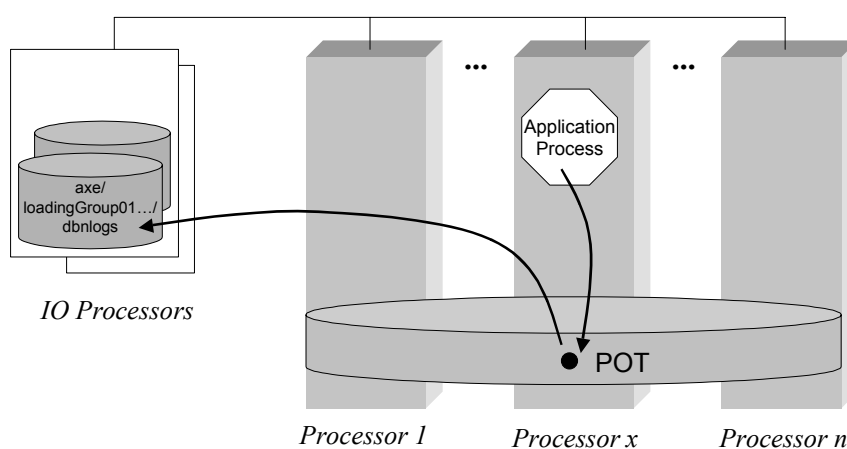


Figure 9-1 Disk Log

Applications can use these features to make sure all updates to critical information, such as subscriber account balances in a prepaid system, is recovered even if the entire system reloads. (Reference to application product description to get more information about which application data that uses this capability.)

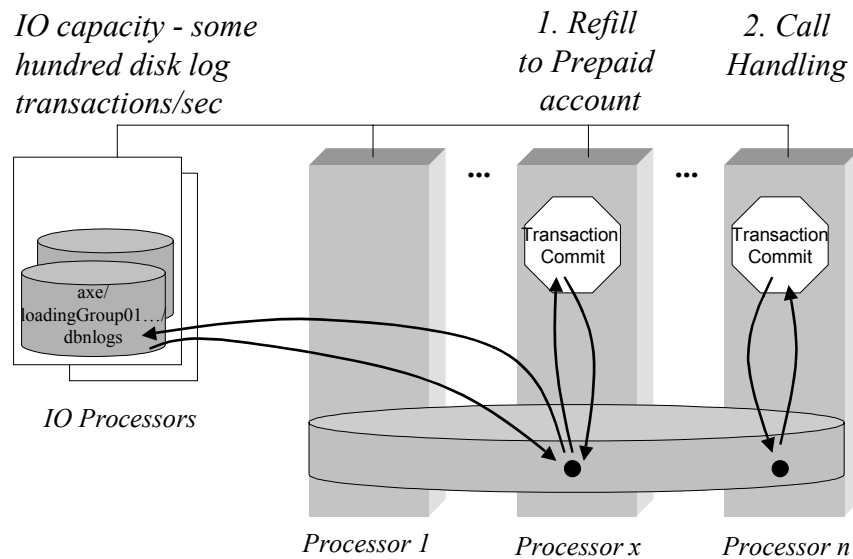


Figure 9-2 Disk Log of Application critical data

A transaction that updates such data will store the changes in a log file on the IO processors when the transaction commits. If the zone should reload then the active backup is loaded (as always) and after that the database log files will be read to restore all disk persistent data.

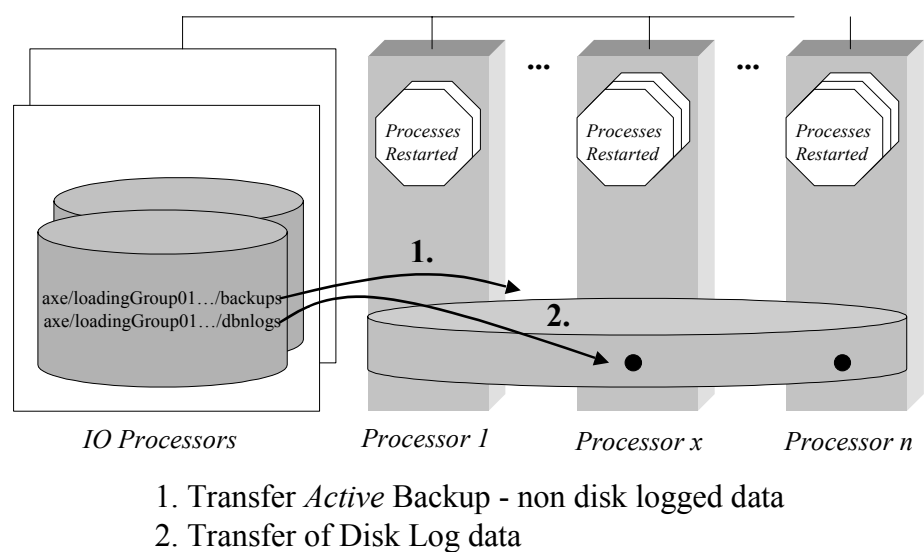


Figure 9-3 Zone Reload with Disk Log Enabled

The duplicated disks are still used to even further secure the persistency of the data in case of hardware failure.

The Disk Log feature is only available for configurations that do not have the optional geographic redundancy option. Installations that do have geographic redundancy do not need the database disk logging option since their data is already being backed-up on another physical cluster.

This improved persistency has some limitations:

- All transactions that update disk persistent data will of course take a longer time to commit. The commit call will not return until all changes have been secured on redundant disks. Notice that this increased latency only applies to write operations, reading disk persistent data has the same characteristics as for normal POTs.

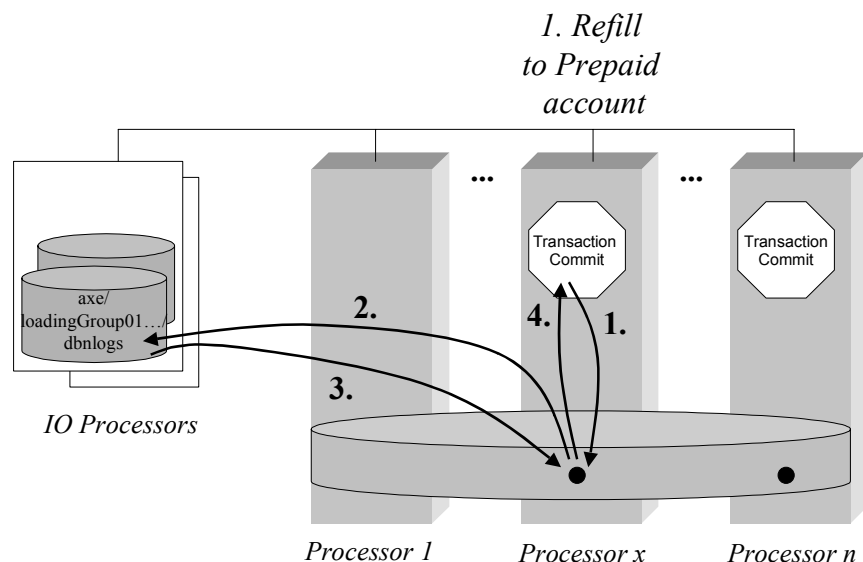


Figure 9-4 Disk Log transactions

- POTs can **not** be both net-shared and have disk persistent at the same time. Transactions should only open one kind of POT (net-shared, disk persistent, or normal), different kinds can not be mixed in the same transaction. This limitation may be removed in future releases.

CONFIGURING DISK LOG

In order to incorporate the Disk Log functionality onto the TSP site the Disk Log_IDP (IDP/D24) needs to be included in the site's *epct* configuration file.

The Disk Log function can either be disabled or enabled on the TSP node. If it is disabled, then no data will be logged and the alarm *Disk Log Incomplete* will be raised. Otherwise the POTs configured for disk logging will be logged to one or more logging group file sets.

In a similar way to network redundancy, these Persistent Object Types (POTs) can be configured (in runtime) to have disk persistence.

ENABLE DISK LOG

The Disk Log function is enabled, disabled and the disk logged POTs are configured via the *Disk Log* tab on the *TelORB Manager*.

Pressing the tab, *Disk Log*, takes you to the following screen (note that this tab is only available if your system contains Disk Log functionality)

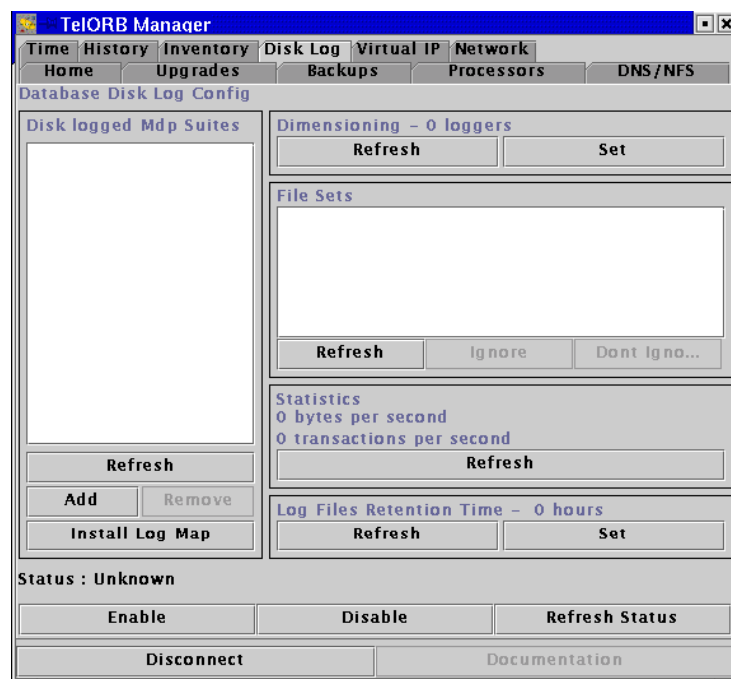


Figure 9-5 TelORB Manager Disk Log tab

The *Disk Log* function contains the following fields:

- *Dimensioning*
- *File Sets*
- *Disk Logged Mdp Suites*
- *Statistics,*
- *Log Files Retention Time,* and
- *Status* Information field

Dimensioning

Displays how many loggers are currently set. The recommended value here is '2'.

File Sets

The loading groups available for logging will be shown. Active loading groups are indicated by the color green and ignored loading groups are shown in red.

You can use the buttons "Ignore" and "Don't Ignore" to specify on which loading groups you want to log. Normally you should always log on both loading groups. This will prevent a loading group from being a single point of failure in the Disk Log function.

Disk Logged Mdp Suites

This field allows you to enter the POTs available to the Disk Log function. The POT is entered under its Class Id.

The "Log Map" function contains a list of POTs which should be Disk Logged. Logging for a POT does not actually start until it is entered into the "Log Map". By clicking "Install Log Map" you enter all of the POTs listed in the window "Disk Logged Mdp Suites" into the "Log Map".

When you modify the "Log Map" in this way you need to make a new Backup. This is needed since the Disk Log only contains incremental changes to some known state (i.e. a backup).

Statistics

The statistics shown here are the number of transactions per second currently being logged and the number bytes being written as a result of these transactions. Clicking on the *Refresh* button returns the latest values.

Log Files Retention Time

The Disk Log retention time feature is used to remove old Disk Log files. The “refresh” button is used to get the current value for retention time and the “set” button is used to change the value.

Every time a backup is generated with Disk Log enabled, all disk log files are examined for time of creation. If their time of creation is older than the current retention time, those disk log files will be deleted.

Note that the retention time can be set to 0. This means that all disk log files will be deleted with the exception of the ones currently open and the ones of the previous backup.

Status Information field

The Status Information field allows you to verify the current status of the Disk Log function, i.e. *Enabled* or *Disabled*. The *Refresh Status* function returns the true status at any time.

SUMMARY

- The Disk Log function on the TSP platform supports higher system persistency by providing a function where transaction updates to objects on the TelORB database can be logged to a log file on the IO processors when the transaction commits.
- If the TSP platform should reload then, the active backup together with these database log files are read to restore all disk persistent data on the node.
- The Disk Log function is made available to the TSP platform by loading the DiskLog_IDP software, IDP/D24, at site configuration and is enabled, disabled and configured via the *TelORB Manager*.

10 SS7 Signaling Management

CHAPTER OBJECTIVES

After this chapter the student will be able to:

- Describe the protocols available for SS7 communication on the TSP Platform
- Detail the component structure of the SS7 stack on the TSP Platform
- Manage the SS7 function on the TSP Platform

SS7 SIGNALING

The Signaling subsystem within TSP provides connectivity with external telecommunications and data communication networks. This connectivity is provided at the hardware level by specialized plug-in units, and complemented at the software level by various supported protocols. The software further enhances usability by providing such features as load sharing and high availability.

TSP as a platform supports both traditional Signaling System no.7 and several new IP-based network signaling protocols.

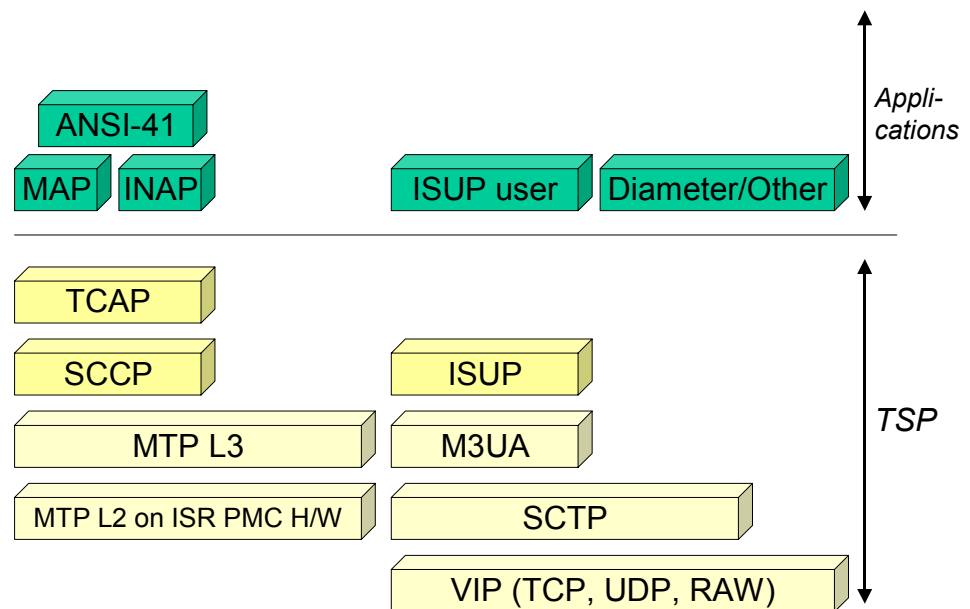


Figure 10-1 Overview of Signaling Stacks in TSP 4.0

SS7-Based Protocols

SS7 is an integrated signaling solution on the TSP platform. The following narrow-band (56 or 64-Kbps per channel) SS7-based protocols are supported.

TCAP:

The Transaction Capabilities Application Part (TCAP) protocol is offered in three different standards/markets: ITU, ANSI, and Chinese.

ISUP:

The ISDN User Part (ISUP) protocol is offered as a SS7 solution in two different standards/markets: ITU and ANSI.

IP-Based Protocols

The main function of the IP connectivity is to support IP-based payload and IP-based operation and maintenance. Accordingly, TCP/IP, UDP, and IP are all supported. In addition, the TSP platform offers the following IP-based protocols and broadband SS7 solutions over IP:

SCTP:

The Stream Control Transmission Protocol (SCTP) according to IETF RFC 2960 (October 2000) is supported.

ISUP:

The ISUP protocol is offered as a broadband SS7 solution over IP in two different standards/markets: ITU and ANSI.

Physical interfaces

External communication is provided by the following physical interfaces:

- PM-E1/T1 Processor Module
- PM-ETH Processor Module

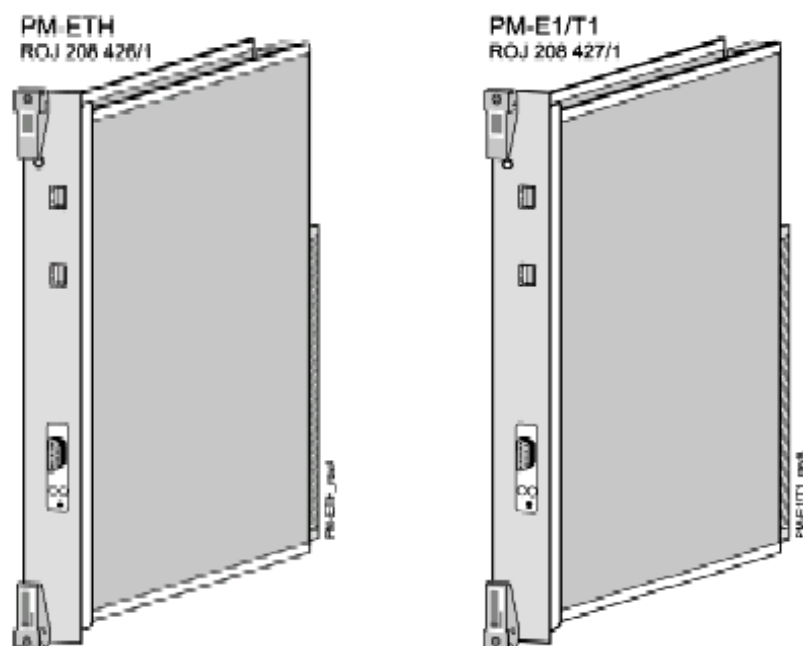


Figure 10-2 SS7 Physical Interfaces

The **PM-E1/T1** processor module with E1/T1 interface terminates Signaling System #7 (SS7) signaling connections. The module is a Pentium 700 MHz traffic processing board outfitted with E1/T1 connectors on a PMC card. From 2 to 6 PM-E1/T1 modules can be supported on the platform. Each E1/T1 PMC-board has 32 signaling channels (32 timeslots) which can be distributed on four physical E1/T1/J1 connectors.

The **PM-ETH** processor module with Ethernet interface terminates Ethernet connections for IP signaling. The module is a Pentium III 700 MHz traffic processing board outfitted with IP interfaces on a PCI Mezzanine Card (PMC). 4 to 10 PM-ETH modules are supported on the platform. Each IP PMC has 2 100 M-bit connections, and can be configured for VIP to handle external IP communications.

SS7 SIGNALING ARCHITECTURE

General Architecture

The SS7 stack uses a 2-tier or *distributed* design. These 2 tiers are called the Front End (FE) and the Back End (BE). The FE of the stack terminates the SS7 or IP connections and performs protocol handling for an MTP or SCTP stack. The BE of the stack contains the upper protocol handling layers, and handles the interface with the application. The FE and BE are interconnected by an Ethernet LAN using TCP/IP.

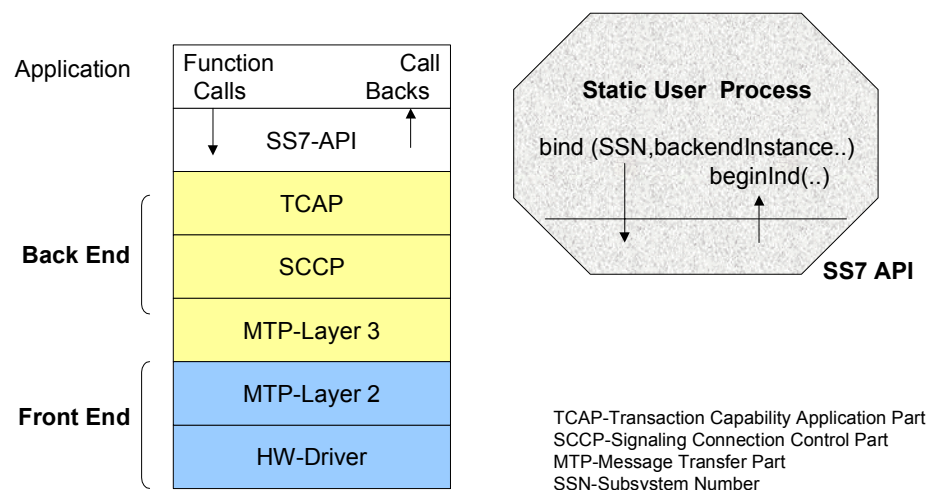


Figure 10-3 SS7 Stack – Front End and Back End Processes

SS7 Component Description

All signaling stacks of the TSP platform are composed of a number of identifiable stack components. A feature of these stack components is their reusability and adaptability to different hardware and operating systems. This section describes the components within each of the TSP SS7 signaling stacks:

- ISUP Narrow-band Stack

- ISUP SIGTRAN Stack
- TCAP Stack
- SCTP Stack

The following components are common to all stack architectures:

All components within the SS7 stack architecture can be configured with a graphical user interface tool called **Dynamic Configuration Manager (DCM)**, also running in the Linux environment.

All management over the stack is accomplished through the **Operation and Maintenance Management Interface (OAM I/F)**. Alarms are also received through this interface.

Important warnings and errors are also logged by the stack to a log file on the Linux side of the platform. This is handled by a **log daemon** process via UDP/IP packets from the various stack components. The **Management Module (MM)** is the interface between the OAM I/F and the stack components.

The various stack processes are initialized and monitored by a dedicated agent called the **Process Control Function (PCF)**. PCF is responsible to ensure that stack processes are in an operational state at all times. The PCF restarts any failed stack process in order to minimize downtime. The PCF also enables the reconfiguration of the stack software (e.g. adding new processes, removing processes, and upgrading of software). PCF itself is monitored by the DICOS operating system of TelORB. Should the PCF process fail, DICOS immediately re-initializes a new PCF process to take its place.

The **Common Part (CP)** is an encapsulation of the operating system in which the stack processes execute.

ISUP Narrow-band Stack

The ISUP Narrow-band stack provides an ISUP Application Programming Interface (API) in the C++ language to an application running in the Linux part of the TSP platform. The diagram describes these components as:

- ISUP Application

This is the SS7 handling process belonging to the application that is using the ISUP Narrowband stack. It is bound to the stack via the ISUP API. The application is written in the C++ language.

- ISUP API

The ISUP API provides the interface by which the ISUP application can receive and send traffic through the stack. The API executes in the Linux environment, and is in the C++ language to enable it to be code-compiled with the application. Note that the API is also generic, and relies on the services of CP for its execution environment needs.

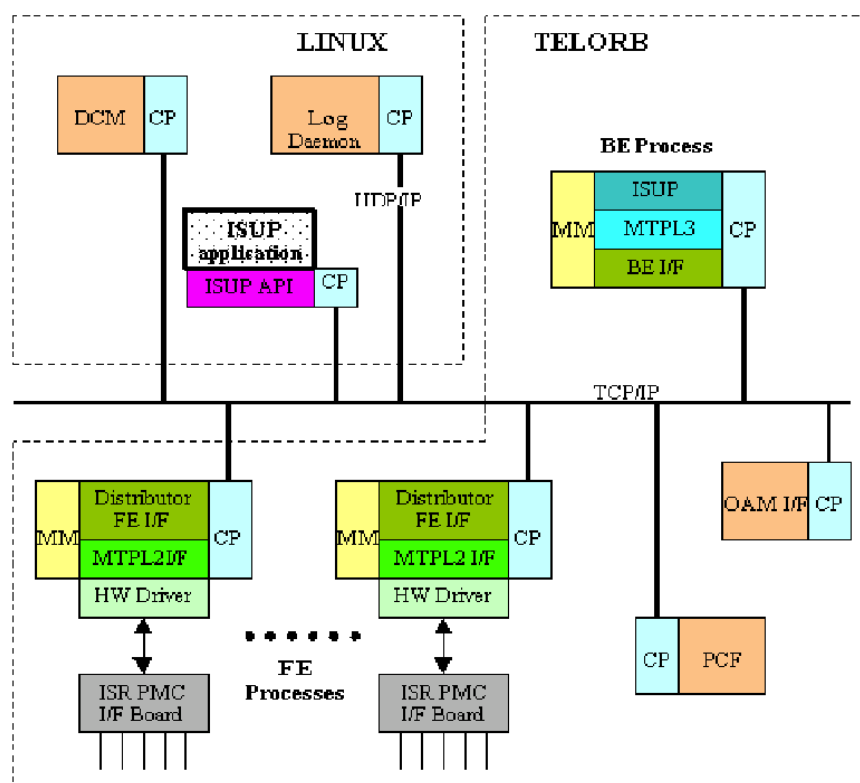


Figure 10-4 Architecture diagram of the ISUP Narrowband stack

The stack itself will be executing on the TelORB part of the platform.

The Front End (FE) of the ISUP Narrowband stack terminates the SS7 connections and performs link handling up to MTP-2. The Back End (BE) of the stack contains the MTP-3 and ISUP stack layers, and handles the interface with the application.

A feature of the ISUP Narrowband stack is the distributed and redundant nature of the FE. More than one FE can be used, each with its own hardware link termination, thus providing SS7 traffic handling even should a FE process or SS7 board fail. However, the BE will consist of a single instance in this release (also called a *monolithic* configuration).

The Front End (FE) processes comprise the following main components:

- Hardware Driver

The Hardware Driver enables the MTPL2 I/F to interface with the SS7 board.

- MTPL2 Interface (MTPL2 I/F)

The MTP level 2 I/F forms an interface layer between the FE I/F and the SS7 device driver. This interface layer is needed because the MTP level 2 protocol handler resides on the SS7 board as firmware.

- Distributor Front End Interface (FE I/F)

The Distributor FE I/F is a combined component. The distributor part contains a distribution algorithm to distribute traffic from the FE to the BE. Since the BE is monolithic for the moment, this distribution function is left as a stub.

The FE I/F part of this component handles the communication with the BE processes (via the BE I/F component).

The Back End (BE) process comprises the following main components:

- Back End Interface (BE I/F)

The Back End Interface handles the communication between the ISUP BE process and the one or more FE processes. Its communication counterpart in the FE is the Distributor FE I/F component.

- MTPL3

The Message Transfer Part (MTP) level 3 component fulfils the MTP-3's point-to-point message transfer functions of SS7. There are two variants – one supporting the ITU standard (corresponding ITU-T Q.701-Q.704, Q.706, Q.707, July 1996) and the Chinese standard (GF 001-9001, 10/94), and the other supporting the ANSI standard (ANSI T1.111, 1996).

- ISUP

The ISUP stack component layer fulfils the ISUP stack signaling function. Two variants of this component are supported on the platform. The ITU variant conforms to the ITU-T 1997 standard (12 September 1997). The ANSI variant conforms to the ANSI T1.113-1995 standard. Note that the same ISUP component is used in both the narrowband and broadband ISUP stacks.

ISUP SIGTRAN Stack

The ISUP SIGTRAN stack provides an ISUP Application Programming Interface (API) in the C++ language to an application running in the Linux part of the TSP platform. The diagram describes these components as:

- ISUP Application

This is the SS7 handling process belonging to the application that is using the ISUP SIGTRAN stack. It is bound to the stack via the ISUP API. The application is written in the C++ language.

- ISUP API

The ISUP API provides the interface by which the ISUP application can receive and send traffic through the stack. The API executes in the Linux environment, and is in the C++ language to enable it to be code-compiled with the application. Note that the API is also generic, and relies on the services of CP for its execution environment needs.

- SCTP Application

This is the application process that is using the SCTP stack. It is bound to the stack via the SCTP API. The application is written in the C++ language.

- SCTP API

The SCTP API provides the interface by which the SCTP application can receive and send traffic through the stack. The API executes in the Linux environment, and is in the C++ language to enable it to be code-compiled with the application. It should be noted that the current release of the API is not patterned after a Unix socket interface, but based on a proprietary design. This API is also generic, and relies on the services of CP for its execution environment needs.

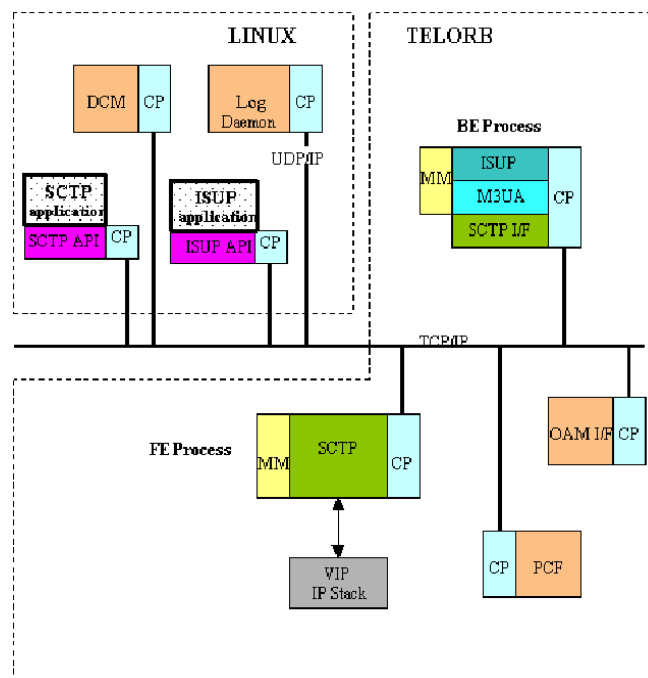


Figure 10-5 Architecture of ISUP SIGTRAN Stack

The stack itself will be executing on the TelORB part of the platform.

The SIGTRAN stack uses an IP-based network as the bearer channel, instead of an SS7 network as in the case of the ISUP Narrowband stack.

In the ISUP SIGTRAN stack configuration, it will also be possible to have an application use the SCTP stack, using a C++API in the Linux environment.

The FE of the stack terminates the IP connections and provides protocol handling for a SCTP stack.

The BE of the stack contains the M3UA adaptation layer and the ISUP stack layer, and handles the interface with the application.

Both the FE and BE execute as single instance in this release (also called a *monolithic* configuration).

The Front End (FE) processes comprise the following main components:

- VIP IP Stack

The Virtual IP, or VIP stack provides a raw IP socket interface upon which the SCTP stack is built.

- Sctp

The Sctp component is the main Sctp protocol stack running as a FE process. Its function is to handle the Sctp protocol and distribute data to the BE according to the associations that have been established. In the current monolithic configuration, all Sctp associations are distributed to the unique BE process.

The Back End (BE) process comprises the following main components:

- Sctp I/F

The Sctp I/F handles the communication between the ISUP protocol stack BE process and the Sctp FE process.

- M3UA

The MTP Level 3 User Adaptation component adapts the SS7 MTP Level 3 protocol to the IP-based bearer service. M3UA performs the translation between SS7 point codes and IP addresses. Towards higher-level protocols, M3UA behaves exactly like a normal MTPL3 stack. The M3UA supports two variants corresponding to the ITU (ITU-T Q.701-Q.704, Q.706, Q.707, July 1996) and ANSI standards (ANSI T1.111, 1996).

- ISUP

The ISUP stack component layer fulfills the ISUP stack signaling function. Two variants of this component are supported on the platform. The ITU variant conforms to the ITU-T 1997 standard (12 September 1997). The ANSI variant conforms to the ANSI T1.113-1995 standard. Note that the same ISUP component is used in both the ISUP SIGTRAN and ISUP narrowband stacks.

TCAP Stack

The TCAP stack provides a TCAP Application Programming Interface (API) in both the C++ and Java languages to an application running in the TelORB part of the TSP platform. The diagram describes these components as:

- TCAP API

The TCAP API provides the interface by which the TCAP application can receive and send traffic through the stack. The API executes in the TelORB environment, and is code-compiled with the application. Two versions of the API exist – in C++ and in Java. Note that the API is generic, and relies on the services of CP for its execution environment needs.

- TCAP Application

This is the SS7 handling process belonging to the application that is using the TCAP stack. It is bound to the stack via the TCAP API. The application is written in the C++ language or the Java language.

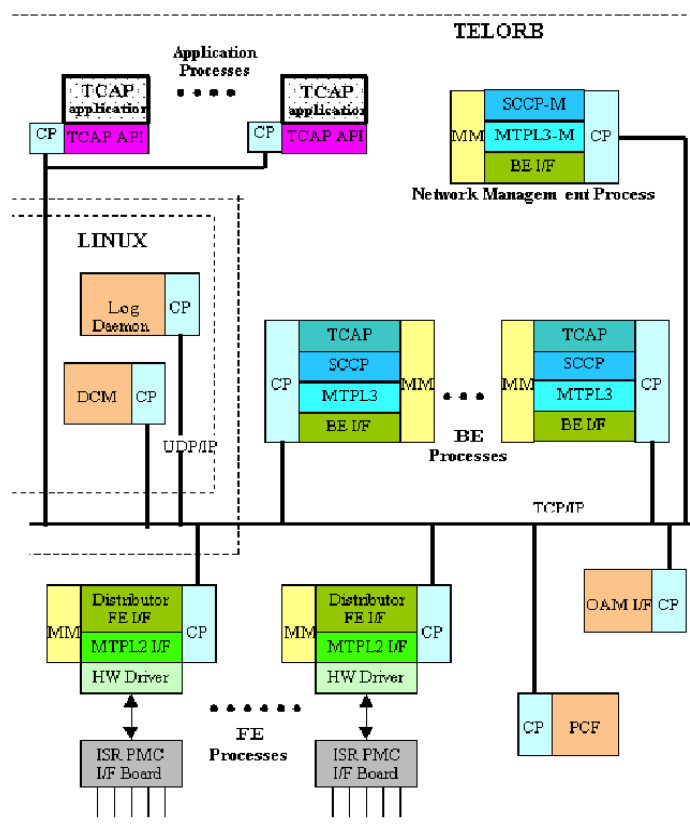


Figure 10-6 Architecture of TCAP Stack

The stack also executes on the TelORB part of the platform.

The FE of the stack terminates the SS7 connections and performs link handling up to MTP-2. The BE of the stack contains the MTP-3, SCCP, and TCAP stack layers, and handles the interface with the application.

A feature of the TCAP stack is the distributed and redundant nature of both the FE and the BE. More than one FE can be used, each with its own hardware link termination, thus providing SS7 traffic handling even should a FE process or SS7 board fail. Similarly a multitude of BEs can be used for redundancy and load sharing.

The Front End (FE) processes comprise the following main components:

- HW Driver
- MTPL2 I/F
- Distributor FE I/F

All of which are the same as in the *ISUP Narrowband stack* and described under this section.

The Back End (BE) process comprises the following main components:

The following two components are also the same as in the *ISUP Narrowband stack* and described under this section:

- BE I/F
- MTPL3

In addition the following components apply to the *TCAP Stack* architecture:

- SCCP

The SCCP stack component fulfils the SCCP signaling function. The ITU and Chinese variant conforms to the ITU-T 1997 standard (12 September 1997) and the Technical Specification of TCAP from the Chinese Directorate General of Telecommunication, Oct 1994. The ANSI variant conforms to the ANSI T1.111 – T1.112 1996 standard.

- TCAP

The TCAP stack component layer fulfils the TCAP stack signaling function. Two variants of this component are supported on the platform. The ITU and Chinese variant conforms to the ITU-T 1997 standard (12 September 1997) and the Technical Specification of TCAP from the Chinese Directorate General of Telecommunication, GF 001-9001, Oct 1994. The ANSI variant conforms to the ANSI T1.114-1996 standard.

- MTPL3-M

The MTP-3 Management module is part of the Network Management Module (NMM) of the BE. Its purpose is to coordinate MTP-3 network traffic management messages to/from the external network so that the stack appears externally as one stack (when in fact there may be several horizontally distributed stacks).

- SCCP-M

The Signaling Connection Control Part (SCCP) Management module is part of the Network Management Module (NMM) of the BE. Its purpose is to coordinate SCCP network traffic management messages to/from the external network so that the stack appears externally as one stack (when in fact there may be several horizontally distributed stacks).

SCTP Stack

The SCTP stack provides an SCTP Application Programming Interface (API) in the C++ language to an application running in the Linux part or TelORB part of the TSP platform.

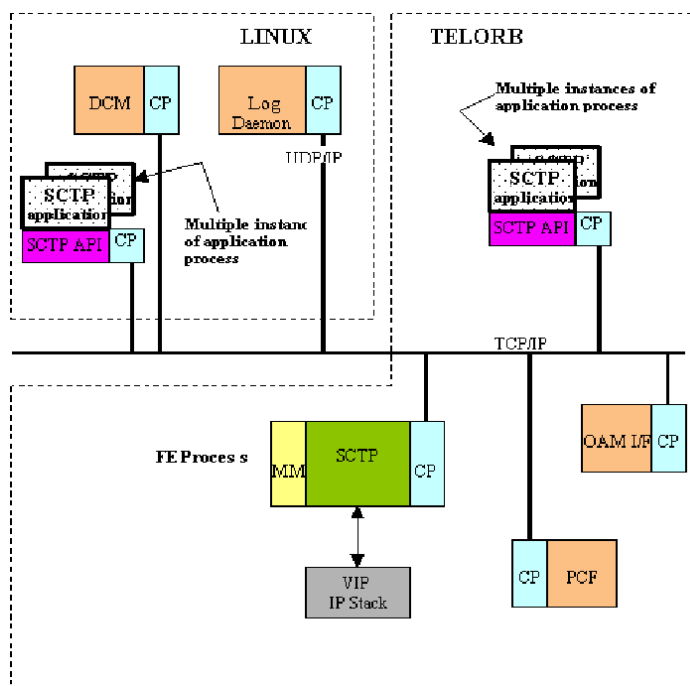


Figure 10-7 Architecture of Sctp Stack

In the current release of the Sctp stack, only the FE has been developed; the BE for the moment contains only the Sctp application bound together with the Sctp API. The FE of the stack terminates the IP connections and provides protocol handling for a Sctp stack. Messages corresponding to associations are then distributed to the application instance that owns the registered association. It is possible to have multiple instances of an application running concurrently (for the purpose of load sharing). However, all the instances must share the same association(s), and messages are forwarded to the multiple application instances in a round-robin manner.

SS7 PROVISIONING

The purpose of provisioning is to configure the TSP platform so that it can interact with the SS7 network. MTP data (signaling points, signaling links, link sets and route sets) and SCCP data (Global Titles and SSNs) are created for each of the nodes with which the TSP platform communicates.

The MTP and SCCP routing and administration functions use internal tables created by the provisioning tools. New nodes must be configured as they are added or changed.

MTP Provisioning

An SS7 network consists of connected **signaling points** (SPs).

A signaling point can be a physical node, but it is possible for a physical node to be defined as more than one signaling point. Each signaling point is identified by a unique code called the **signaling point code** (SPC).

Messages are transferred from one node in the network to another by means of **signaling links**. A signaling link connects two signaling points. For capacity and redundancy reasons, there are usually several links that connect two nodes. This is called a **link set**. If messages must be delivered in sequence, a specific link in the link set is used, otherwise load-sharing can be applied.

A message that is transferred from one node to another node “travels” via a series of link sets. The series of links is called a **route**. If there are several possible routes between two nodes, these routes constitute a **route set**. Each route in a route set has a **priority** associated to it. The priority is used in order to determine which route a message will use. If several routes have the same priority, load sharing will be applied.

The nodes that a message passes on its way to the destination node act as **signaling transfer points** (STP).

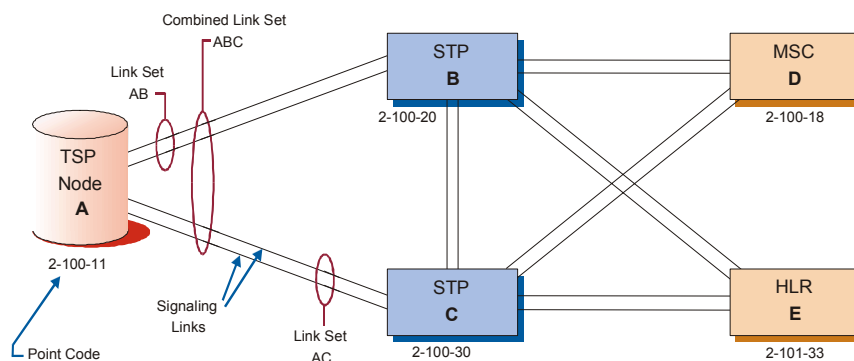


Figure 10-8 Example of ANSI SS7 Network Configuration

Signaling Point Code (SPC)

Each node defines its Own Point Code (OPC). For example, 2-100-11. This is used as the OPC in outbound messages. The SPC values in ANSI follow this format, SP=a-b-c, where

a = Network Identifier (ID) which identifies the network to which a point code belongs.

b = Network Cluster which identifies groups of signaling points and individual STPs of a signaling network.

c = Network Cluster Member which identifies individual signaling points within a cluster. The cluster member code 0 is reserved for addressing STPs.

Signaling Link (SL) and Link Set (LS)

MTP maintains a link table, containing data on each signaling link connected to the node. A signaling link is associated with a link set and is usually defined as a time slot (or channel) on a physical T1 or E1 link.

MTP maintains a link set table, containing the name and adjacent point code of every link set to which the node is connected. In the following diagram link sets AB and AC are defined and associated with the point code for nodes B and C, respectively. Node B defines link sets BA, BC, BD and BE.

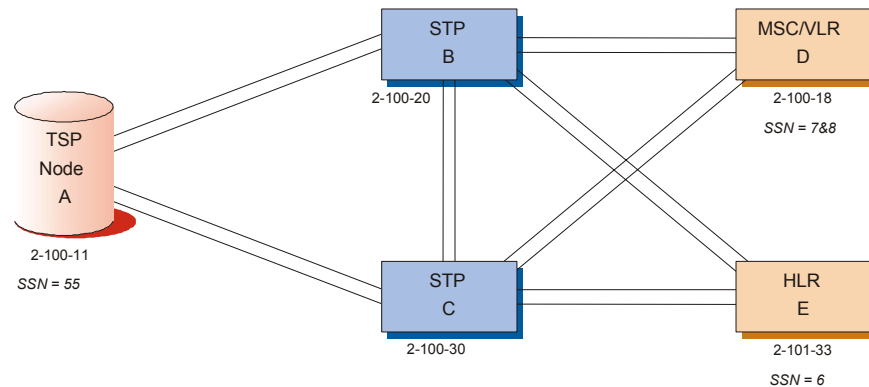


Figure 10-9 Types of links in an ANSI SS7 Network

Combined Link Set

ANSI SS7 specifies different types of links connecting SPs and STPs. There is no functional difference between the links. The type of link only indicates the type of traffic carried on the link. STPs are generally deployed in load-sharing mated pairs, with the traffic shared between each node.

End points are connected to the STP pair using *A links*. Each A link is defined as a link set and the pair of link sets containing the A link is called a Combined Link Set.

A direct link between SPs is an *F link*.

A direct link to a remote STP is an *E link* (these can be used if massive failures occur in the normal links).

STPs are connected together by *B* and *C links* only.

Route Sets

A route set is defined for every node with which the TSP platform communicates. The route set table contains:

- the Destination Point Code (DPC),
- the route set name and,
- all routes (link sets) which can lead to the destination.

In the previous figure a route set for nodes B, C, D and E is defined in node A.

SCCP Provisioning

The **sub-system number** (SSN) identifies a user of SCCP or TCAP within a node. The SPC in combination with the SSN can be used to route SCCP messages.

SCCP also offers the possibility to route on **GT** (Global Title).

Data to support the SCCP addressing needs to be defined. The Subsystem Number (SSN) of each local subsystem and the point code of each communicating node is defined. The remote SSNs must be defined and associated with the point codes of the communicating nodes. Subsystems are informed of the availability of other subsystems by SCCP management messages between the nodes.

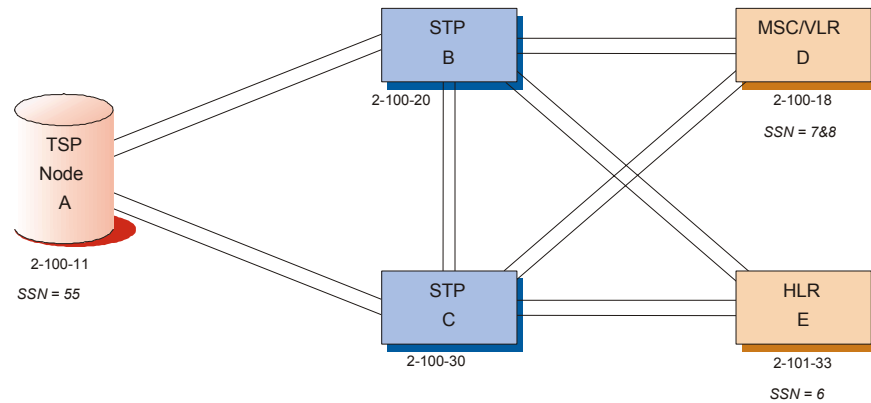


Figure 10-10 Example of SCCP network

Remote Subsystem Numbers

Applications in a node are identified by the subsystem number. Subsystem numbers are in the range of 2 to 254. For example:

SSN	SCCP User Function
5	MAP
6	HLR
7	VLR
8	GMSC or MSC
10	AUC
11	SMS
55	SCP
149	SGSN
150	GGSN

Table 1 SSN numbering

In the example network in the above figure Node D has MSC and VLR subsystems. If node A needs to know when these subsystems are accessible to node D, it defines the remote SSNs with the point code of node D. SCCP management messages inform node A when the MSC and VLR applications are accessible or inaccessible.

Concerned Point Code (CPC)

This is the point code of a node that communicates with a local subsystem. The local node must inform concerned nodes when subsystems are unavailable. For example, in the above figure, Node A defines Node D as a Concerned Point Code (CPC) for subsystem 55. Thus, Node A informs Node D whether or not subsystem 55 is accessible or inaccessible.

Global Title Translation (GTT)

Global Title Translation data can also be defined in the TSP platform. However, this type of routing is mostly used to route messages **to** the node rather than **from** the node. GTT will be used more frequently when international roaming becomes commonplace.

Provisioning Example – ITU-T SS7

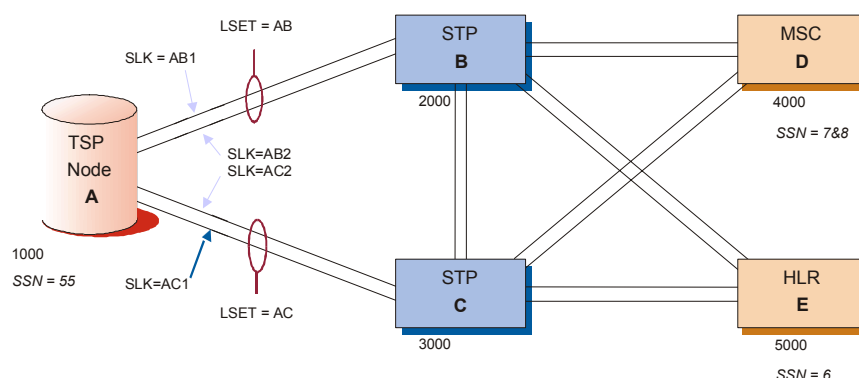


Figure 10-11 Example ITU-T SS7 Network

The procedure for configuring Node A in the SS7 example network above are shown in the next figure. This configuration is done by using the DCM GUI.

The following figure indicates the order in which provisioning is performed.

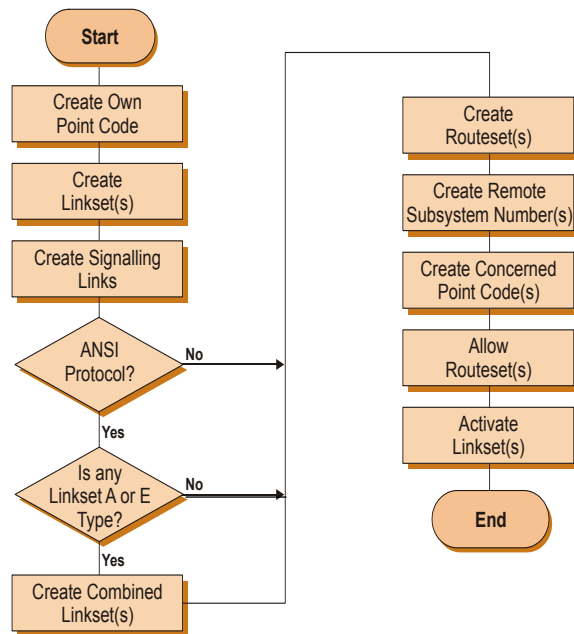


Figure 10-12 Order to ADD SS7 data

The sequence of operations to remove provisioning data is shown in the following figure.

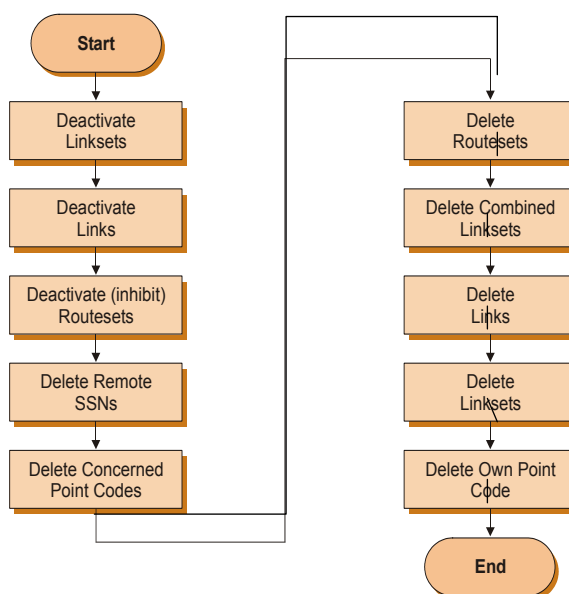


Figure 10-13 Order to REMOVE SS7 data

The Dynamic Configuration Manager (DCM)

The Dynamic Configuration Manager (DCM) is a tool providing a graphical user interface for the configuration and operation of Ericsson SS7 stacks.

The Dynamic Configuration Manager (DCM) facilitates the configuration of SS7 by displaying (and letting you edit) the stack and its components and their associations in a (mostly) hierarchical relationship. In addition, DCM contains functions that can check the logic of this structure for you while you work.

The Main DCM Window

The main DCM window is split into three parts:

- The *DCM Navigator* contains all elements that make up the currently opened SS7 stack configurations.
- The *DCM Desktop* is a workspace used for viewing and editing properties with an element's Property Sheet.
- The *Menu Bar* provides access to basic DCM functions, such as opening and saving configurations.

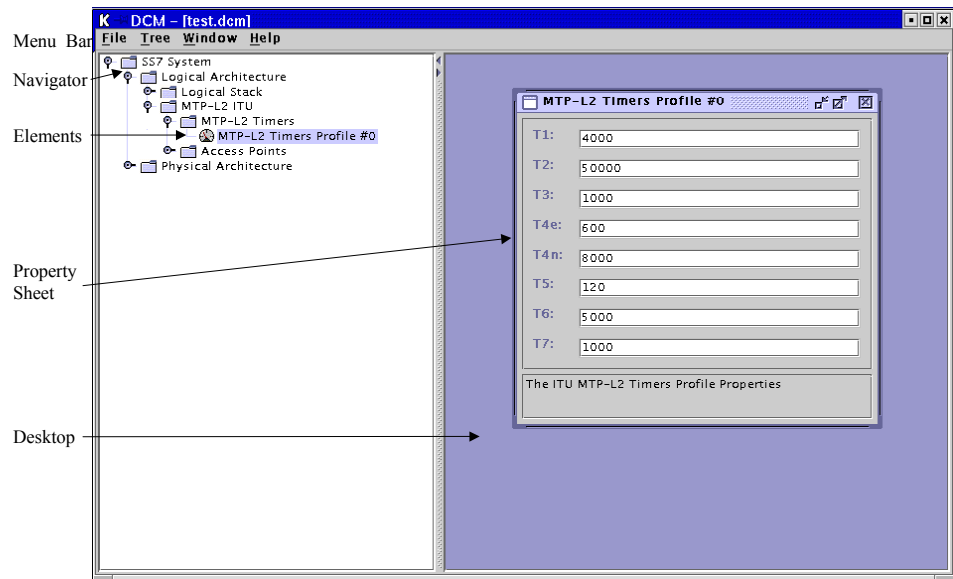


Figure 10-14 DCM Main Window

The left-hand side of the DCM main window contains the *DCM Navigator*. It uses a tree-like view to show the structure of the SS7 System currently loaded into DCM. The entire configuration of a stack is performed by using the Navigator to add, delete, and change the properties of the elements that make up the stack configuration.

The space on the right hand side of the DCM window is called the *DCM Desktop*. This is where you view and edit the properties of an Element shown in the Navigator tree using its Property Sheet. You can have several Property Sheets opened on the DCM Desktop at the same time. A Property Sheet displays all properties (configuration parameters) for a certain configuration Element. It is opened with the "Properties" option in an element's pop-up menu.

To close a Property Sheet, click the "close window" icon in the upper right corner, or use "Close all" in the "Window" menu. Changes will not be permanently saved until you use **Save** or **Save as...** in the **File** menu (but they will still show if you re-open the Property Sheet).

DCM Control Panel

The DCM Control Panel allows you to apply the configuration to an SS7 Stack, and also send other stack operation commands to the stack process(es).

To open the DCM Control Panel, Right-Click the Physical Architecture element in the DCM Navigator and select the Control panel... option from the Physical Stack Element.

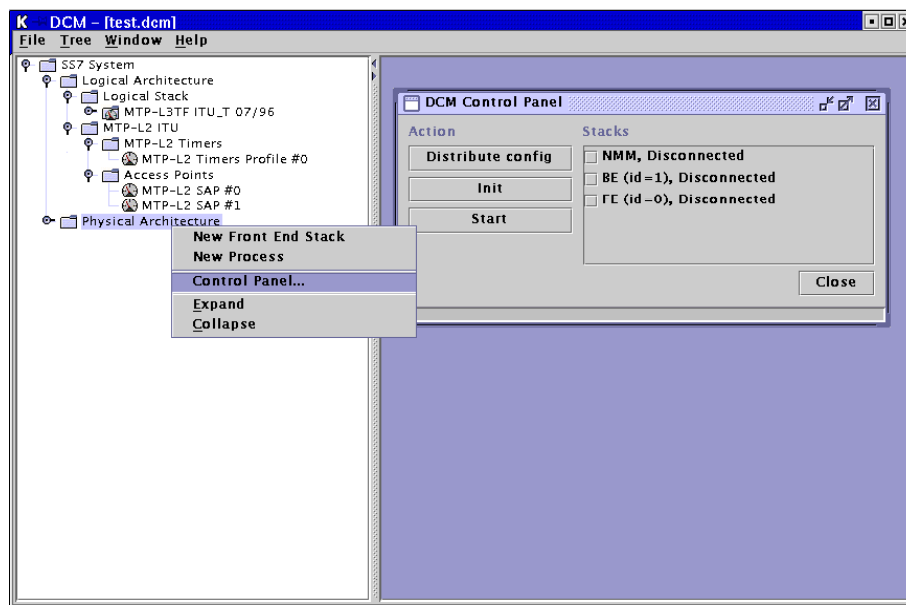


Figure 10-15 The DCM Control Panel

The DCM Control Panel allows you to apply the configuration to an SS7 Stack, and also send other stack operation commands to the stack process(es).

To open the DCM Control Panel, Right-Click the **Physical Architecture** element in the DCM Navigator and select the **Control panel...** option.

In the *Stacks* box, there is a list of connection points to the stack processes making up the complete SS7 stack.

The *Stacks* box contains a list of all defined Front End (FE)- and Back End (BE)-processes together with The NMM Process (Network Management Module, which is the part of an SS7 stack which handles all MTPL3 and SCCP management messages, such as link management and subsystem handling). In the SIGTRAN stack there is no NMM process.

The SS7 Stack process(es) must be in an "Idle" state before using the DCM Control Panel. The first state shown when opening the Control Panel – "Disconnected" – just indicates that DCM is currently not connected to the SS7 Stack process. When the connection is made, the display changes to show the actual stack status. For a newly started stack this should be "Idle".

To give a command to the stack, first select which stack process(es) you want give the next order to. To select a desired process(es), select the Checkbox next to the process in the list.

The buttons in the *Action* column shown on the left side of the Control Panel will always affect the currently selected stack processes in the *Stacks* list.

Distribute config button

Distribute config sends the current DCM stack configuration where a selected stack process can read it (when it gets an *Init* or *Reinit* command). This should be done each time the configuration is changed before *Init/Reinit* is pressed.

Init / Reinit button

- *Init* will make a stack process that is not yet in the "Running" state read the currently distributed configuration data.
- *Reinit* will make a running stack process read the currently distributed configuration data and apply changes to run-time re-configurable properties. If you have changed the configuration, do not forget to click the *Distribute config* button before reinitiating. If you don't do this, the stack process reads the previous distributed configuration data.

Note: Not all properties are re-configurable in run-time. For information about run-time re-configurable properties, see the DCM on-line help section for each SS7 Protocol Layer.

Start / Terminate button

- *Start* starts all the SS7 layers in the stack process. A successful start results in the "Running" state.
- *Terminate* terminates a running stack process. The only way to get the process running again is to restart the stack process itself (put it in "Idle" state), and then issue the "Init" and "Start" commands again. This may only be used to remove processes in horizontally distributed systems.

Order

Orders are used to change the behavior of the stack, get status information, and so on, while the stack is in running state.

1. Right-Click on an element in the tree structure to open the **pop-up menu**.
2. Choose Request Order.

The Request Order Dialog box will be displayed.

Note: Remember the process you want to send the Order too needs to be connected via the Control Panel.

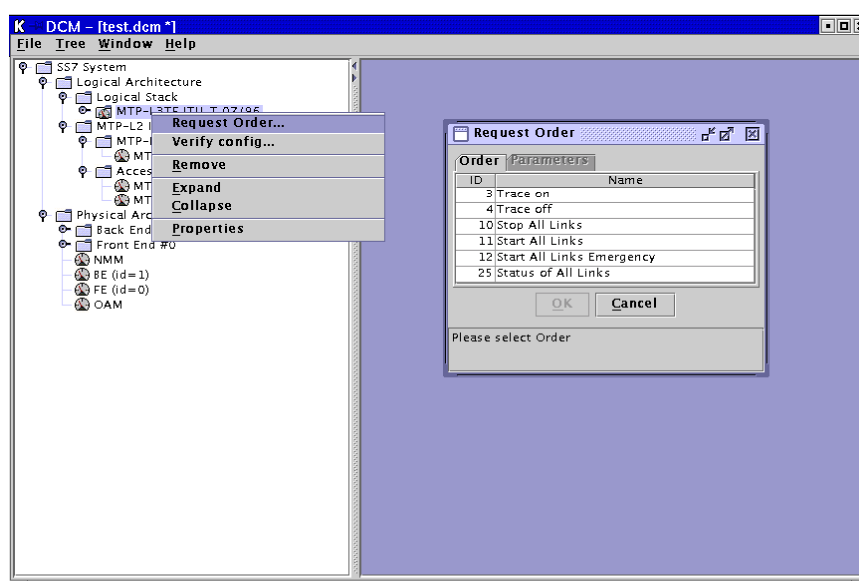


Figure 10-16 An example of a Request Order Dialog

3. Choose the **order** you which to perform.
4. If the order takes parameters the tab **Parameters** will be enabled. Click on this and enter the values.
5. Press **OK**.

When the order has been performed by the stack you will see the result in a **Order Result dialog**. For instance, if you select *Status of All Links* in the previous example, the order result will be displayed in the Order Result Dialog Box.

SUMMARY

- The TSP Platform supports both traditional Signaling System no.7 and several new IP-based network-signaling protocols. These include TCAP, ISUP Narrowband, SCTP and ISUP Broadband.
- The SS7 stack uses a 2-tier or *distributed* design. These 2 tiers are called the Front End (FE) and the Back End (BE). The FE of the stack terminates the SS7 or IP connections and performs protocol handling for an MTP or SCTP stack. The BE of the stack contains the upper protocol handling layers, and handles the interface with the application. The FE and BE are interconnected by an Ethernet LAN using TCP/IP.
- Configuration and operation of the Ericsson SS7 stack on the TSP Platform is performed via the DCM (Dynamic Configuration Manager) management GUI.

11 Virtual IP

CHAPTER OBJECTIVES

After completing this chapter you will be able to:

- Describe the Virtual IP function on the TSP platform
- Explain the distributed IP stack on the TSP platform
- Perform management functions of Virtual IP on the TSP platform via the Node Management interface

INTRODUCTION

Virtual IP (VIP) is a platform embedded function in TSP built for interfacing IP networks.

VIP simplifies the network building and lets the TSP platform, in an IP network, be addressed by a common single IP address representing the actual traffic node. Thus, a VIP address in this respect is similar to a connection point in SS7 networks. Furthermore, a TSP platform can have several VIP addresses. Up to 40 VIP addresses in a single TSP node is the limit, as set in the current TSP 4 release

VIP is an IP stack distributed among a number of processors that allows access to a cluster of processors via a single shared IP address. Therefore the entire cluster becomes addressable. VIP solves common load balancing and addressing problems in IP networks for applications that require both a high degree of reliability and agility. The VIP function also provides high fault tolerance of Internet access and scalability. Since the stack is distributed and replicated in a number of processors, the load of the system can be distributed more evenly.

It is important to note that VIP is a TSP concept and implementation and that it is not based on any IP address translation schemes similar to NAT⁵.

⁵ Network Address Translation (NAT), an Internet standard that enables a local-area network (LAN) to use one set of IP addresses for internal traffic and a second set of addresses for external traffic.

VIRTUAL IP (VIP) ARCHITECTURE

An external IP network is physically connected to a TSP node via an intermediary external gateway router.

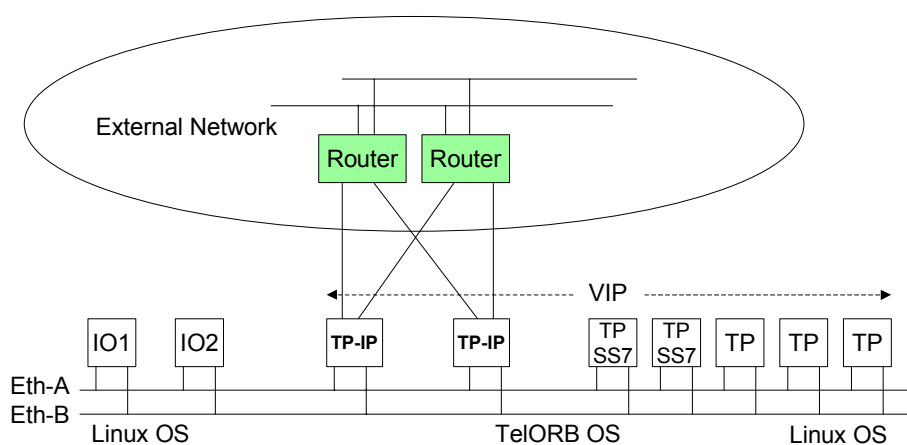


Figure 11-1 VIP physical interfaces

The gateway router is thereafter connected to the TSP node via communication enabled processors denoted PM-ETH boards.

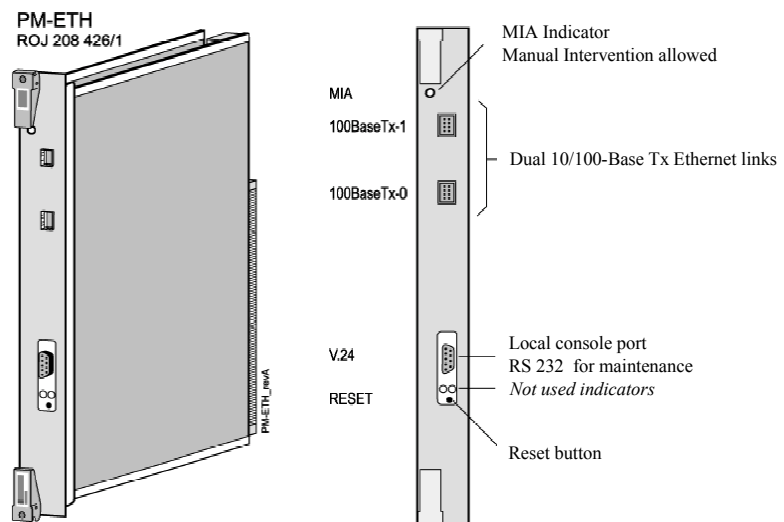


Figure 11-2 PM-ETH interface board

The PM-ETH boards serve as communication enabled front-end processors. PM-ETH boards are equipped with two front mounted soffix connectors, one for each network interface (Ethernet). With duplicated gateway routers each PM-ETH boards is connected to twin gateway routers.

The VIP architecture is based on a 3-level distribution model as follows:

- Gateway Distribution
- Front End Distribution
- Back End Distribution

Level 1 distribution is achieved by the external Gateway Router. **Levels 2 and 3** are system internal distribution stages, where level 3 is the innermost stage, distributing towards the socket holding processor.

Internal packet transport in the TSP node is done by an internal communication service that is part of the VIP implementation. This internal service is called Distributed Communication Function (DCF) and is carried by IPC. It has a particular name service and provides special services that are related to the VIP implementation.

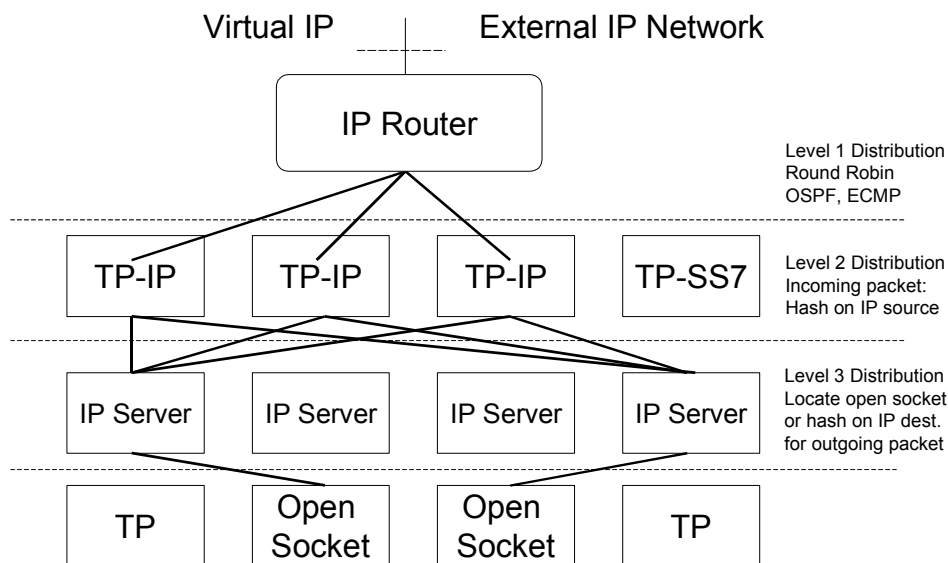


Figure 11-3 VIP 3-level distribution

LEVEL 1 – GATEWAY DISTRIBUTION

IP networks are built up with routers that have their own IP address. VIP, in ordinary network cases requires a front-end Gateway Router that is facing the external IP network on one side, and the TSP equipment on the other side. In TSP 4 this gateway router is assumed to be a commercial (third party) product that is located on-site but housed outside the proper TSP cabinets.

A TSP node is given a public⁶ IP address which is the VIP address. This VIP address is visible in the external network.

The gateway router is given a different IP address (or several). This IP address is also a public address that is visible in the external network but not a VIP address.

VIP addresses are always associated with a TSP node, but a TSP node may have several VIP addresses.

⁶ Reference IETF RFC 1918 "Address Allocation for Private Internets" - Technically VIP addresses can be private addresses but that requires a strict address plan in the external network to avoid that any IP address used internally in TSP is allocated in the external network. The internal address structure in TSP must be fully understood before other addresses than public IP addresses are used as VIP addresses.

OSPF is the routing protocol typically used between the external network and the interfacing gateway router.

The router is normally duplicated to achieve high availability. Each router has a unique public IP address. The two routers should be directly linked e.g. by a separate 100 or 1000 MB/s Ethernet link.

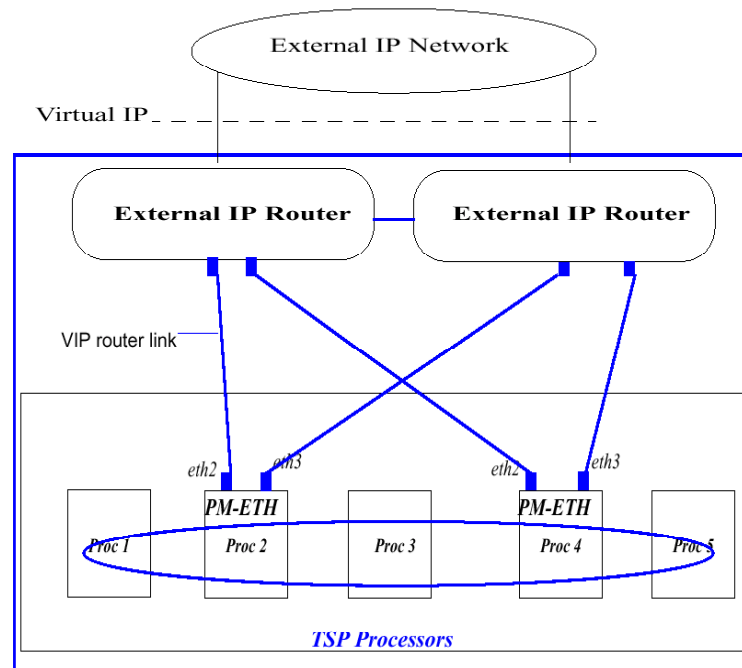


Figure 11-4 Duplicated routers are used to achieve high router availability

VIP Router Links

The VIP function communicates with the external IP network through a number of dedicated Ethernet links, so-called “VIP Router Links”. Each VIP router link exists in its own very small IP network. Each VIP router link “network” contains ONLY one port on the external router, and one port on the PM-ETH processor.

There should be at least 2 links allocated to the VIP function, in order to achieve redundancy. These two links MUST go to different TSP processors (i.e. one must have at least two PM-ETH's in a system to achieve redundancy).

The next figure shows a TSP system with 4 VIP router links, two towards processor 2 (*Proc 2*) and two towards processor 4 (*Proc 4*). It therefore has four small IP networks or IP subnets.

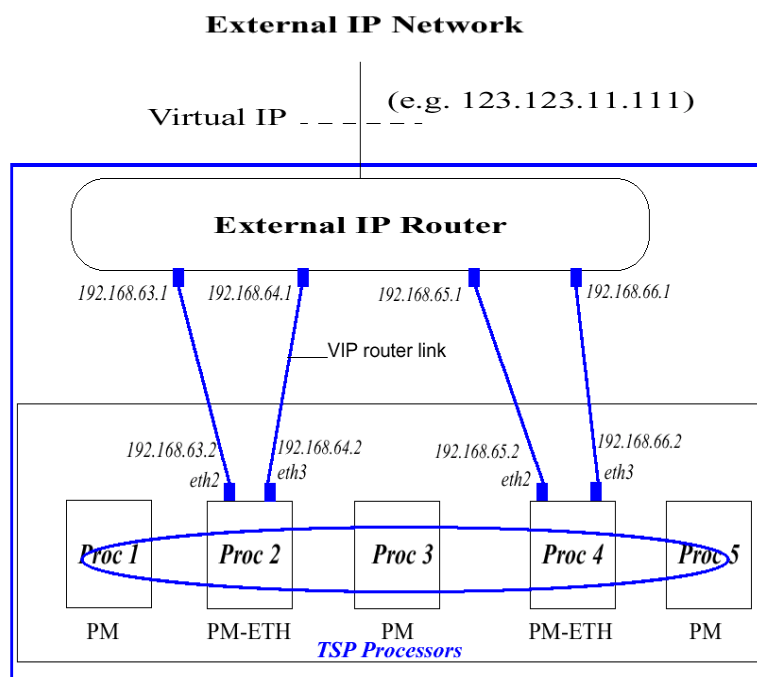


Figure 11-5 VIP Router Links

The external IP router is part of the external network domain, but also a part of the internal node domain with internal IP addresses. If you have two links per PM-ETH, i.e. two network interfaces (like in the figure), then these MUST reside in different networks.

In the example shown, one of the VIP router links has:

- VIP router link (router end) 192.168.63.1
- VIP router link (TSP end) 192.168.63.2

The VIP address of the TSP node in this example is 123.123.11.111.

All of these IP subnets have a netmask of 255.255.255.252. It is this that separates them into a series of mini IP networks, rather than sitting on the same network as it might appear.

The VIP router links are identified by:

- Processor name
- Logical name of Ethernet address of external interface on PM-ETH board, e.g. eth2

- The IP address of the external router end point.

The following example shows how a VIP router link can be defined:

Processor2, eth2, 192.168.63.2.

Distributor Device (DD)

Realized in the Device Distribution Layer, a Distributor Device is a logical abstraction of a collection of VIP Router Links, which explicitly is a group of network interfaces that are linked to ports in a gateway router. A VIP address is associated with a Distributor Device and thereby its network interfaces. For IP version 4, the Distributor Device is denoted "i4DD". Only interfaces "belonging to a VIP address" through their Distribution Device can use the VIP address.

The Device Distribution Layer is indeed distributed, it spans all processors in the TSP cluster. Part of the Device Distribution Layer is located on the front-end PM-ETHs which receives Ethernet frames from the gateway router, while part of the layer is located on the IP Servers, where segmented IP packets are put together.

Only "interfaces belonging to a VIP address" through their Distributor Devices can use the VIP address.

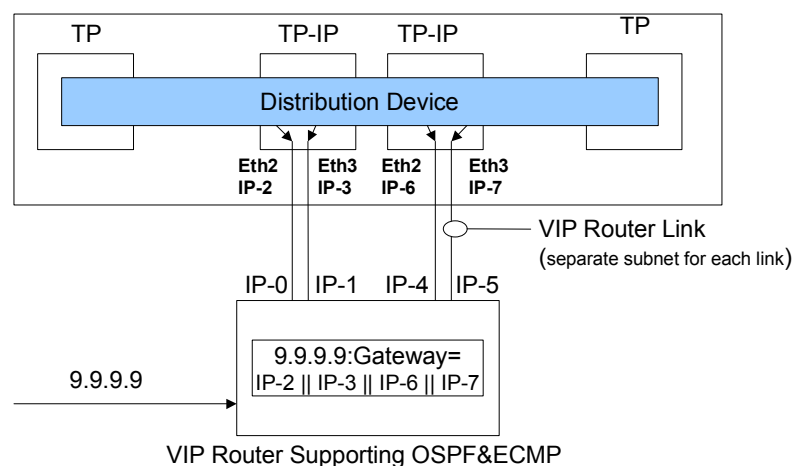


Figure 11-6 VIP Distributor Device (DD)

For Example in the figure, the Distributor Device can contain VIP Router Links with different processor numbers. By a configuration command a DD is associated to a VIP address. Packets entering or leaving the TSP node through the DD in this example, can thus pass through any of the four VIP Router Links.

Using ECMP with VIP

For the purpose of load balancing incoming IP traffic over several physical interfaces, the external routers serving as gateway routers, must on an individual packet basis, truly support a feature, known as Equal Cost Multi Path (ECMP). ECMP is used for spreading the incoming traffic to different PM-ETH boards.

Using OSPF with VIP

OSPF is used in the “External Router” in a VIP system to ensure that full redundancy of VIP router links is obtained. OSPF is an open protocol defined by IETF RFC 2328.

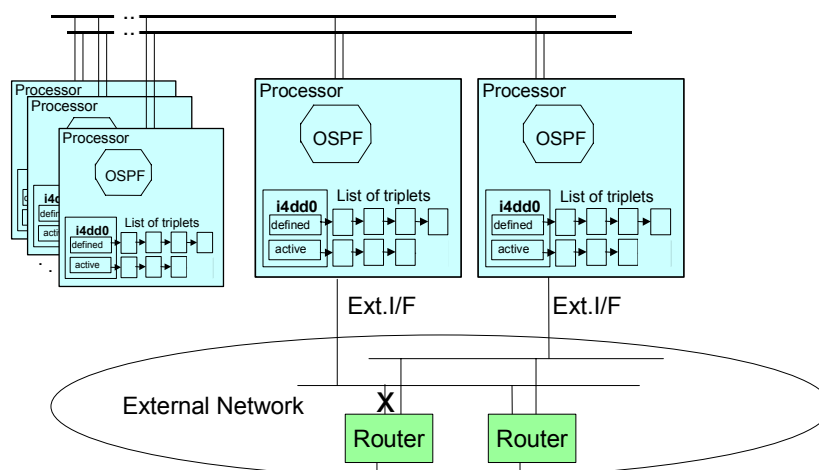


Figure 11-7 OSPF

OSPF takes care of link supervision. Both incoming and outgoing packets will be re-routed to alternative VIP Router Links in case of a link failure. The actual Ethernet environment between Gateway Routers and TSP node may differ from what is depicted above e.g. it may contain other additional equipment such as network switches. However, it is important that the Ethernet network is arranged so that no single point of failure in the additional equipment can be service stopping.

OSPF Configuration

OSPF uses the concept of an area which is a small routing domain. The OSPF area required by VIP is configured in the Gateway Router as an "OSPF stub area".

The OSPF stub area has an identity (ID), which is configured both in the TSP and in the Gateway Router. Configuration of the area ID in TSP is done by a GUI (or in the initial *epct* configuration file). It is common practice that OSPF area IDs are given in the familiar "IP address dot format" (a.b.c.d) though this is not strictly necessary. Any positive integer higher than zero is OK with TSP, this is in accordance with OSPF specifications. However, it is good to use the "IP dot format" as it is common practice and even the only possible way on many commercial routers. Moreover, it is customary in many networks to name the areas (OSPF area ID) by the same name (same value) as an existing IP address. So, the OSPF stub area required by VIP could e.g. be named after one of the private IP addresses of the external interfaces. But an area ID could in principal be identical to some fictitious IP address not used in the actual network.

(The value 0 (in dot notation 0.0.0.0) in TSP is a special case that would give each VIP router link an own OSPF area with an area ID equal to the link's own network id (private IP address of interface). It is an obsolete method not recommended in normal networks).

Routers configured with OSPF must have a priority related to the router defined. Most routers give a default priority 0 (zero). This must in our case be changed for the routers acting as Gateway Routers. The Gateway Routers MUST by configuration be given a priority higher than zero.

For the TSP side there is no need (and no option) for any configuration of router priority. It is automatically set to 0 (zero).

LEVEL 2 - FRONT END DISTRIBUTION

Once an incoming packet to the TSP node arrives on the PM-ETH boards, TSP's distributed IP stack will arrange the passage to its final destination within the node. Here it is expected to be handled by an application software unit through a socket API.

An outgoing packet originating in an application, through its protocol socket, will invoke the sending of an IP packet. The VIP and TSP internal machinery will carry the packet to a PM-ETH board. Further on, the Gateway Router will direct its onward journey.

All transportation and if needed segmentation or de-segmentation, is taken care of by "VIP and TSP internal machinery". It is the machinery of a distributed IP stack.

The DCF function and the underlying IPC mechanism take care of the internal transportation.

The Ethernet devices on the PM-ETH are location wise, within the cluster, de-coupled from the IP functions. Hence, packets arriving on different PM-ETH boards may be assembled (de-segmented) on yet another processor inside the cluster.

A function called IP Server is a key part of the VIP internal machinery. The IP Servers and the concept of a Distributor Device are central for realizing the location transparency provided in VIP.

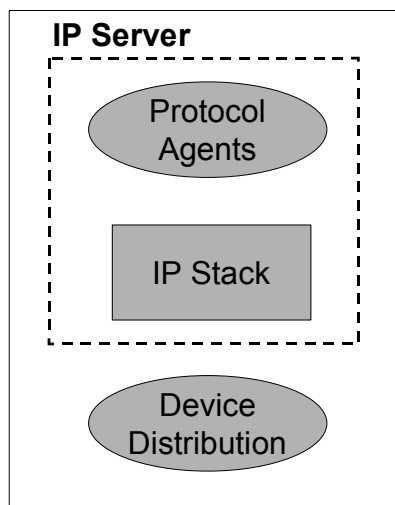


Figure 11-8 IP Server internals

In the figure, Protocol Units serving TCP, UDP, or other IP based protocols will through their Protocol Agents handle the interface towards the socket-end. The Device Distribution Layer takes care of interfacing towards the front-end PM-ETHs.

IP Server

In a TSP 4 system there are 255 IP Servers which are automatically distributed over all TelORB processors (and Linux, in a future software release). Hence, several IP Servers are allocated to a processor.

IP servers are used both for incoming and outgoing IP traffic. On the PM-ETH board, all incoming IP packet headers are examined, the source address is read, and hashed over the total number of IP Servers in the cluster, to point out an IP server in the node. For an outgoing IP packet, an IP server is instead pointed out by hashing on its destination address.

The IP server forwards complete IP packets to the application through it's socket, possibly after first assembling the packet from several segmented packets.

Device Distribution

A Distributor Device (DD) is a logical abstraction of a collection VIP Router Links, which is explained earlier in the section dealing with *Gateway Distribution*.

A DD is realized in the Device Distribution Layer which spans all processors in the TSP cluster. Part of the Device Layer Distribution is located on the front-end PM-ETHs which receives Ethernet frames from the gateway router, while part of the layer is located on the IP Servers, where segmented IP packets are put together.

LEVEL 3 - BACK END DISTRIBUTION

In the "socket end" a number of protocol handlers (socket users), spread out on several different processors would typically be listening to a certain port number (e.g. a TCP port). Thus, they are in this example all listening to the same particular port.

As incoming traffic to the particular port number arrives, the individual port instances will be connected, one by one in a random or round robin order, and the protocol handlers would go into action. In this way the traffic to a port number used for a particular application protocol, can be spread out to several different processors in the cluster, and traffic load can thus be shared.

To realize the distribution transparency towards the "socket end", each individual IP Server keeps track of all it's own open sockets in hash tables. In the case of TCP it is a connection list. While for "connection less" UDP, tables with a similar set of address and port associations is used. These tables can be used to locate the appropriate socket for an incoming packet.

Conversely, in the opposite direction, on each processor with open sockets, tables are kept relating the open sockets to a specific instance of an IP Server.

	Source		Destination	
	IP Addr	Port No	IP Addr	Port No
TCP connections				
UDP connections				

Figure 11-9 IP Server Hash Tables

The high availability functions of TSP protects VIP traffic against cases of a processor outage. In the event of a processor failure, IP Servers on the faulty processor would automatically be transferred to other working processors and their TCP connection tables would virtually survive as they are reconstituted on the new processors. However, UDP associations would be left to expire.

Socket Groups

With VIP, the socket user discrimination is done by configuring Socket Groups. Before explaining the concept of a “Socket Group” let us first establish what is here meant by a “socket”. Socket, is a widely used term in general IP network programming (e.g. on Unix). For programmers it is an abstract device for communication, visible in the API. The metaphor is taken from the “familiar” world of telephony. Ordinary “telephone sockets” on the wall, enables subscribers to receive telephone calls whence called upon. But for this to work the socket needs to be bound (connected by wire) to a port (line circuit) in the telephone exchange, and of course an address (telephone number) must have been assigned to the port in question.

This is a world well known to all IP network programmers. Now, to this VIP brings the additional concept of a “Socket Group” which also have an equivalent in the world of telephony. It is similar to a feature (common in many offices and work places) called a “Group Hunting Group”. Which means a group of telephones that can be called upon by a common group number and to which calls are distributed according to some defined scheme (round robin, random etc.).

With VIP, IP packets can from the IP Server towards the socket, be distributed to sockets on different processors, which belong to the same Socket Group.

A remarkable feature in VIP is that within a Socket Group, the socket API allows sockets to be moved from one processor to another while in service. It is an option available to applications requiring smooth, programmer controlled in-service re-configuration, or it can be used in conjunction with in-service upgrading or other “session preserving” activities.

A Socket Group in TSP 4 is specified by the following quadruplet:

- IP type
- Protocol type
- Port number
- Distribution scheme

The only currently supported IP type is IPv4, but IPv6 is planned to be included later. Protocol types TCP, UDP and IP are today supported but will in the future be expanded with other IP based protocols. Port number is the port number used by a specific application’s socket.

Moreover, to support multiple VIP addresses towards the same port, provisions have been made for an “aliasing technique” that allow applications to bind to particular combinations of VIP addresses and sockets of a Socket Group. All this will be explained later in this chapter.

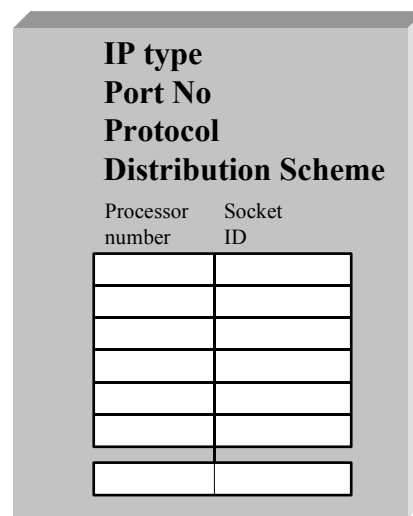


Figure 11-10 Socket Group listed as Processor Number and Socket ID

It is the IP Server that keeps track of the Socket Groups and their open sockets. This data is kept in hash tables for fast response.

The UDP associations are time limited. Time limitation is based on fixed, one minute leasing periods that are prolonged automatically as long as UDP packets keep arriving to the particular association, within a non-expired leasing period.

Distribution Schemes

A variety of distribution schemes, in the socket-end, are offered to applications in TSP 4. The actual selection of scheme is done through the VIP configuration GUI.

We are here mentioning only the most important schemes.

- RandomDistributor
- RoundRobinDistributor
- PreferLocalUdpDistributor; PreferLocalTcpDistributor

Random and round robin refers to the distribution over the Socket Group. For UDP packets, the distribution is always "Sticky". The term Sticky refers to sticking to the same socket within a Socket Group.

Some distributors are of the type "prefer local". The term "prefer local" refers to a first choice of a socket on the same processor as the IP server, and if several sockets exist a sticky choice is made within the sub-group.

Furthermore, it is possible for application designers to tailor their own specific distribution schemes, by implementing a specialized "driver" for the desired distribution algorithm.

Aliasing

As stated before, a TSP node can have several VIP addresses. Moreover, it is possible for a programmer, through the TSP socket API to bind individual sockets in a Socket Group, (or all sockets belonging to a Socket Group for that matter), to a particular combination of port number and VIP address. The binding is done by means of a function call "bind" provided in the socket API.

Aliasing in VIP gives application builders a unique possibility of using the full power of distributed processing combined with application co-location and port naming flexibility.

SPECIAL CASES OF VIP

USING VIP INTERNALLY INSIDE A TSP CLUSTER

Applications can also use VIP internally in a TSP node. A case that would suggest this type of design solution, is when parts of an application are designed for possible external relocation to non-TSP computers, e.g. partially using other platforms outside the TSP node. A similar case appears when co-locating different independent applications within a TSP node. A potential advantage with VIP is that the applications would be better prepared for node-wise separation. However, capacity considerations should always be taken into account before opting for node internal TCP/IP solutions. TSP's own communication techniques e.g. Dialogues (IPC) are normally superior.

VIP ON LINUX

Full implementation of VIP together with Linux is planned.
However, in the first TSP 4 partial releases the PM-ETH boards
interfacing the gateway routers will require TelORB/Dicos as OS.

MANAGING VIRTUAL IP

In order to incorporate the VIP function onto the TSP node installation the site configuration should include the following IDPs

- IDP/D32
- IDP/ D35,

And, the following if the site contains Linux processors

- IDP/D48
- IDP/D49

These can also be applied as an upgrade.

TelORB Manager contains the following tab, *Virtual IP*, in order to manage the VIP function.

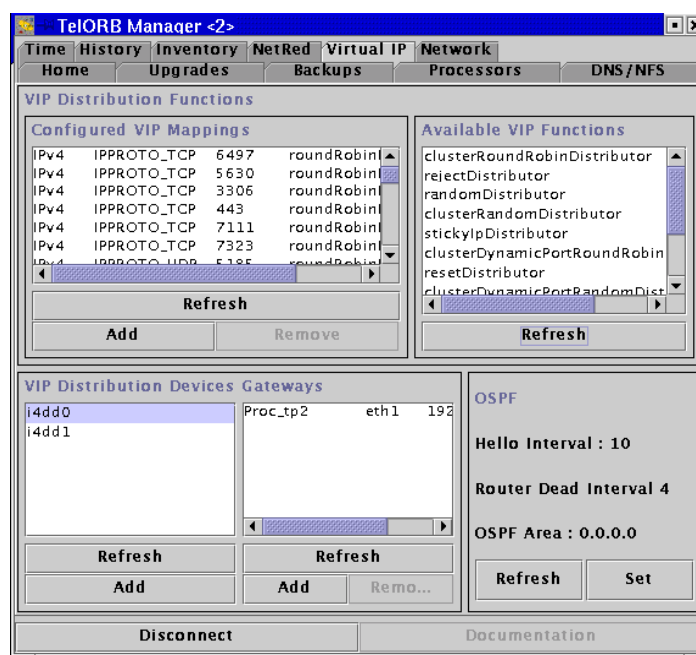


Figure 11-11 TelORB Manager VIP tab

This card is used to configure the Virtual IP. It is split up into three sections, “VIP Distribution Functions” on the upper half of the screen, “VIP Distribution Devices” and “OSPF” on the lower half.

VIP DISTRIBUTION FUNCTIONS

The VIP Distribution Functions section shows on the right-hand side, a list of the available Distribution functions that have been loaded into TelORB, and on the left-hand side a list of what Distribution function mappings have been configured.

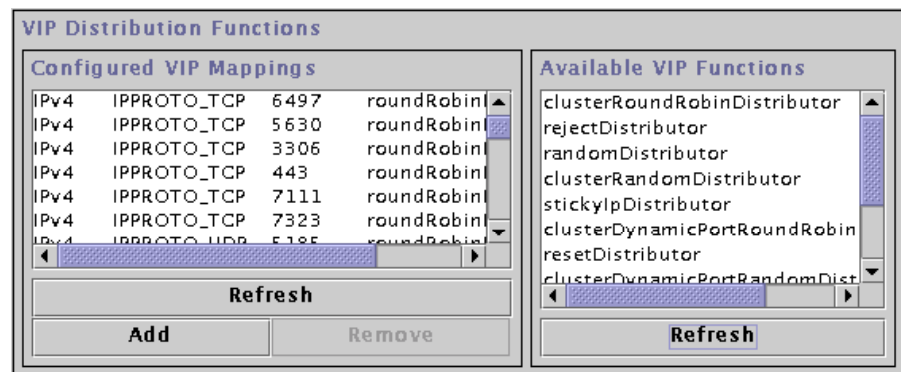


Figure 11-12 VIP Distribution Functions

The Distribution functions or algorithms can be used when configuring mappings of Distribution functions to port numbers.

The list of configured Distribution function mappings shows:

- The protocol configured (e.g. IPv4 or other protocol)
- TCP or UDP
- Port number
- Distribution Function to be used

When adding a distribution mapping, a dialog box will pop up, requesting a number of parameters for the new Distribution Mapping.

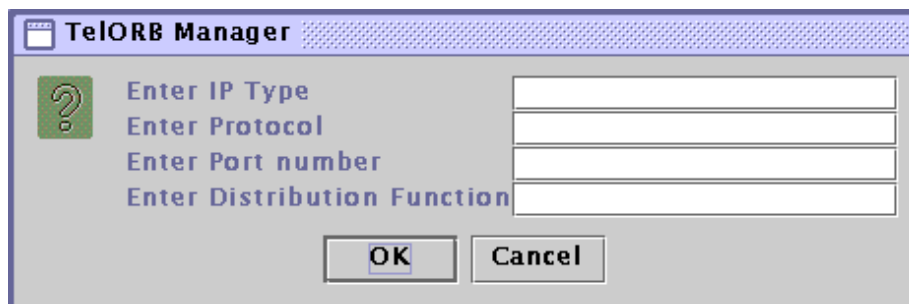


Figure 11-13 Adding a Distribution Mapping

These should be entered as following:

- The IP type (should be IPv4 for IP version 4)
- The protocol (Can be TCP or UDP or 0 for all protocols)
- The port number to be mapped
- The distribution function from the list of available functions.

Note: Distribution mappings may also be configured at site configuration by means of the 'Initial Data' settings in the epct file.

VIP DISTRIBUTION DEVICES

The VIP Distribution Devices section shows a list of "VIP Distribution Devices". These are effectively IP devices that are implemented using the patented VIP scalable IP technology. Each VIP distribution device is defined with a device name (normally starting with i4dd) and has a number of defined gateways which are the physical inputs and outputs from the cluster which will be used for IP traffic to and from these devices.

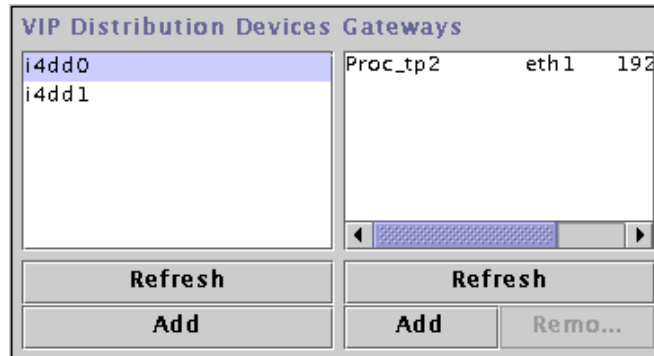


Figure 11-14 VIP Distribution Devices

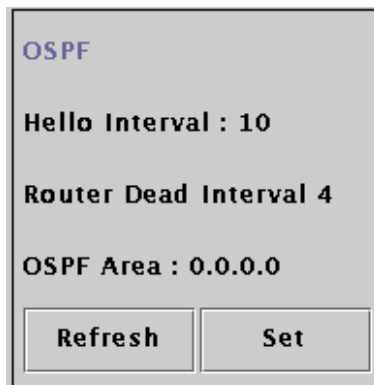
By clicking on an existing Distribution Device, the gateways for that device are shown in the list on the right hand side.

Each gateway contains the name of a TelORB processor on which the gateway is placed, the name of the Ethernet device through which the IP traffic should travel and the IP address of the external gateway to which it should be directed.

OSPF

The OSPF section shows a list of the parameters relating to the OSPF

- Hello interval,
- Router Dead Interval and,
- OSPF Area.

A screenshot of a configuration window titled "OSPF". The window has a light gray background and a thin black border. Inside, the text "OSPF" is at the top left. Below it, the text "Hello Interval : 10" is displayed. Further down, "Router Dead Interval 4" is shown. At the bottom of the text area, "OSPF Area : 0.0.0.0" is displayed. At the very bottom of the window, there are two buttons: "Refresh" on the left and "Set" on the right, both with black text and light gray backgrounds.

OSPF

Hello Interval : 10

Router Dead Interval 4

OSPF Area : 0.0.0.0

Refresh Set

Figure 11-15 OSPF

The only configurable OSPF parameters in TelORB Manager *are Hello Interval, Router Dead Interval and Area Identifier*. These parameters must have the same values as in the external router.

- *Hello Interval* should be specified to any value greater than or equal to 3 seconds. TelORB has a default value of 10 seconds.
- Number of Hello Intervals for *Router Dead Interval* should be specified to any value greater than or equal to twice the hello interval. Default in TelORB is four times the Hello Interval.
- *Area Identifier* should be specified as number > 0 or as an internet address (a.b.c.d). The value 0 (0.0.0.0) is special. If specified, the used area id will be equal to the network id for respective VIP router link.

SUMMARY

- Virtual IP is a TSP Platform function that simplifies the network building by allowing the TSP Platform to be addressed by a common single IP address representing the actual traffic node.
- Virtual IP architecture is based on a 3-level distribution model:
- Gateway Distribution achieved by the External Gateway Router
- Front-End Distribution and Back-End distribution which are system internal distribution stages performed by the Traffic Processors.
- Management of the Virtual IP function is performed via the *TelORB Manager*.

12 Performance Management

CHAPTER OBJECTIVES

After this chapter the student will be able to:

- Describe the function of Performance Management on the TSP platform
- Explain the Performance Management Framework (PMF)
- Manage the TSP Platform Measurements
- Configure and analyze Performance Management data via the Alarm Interfaces, LDAP Browser and the Performance Management Dashboard (PMD)

PERFORMANCE MANAGEMENT

Introduction

Performance Management provides the functions needed for reporting statistical measurements for analysis and tuning of the network element. It has functions for collecting performance data and a management interface to the external operators.

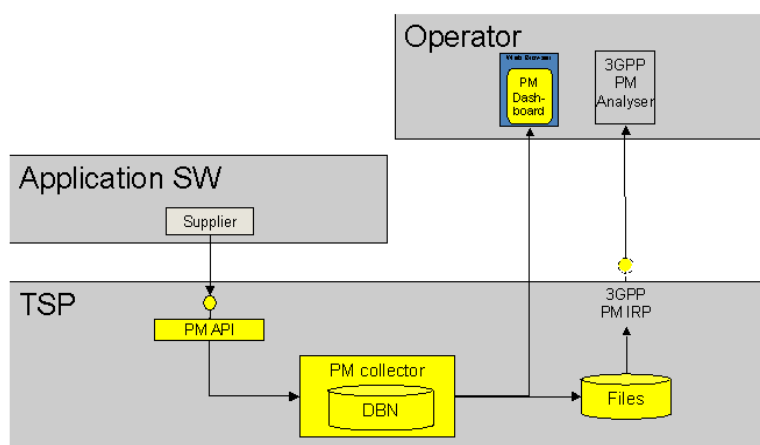


Figure 12-1 Overview of Performance Management

Applications (as well as platform components) that are to provide statistical measurements, reports their values via the reporting interface to the Performance Management Framework (PMF) of the platform. TSP provides both C+ and Java interfaces on both Dicos and Linux. PMF collects the values and stores them in the database.

The supported measurement types are:

- **Counters.** A counter is a measurement type that is used to report cumulative, incremental integer variables. The counter is incremented by the occurrence of an event.

- **Gauges.** A gauge represents a dynamic variable that may change in either direction. A gauge is real valued. Within a granularity period, a gauge maintains the minimum value, the maximum value and the mean (average) value. The current value of a gauge always represents the average of all the previous values for a given scan period.
- **Status Inspection.** Status inspection is a mechanism for high frequency sampling of internal counters at pre-defined rates.

At each reporting occasion PMF will collect all values from the database and create a snapshot of the measurement data and store this in a measurement file in the file system in XML format.

The platform can be configured to automatically generate threshold-based alarms. If a measurement value exceeds a configurable threshold value, then a (configurable) alarm will be raised, and if, for a subsequent measurement, the value goes below another threshold value, then the alarm will be cleared.

The PMF supports the reporting and collection of statistical measurement from any application designed upon the PMF service development kit. The TSP Platform also provides measurements to PMF for critical system resources. The applications reporting measurement values are as follows:

These can be grouped into:

- **TSP or Platform statistics.** Supports statistical measurements for both TelORB native and Linux processes on every processor within the cabinet. These measurements include *CPU load*, *memory usage* and *Cluster Uptime*. The measurements can be on each processor or on the total system.
- **Node Management provisioning statistics.** The O&M Provisioning counters measure the total provisioning transactions per second (TPS) in the TSP. The number of incoming LDAP and SNMP transactions are counted for this interval and the number is converted to TPS.
- **Signaling statistics.** Signaling measurements report data gathered from the EIN Stack (e.g. SS7 throughput, components sent and received at TCAP level). They are available in two types, ITU and ANSI.
- **Redundancy statistics.** This data is available through the notification handler of TelORB, for example, *Date and Time of Last Backup*.

The platform provides PM files in XML file format according to:

- The 3GPP PM IRP XML SS (3GPP TS 32.104 v3.4.0)

These files can be fetched or pushed from the node to the support systems using secure ftp

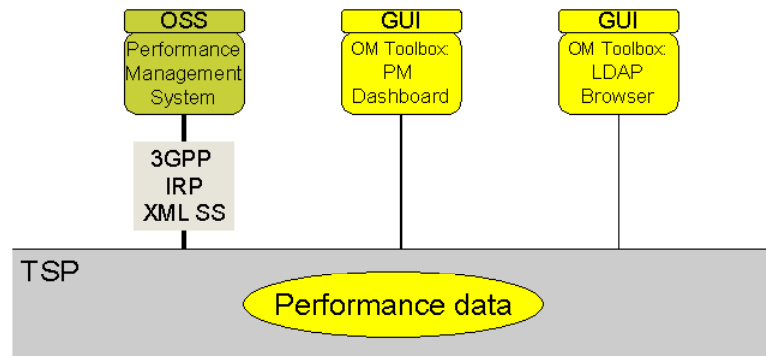


Figure 12-2 External Interfaces and Protocols for Performance Management

The PM function uses the following Operations & Maintenance components:

- Fault Management - for reporting of threshold based alarms.
- Configuration Management - the configuration of measurements is done by means of the LDAP browser through the JIM interface
- Logging Support – uses the Network File System (NFS) to log scan reports as XML files.
- Performance Management Dashboard (PMD) – For element management purposes the platform provides the Performance Management Dashboard that is an application of the OM Toolbox. The PM dashboard can be used to view subsets of the counters, gauges and status inspections that the platform provides.

PmSimple

The PmSimple is a module that is created as part of an instance of a client application process. It is responsible for collecting the various measurements from the client application terminates, it must explicitly request the PmSimple to send all the current scan data.

The PmSimple forwards the scan data to a PmIntra running on the same processor.

PmIntra

The role of PmIntra is to collect scan data from all the PmSimples in the same processor and to forward the data to the PmServer at the end of its scan period.

PmServer

The PmServer is responsible for accepting scan data from all PmIntras and persistently storing them after being aggregated. It is also responsible for pulling the scan data from the TelORB database when requested by the PmReporter.

PmReporter

The PmReporter reports the scan data using the Network File System (NFS) at the end of its scan period. The scan data is reported in a XML format. The PmReporter scanning period is configurable.

PMC – PmConfigurator

The PMS is responsible for retrieving PMF configuration data from the configuration manager through JIM. It is also responsible for registering with the notification service, consequently it is inform when any of the PMF configuration parameters have changed. Once it receives a notification, it informs the appropriate PMF modules of the change.

CONFIGURABLE ATTRIBUTES

Configuration data is stored in the JIM (Jambala Information Manager⁷) managed object hierarchy. The PMF application serves the purpose of being the root of the sub-hierarchy of objects that belong to the application. An external operator can configure PMF parameters using the TSP configuration manager.

This section describes the PMF configuration attributes together with the Base Distinguished Name (DN) form which they can be accessed using an LDAP browser. The Base DN represents the root node object and the Performance Management (application) object.

The Base DN for all PMF applications is as follows:

`applicationName=PmApplication,nodeName=jambala`

Under the PM application, there are three application MO classes, each with specific configurable attributes:

- PmConfig class identified by **PmScanName**
- PmScanConfig identified by **PmScanKey**
- PmMeasureConfig identified by **PmMeasuresName**

These three templates contain a preset list of attributes, which are applicable to each object. These PmApplication templates are included with the delivered Jambala Information Manager (JIM) and the LDAP Browser.

PmScanName attributes

The scan type attributes define the top-level attributes for a given PMF application. Assuming that the PMF application is identified with <scanName>, the DN used to access these attributes is as follows:

`PmScanName=<scanName>,applicationName=PmApplication,nodeName=jambala`

⁷ JIM (Jambala Information Manager) is the generic provisioning framework for the TSP.

PERFORMANCE MANAGEMENT DASHBOARD

TSP Node Management provides the Performance Management dashboard (PMD) for on-line inspection of performance management data. The PMD is a troubleshooting tool available from the Node Management Toolbox.

The PMD can be used to view data from any application, which uses the Performance Management Framework (PMF) to report its measurement values.

The PMD registers with each TSP application from which data is required for display. At registration time, the PMD application reads the maximum and minimum value of each gauge and status inspection. It also reads the unit type of each gauge, counter and status inspection.

The PMD collects data from the PMF at the end of the sampling period. This information is stored and sorted. Then the available measurements appear within the PMD where they can be selected for display. Measurements display their previous values and remain unelectable until they are updated at the end of their respective sampling period.

Note: The sampling period supported by the PMF is configurable in the PMF and available in increments of 5, 10, 20, 30 and 60 seconds.

Starting the PMD

1. Open a web-browser window and type the URL of the Node Management interface. The **Node Management Toolbox login** window appears.
2. Login with a valid Node Management username and password combination. Upon successful login authentication, you are presented with a Toolbox menu applicable to their authorization level.

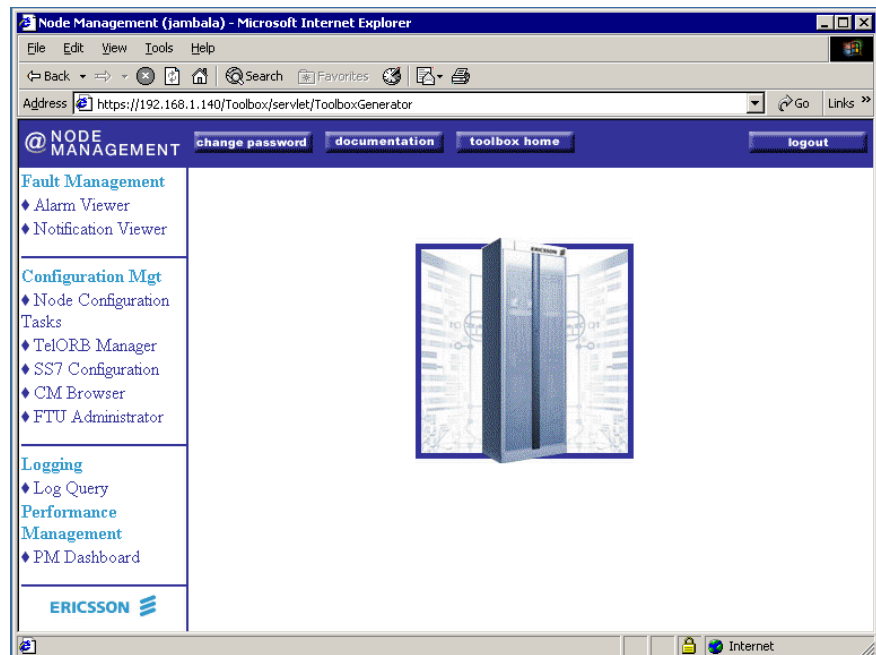


Figure 12-4 Node Management Window

3. Under *Performance Management*, click **PM Dashboard**. The PMD work area appears in the right browser pane.

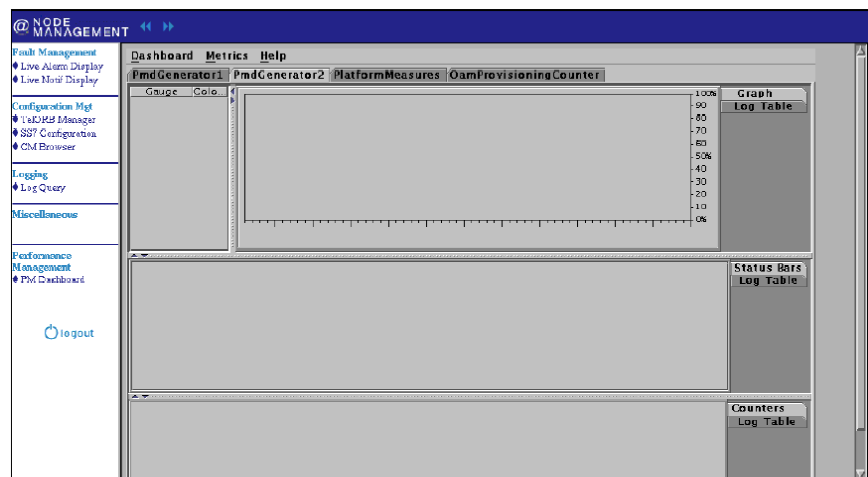


Figure 12-5 PM Dashboard work area

Using the PMD

When you open a PMD, it consists of

- the menu bar,
- the available platform measurements and,
- three visual display areas for each of the possible measurements types.

The **menu bar** contains three menus:

- the **D**ashboard commands allows the user to exit or to reconnect to the PMD Server.
- the **M**etrics command allows the user to view and configure each available measurement.
- The **H**elp command gives the user access to the latest on-line help for the PMD.

The **platform measurements** are displayed below the menu bar as labeled groups. A separate tab is created for each platform measurement and the user can access different applications by selecting the appropriate tab. The **Metrics** menu allow the user to configure the available measurements for each application for viewing.

There can be a waiting period before measurements have been sampled by the PMF and reported to the PMD. In this case, the selected measurement is not available until the values are received by the PMD and a warning message is displayed to the user.

The **visual displays** show measurement entries, which have been selected in the **Metrics** menu. In the case, where none of the selected measurements are of a certain measurement type the display will be empty. The user can resize and adjust the individual display panes.

There are three visual displays available within the PMD window: Gauge, Status Inspection and Counter.

In addition to the default display for each type of measurement, the user can select a log table to view the data. The log table presents a historical record of the data received over a period of time. It consists of the timestamp of the sample and its value.

TSP (or Platform) Measurements

The PMD displays data for each TSP Measurement that is registered with the PMF. The PMF sends data to the PMD, which sends the measurements to the PMD interface. During the initialization and start up of the PMD, a unique tag is created for the each TSP Measurement. The user can view the available measurements by selecting the appropriate tab.



Figure 12-6 Available TSP Measurement Tabs

Initially, when the PMD starts, none of the measurements are enabled and available to the user. Measurements are only available for selection once a value has been sent from the PMF. There can be a waiting period before they have been sampled by the PMF application reported to the PMD.

The measurements units available from the PMF are percentage, kilobytes, and transactions per second (TPS).

Setting and Displaying measurement Types

1. In the PMD window, click the Application Tab you want to view.
2. On the Matrix menu, click **Gauges**, **Status Inspections** or **Counters** as required. The Metrics menu allows the user to view and configure each measurement.

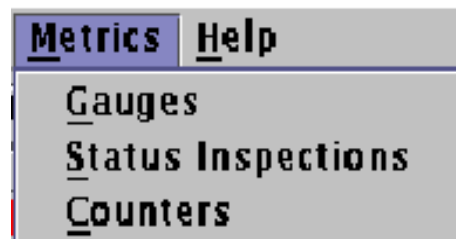


Figure 12-7 Metrics Menu

The appropriate check box appears.

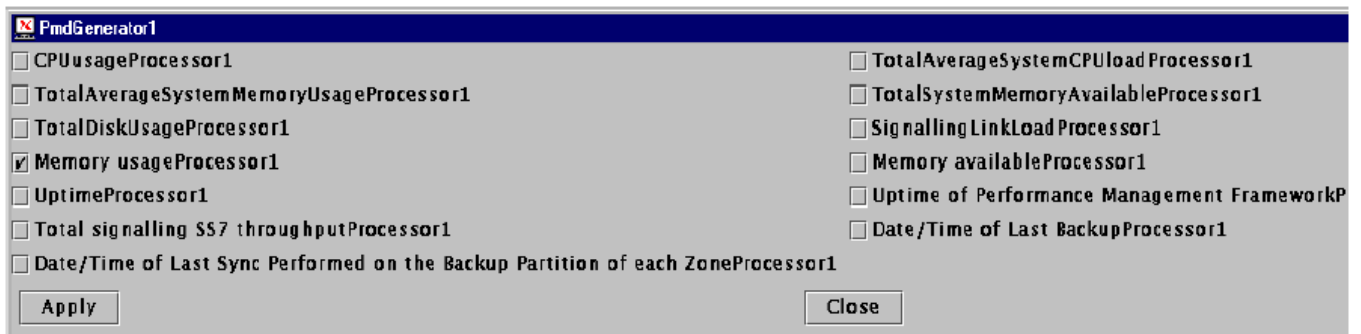
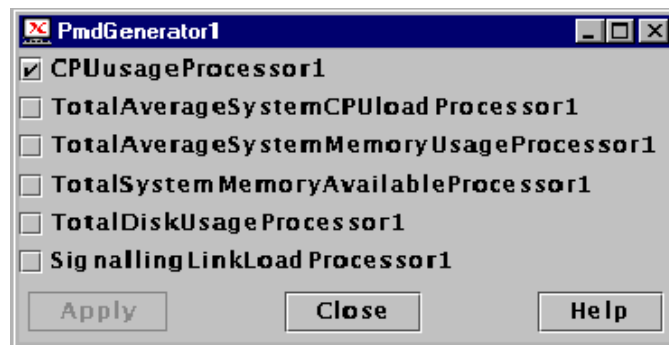


Figure 12-8 Check boxes

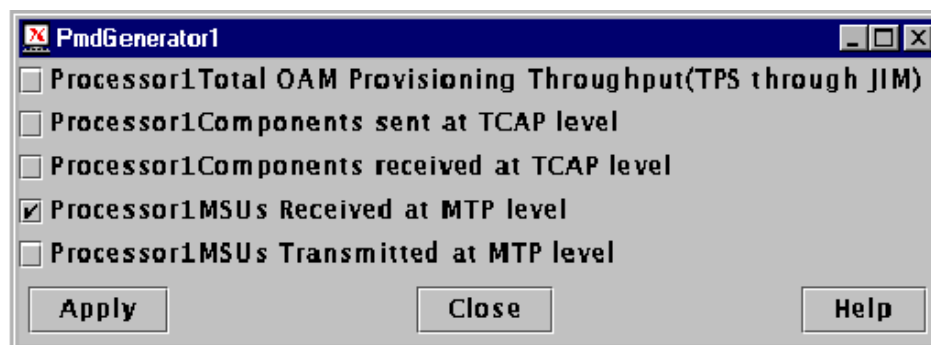


Figure 12-9 Counter Check box

3. Select the appropriate measurements. Click **Apply** to accept the configuration.
4. **Close** the check box. The measurements are displayed in the PMD display area.
5. Click **Graph** to view the Gauge Graph display, or

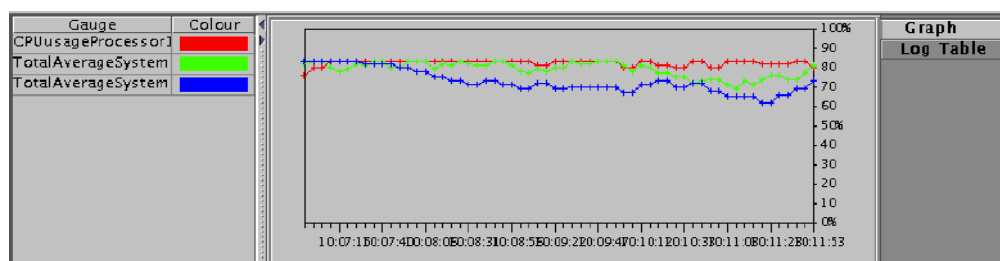


Figure 12-10 Gauge Graph Display

Click **Status Bars** to view the Status Inspection display, or

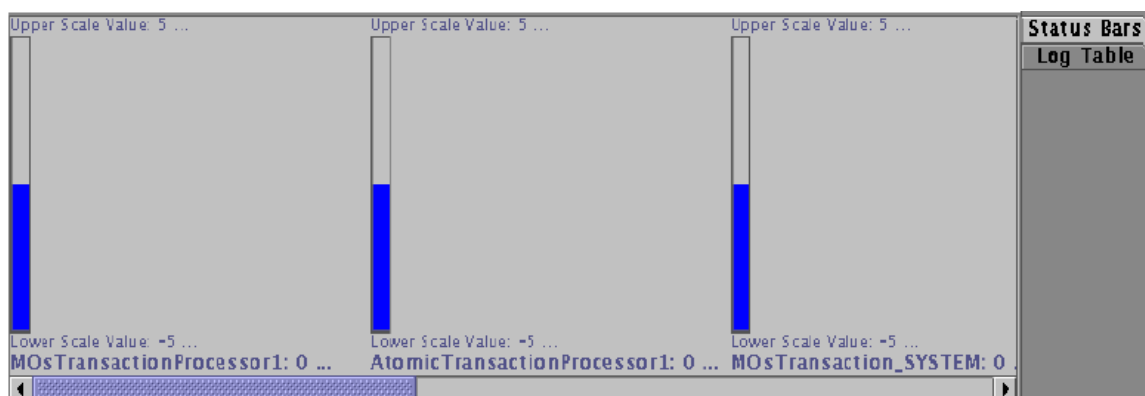


Figure 12-11 Status Inspection Display

Click **Counters** to view the Counter display, or

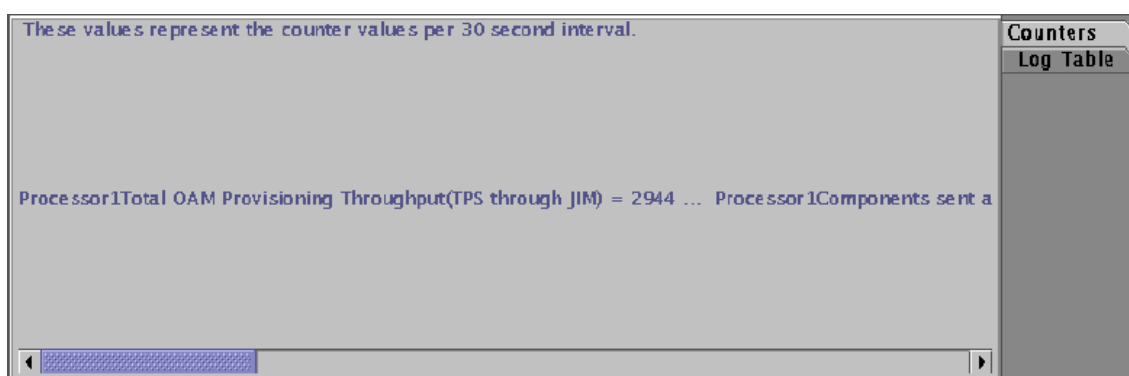


Figure 12-12 Counters Display

Click **Log Table** to view the Log display for all measurements.

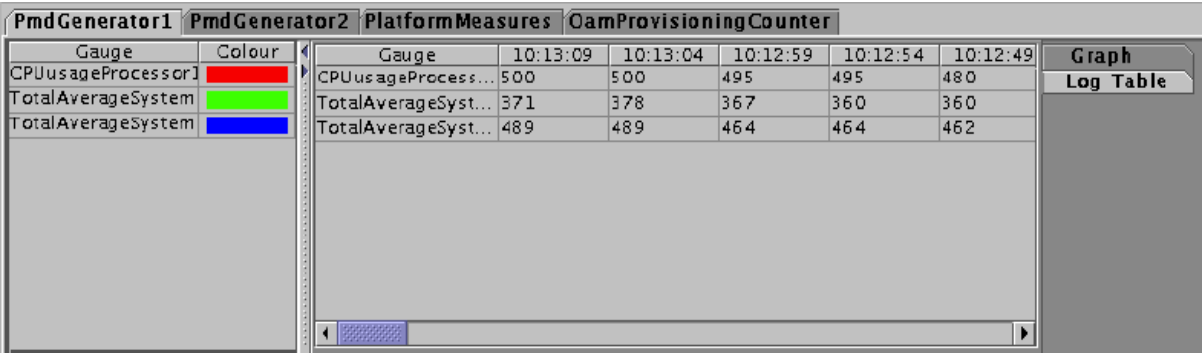


Figure 12-13 Gauge Log Table display

SUMMARY

- Performance Management (PM) measures the TSP platform statistics using counters, gauges and status inspections measurements
- Statistical measurements from the TSP Platform and the TSP Applications are reported to the Performance Management function via the Performance Management Framework (PMF) on the TSP.
- Performance Management uses the Fault Management interfaces for reporting alarms, the LDAP Browser for configuration of measurements and the XML file function for reporting scan files.
- Built-in to the TSP Node Management interface is the Performance Management Dashboard (PMD), which provides on-line inspection of PM data.

Intentionally Blank