Distributed Test and Maintenance Methodology Report

October 21, 2005

Status: Final

Preface

TEMPLATE CHANGE LOG

Submission	Date
Preliminary	N/A
Draft	10/13/2005
Final	10/21/2005

Table of Contents

1 Engineering Team

Below is the list of engineering team members:

IIF DTMM Team		
Zack Bocelle	609-485-8761	ATO-P
Chris Malitsky	609-485-7921	ATO-P
Bill Monsour	609-485-7929	ATO-P
Steve Souder	609-485-6170	ATO-P
Steve Bakanas	609-485-7916	L-3 Communications
John Gauntt	609-485-8570	J&N Technologies
Derbin Hamler	609-485-7914	L-3 Communications
Tom Morell	609-485-7907	LMATM

2 Executive Summary

Test and maintenance methods for modern distributed network environments were researched as a part of the Integration and Interoperability Facility's Fiscal Year 2005 tasking from their primary sponsor, Air Traffic Operations for the En Route domain. This report outlines the research conducted and products/concepts developed that resulted from this activity. These results fall into four categories:

- Laboratory operations management
- Centralized system maintenance
- Maintenance approaches for commercial off the shelf products
- New En Route air traffic control simulation environments

3 Reference Documents

- **Tivoli Netview for UNIX Programmers Guide Version 6**
- **Tivoli Netview for UNIX Programmer's Reference Version 6**
- **RFC 1227 SNMP Multiplexing Protocol (SMUX) Specification**
- **RFC 1228 SNMP Distributed Protocol Interface (DPI) Specification**
- **RFC 3954 Cisco Systems NetFlow Services Export Version 9**
- **Network Performance Measurement Tools: An Internet2 Cookbook**
- **FAA Order 1370.94, Wireless Technologies Security, 2/3/05**

4 Purpose

The purpose of this report is to document several tasks conducted at the Integration and Interoperability Facility (IIF) that pertain to new approaches for distributed test and maintenance methodologies in the FAA's En Route environment. This work was performed for the FAA Air Traffic Operations for the En Route domain Program Office (ATO-E).

Most of the approaches discussed in this report have been implemented and are functional. In some cases, the capabilities have been added to the IIF baseline and are used in the IIF business practice.

5 Definition of Terms

The following list is a definition of terms provided for use in the context of this report

- **Graphical User Interface** – The common practice of providing a graphic (i.e. visual object) to a user for interaction with a software application.
- **Client / Server Model** – An industry standard system design that allows multiple user applications (clients) to connect to a common service application (a server) to acquire common data types.
- **Relational Database** – A table based data storage model that provides quick and efficient storage and access capability of various data types.
- **Structure Query Language** - An industry standard language for querying, storing, or modifying specific data within the context of a relational database.
- **Application Programmer Interface** – A well-defined method typically provided with an application, for communicating or exercising functions within the application. The interface usually comes in to form of software routines that may be linked to by external applications so that the routines may be used by the application.
- **Berkeley UNIX Socket Based Communication** – A method of digital communication originally developed at the University of California Berkeley that utilizes Internet Protocol (IP) to establish a data socket between two software processes. This socket can then be used to share data between the processes

6 Background

6.1 Network Based Systems

Network-based systems use an underlying network infrastructure for subsystem communication. Most network based systems are considered distributed systems because system functions are separated across the various nodes that reside on the underlying network. Network-based systems are characterized by a set of computers interconnected, for example, by IEEE 802.3 Ethernet Local Area Network (LAN) technology, for the purposes of communicating, monitoring and controlling. The advantages of this technology include flexibility in physical location of system components, opportunity to implement security policy, standardization in communication protocols, scalability in design, opportunity to easily integrate COTS applications and more. The IEEE 802.3 Ethernet technology, as well as the IEEE 802.5 Token Ring technology used in the Display System Replacement (DSR) system, and other network technologies, are represented by a comprehensive set of hardware appliances and components, software applications, communications protocols, and design methodologies. These characteristics can be incorporated to affect a superior system design.

By designing a system that emphasizes the strengths of the network technology, the distributed system processing hosts can be optimally designed for their specific tasks while relying on the network for the services the network optimally provides. This modularized approach allows for designing one type of system component to perform a specific set of functions, and for designing another system component type to perform a different set of functions. Modularity in design provides clear boundaries to implement security and to provide logical and physical separation. These desirable characteristics allow design changes to be made in one module without necessarily affecting the design in other modules; the emphasis is on stability of design at the module boundaries. This modular design concept is scalable and can be applied to groups of hosts. In network design a set of hosts can be logically separated into various groups or broadcast domains (see Section 9.1.3 VLANs). These broadcast domains provide group boundaries where security can be implemented, and where resources can be optimally allocated for the group.

A feature of this broadcast domain concept is that for a host in one broadcast domain to be able to communicate with another host in another broadcast domain, the traffic must pass through a network router. The router performs the reconciliation of the logical broadcast domain addressing (Internet Protocol (IP) addressing), source and destination IP addresses, with the physical connections, or ports, to the router where the broadcast domain hardware connects to the router. So if one host in a broadcast domain represented by an IP address belonging to that domain sends a message to another host in a different broadcast domain identified by an IP address from that different domain, the reconciliation would necessarily be performed by the router. The router would receive

the message and examine the destination IP address. The result of the examination would be the decision that the path to the destination, derived by path (routing) information contained in the router matched to the destination IP address, is through the router port that connects the destination broadcast domain hardware. This simplistic example is scalable such that many routers where the IP address – routing information matching process occurs, could separate the source and destination hosts. This process is called network routing.

At each router security measures can be implemented to permit or deny traffic based on IP addresses, network protocols, and many other criteria. These broadcast domains and interconnecting routers can be used to optimally group resources and to provide an effective means to integrate new subsystems, or decommission old subsystems.

6.2 The Near Term Future of En Route ATC

6.2.1 Display System Replacement

The DSR is one the first FAA En Route domain system that is a distributed network-based system. The system uses multiple network architectures including Ethernet (IEEE 802.3) and Token Ring (IEEE 802.5) technologies. DSR applications are distributed across the network in logical groupings; however the applications do not take advantage of true routing capabilities as described in Section 5.1. This is because at the time that the DSR design was established, the available routing protocols were not considered resilient enough to meet the system's specifications. Instead, networks were treated as pipes. In the DSR design, if a particular data stream is required by an application, the machine housing that application is fitted with the pipe that provides that data stream.

A good example of this is the DSR Local Communications Network (LCN) Token Ring design. This design is comprised of a backbone ring set consisting of four token ring networks. Each network represents a different data type coming from and/or going to the mainframe Host Computer System (HCS); Broadcast (radar), Point-to-Point (flight data), Recording (System Analysis and Recording), Spare. There are then four access ring sets (four networks a piece) that are interfaced with the backbone set. Application nodes reside on all of these access ring sets (See Figure 6-1).

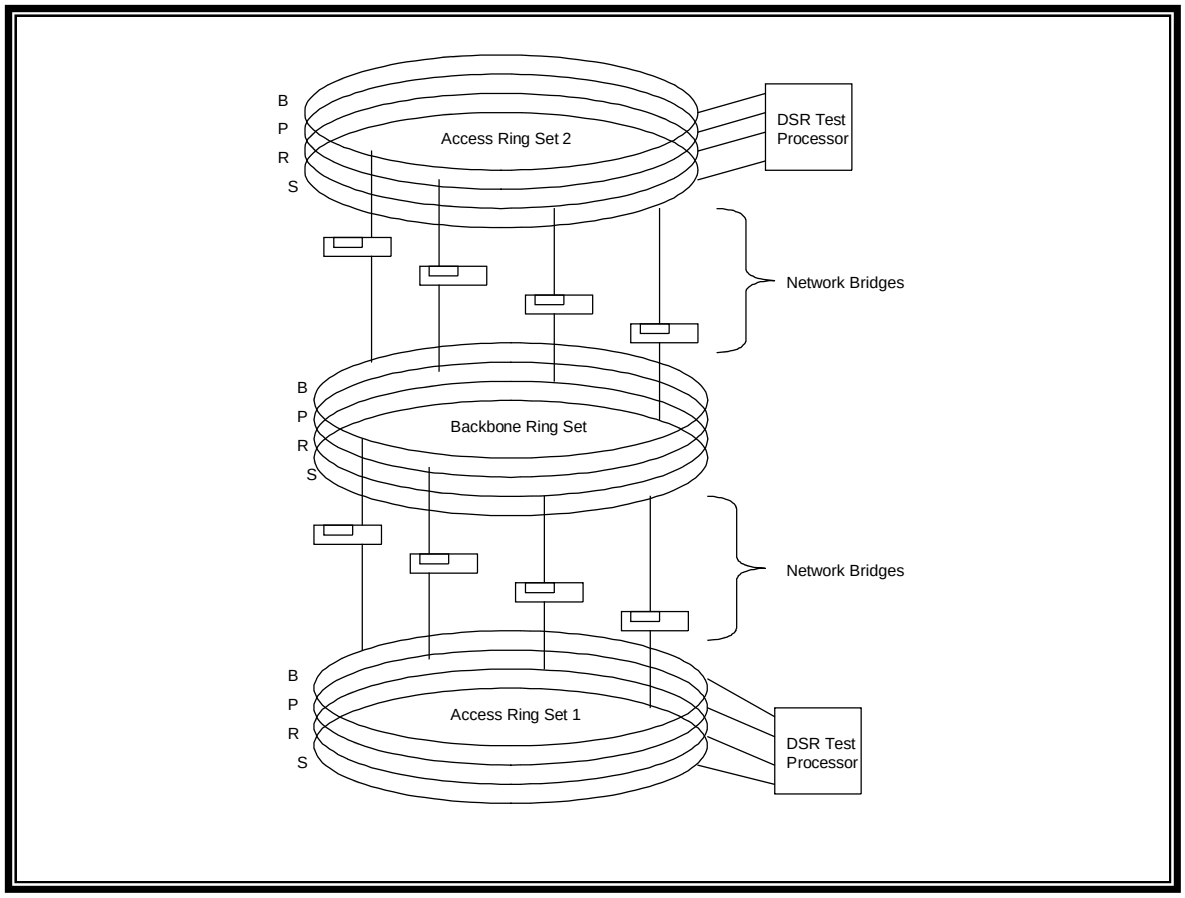

Figure 6-1 DSR Local Communications Network Architecture

More recently, there have been several technology refresh iterations of DSR that include the introduction of the User Request Evaluation Tool (URET) and the Data Position Technology Refresh programs. These programs have moved the DSR system away from the antiquated Token Ring network technology to more modern network protocols, but have not introduced routing protocols into the core systems.

6.2.2 En Route Automation Modernization System

The En Route Automation Modernization (ERAM) Program is currently under development under the management direction of the FAA ATO-E. This program is being developed and deployed in an incremental fashion, using existing infrastructure to facilitate migration. In particular, the DSR technical refresh initiatives and the URET program are providing the building blocks for ERAM rollout.

Like the DSR, the ERAM system will also be a modern network-based system unlike its predecessor the HCS.

Unfortunately, the ERAM architecture design incorporates few instances of network routing using broadcast domains separated by routers as described in Section 6.1. An ERAM design evolution that includes more instances of broadcast domains separated by

routers would enhance system security and facilitate the evolution of the ERAM System to the FAA's envisioned 2025 system architecture.

6.3 Working Within the New En Route Environment

6.3.1 Network-based Architecture

The URET National Deployment program has introduced Cisco Systems 4000/4500 multi-layer switches into the En Route infrastructure. These switches currently satisfy port density requirements for operational and maintenance functionality. More importantly, however, they introduce the capabilities for transitioning the current En Route design into a route-based, network centric model. The introduction of enterprise level switches allows the collapsing of capabilities and systems into fewer platform(s), thus providing a means to simplify infrastructure, integrate capabilities, reduce hardware and training costs, increase system availability, and provide benefits such as an increase in performance.

6.3.2 Open Systems Protocol Capabilities

The introduction of open systems protocols within the En Route environment enables system owners to add extensibility to certain aspects of the system. In particular, the maintenance of an En Route environment using open systems protocols allows the FAA to enhance its maintenance environment in an efficient and cost-effective manner. It also allows greater interfacing between the system developer and the user of the system.

More detailed information regarding these types of approaches is discussed in subsequent sections.

6.3.3 Potential Remote Access Capabilities

Second-level maintenance support activities are planned to be centralized in ERAM, much in the same fashion of recently deployed En Route systems. Access to certain data at the centralized site (WJHTC) from the En Route Centers at times may be useful in problem determination and resolution. This type of data access is feasible with the existing DSR and URET systems and will be extensible and/or capable with the ERAM system as well. Further discussion in this area is provided in Sections 7 and 8.

6.3.4 Security and the New Distributed En Route Design

The approaches addressed in this report have utilized, to the best extent possible, existing En Route infrastructure. Ensuring data security and integrity, a hacker-proof infrastructure, and a virus-free environment is the standard for all En Route communication. This standard must be met in any system and capability utilized within the En Route environment. Security requirements of programs and domain level Security Certification and Authorization Packages (SCAP) must be

followed. An industry standard approach to securing networks and networked data is to apply layered security. Examples of layered security principles applied in the approaches taken within this report are noted in the appropriate Sections. While security principles are understood by the IIF technical team, any early engineering assessment or proof-of-concept activities performed by the IIF should be considered functional exercises only. It is the responsibility of the program office to ensure a secure implementation of any capability that is put into operational use.

6.3.4.1 En Route Interfacility Wide Area Network

The current En Route architecture contains various methods to distribute its numerous sources and sinks to and from an En Route Center to various external sites. It is assumed at some point in the future that the FAA Telecommunications Infrastructure (FTI) program will provide all FAA communications infrastructures for all Wide Area Network (WAN) communications. In the interim, any approach taken by the IIF to demonstrate capabilities will utilize the existing infrastructure. Most notably are the DSR Frame Relay network and the Bandwidth Manager WAN. Any security requirements for WAN communications will be assumed to be covered by the WAN program requirements. It is also assumed that a trusted En Route-based host will exist on the other side of the communication link to receive or send data over what is considered a trusted WAN link.

6.3.4.2 En Route Intrafacility Local Area Network

The operational En Route network infrastructure will be based heavily on the design of the URET and ERAM architectures. The designs of these systems are constructed to meet subsystem security SCAP processes, which include subsystem level firewall and intrusion detection systems. The design principle also segments data into multiple types which equate to levels of trust in networking terms. Physical separation of these data types is held true in the network design, but commingling of the data types is allowed at the server level. Security principals are applied not only in the firewall and intrusion detection systems, but also in the network appliances and server systems themselves. Data transitioning from a higher level of trust to a lower level of trust is generally ‘pushed’ from the higher level of trust to the lower level of trust or stored first in a common data storage where it is accessed from the system in the lower level of trust. This model and the existing subsystem SCAPs should be adhered to by any maintenance systems. The approaches discussed in this report would all have to meet domain and subsystem requirements prior to implementation, and in general, security considerations are outside the scope of this report. Where practical, the approaches will abide by the security principles of the En Route systems used for the proof of concept exercises.

7 Lab Operations Approaches using SNMP Agents and a Network Node Manger

7.1 Current WJHTC DSR Lab Operations

Currently the daily DSR Lab operations activities at the WJHTC are facilitated via the menu-based Laboratory Operations Management Laboratory Environment Tool (LOMLET). Prior to and after every DSR lab run, lab users or lab operations personnel are required to obtain the status of all DSR processors within their assigned lab using LOMLET. LOMLET gathers this data using a custom method of Distributed UNIX Korn Shell scripting over broadcast User Datagram Protocol (UDP) / IP connections. LOMLET obtains network access to all of the Lab Test Processor via the Ethernet/IP based DSR Maintenance LAN. Running these Korn shell templates (e.g. CHECKPROC.template) against a configuration list containing every processor in the lab ensures that the lab is initialized from a known state for each test run. The LOMLET tool requires the user to execute remote template scripts against a laboratory configuration of processors. As each processor completes its tasks it then reports the status back to the LOMLET software running on a DSR Lab Controller processor, where the results for each processor are displayed to the user as a listing in a textual report format (See Figure 7-1).

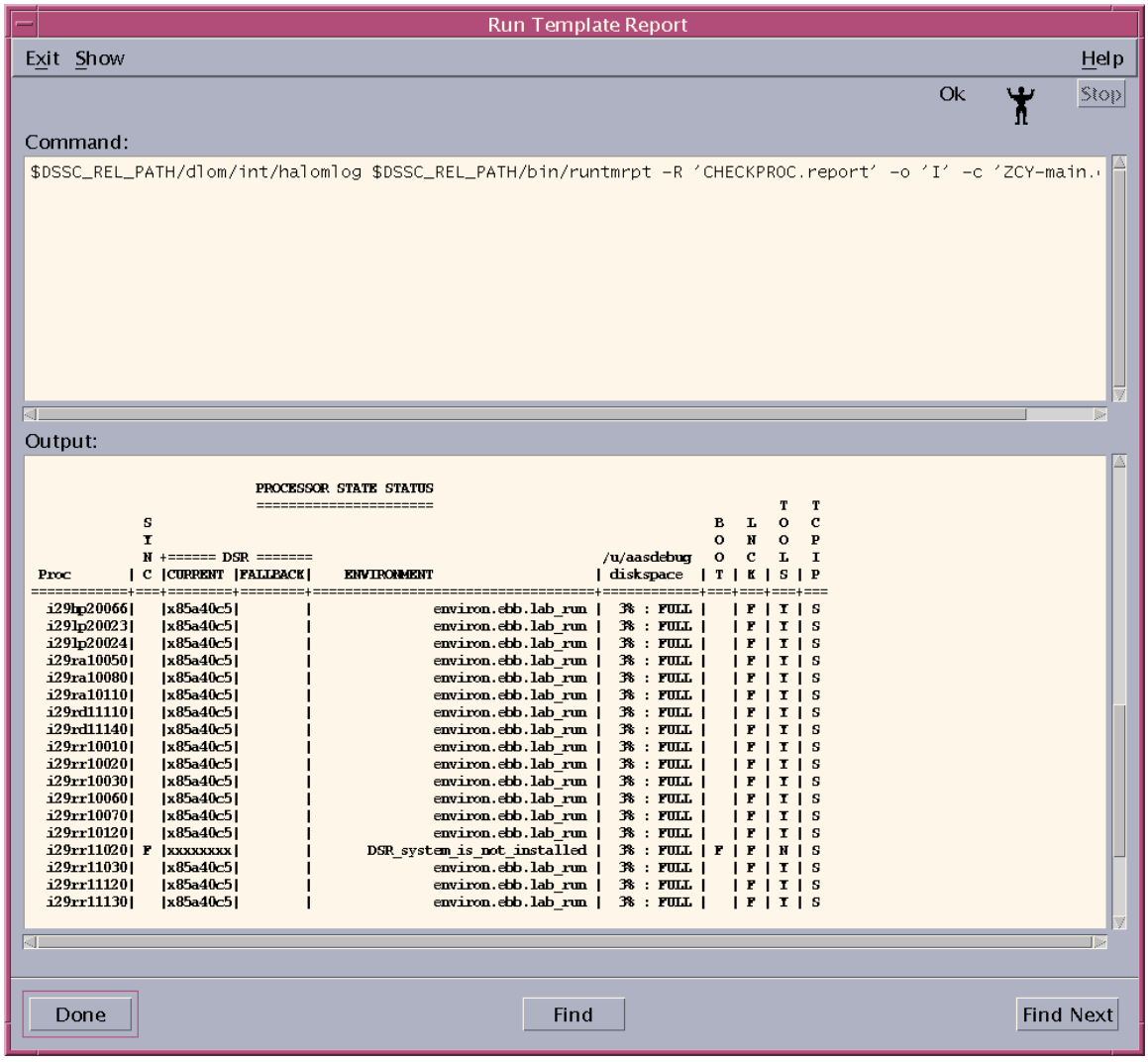

Figure 7-1 LOMLET Checkproc Report

Ongoing DSR Test Processor hardware and software status is obtained using LOMLET in the previously described method. Using a method such as this is not an optimal approach to lab management. It is a reactive, not proactive approach. Problems are discovered only when a users solicits test processor status manually. Automated, routine system monitoring is much more suited to a large laboratory environment such as the WJHTC DSR labs.

7.2 Lab Operations with an SNMP Architecture

7.2.1 Description

Lab operations utilizing an SNMP architecture enables lab personnel and users to view laboratory status (DSR software running, Filesystems utilization, current/fallback release status, etc.) graphically in near real-time without solicitation (i.e. LOMLET template execution). This could provide a considerable time savings to the Lab customer. Currently the IIF employs SNMP as a means of displaying near real-time status of DSR

processors. SNMP is also used to execute processes remotely on the DSR processors and collect the necessary data to display lab status in the format of the current DLOM CHECKPROC report. This technology, if developed further, may be utilized to accommodate the additional functions necessary to run and maintain the DSR labs.

7.2.2 IIF SNMP Architecture

The base architecture needed to support SNMP lab operations requires one Lab Management Processor (LMP), located outside of the DSR operational networks. This processor requests/receives data from DSR Test Processors and can be placed on the DSR maintenance LAN to gain Test Processor connectivity.

The LMP at the IIF was installed on the IIF Management LAN which has routed connectivity to the IIF DSR Maintenance LAN. The IIF LMP platform is a Sun Blade 150 running Solaris Version 8. Tivoli Netview Node Manager Version 5.0 is installed as the SNMP Management package on the IIF LMP.

7.2.2.1 Netview

Tivoli's Netview Node Manager is a powerful, flexible tool for managing networks. Because network management may be quite dynamic, Tivoli Netview provides several mechanisms with which you can extend its function or tailor it to your specific needs.

The Netview Application is the centerpiece of much of the Distributed Test and Maintenance activities described in this report. It provides two vital elements; SNMP message processing capabilities and database capabilities for retention of object state data. Netview maybe configured to solicit data from network nodes (in this architecture DSR Test Processors) on a routine basis using standard SNMP messaging. This data may then be indicated graphically to the user with the OpenView Windows (OVW) Netview subsystem.

The primary method for using these Netview capabilities is via the Netview Application Programmers Interface (API). The Netview API allows the user to access the Netview object database where state data pertaining to objects or network nodes are stored. The End User Interface (EUI) portion of the Netview API allows for customization of the Netview Graphical User Interface (GUI). The API also allows the developer to harness the Netview SNMP capabilities to request specific data from network nodes. Figure 7-2 depicts the Netview API infrastructure.

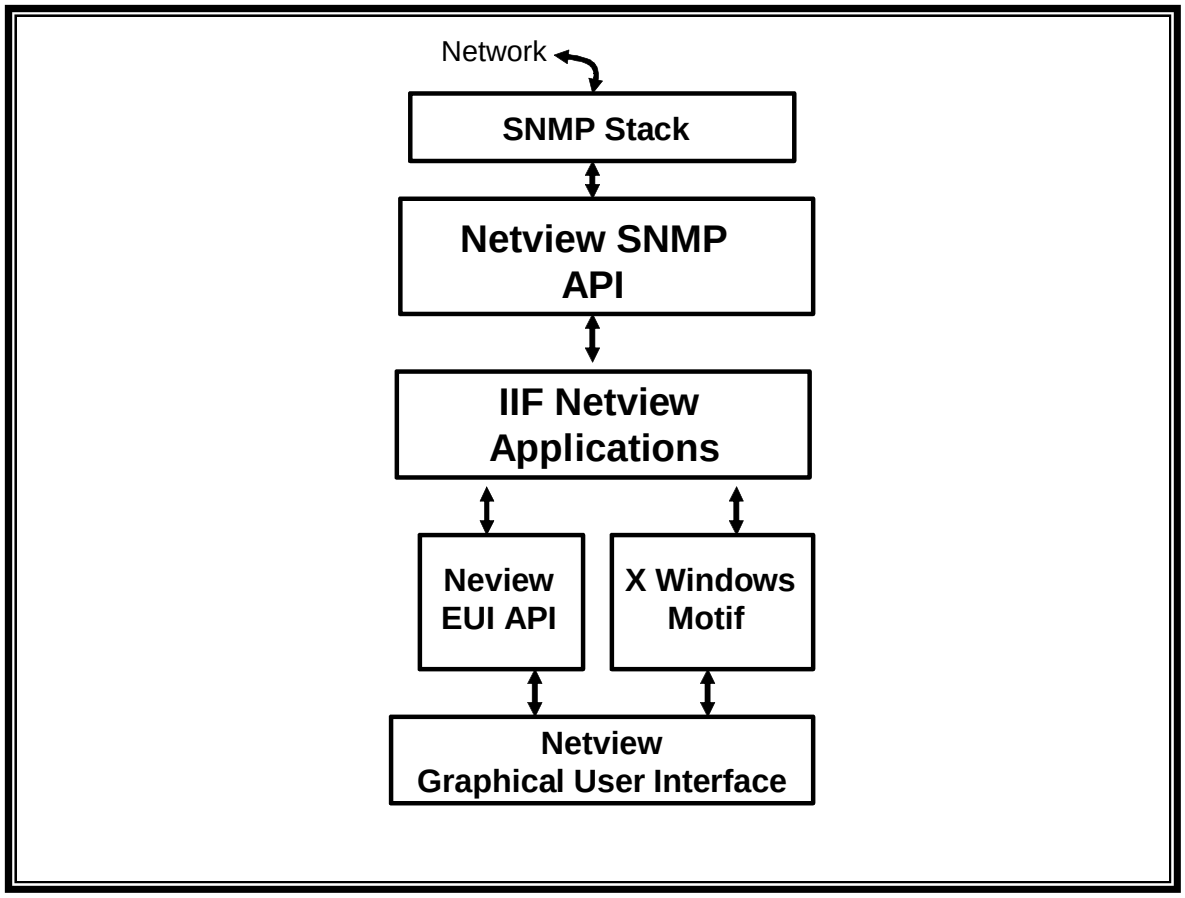

Figure 7-2 Netview API Infrastructure

7.2.2.2 Extensible SNMP Agents

SNMP agents running on each Test Processor must also be implemented to support SNMP lab operations.

Extensible SNMP agents broaden the capabilities of SNMP-based management applications. Extensible agents allow developers to quickly change and add Management Information Base (MIB) item definitions to accommodate customized SNMP requests and directives. Specific management needs, such as getting and setting remote node data values, triggering processes and generating alerts (sending SNMP traps), may be rapidly developed through the use of extensible SNMP agents.

While extensible SNMP agents provide great flexibility for rapid development in Laboratory environments it is worth noting that in many cases, agents of this type could create a performance issue on processors in certain situations, where as true MIB developed (i.e. non-extensible) SNMP agents would not. Most extensible SNMP agent implementations can be further developed into custom MIB-based SNMP agents if performance becomes an issue.

There are many COTS extensible agents. The following agents have been used at the IIF for this task as well as others.

7.2.2.2.1 NetSNMP Extensible Agent

Net-SNMP is an open-source third-party SNMP suite that is extremely modular in nature. The suite of applications implements SNMP v1, SNMP v2c and SNMP v3 using both IPv4 and IPv6. The suite includes:

- Command-line applications to:
  - Retrieve information from an SNMP-capable device, either using single requests (snmpget, snmpgetnext), or multiple requests (snmpwalk, snmptable, snmpdelta).
  - Manipulate configuration information on an SNMP-capable device (snmpset).
  - Retrieve a fixed collection of information from an SNMP-capable device (snmpdf, snmpnetstat, snmpstatus).
  - Convert between numerical and textual forms of MIB OIDs, and display MIB content and structure (snmptranslate).
- A graphical MIB browser (tkmib), using Tool Kit (TK)/Perl.
- A daemon application for receiving SNMP notifications (snmptrapd). Selected notifications can be logged (to syslog, the NT Event Log, or a plain text file), forwarded to another SNMP management system, or passed to an external application.
- An extensible agent for responding to SNMP queries for management information (snmpd). This includes built-in support for a wide range of MIB information modules, and can be extended using dynamically loaded modules, external scripts and commands, and both the SNMP multiplexing (SMUX) and Agent Extensibility (AgentX) protocols.
- A library for developing new SNMP applications, with both C and perl APIs.

Net-SNMP is available for many UNIX and Unix-like operating systems and also for Microsoft Windows.

7.2.2.2.2 Sun Solstice Enterprise Agent

Solstice Enterprise Agents is a comprehensive development and runtime environment enabling the creation of custom, extensible SNMPv1 or Desktop Management Interface (DMI) 2.0 agents to enable Solaris-based network and system management.

Key benefits include:

- Runtime integrated into next Solaris release. Enables improved manageability of Solaris systems.
- Deployed SNMP and DMI-based subagents can be managed from any SNMP management application including Solstice Enterprise Manager
- Extensible architecture allowing simultaneous execution of multiple SNMP and DMI subagents
- Provides full support for existing SNMP legacy agents
- Enables both static and dynamic registration of subagents

Note that the Solstice SNMP agent is only available for Sun Solaris platforms.

7.2.2.2.3 IBM SMUX (SNMP Multiplexing Protocol) Extensble SNMP SubAgent

SMUX is an extensible subagent that may be used by the IBM SNMPD agent to query data points maintained by other user-level processes currently running on a machine. SMUX lets the SNMP agent listen for and respond to all management requests for a single device, but it multiplexes the handling of requests among (possibly several) local entities, called SMUX *peers*, each of which is responsible for instrumenting some part of the device's MIB. A common SMUX peer is the SNMP Distributed Protocol Interface (DPI) peer. Instead of using SNMP PDUs between the SNMP agent and the subagent (which is the SMUX specification), DPI specifies its own lightweight protocol for agent-subagent communication.

The SMUX subagent functionality is built into the IBM SNMPD agent and comes with the AIX operating system (including the legacy AIX 3.2.5 – the DSR Test Processor Operating System level). SMUX and DPI, in particular, were used for this portion of the DTMM task.

7.3 Lab Operations Proof of Concept Tasks

The following examples were developed to illustrate how DSR Test Processors could be presented in an SNMP based architecture where users can be aware of pertinent information in near real time using routine polling. Users may also solicit the same type of data that the DLOM LOMLET tools provide by using graphical display techniques. Both examples utilize SNMP for network transactions.

7.3.1 DSR Test Processor Polling Using SNMP

The following data points were selected for Polling within the IIF DSR Laboratory:

- **DSR Current Release Name** – This data point is useful in the event of an ad-hoc user who has specific functional requirements, but does not care what DSR system release is running in the configuration. Assessing what release is currently distributed into the lab may prevent a subsequent distribution, thus saving lab time.

- **DSR UP or DOWN –** DSR Test Processors may only be manipulated in the DLOM environment if DSR software is not running, thus it is important to verify that all DSR software is shutdown prior to commencing a DLOM software distribution. Having this information readily available without solicitation would save time.
- **AASDebug Directory Size** – DSR software is sensitive to disk space availability, if there is not enough space on a processor (per DSR specification) the software will place the machine into a diagnostic role, preventing it from becoming fully operational. The "aasdebug" directory is where most debug data is stored when testing in the WJHTC DSR labs. The directory is routinely cleaned up, but in some cases when cleanup is overlooked, this directory can cause the processor to be above the threshold for disk utilization.

7.3.1.1 Netview Configuration

To execute this task several Netview configuration items were modified and a small software module was developed using the Netview SNMP API and EUI API to poll and status the DSR Test Processors.

One of the basic Netview functions is network node discovery. Because of this function, the IIF LMP Netview application was already aware of the IIF DSR Test Processors through the LMP's network drop on the IIF DSR Maintenance LAN. The Test Processor Nodes were indicated on the Maintenance LAN Subnet within the Netview application (See Figure 7-3).

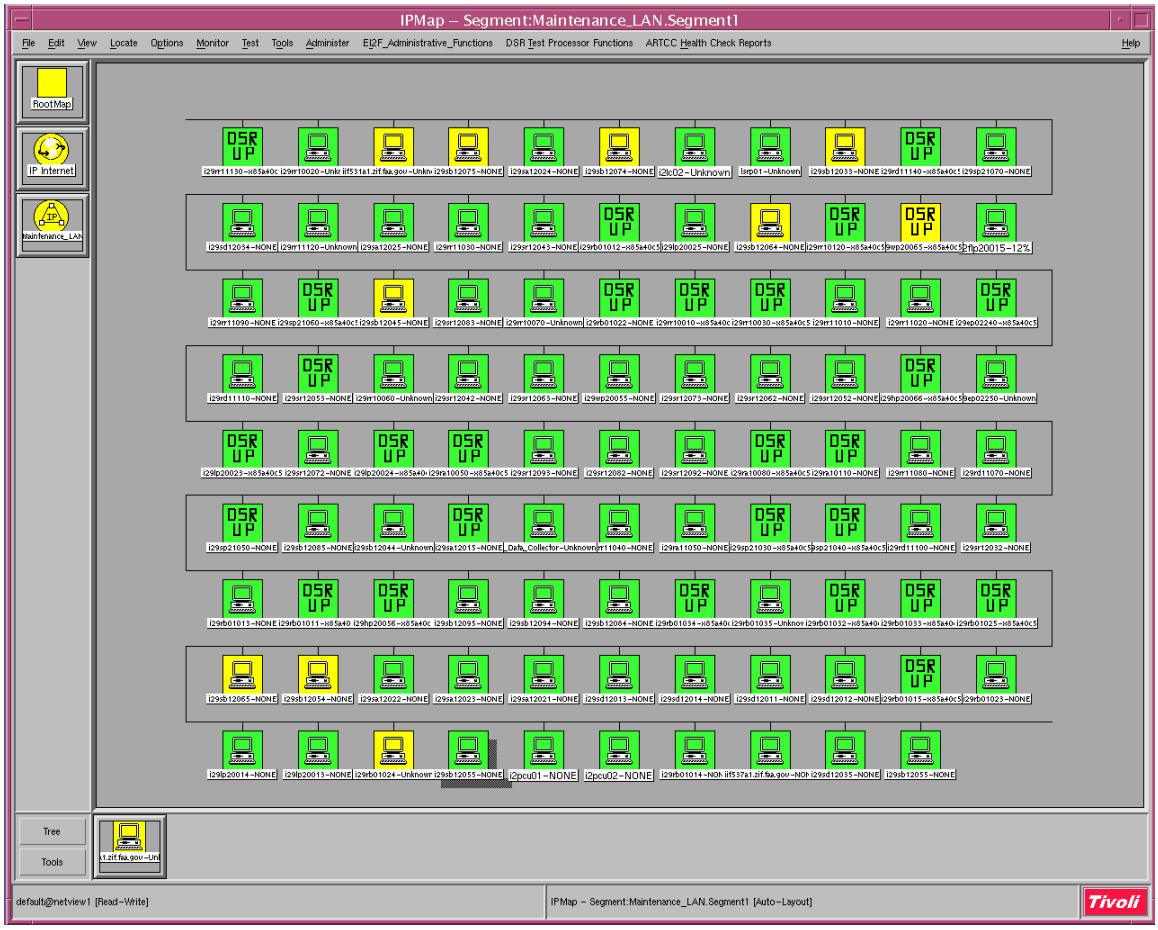

Figure 7-3 Netview DSR Maintenance LAN Submap

To be able to discern Test Processors from other network nodes that Netview is aware of, a new node field type was added to the Netview configuration called “Test_Proc”. This new type was added by inserting a registration file containing the Test_Proc type definition to the base set of Netview fields files (See Appendix C for Test_Proc fields file). Each DSR Test Processor was then “tagged” with this field. This field tag allowed the polling software to gather a list of Test Processor nodes for polling.

The DSR Test Processor polling software was set up to poll each network node tagged as a Test_Proc on a thirty second cycle. Using the Netview SNMP API, the Test Processors were polled for the three example data points (DSR current release name, DSR software up or down, and aasdebug directory size). Responses to these requests are received by the poller and are represented in the Netview GUI using the Netview EUI API. Display methods are as follows:

- **DSR Current Release Name** – The Current DSR release is received and placed into a configured Netview field called “Current_Release” (See Appendix C for Test_Proc fields file). A menu item was added to the Netview OVW registration files (Netview X-Windows Motif API) to allow the user to View/Hide the DSR current release (See Appendix C). When “View” is selected, the corresponding current release name is placed into the symbol label of each symbol that represents a Test_Proc tagged node. When “Hide” is selected, the current release name is removed from the label (See Figure 7-4).

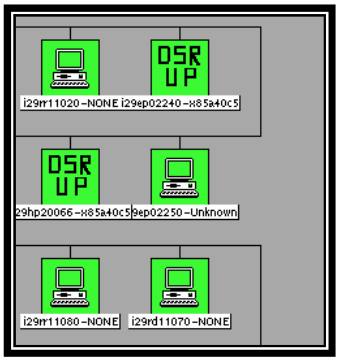

Figure 7-4 Netview DSR Test Processor Symbols

- **DSR UP or DOWN –** The current DSR software status is received and if the software is running on a Test_Proc node, each symbol representing that node is modified with a custom symbol bitmap that has “DSR UP” displayed on it using the Netview EUI API. If the software is not running, each Test_Proc node symbol is set to the default symbol for the node (i.e. no “DSR UP” indication - See Figure 7-4).

- **AASDebug Directory Size** – The directory size is received at the poller application in the form of a percentage. A symbol was added to each Test_Proc network node’s submap. This symbol represents the status of the aasdebug directory for that Test Processor. If the received percentage is less than or equal to 50%, then the aasdebug symbol color for that Test_Processor is set to Green (Netview Up color) using the Netview EUI API. If the percentage is greater than 50%, then symbol color is set to Yellow (Netview Marginal color See Figure 7-5).

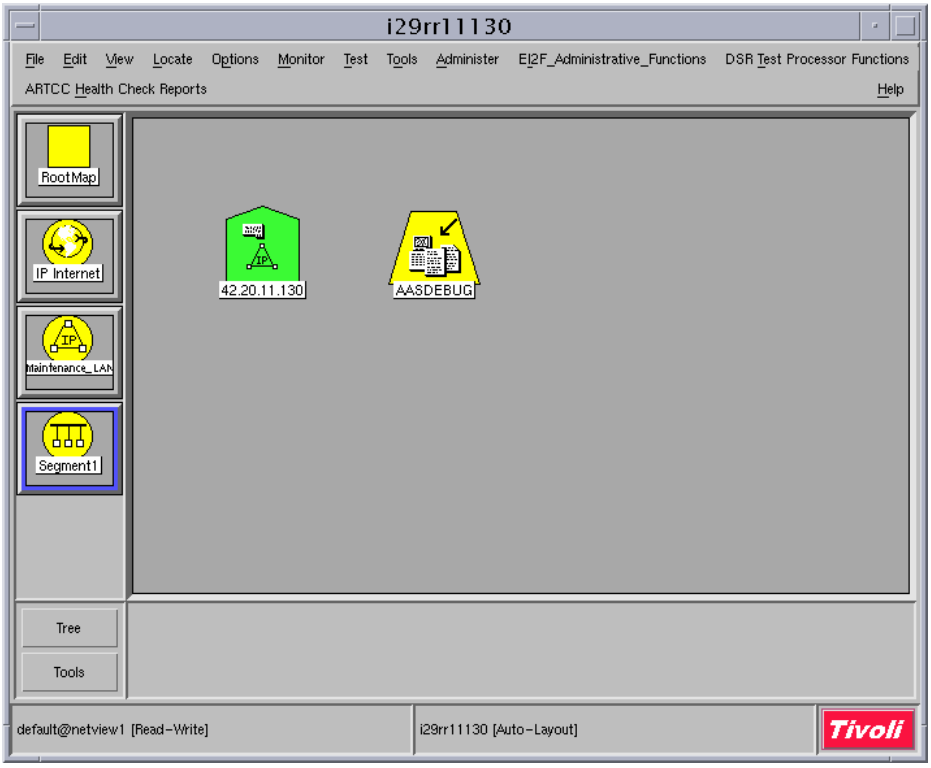

Figure 7-5 Netview DSR Test Processor Submap With aasdebug and NIC Symbols

The Test Processor symbol which contains the submap with the aasdebug symbol was also configured to be "Compound Propagated" using the Netview EUI API. This configuration item causes the Test Processor symbol to take symbol color (status)

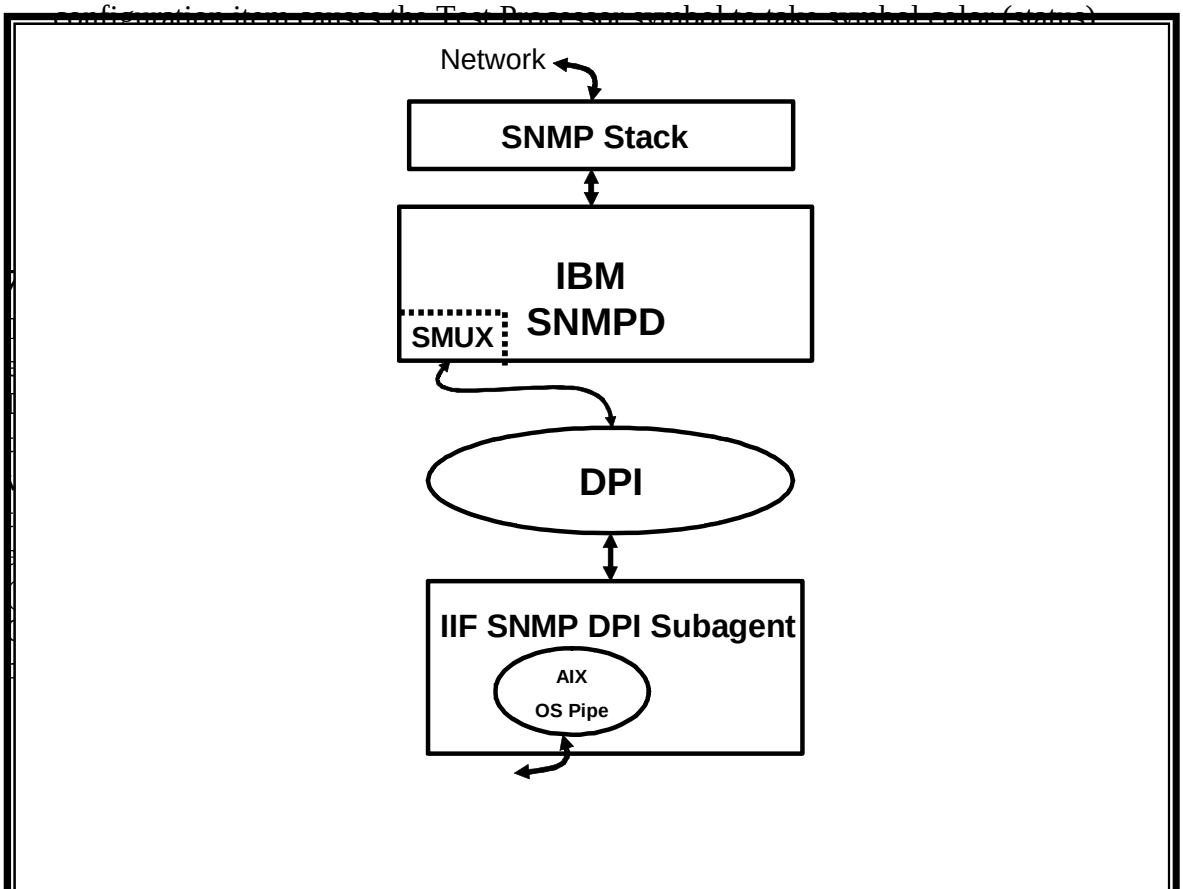

Figure 7-6 SMUX/DPI Infrastrucutre

7.3.2 DSR Test Processor Information Requests Using SNMP

In addition to polling for DSR Test Processor status, a data/status request mechanism was implemented to provide an example of querying DLOM type data using Netview.

As mentioned previously, a common status report that is run as a part of current DLOM Laboratory operations is the Checkproc report. This report may be run on a specified group of Test Processors simultaneously. The report provides information regarding the current condition of the DSR Test Processors contained within the specified configuration. This task essentially implemented the Checkproc report using SNMP as a means of communication and Netview as a means of soliciting and displaying the report (See Figure 7-7).

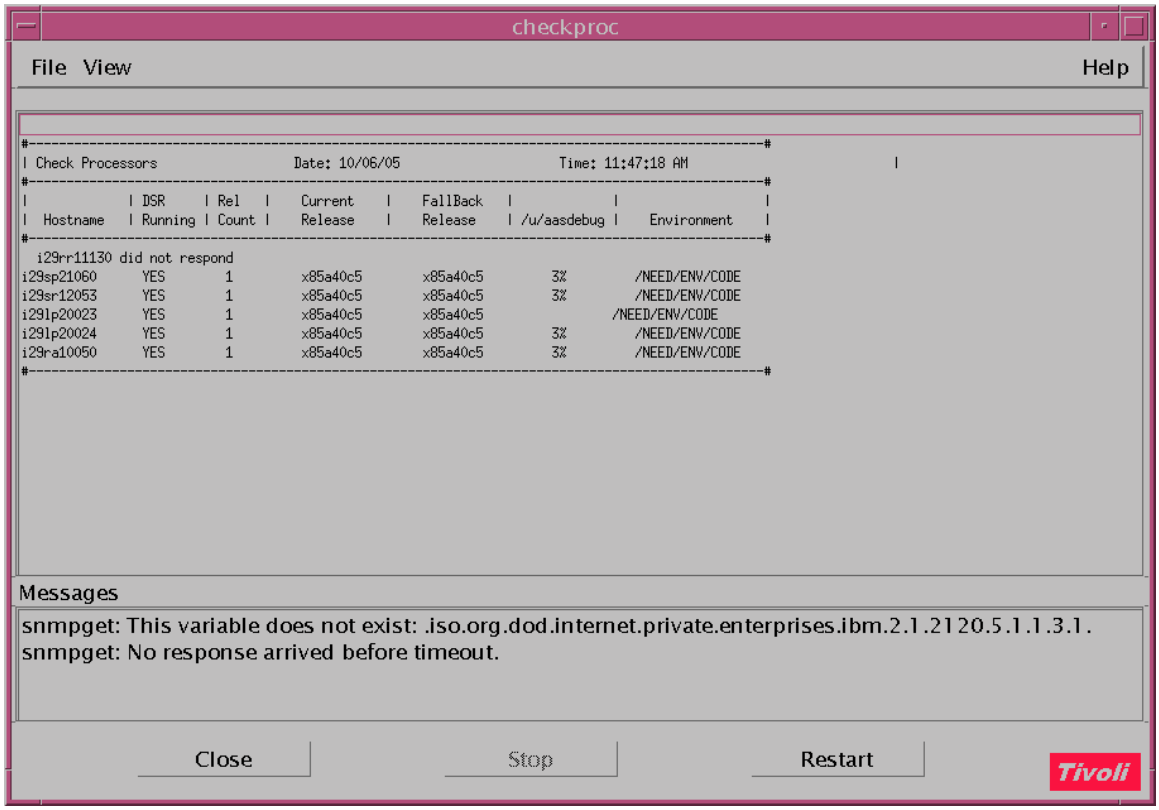

Figure 7-7 Netview Checkproc using the Application Monitor

7.3.2.1 Netview Configuration

The Netview configuration is the same as described in Section 6.3.1 with the following additions:

- A shell script was developed to request a set of OIDs via SNMP, using the Netview provided SNMP tools. The OIDs represent the required data points at the Test Processor to build the Checkproc report (See Appendix A for snmpd.conf listing).

- A menu item was added to the Netview OVW registration files to allow the user to trigger a checkproc report. This menu item may be selected only if one or more symbols representing Test_Proc tagged network nodes are selected within the Netview GUI. When a Checkproc report is requested, the output is displayed using the Netview Application Monitor software (See Figure 7-7).

7.3.2.2 Agent Setup

The SNMP agent configuration is the same as described in Section 6.3.1 with the following addition:

- A set of OIDs comprising the required data points for the Checkproc report were added to the SNMPD agent. The OIDs were mapped to the developed DPI SMUX peer.

- A set of calls to the operating system pipe within the developed DPI SMUX peer were added to support the additional OIDs required for the Checkproc report.

8 Methods for Centralized/Level Two En Route Maintenance Support

8.1 Background

The WJHTC DSR In Service Management Team (ISMT) has a mission to perform second level maintenance support for the FAA En Route DSR System. Other teams are chartered to perform second level maintenance on the NAS HOST and other legacy systems as well. The second level maintenance activities performed at the WJHTC rely on methods of data transfer and communication that delay the actions of the problem solvers and the implementation of effective problem resolutions. After a problem is encountered, recorded NAS data that is problem related is sent from the ARTCCs to the WJHTC via cartridge tape. This transportation method can take a day or more and represents at least this much of a delay before the problem solver can begin to analyze the problem. Also at the first indication of an ARTCC system problem a WJHTC problem solver attempts to contact a representative at the ARTCC to start a dialogue concerning the problem. The success of the dialogue using this communications convention is contingent on the appropriate people engaging each other in a timely manner. The effective outcome of this effort can take several days representing additional delay. Getting timely information and data to the appropriate people at the WJHTC is a desirable goal that will enhance the effectiveness of the problem determination and resolution process, and increase the success of the ISMT in the performance of their mission.

NAS generated data and information from the ARTCCs is not only necessary to resolve problems, but is very useful and necessary for performing system trend analysis, identifying NAS wide systematic anomalies, performing resource utilization analysis for planning purposes, performing statistical analyses of system functions and other valuable activities. Currently, the minimal scope and frequency of this data and information that is available to the WJHTC precludes these activities to any significant end.

Not only are these activities of value to the system planners and problem solvers, their implementation in the WJHTC support domain is identified as a goal to assure the viability of the WJHTC in this support role. The implementation of these activities, as well as the complimentary changes to infrastructure, process and policy, enhance the awareness and effectiveness of the planners and problem solvers such that a superior air traffic control system can be designed, implemented and maintained.

8.2 Centralized DSR Level Two Maintenance Proof of Concept Definition

This centralized maintenance concept highlights the opportunity for enhancing the ARTCC-to-WJHTC data and information transfer that facilitates the system planning and

problem solving activities performed at the WJHTC. The current infrastructure, information types, and analysis procedures are recognized while enhancements to these attributes have been prototyped to present a possible improved alternative to the status quo. Getting the relevant information and data to the appropriate people in a timely manner is the fundamental necessity for effective planning, problem determination and resolution. A centralized subsystem that includes a data repository and presentation facility accommodates this need and is characteristic of the concept architecture.

A method of processing the DSR SAR data is the System Wide Analysis Capability (SWAC). This tool can format SAR data and provide intermediate data that can be further processed to summarize event statistics for the entire system. An example of this summary information to be used for this concept is the DSR System Health Report (SHR).

While the recording of SAR data, the execution of the SWAC tool, and the generation of the SHR are currently part of the DSR sub-system, there are systematic and procedural limitations in their use. DSR SAR is recorded real-time and archived on IBM 3490 cartridge tape. On operational systems, SWAC cannot run by design because of resource contention. Cartridge tapes are uploaded and data is sent to the WJHTC for the ISMT to process, using Offline SWAC and other offline tools. These activities are separated by time and location, thus reducing their utility.

The centralized level two maintenance concept automates the steps that include generation of SHRs for problem resolution and for planning activities. Additional infrastructure will allow for the ARTCC operation of real-time SWAC on real-time operational SAR data, SHR generation using this data, real-time notification to the WJHTC ISMT of SHR-based alert conditions and the presentation of these SHRs on-demand to the WJHTC ISMT. In addition daily generated ARTCC SHRs will be archived at the WJHTC and be subjected to data mining activities for trend analysis and planning statistics. Also a real-time SWAC client allows for selective viewing of the SWAC output for the purposes of debugging.

8.3 Infrastructure Discussion

The NAS ARTCCs and the WJHTC are all interconnected via the DSR Frame Relay Wide Area Network (WAN). This WAN provides a medium that can facilitate the automated data and information flows previously identified. To illustrate this concept feature, a set of functions was designated at the IIF to represent the WJHTC node. These functions complemented the existing IIF DSR infrastructure, which acted as the Air Route Traffic Control Center (ARTCC) node in the example concept. The two nodes (WJHTC - ARTCC) communicated through the DSR Frame Relay WAN.

The WJHTC node accessed the DSR Frame Relay WAN indirectly through the AOS/ISMT Frame Relay Router; Network fibre connections from the IIF concept router to the ACB800 Job Shop router, and a further connection to the WJHTC DSSC Token Ring Support LAN. Figure 8-1 Represent the IIF Centralized Maintenance Architecture.

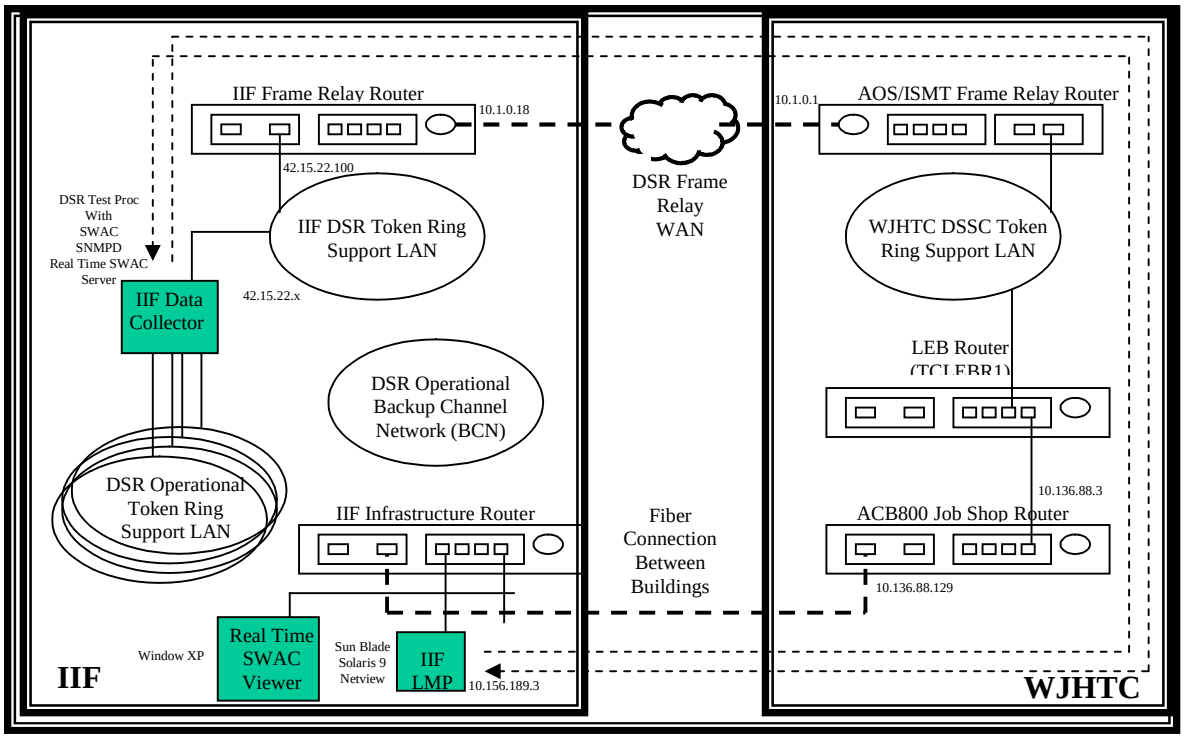

Figure 8-1 IIF Centralized Maintenance POC Architecture

8.4 Data Collection Methods

8.4.1 DSR SAR Operations

8.4.1.1 System Wide Access Capability

System Wide Access Capability is a DSR data collection tool that was developed as a part of the Demonstration, Test, and Evaluation (DT&E) phase of the DSR acquisition. SWAC is a first level of filtering to raw DSR SAR. SWAC may be run in online or offline mode. In offline mode, DSR SAR can be presented to the SWAC application in the form a disk file or in IBM 3490 tape format. In online mode, SWAC attempts to tap the operational network, either the LCN Token Ring network or Backup Channel Network (BCN) which is Ethernet. In either mode, SWAC requires an input filter set identified by a file named “swac.mdx”. This filter file represents the data of interest that SWAC should pull from the raw DSR SAR stream/buffer. The filter file also provides formatting information for the various data types to be collected. SWAC submits formatted data to UNIX Standard Output (i.e. the command line prompt).

As stated above, online SWAC requires a DSR SAR stream to be available on the operational network for processing. Note that when DSR SAR tapes are not mounted, processors begin to save data that is bound for SAR locally. In this situation, online SWAC would not have access to this SAR data. Therefore, at least one SAR tape must be mounted to the DSR SAR processor to obtain a DSR SAR stream.

8.4.1.2 System Health Report

The System Health Report (SHR) is a set of statistical reports that are generated by processing SWAC output data. The SHR is used widely in the WJHTC DSR System Verification test groups which are also a part of the DSR ISMT. These reports identify various metrics and errors that collectively represent the utilization and performance of the DSR System. This information can be used to create a baseline of performance for the DSR System, to diagnose and resolve problems and to enhance the awareness of the maintenance staff in their system maintenance, system planning and problem resolution duties.

8.5 Centralized Maintenance Proof of Concept Tasks

8.5.1 Software Architecture

For this task, an additional standard DSR test processor was installed in the IIF to act as the Data Collector (DC). The IIF DC was networked to the DSR LCN, the DSR BCN, and the IIF DSR Support LAN connection. The LCN and BCN connection provided connectivity to operational DSR SAR streams so that online SWAC could be executed. The Token Ring Support LAN provided remote connectivity to the WJHTC though the DSR Frame Relay WAN cloud. Using this configuration, the IIF DC, which represented an ARTCC SWAC Data Collector, could communicate with the IIF LMP, which represented an SNMP-based WJHTC Second Level management/maintenance entity. The IIF LMP - DC connection facilitated all remote System Health Reporting operations. Real Time SWAC Viewing was implemented using a Microsoft XP Personal Computer with a client application that connected to a SWAC Server running on the IIF DC (See Figure 8-2 below).

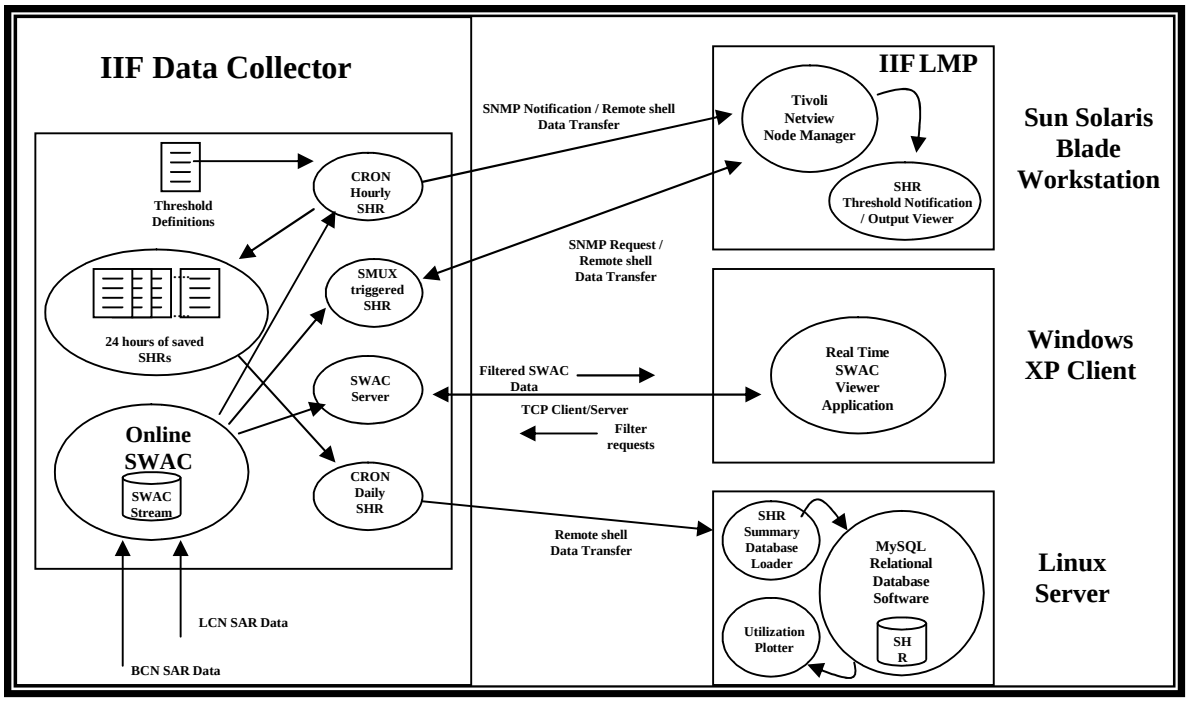

Figure 8-2 IIF Centralized Maintenance Software Architecture

8.5.2 Using Health Reporting for Centralized Maintenance Notification

The System Health Report has been identified by the DSR ISMT leadership as a valuable set of information for system maintenance and problem resolution. Remote access to the ARTCC subsystems by DSR ISMT personnel located at the WJHTC is limited. A reliance on the local support personnel is necessary. In addition, there is no provision to provide real-time system data from the ARTCCs to the ISMT staff for problem diagnosis and resolution. While SHRs are valued and provide a great deal of information, the opportunities to create the SHRs are limited. The goal of this task was to demonstrate possible opportunities to create SHRs, to make them readily available to ISMT staff and to increase the awareness of the ISMT staff of the DSR Systems of all of the ARTCCs. There are other types of information that have value in these activities but the SHR was selected to demonstrate this concept.

8.5.2.1 Periodic Health Reporting with Event Driven Notification

The SHR contains numerous information types that represent various subsystem performance metrics and errors. Some of this information (or particular values of some of this information) represent subsystem problems or anomalies that are of particular interest to the ISMT maintenance staff. The developed concept presents a means to alert the ISMT staff of these events and provide SHRs on demand for further analysis.

The IBM Tivoli Netview network management application was the basis for the concept representative ISMT user interaction. The Netview application provides a graphical user interface (GUI) that logically represents network topology and network object status that enhances the awareness of the user. It therefore represents a logical platform to base this

awareness enhancing concept. Various scripts (i.e., UNIX Korn shell, TCL/TK, Netview X-Windows Motif) are invoked through the selection of prepared Netview menu items, or through Netview SNMP Trap initiated script execution (via Netview Event Correlation).

In this concept, an application at each ARTCC will periodically (hourly) generate an SHR and search the SHR set for errors or items of particular interest to the ISMT staff. Upon detection of these items, the application will send an SNMP Trap to the Netview application located at the WJHTC to alert the ISMT staff member. The configured Netview application will receive the Trap and execute a script that will present a pop-up window alerting the ISMT staff member that a significant event occurred. The pop-up window will also contain some detail about the actual event. In addition, the pop-up window will provide an opportunity to request the set of SHRs to be packaged up and sent to the Netview host for subsequent viewing and analysis by the ISMT staff member.

Figure 8-3 Netview SHR Notification Alert Pop-Up Messase

The request for the SHR is conveyed to the specific ARTCC Data Collector SHR application via SNMP and is initiated by the acknowledgement of the alert pop-up window (See Figure 8-3) or through the selection of a prepared Netview menu item. The ARTCC Data Collector SHR application bundles up the set of SHR reports, sends them to the WJHTC Netview host via the UNIX remote file copy (RCP) utility. The ARTCC Data Collector SHR application then sends an SNMP Trap to the WJHTC Netview application to notify the ISMT staff member that the reports are available for viewing (See Figure 8-4).

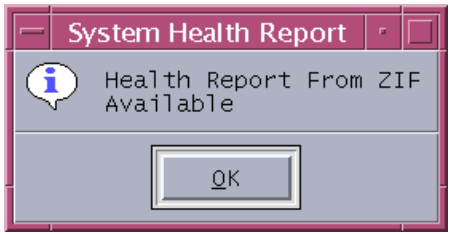

Figure 8-4 Netview SHR Availablility Notification Pop-Up Messase

The acknowledgement of the notification pop-up window invokes a script that unpacks the sent SHR files and places them in the appropriate place in the file structure according to ARTCC and the hour that the SHR represents. The prepared Netview menu selection subsequently presents a submenu of the various constituent reports of the SHR and a viewing window for the reports.

An example of the menu structure is detailed by the following illustrations: Figure 8-5 shows on the main Netview toolbar a menu heading entitled "ARTCC_Health_Check_Reports". A submenu identifies each ARTCC including the WJHTC (ZCY) and the IIF (ZIF). A submenu for each ARTCC allows for the solicitation of the health reports ("Get Health Reports"), or for the viewing of the health reports ordered by the hour ("0000", "0100", "0200", "0300", etc.).

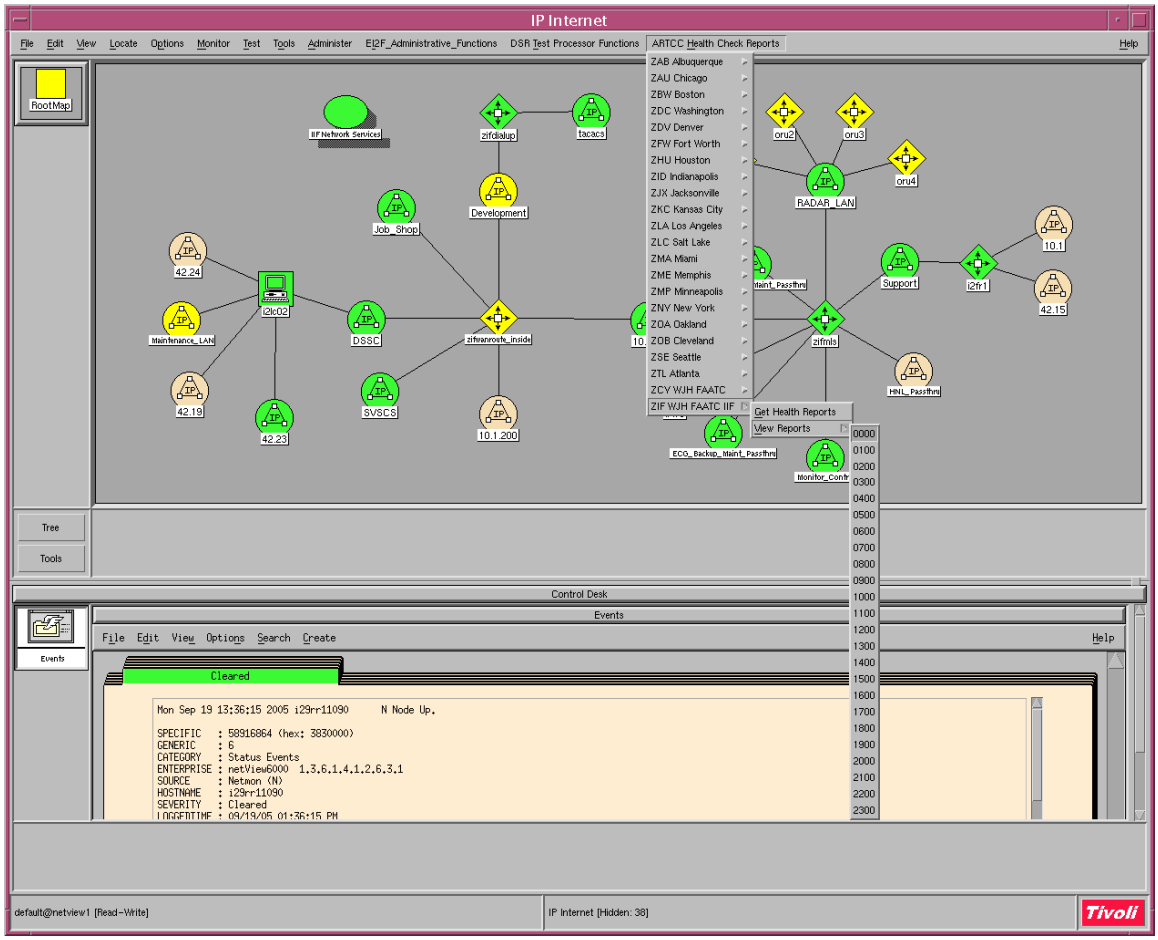

Figure 8-5 Netview ARTCC Health Check Report Menu

The Figure 8-6 shows the report browser window that is presented by selecting a specific start time on which the reports were generated. The menu in the browser window

organizes the reports according to the LCN or BCN networks. In the Figure, the LCN subset is selected and the subset of LCN report names is revealed in a sub menu.

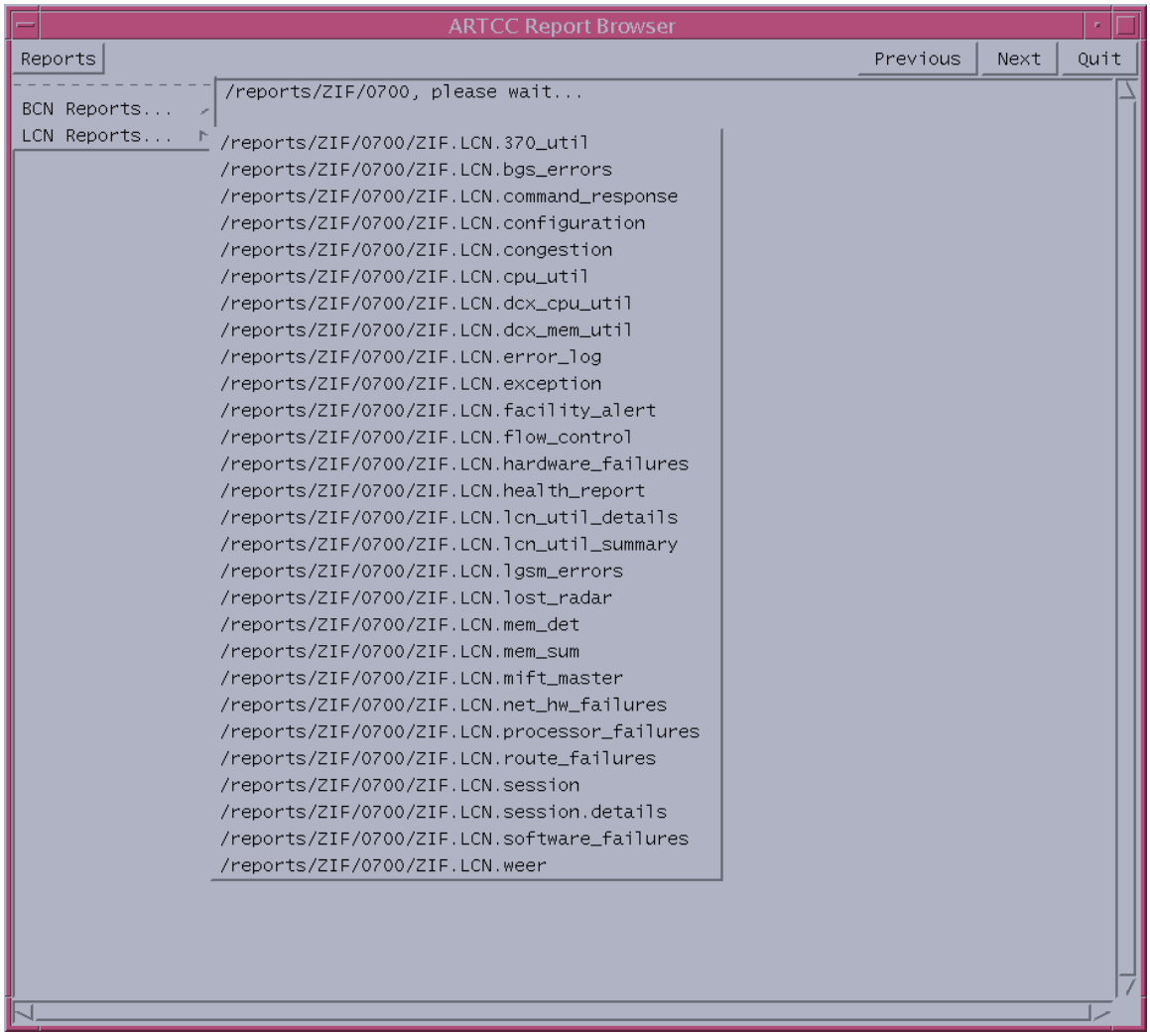

Figure 8-6 SHR Report Browser

Figure 8-7 shows a selected report (the LCN health report) in the browser for viewing. Any report that is larger that the browser window size is presented in a browser window with scroll bars. In addition, the reports can be sequentially selected for viewing by depressing the “Previous” and “Next” buttons of the browser window.

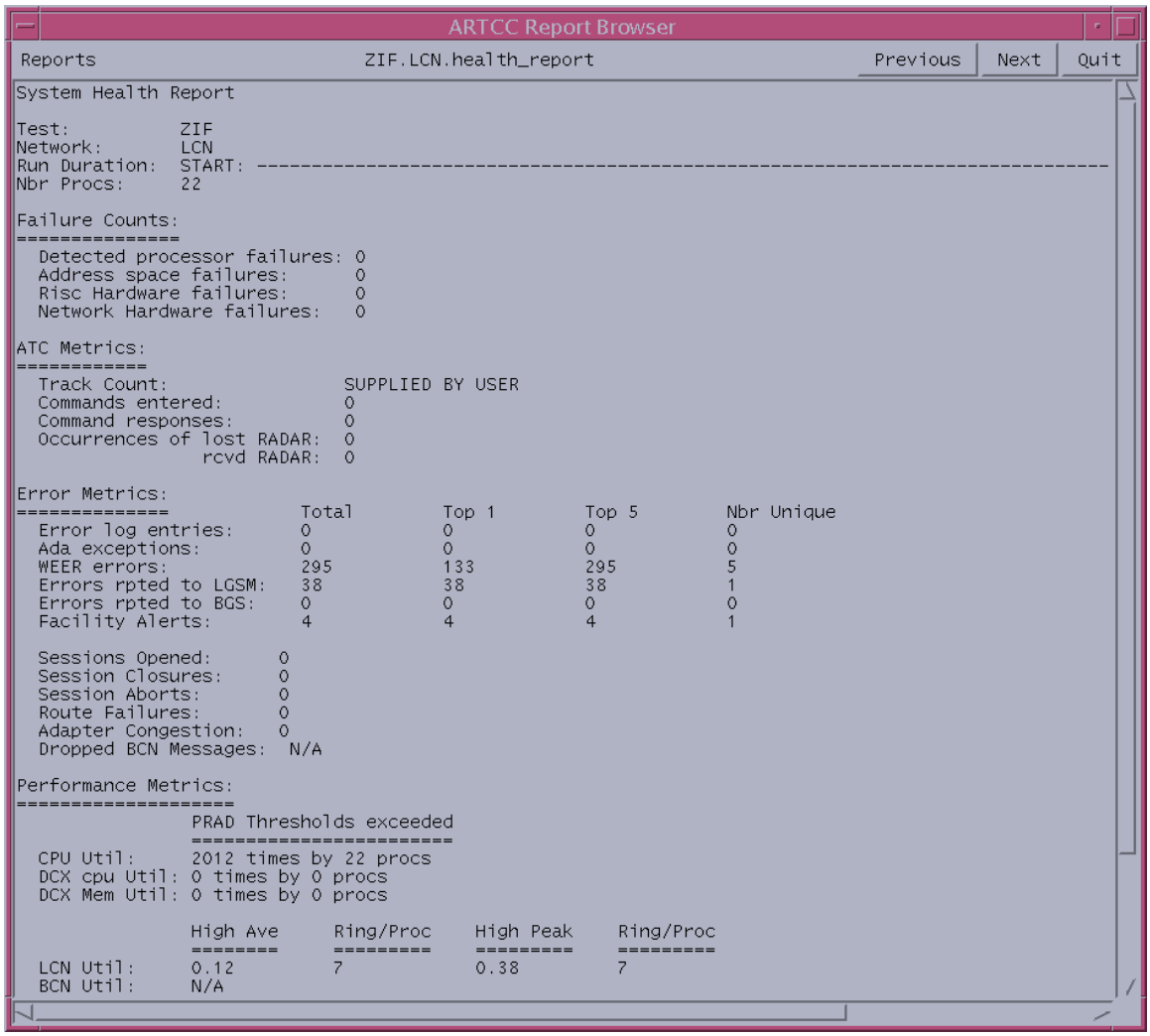

Figure 8-7 SHR Report Viewer

8.5.2.1.1 Selected Notification Threads

An example of how notification of potential DSR health problems could be issued was implemented in this task using an SHR threshold configuration file (See Appendix B shr_thresholds.config). This file contains a listing of SWAC filters mapped to a number of allowed occurrences (a threshold) of that particular filter within an hour's worth of DSR SAR data. When the hourly SHR is run, this threshold file is checked against the SHR output to assess the need for remote notification.

8.5.2.2 Routine Health Report Archival

The System Health Report can be used for trend analysis and subsystem improvement planning. This concept was represented by the daily creation of SHRs that represent 24 hours of processed data. These SHRs get posted to an SQL database using routine database load methods. These data are then available for analysis that could reveal trends or otherwise enhance the awareness of the ISMT staff to normal processing conditions and sensitizes them to anomalous system behavior.

8.5.3 Centralized Real Time debugging using a SWAC Server and Remote Access Client

In some maintenance situations, it may become necessary to remotely view data representing real time software processing. To explore this situation, the IIF has developed a real time online SWAC data viewer. This application may be used to remotely view SWAC output from ongoing DSR software execution.

8.5.3.1 Real Time SWAC Viewer Application

To facilitate a real time SWAC stream, a TCP Client/Server application was implemented. The SWAC Server was implemented on the IIF DC. Upon initialization, the SWAC Server opens a UNIX operating system pipe that executes the SWAC executable using a specified "swac.mdx" filter as input. This pipe outputs a SWAC stream that may be forwarded to connected clients. The SWAC Server also opens a TCP server socket and listens for client connection requests. When a client requests a connection, the server honors the request and closes the listening TCP socket. The SWAC Server then sends the "swac.mdx" filter information to the client and then reads the established client socket for client submitted SWAC filter request data.

The IIF Real Time SWAC Viewer Client was implemented on a Windows XP platform using Visual Basic Version 6.0. Visual Basic was chosen because it allows for rapid GUI development. Upon initialization, the SWAC Client allows the user to connect to a particular ARTCC. Once the connection is established, a SWAC filter set is sent to the client by the server. This filter set is defined by the "swac.mdx" filter file that is currently being used by the SWAC Server. Upon receipt of the filter set, the client presents the available filters to the user. The user may then select a set of filters for viewing and submit that set to the Server. Upon receipt of a Client requested set of filters, the Server begins to examine the SWAC pipe stream line by line against the filter set. If a match is found in the SWAC stream it is forwarded to the Client. The Client, in turn, presents the received SWAC output to the user. The Client User interface also allows the user to apply a third level of filtering (include and exclude) by providing key word(s). The Real Time SWAC Viewer Application Client User Interface is depicted in Figure 8-8.

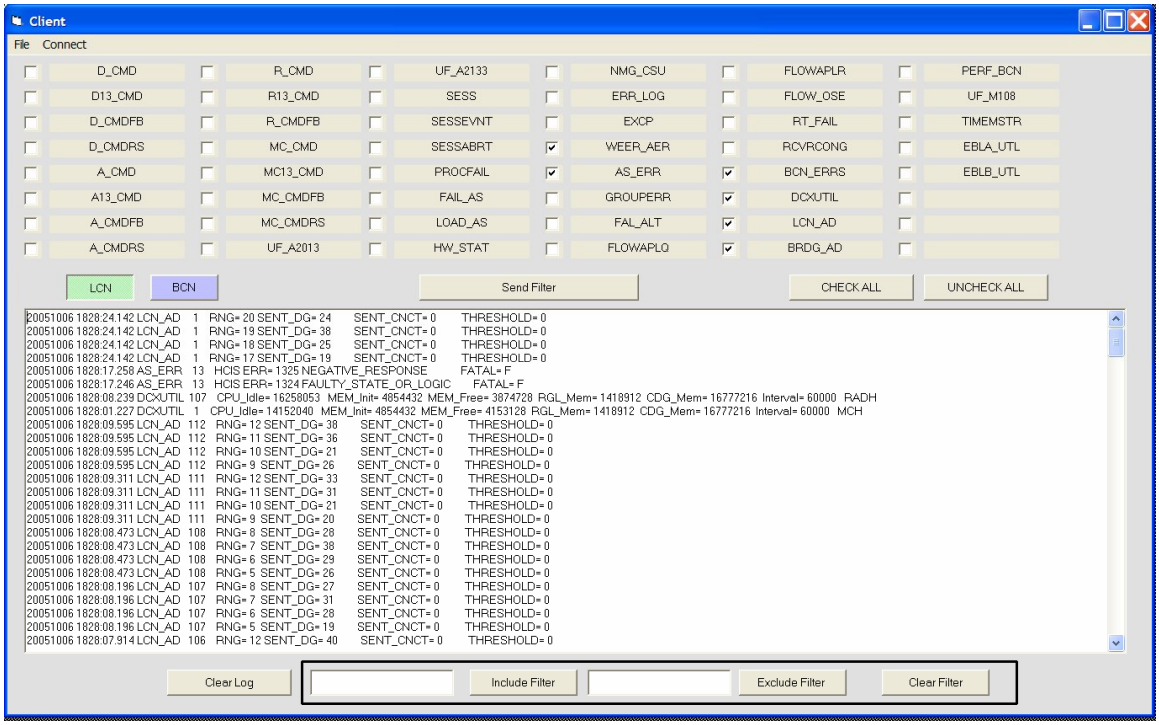

Figure 8-8 Real Time SWAC Viewer Application

8.5.4 Information Trend Analysis Using Data Mining

8.5.4.1 Data plotting using GNU plot

A basic trend analysis example was also provided as a part of the Centralized Level Two Maintenance task. The example uses the LCN Utilization Summary Report, which is part of the SHR set, to provide graphical plots of DSR Token Ring network utilization across multiple days. The LCN Utilization Summary Report data is obtained from archived SHR data that was placed in a MySQL database. The data is extracted using a UNIX Scripting language called Tool Command Language/Tool Kit (TCL/TK). The data can then be plotted using an application called Gnuplot.

Gnuplot is a UNIX plotting utility. It provides for the display of the SHR LCN Utilization Summary data vs. date. A master TCL/TK script is executed on a Sun Solaris Version 9 host to provide the main GUI (See Figure 8-9) for plotting the performance data. The TCL master script contains a software library that allows it to connect to and query the MySQL database application.

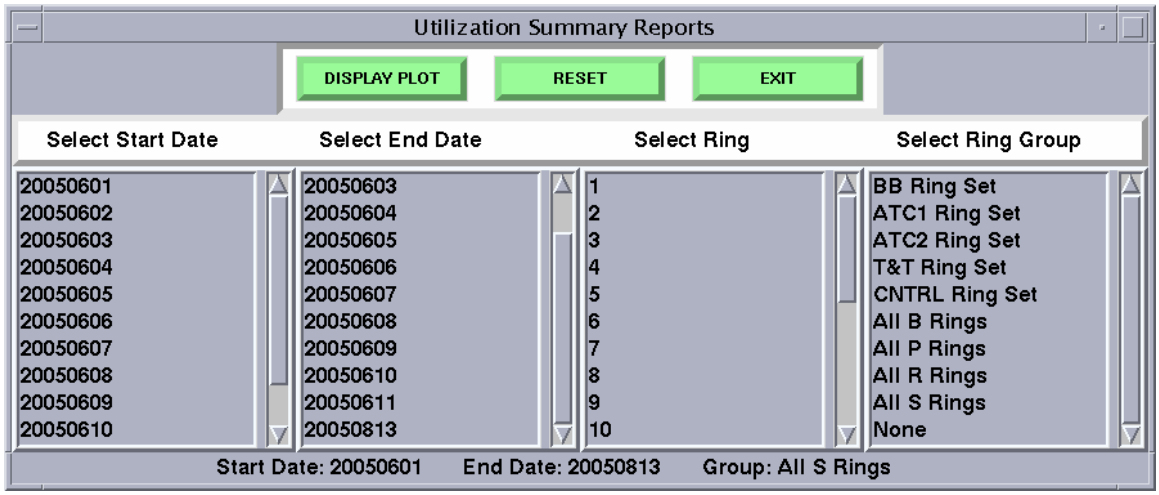

Figure 8-9 LCN Utilization Summary Plotter

From the GUI, the user may select a start and end date which determines the time period for which the plot is displayed. The user may also select an individual Ring and/or Ring group for plotting. The Ring groups include: BB, ATC1, ATC2, T&T and Central Ring Sets and all B, P, R or S Ring sets. The user may then select the DISPLAY PLOT button. The master script then obtains the performance data from the MySQL database and invokes the Gnuplot utility to display the plot. The current plot capability includes the LCN Summary Average and Peak Utility values for the selected Ring or Ring group. An example plot is provided in Figure 8-10.

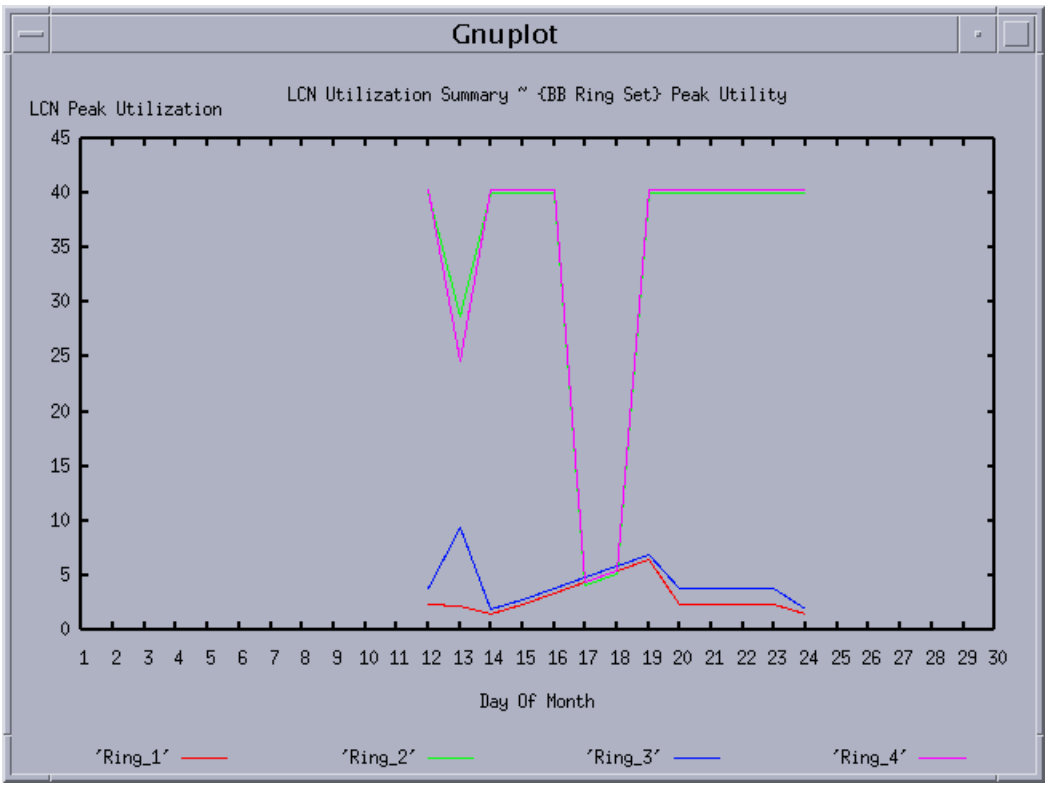

Figure 8-10 – LCN Utilization Summary Plotted with GNU Plot

9 Maintenance Approaches for Commercial off the Shelf Equipment in the En Route Environment

9.1.1 COTS statistics and data flow characteristics gathering

Industry standard practices recommend developing a system baseline to understand the signatures of a 'normal' operational environment. [Ref: ERAM SIG 306]. An awareness of system parameters and behaviors exhibited when the system is performing to expectations and the establishment of a baseline set of measurements, provides a greater sensitivity to anomalous behavior and enhances problem determination and resolution. Periodic analysis of system performance in the context of the baseline measurements can reveal potential problem areas (i.e. resource exhaustion, capacity saturation, failure trends) before problems manifest. Maintaining a network baseline within the En Route infrastructure and tracking deviations in traffic patterns from that baseline as well as actively looking at selective network data can assist the maintainers of the En Route environment in providing a more efficient and proactive maintenance service. There are various mechanisms that could be utilized to gather and analyze the network specific data required to baseline an En Route 'system'. As industry standards and practices continue to develop and mature, there may be further capabilities to perform these types of tasks. Some mechanisms are discussed in the following sections.

Currently within the En Route environment, the DSR ISMT collects baseline data on site configurations on an occasional basis. The ERAM SIG 306, "Centralized System Maintenance Support", identifies the need to collect and analyze trending data to make it easier for ERAM system maintainers to execute the following types of tasks:

- Search for common failures, problem trends, and lower-level system events; understand correlations between workload and resource utilizations
- Develop historical usage rates; maintain current inventories
- Identify changes in site hardware configurations
- Get automated notification, in near real time, of events for assisting in real-time resolution
- Perform site radar analysis.

Some data are identified in ERAM SIG 306 which will be required to be captured for maintenance support activities. Although many details remain to be determined, the general concept of the ERAM maintenance environment has been identified.

9.1.1.1 Scripting Using SNMP

One mechanism that is used to gather network data is the IP SNMP requests to retrieve data from network devices. SNMP is a standard protocol that conveys information contained in MIBs. As stated previously, a MIB is a database of objects that can be monitored and manipulated by a network management system.

Defining a mechanism to track indicators of a healthy network is not an easy task and it is not an area where many industry experts have lent their opinion. However, Cisco Systems has published a White Paper on this topic and the Internet2 E2E piPES Project (http://e2epi.internet2.edu/e2epipes) has made some recommendations in this area that may provide a good starting point for establishing a network baseline. Additionally, the Internet2 working group has written a document entitled, “Network Performance Measurement Tools: An Internet2 Cookbook”. This document specifies the following network problems which the group finds as affecting network performance of most applications: packet loss, jitter, out-of-order packets, excessive latency, and duplicate packets. Cisco Systems recommends capturing the following MIB data: cpmCPUTotal5min (the five-minute decaying average of a Cisco devices's central processing unit) and polling for this data on five minute intervals. Cisco also recommends polling this variable at a minimum over a two-week period so two weekly business cycles of data can be evaluated.

As an example of these types of polling, the IIF developed a function that requests various types of information pertaining to network device interfaces. This information is reported via a Netview GUI representation (See Figure 9-1). This example can be tailored to poll for requests on various device data that can be retrieved through this mechanism. This or similar capabilities can be developed, expanded, and/or tailored as needed, to satisfy En Route specific needs.

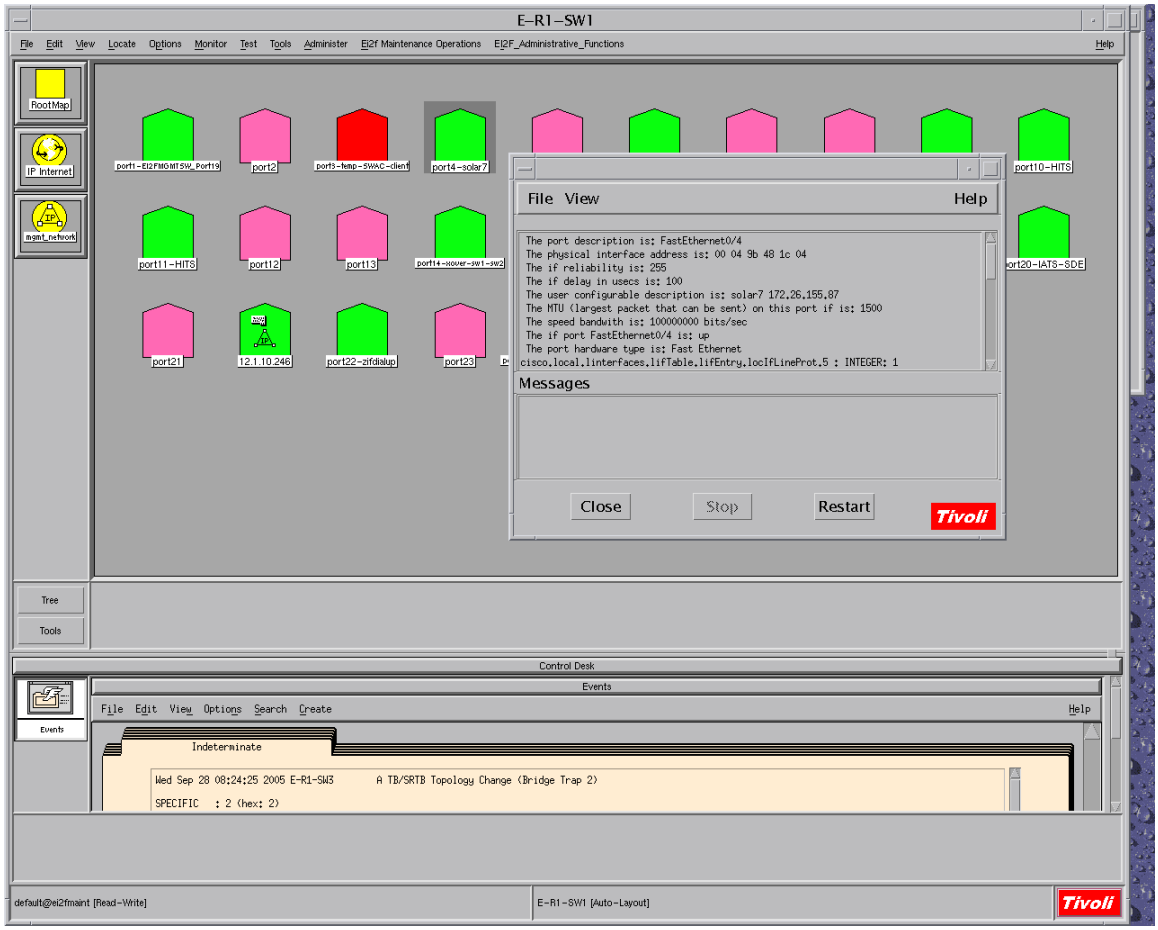

Figure 9-1 Netview Solicited Network Switch Port Statistics

9.1.1.2 Standardized COTS Network Data Collection Mechanisms

One of the means for collecting network appliance data through COTS approaches in the past was a mechanism known as Remote Monitoring (RMON). Statistics from compatible network devices would be queried, via an SNMP Request, to send specific data to a receiving device/application. In the past, RMON was a standardized approach and provided much needed information used for network baseline creation and for generating network snapshots. However, RMON is taxing on the network appliance processors and could affect network performance. Because of this, industry experts have started to look at other avenues of gathering needed network data as RMON is being obsolesced.

The En Route modernization programs have included the introduction of Cisco Systems network switches and routers into the En Route baseline. Because Cisco Systems has been selected as the networking vendor for En Route, it was decided to investigate one of the vendor's new applications designed to collect network data flows called Netflow. Among the capabilities the Netflow application offers are means to evaluate network

bandwidth, performance, quality of service, and security as well as network and application resource utilization. Unlike RMON, Netflow collects these types of data without huge demands on network device processors.

Netflow is a new application which is starting to get attention from multiple network vendors. Although not a standard, the Netflow product stream is being submitted as RFC 3954 "Cisco Systems NetFlow Services Export Version 9". It is also developing and growing and may be very beneficial in the future. As part of an effort to show viability in this area, the IIF has installed a dedicated Netflow platform (Solaris 9) which collects data flows from a Cisco 6509 Catalyst switch engine. These flows are displayed via a support tool in the Netflow application suite of tools. Netflow is an application and concept that is continuing to mature. By the time ERAM is deployed to the field, Netflow, and possibly other tools like it, could provide various baseline ERAM network measurements. In the meantime, the Netflow concept can be explored further to assess its usability and viability within the En Route environment.

9.1.2 SNMP-Based Management Mechanisms for COTS network devices

SNMP is a very useful protocol that can be used to automate network management functions via in-band requests to devices from a management platform. The IIF has experience in utilizing network node managers to initiate management requests to network devices. Currently, the IIF is using Netview Node Managers as part of its management infrastructure. Netview extensions, which use an SNMP API, have been added to the suite of management tools at the IIF. Additionally, the SNMP protocol is utilized within applications written in scripting languages for some IIF management functions. Some the SNMP-based concepts utilized within the IIF as part of its business practice may fit into the En Route environment. Some of these concepts are explained in the following sections.

9.1.2.1 Using SNMP MIBS for COTS Devices

As stated previously, a MIB is a database of objects that can be monitored and manipulated by a network management system. Standard MIBs are the common sets of objects communicated via the SNMP amongst network devices and network managers. MIBs are the primary means for determining network appliance identity and management control of network appliances. Almost all network appliance vendors and network manager vendors provide standard MIBs with their COTS applications and appliances.

The Tivoli Netview application as purchased is loaded with many standard MIBs. Additional MIBs can be loaded and become immediately useable within the standard Netview functionality. Software that enhances Netview capabilities can also be integrated to take advantage of these additional MIBs.

9.1.2.1.1 SNMP Extensibility/Netview Extensible Applications

The Tivoli Netview Network Manager application uses standard network protocols, including SNMP and MIBs, to detect network devices and creates a database of network objects representing those devices. The Netview application graphically displays the set

of network objects, and their inter-relationships, in layers of presentation. This presentation depicts network objects using specific symbology and indicates their inter-relationships by connecting lines or symbols. The level of graphical detail and control function provided by the Netview application in representing these network objects, without customization, is limited to identity and network orientation and status.

As mentioned previously, the Netview Node Manager utility is enhanced by an API to integrate additional management, control, and/or monitor functionality. For example, the IIF staff has developed and integrated software, utilizing SNMP and MIBs, that increases awareness of network switch device ports. The software provides a control function for these ports through a Netview GUI (See Figure 9-2). An additional layer of graphical presentation (i.e. submap) for each switch reveals a set of icons depicting all of the ports of a given switch. Additional polling action discerns the status of each port and the results of this action are displayed through an enumerated color scheme, representing various port states, applied to the identifying port symbols. This software also provides a Netview GUI for port control. This allows the user to alter port descriptions in the switch configuration, to enable or disable ports, and to query and display the status of each port. This additional function extends the common GUI approach to network device management and reduces the need for arcane knowledge of network device command-line syntax.

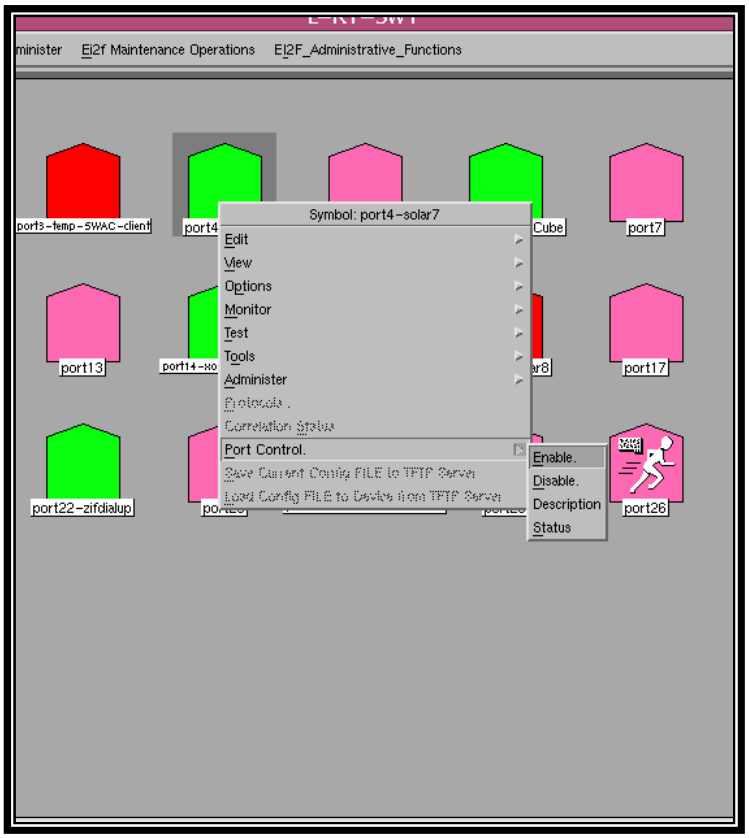

Figure 9-2 Netview Network Switch Port Control Submenu

9.1.2.1.2 Network Devices Backups

Automating scheduled network device backups is a means to improve efficiency, provide cost savings, and centralize storage of the backup files. In the case of the IIF, a portion of the scheduled system backups are saved to a trivial file transfer protocol (tftp) server. More specifically, network device configuration files are copied from Cisco Systems network devices to the IIF tftp server via SNMP requests. This is achieved via scripting and is initiated via UNIX CRON with pop-up window built into the Netview GUI. The IIF Management Netview server reports the status of the weekly device backups. Status is displayed in a pop-up window either successfully completed or unsuccessful in the backup attempt (unsuccessful indication provides specific devices - See Figure 9-3). In addition, email notification of the event completion is sent to key personnel so they are aware of the backup completion and status. All files are saved in a name and timestamp manner that clearly depicts which device configuration the file contains as well as when the backup was made. This process is part of a larger system backup process which is executed on a weekly basis. The files are easily accessible from the network devices and other systems on the management network. If a device configuration is accidentally corrupted, a configuration which is no older than a week prior can be loaded onto the network device from the tftp server and executed.

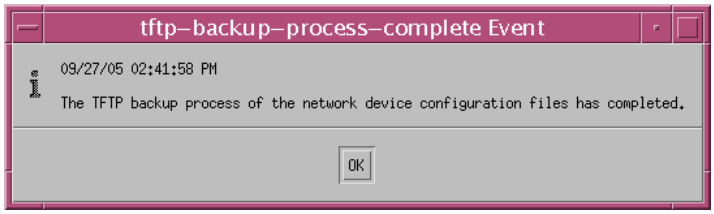

Figure 9-3 Netview tftp Network Device Backup Completion Notification

9.1.2.1.3 Remote Access Methods

Access to Site Data - Second-level maintenance personnel can access remote site data in two fashions. The first would be for direct access to the data required via a remote session into a device residing at the site. The second would be for the data to be sent to a device within the centralized maintenance environment where the personnel can have access to the data. It is assumed that access to all data would require the usage of existing FAA WAN connections (currently DSR Frame Relay and BWM; future FTI). There are no known plans to add additional WAN connections for maintenance purposes.

Access to Site Devices - Access to site devices from the centralized second-level maintenance location would provide maintenance personnel the capability to not only directly access data, but also adds the possibility of remote monitor and control, debugging, configuring, etc. While these types of capabilities may improve the overall effectiveness for troubleshooting and fixing some problems, it is assumed that these types of actions will only be allowed by on-site personnel. [Note: there are provisions within

ERAM where a remote maintainer could be provided access to site support processors for remote maintenance purposes.]

9.1.3 Using a Management VLAN Network for WJHTC Lab Operations

This concept provides for the separation of network communication and data flows from the LAB operations, from those of the management activity of the network, and is referred to as in-band network operations and out-of-band network management. This approach assures the fidelity of LAB operations communications and data flows, and minimizes the potential disruption caused by management activities on the network. The management of the LAN is enhanced by the security, the awareness and the flexibility afforded by the network partitioning, and the focusing of the management domain on the management task.

9.1.3.1 VLAN Description

The technical definition of virtual LANs (VLANs), as set forth by IEEE, is that they represent broadcast domains in an Open System Interconnection (OSI) Layer 2 network. VLANS can be further defined as a logical grouping of network users and resources connected to administratively defined ports on a network switch.

A broadcast domain is a set of network segments in which every host within the set has direct communication access to every other host within that set. A broadcast message, from which a broadcast domain gets its meaning, is a message that every listening host in that domain must process (this usually means wasted processing resources); to contrast, a unicast message is a message that has one sender and one receiver. Inter broadcast domain access requires a routing process (OSI Layer 3) and this process constrains broadcast communication to its source broadcast domain; broadcast traffic does not automatically traverse broadcast domains. This physical partitioning of networks, that is broadcast domains separated by routing processes, is a desirable if not a necessary network design goal. Smaller broadcast domains reduce opportunity for network congestion and wasted network resources, and more broadcast domains partitioned by routing processes provide more points to implement security policy. Broadcast domains are fundamentally a set of physically connected hosts, via network switches (OSI Layer 2), which represent a common network. This set of hosts may be connected to a routing process for inter-domain communication.

VLANs are logical representations of broadcast domains that may span multiple physical networks or multiple portions of multiple physical networks. The logical characteristic of VLANs provides for optimal design of the physical network topology, optimal logical design of security partitions, and logical grouping of geographically dispersed workgroup hosts/servers. In short, VLANs provide a means to create logically separated LANs superimposed over a physical LAN.

9.1.3.2 How Management VLANs Work

Communication among hosts through network switches and routers in a network is characteristic of a LAN. Network appliances such as network switches and routers can be

configured to be objects of communication as well. Each appliance can be configured with an Internet Protocol (IP) address that becomes its identity for network communication. Typical network design results in a common set of hosts being grouped in a broadcast domain. The host's unique IP addresses are represented in a defined range of addresses, called a sub-network. In the Management VLAN model, all network appliances are grouped as members of a Management broadcast domain and the Lab operations hosts are members of one or more other broadcast domains. This separation allows for communication in the management broadcast domain to occur without disrupting the Lab operations activities.

The configuration of various network protocols allows for subsets of network switch ports or individual ports, to be assigned to specific broadcast domains called VLANs. This means that while physically co-connected to the same switch, communication among hosts of a VLAN on a network switch are logically separated from the communication among hosts in another VLAN on the same switch. In this model, the network switch is given an IP address in the Management VLAN and the network switch device ports are assigned accordingly to other Lab operations VLANs.

When the network traffic of multiple VLANs flow over a common network connection between interconnected network switches, additional network protocols bundle the VLANs into a trunk connection, to take advantage of the physical topology, while keeping the VLANs' data separate.

This bundling, while maintaining separation is analogous to airline travelers passing through a common security checkpoint corridor. While groups of people are separate from other groups as they check-in at specific airline counters, and are destined for corresponding airline gates, they all traverse the same security corridor. The logical separation mechanism, in this case an airline boarding pass, logically separates individuals from a specific group from other individuals from another group as they traverse a common physical path, the security corridor. As a group, a specific airline's passengers reemerge at the airline's concourse. In this analogy each airline represents a broadcast domain or VLAN. The security corridor represents the trunk connection physically connecting the airline counter area (source switch) to the concourse area (destination switch). The boarding pass represents the temporary VLAN identification for each passenger (data frame) during the traverse of the security corridor (common trunk interconnection). In a similar way, the various VLANs' traffic are kept separate during the inter-switch traversal by pre-pending a special header to each data frame that identifies the frame's VLAN. As the data frame reaches the destination switch, this header is stripped off and the frame is sent to the switch ports that represent the appropriate VLAN. According to airline analogy, as each passenger prepares to enter the aircraft at the destination gate (or destination switch port), the boarding pass (the temporary VLAN header) is surrendered and the passenger enters the passageway (switch port) to the aircraft (the destination HOST). This mechanism allows the trunk connection to convey all VLANs' traffic from one switch to another, while denying inter- VLAN crosstalk (one airline's passenger boarding another airline with the former airline's boarding pass), thus keeping broadcast domains separate.

The Management VLAN includes a network management server running the Tivoli Netview network management application. By its location in the Management VLAN this platform has direct IP accessibility to all network appliances. It is strategically placed to perform Monitor & Control management activities. This architecture enhances awareness of the physical topology of the network, the logical topology created by VLANs, and the normal states of all of the network objects identified by Netview. This enhanced awareness facilitates the quick detection/resolution of problems and aids in the implementation of network modifications. Figure 9-4 is the Netview presentation of all the network switches and routers residing in the IIF Management VLAN. All of network traffic for managing these devices is conducted on the Management VLAN and does not interfere with network traffic on other network switch device VLANs. The management is thus is called 'out-of-band'.

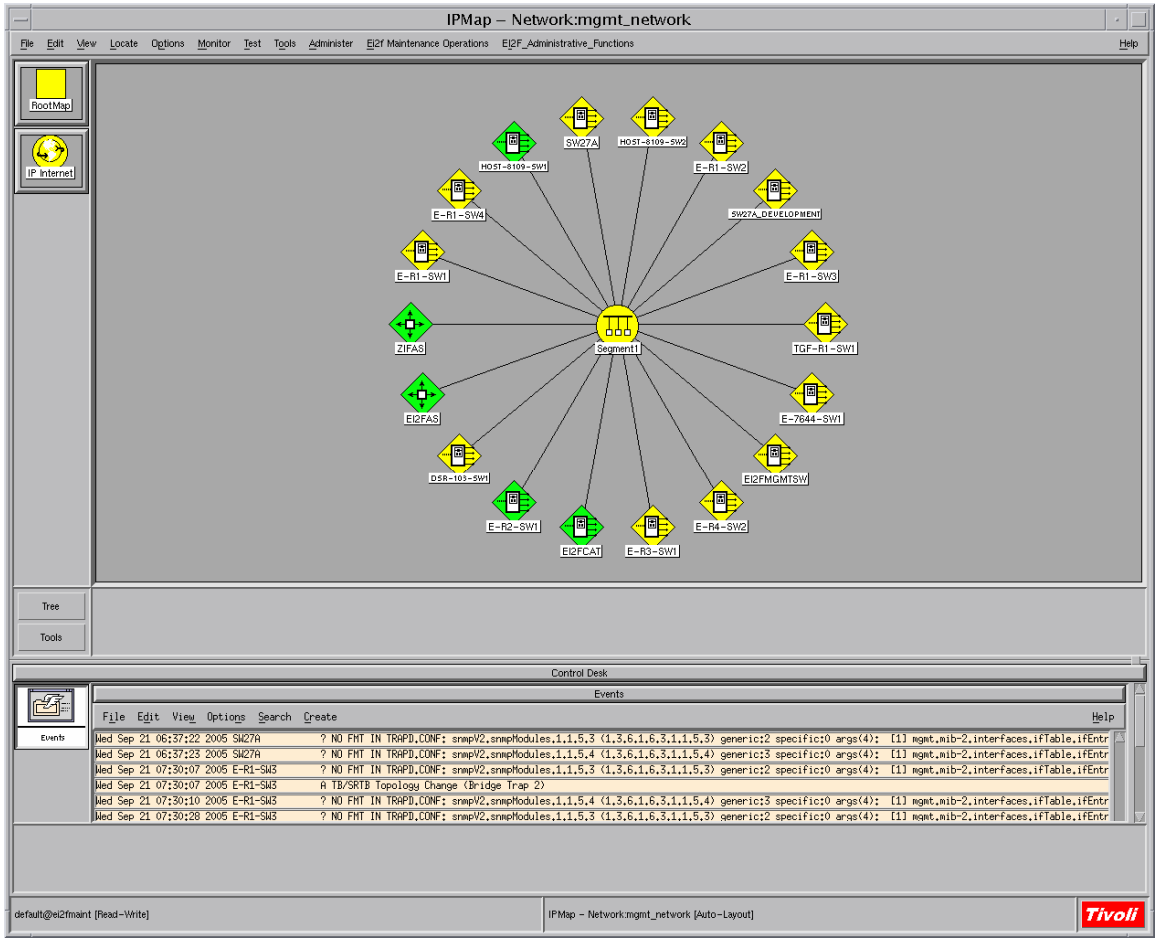

Figure 9-4 Netview Network Switch Port Control Submenu

9.1.3.3 VLANs for Laboratory Switching and Port Management

VLANs are established and modified by manipulating the configurations of the network switches and routers. This characteristic of VLAN management provides great flexibility and scalability in accommodating network resource reallocations. VLAN control is at the

switch port level within the device configuration. An out-of-the-box physical switch that normally has all switch ports configured onto a single VLAN can quickly be partitioned into multiple VLANs.

Cisco devices identify VLAN number 1 as the Management VLAN by default. This VLAN, by design, includes the sub-network that represents all of the IP addresses of the network appliances to be managed. Through this sub-network, any of the network appliances can be accessed and their respective configurations may be modified. As programs and projects appear in the IIF, the physical network topology can quickly be partitioned into the appropriate number and size of VLANs to accommodate the project's network requirements. While the Management VLAN infrequently changes, other project VLAN changes can occur as often as needed. These project changes usually require a switch port or set of switch ports be configured to another existing VLAN or to a newly created VLAN. The practical use of VLAN switching may be applicable in the ERAM test lab environment. If the ERAM test program requires a need of quickly switching test processors or lab equipment between specific roles, labs, and/or functions, VLAN switching within a management VLAN concept may be appropriate. A management VLAN may also be helpful in splitting lab resources using VLANS to accommodate multiple users within the same laboratory.

9.1.4 Wireless Technology and the En Route Laboratory Management Environment

Wireless LAN technology uses radio transmission, rather than physical wires, to interconnect network hosts and appliances, and facilitate communication. This characteristic provides great mobility potential for the network components and the users of those components. This is a desirable and popular feature because it provides greater freedom for a user to exercise his skills in the conduct of a computer based activity. This freedom shifts the emphasis from the tool to the user of the tool.

The current En Route laboratory management environment consists of many host computers that are interconnected by physically wired networks. This topology is permanent for the life of the subsystem. System maintenance personnel must access the subsystem components at designated access points or directly at a particular component's location. This constraint demands a focused effort by the system maintenance personnel that can preclude other relevant activities or awareness. Maintenance procedures and security policy are heavily influenced by this physical characteristic and therefore incorporate forms of systematic inefficiency.

9.1.4.1 Where is Wireless Appropriate and Useful

Wireless network technologies offer a system maintainer a flexibility in operations that has yet to be established within the design of the En Route architecture. Traditionally, each equipment rack has either a serial-based maintenance console and/or a serial-based console on a maintenance cart for system accessibility for maintenance actions. The system maintainer must go to the specific access console, or push the console cart and connect to the equipment of interest. However, with wireless technology, system maintainers can quickly access the appropriate equipment from their current location

(assuming the maintenance console functions are based on what will be termed the maintenance LAN). They also have the flexibility of working from a laptop computer which is mobile and can access the network while it exists within the radio range of the access points. The significance of this flexibility is not limited to convenience and savings of time. It also allows the system maintainer to interact with other location displaced, relevant system components while wirelessly accessing the equipment of interest. This new ability enhances the awareness of the system maintainer and provides opportunity for more effective problem determination and resolution methods. Wireless network access points, the radios that connect the wireless LAN to the wired LAN, can be placed strategically to allow the optimal amount of access coverage for access to a system's maintenance LAN. Additionally, all network appliance and En Route system maintenance LAN connections can be made without disturbing other critical system operations which utilize other LANs.

9.1.4.1.1 Cost Effectiveness

In a traditional En Route maintenance environment, there are no distributed wireless access points. A technician in this environment can log into a device's maintenance console at the device rack or a fixed point through an access server. However, a technician with a laptop outfitted with a wireless connection to the maintenance (or management) LAN has immediate access to all devices on the maintenance LAN. More importantly, the technician has the freedom to move about uninhibited. This type of portability allows a technician the benefit of going directly to the problem without having any time impacts on gaining access to the maintenance LAN from some physical location. Also a technician can interact with a simulation host while concurrently performing a maintenance or problem determination task on a wireless maintenance laptop. This kind of synergistic activity enhances the abilities and value of the maintenance staff. In addition, any tools required to assist the technicians in doing their jobs can be housed directly on the maintenance laptop computer. This type of approach represents a significant cost savings that spreads to equipment and equipment usage costs, training for maintenance technicians, etc. Specific equipment that could be eliminated in this type of environment are the maintenance consoles in the racks (laptops), access servers, etc.

9.1.4.2 A Secure Wireless Network Implementation

For purposes of applying a wireless solution within the context of an En Route maintenance LAN, the focus must be to provide protection from unauthorized users and to ensure accessibility to the functions required by the system maintainers. Wireless networking is a relatively new and continually developing technology. In order to exercise the most secure environment available, a layered approach to security is an industry standard practice. The assurance of a secure wireless infrastructure also implies the implementation of a secure wired infrastructure. Therefore, the implementation of a management LAN must ensure that security and data integrity are not impeded. As discussed in Section 9.1.3.2, an out-of-band management VLAN ensures maintenance data flows are logically separate from all other data flows on every switch. FAA Order 1370.94, Wireless Technologies Security, 2/3/05, provides guidance for the usage of wireless networks within FAA inter-networks.

FAA Wireless networking policy developers have proposed a wireless system architecture that represents a layered approach to security. Ideally, wireless devices such as laptops and Personal Digital Assistants (PDAs) would include anti-virus protection, personal firewall application, file encryption, a mobile cryptographic module and a policy manager. These measures protect the network against virus infection, protect mobile devices from network based attacks, protect against disclosure of sensitive data, and ensure currency of security policy measures. The constellation of wireless access points, that provide optimal network accessibility, would be centrally managed to ensure currency of security policy measures and would represent the beginning connections to the wired portion of the LAN. An encryption/decryption module would appear next and complement the same function on the mobile devices. A firewall would inspect wireless packets before entering the wired LAN, and wired side packets not destined for mobile devices would be blocked. Access attempts from mobile units would be challenged by Authentication, Authorization, and Accounting processing of userids and passwords. And to complete the model, an intrusion detection device would monitor the radio spectrum for intruders and unauthorized wireless devices, and ensure trusted users are operating securely.

9.1.4.2.1 IIF Maintenance Model

The wireless maintenance environment within the IIF was engineered to provide the greatest amount of layered security available at the time of implementation. The implemented design chosen was done so only after careful study of available protocols within the IEEE standard for wireless capabilities. This standard is known as 802.1x for security and wireless access within the context of a supplicant/authenticator/authentication server relationship. A supplicant is defined as the device which is requesting access to the wireless network. The authenticator is the wireless access point (Cisco Systems 1100), which provides the network connection. The authentication server in the IIF implementation is the Remote Access Dial in User Service (RADIUS). The 802.1x protocol uses Extensible Authentication Protocol (EAP) as its authentication framework. EAP is the transport protocol for the authentication method and supports various authentication methods. The protocols used in the IIF design include the Microsoft Supplicant on a Windows XP laptop computer, EAP-Protected EAP (EAP-PEAP)-Microsoft Challenge Handshake Authentication Protocol, version 2 (MSCHAPv2), Wired Equivalent Privacy (WEP). The design also uses a MySql database for EAP-PEAP-MSCHAPv2 management. PEAP is a specific method of tunneling EAP using an encrypted tunnel. This method authenticates a wireless user against a user relational database accessed by a RADIUS server through a wireless-to-wired bridge (authenticator or access point).

The protocols chosen for the concept wireless network were selected to ensure a high-degree of security, while also allowing flexibility for user accessibility. The concept model is built around a maintenance user requiring access to the maintenance (or in the IIF case, “management”) LAN, an access point, and an authentication server. The user, assumed to be on a laptop computer, utilizes a wireless network interface card (NIC) and a wireless client and protocols bundled into what is termed a supplicant. In the IIF

design, a Microsoft supplicant was used. In order to gain access to a wireless network, the prospective user must be authenticated and authorized through negotiations between the supplicant, the access point, and the RADIUS server. The IIF wireless implementation is presented to illustrate the utility of the concept, rather than to showcase an ideal architecture. If the value of the utility is realized, then the integrated wireless subsystem can be architected to ideal specifications for the En Route environment.

As discussed in paragraph 9.1.3.2, the IIF has integrated an out-of-band management VLAN that resides in the most trusted partition of the IIF firewall. The IP addresses of the switches and access server routers reside in the subnet represented by this VLAN. In addition, the wireless subsystem is configured into this VLAN. Through this VLAN, IIF staff equipped with a properly configured laptop, can access the switches and routers of the IIF to quickly perform configuration changes and monitor network activity. This ability is very useful when configuring for and conducting simulations, and for problem determination.

9.1.4.2.2 Free Radius using MySql

RADIUS is an access-control protocol that uses a challenge/response method to verify and authenticate users. It is a standard used in environments which require authentication, authorization, and user accounting. Collectively, these three requirements are known as an AAA process. RADIUS was the first AAA-based protocol to earn industry acceptance and to be widely use. The RADIUS protocol is defined fully by the IETF RFC 2865. FreeRADIUS is an open source RADIUS server project. The FreeRADIUS server was chosen for the IIF secure wireless network because of configuration flexibility, cost effectiveness (no cost), and user customization capabilities. FreeRADIUS provides the required protocols for the IIF wireless network approach.

The FreeRADIUS server can be configured to identify and process the various components of the security system (access points, AAA attributes, userid/password, etc.). However, this approach does not scale well and is difficult to manage. The interface option to access an SQL database, as a repository for this information, provides the scalability and flexibility necessary to accommodate midrange to enterprise systems. MySQL, which is used in many IIF business practices, is a freeware alternative SQL database application that is easily integrated with FreeRADIUS.

10 Transitioning to the New En Route Simulation Environment

10.1 Background

ERAM replaces legacy main frame centralized computer systems (HCS) with a modern distributed network-centric system and will significantly change the methods for generating and executing ATC simulations. These simulations are used during the entire lifecycle of the system. Simulation products will be required for formal system testing for government acceptance at the early stage but more significantly, integration and regression testing will be ongoing as the system is maintained and improved throughout its lifecycle. Also, with the transition to new surveillance sources and increasingly seamless airspace definitions, human-in-loop simulations will be needed for user familiarizations of new functionality and for the development of new ATC procedures. The simulation methods required to perform all of these functions on the existing system require many computer platforms that are loosely integrated, require many user skill sets, and are generally oriented toward a single ATC site. In contrast, the ERAM system will provide a simulation function that is more open, efficient, better integrated, with improved user friendliness and, will need to handle multi-site simulations to accomplish these tasks. The predecessor of this ERAM simulation function exists in the form of the Graphical Scenario Generation Tool (GSGT) and the Simulation Driver Replacement (SDRR). These tools have been utilized to support a wide variety of projects at the IIF and provide a preliminary look at the new En Route simulation environment.

What follows is a description of what constitutes an ATC simulation in terms of surveillance and flight data flows and some necessary conditions for successful execution, (such as adaptation, interface protocols, and timing), an enumeration of the various simulation sources and products (inputs and outputs), and a description of the methods for creating and executing simulations on both existing and future ATC automation systems. Additionally, a discussion on simulations using IP interfaces, and some future En Route design considerations are provided.

10.1.1 Elements of ATC Simulation and the Current NAS/En Route Simulation Environment

An ATC simulation consists of a flight data stream and a surveillance data stream. These data streams must be synchronized with each other and the automation system, and, the data content must be consistent with the adaptation of a particular airspace. To execute properly, these data streams must also be consistent with equipment adaptations. Possible sources of surveillance data include long range radar, and short range radar. Possible sources of flight data include National Airspace System (NAS) interfacility messages, NADIN, FDIO, HCS bulk tapes, controller keystroke messages and HCS keyboard messages. Surveillance data are one-way streams from the sensor to the automation system and flight data is a two-way stream that requires point to point acknowledgements. Flight data primarily consists of flight plans, amendments,

cancellations, intrafacility and interfacility handoff information. Surveillance data consists primarily of aircraft position messages and secondarily, real time quality control messages, along with other surveillance messages.

10.1.1.1 Scenario Generation/NAS SIM program

The current NAS/En Route Simulation environment consists of an offline program called SIM which runs under the IBM Multiple Virtual System (MVS) operating system on a support system mainframe (See Figure 10-1). The SIM program takes as input user Extended Binary Coded Decimal Interchange Code (EBCDIC) text messages and generates as output radar, non-radar, and combined radar/non-radar (called MERGE tapes) for use by the NAS as a source of real-time data inputs. These tapes simulate Common Digitizer (CD) radar returns and also any device capable of supplying flight data input messages to NAS or simulated keystrokes for injection into the DSR consoles. The messages on the tape are time stamped and injected into the system in a time ordered sequence. The tapes are of type IBM 3480 and are sometimes referred to as square tapes.

Note that the SIM program is used to generate or create an ATC simulation and runs offline. There is no way to preview or to test the simulation other than executing the simulation on a NAS system, either an Operational or a Test NAS. Errors are only detected when the user attempts to execute the simulation. Most often errors are detected by reducing and analyzing NAS SAR data. The user must then return to the support environment to modify the simulation using the SIM program in an iterative fashion. This is inefficient as it requires access to both the support system and online operational NAS system and is a time consuming process. Access to NAS automation laboratories is limited and exacerbates contention issues in the laboratory user community.

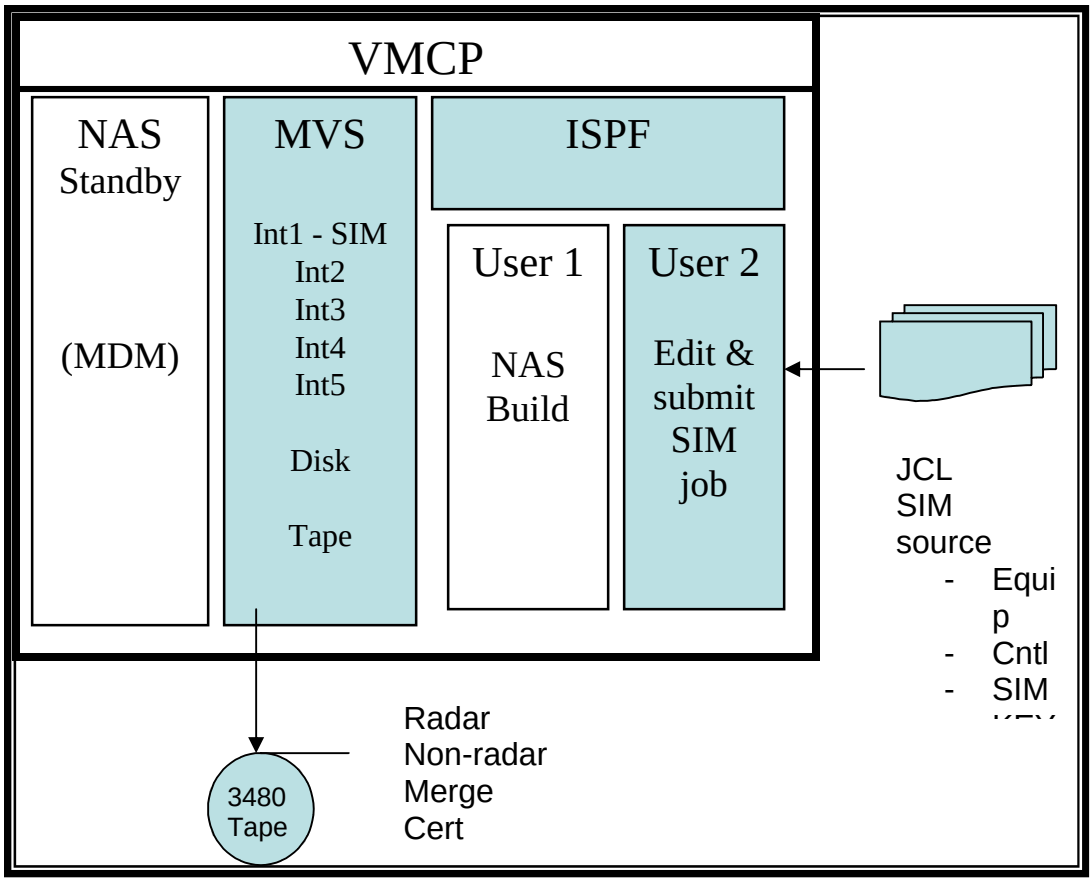

Figure 10-1 Current Simulation Generation Environment

To generate the simulation, the current system requires many user skills to utilize the SIM program. Knowledge of the airspace, the adapted equipment and the operational air traffic flows as well as the general operational knowledge are needed to use the SIM program to create and edit messages. The process also requires manual tape logistics and configuration management procedures. Often the support environment may be in a remote location from the operational environment, which further complicates the process.

10.1.1.2 Scenario Execution/SIM Tapes

Simulations are executed in a direct or looped-SIM mode. Direct SIM (See Figure 10-2) uses a merge tape which is a combination of surveillance and flight data messages. External radar and interfacility interfaces are not exercised in this mode. It should be noted that there is a limited automatic response capability when using SIM tapes, especially from NAS interfacility interfaces. This reduces the fidelity of multi-site simulations to a large degree.

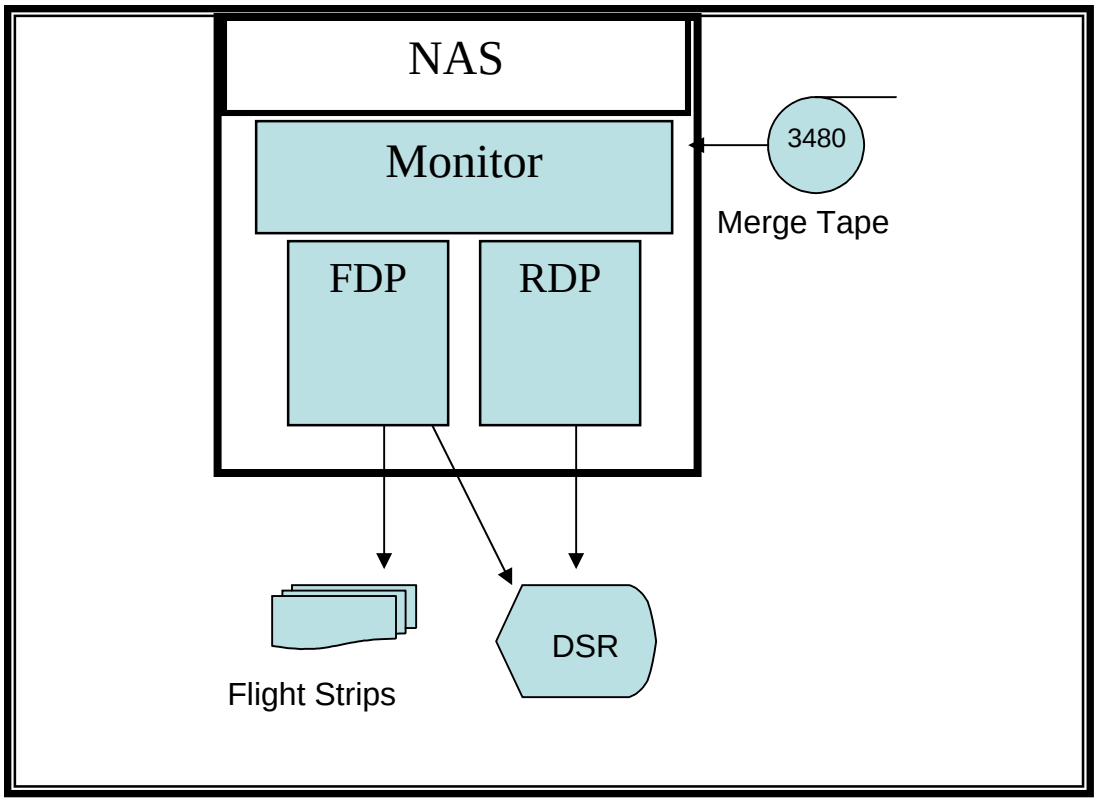

Figure 10-2 Current Simulation Execution – Direct Sim

In Looped SIM mode (Figure 10-3), the non-radar messages (such as flight data entries) are injected via simulation tape but the surveillance messages and NAS interfacility messages are injected by the SDRR processor through the En Route Communications Gateway (ECG). SDRR can also be used to inject DSR console keystroke and trackball entries. In Looped SIM mode, the external simulator messages must be synchronized with the Host time. This is usually done at the NAS 'go' command which in turn generates a General Information (GI) message that is automatically sent to all connected interfacilities. The SDRR simulator, which is simulating all the external radar and interfacility interfaces, receives this signal and automatically starts the simulated data streams.

Looped SIM mode is utilized exclusively at the WJHTC and not at the operational sites. Operational sites are limited to Direct SIM and do not support multi-site interoperability.

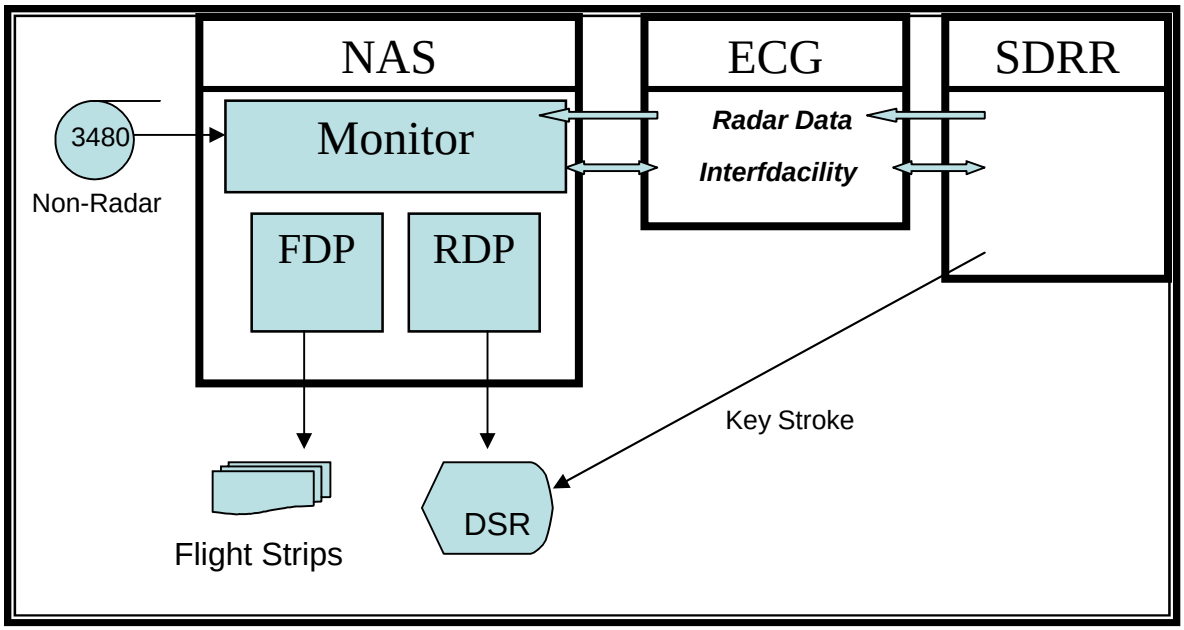

Figure 10-3 Current Simulation Execution – Loop Sim

10.1.2 IIF Simulation Environment

At the IIF, the NAS SIM program has been replaced by the Graphical Scenario Generation Tool (GSGT) which is tightly integrated with SDRR. These processors are Linux-based PCs that share an Ethernet local area network, which is similar to the network-centric architecture of the ERAM simulation environment. The tools themselves are very similar in function to the ERAM simulation tools which will be discussed in a later section.

10.1.3 Simulation Generation/GSGT

GSGT is a powerful tool developed at the WJHTC that stores the entire adaptation data sets, called Host adaptation (ACES) records, of all twenty ARTCCs and has the capability to graphically display this information for interactive sessions with a user. Maps, sector boundaries, fixes, airports, radar coverages, sort boxes and routes are all selectively displayable. By pointing and clicking, a user can draw routes of flight while the tool automatically generates NAS formatted flight plans and other applicable non-radar messages and associated surveillance information from all applicable radar. Flights are easily replicated and large scenarios can be built in a very efficient manner. The created scenarios can be previewed immediately and changes can be made without the need to use the NAS automation system thereby saving time and expensive laboratory resources. In addition to user inputs, scenarios can be generated from other sources that are enumerated below. Finally, various simulation output products can be generated for use on various simulator platforms as described in the next section.

10.1.3.1 Simulation Output Products from GSGT

When the scenario is completed it is exported, producing all the simulation products necessary to execute the ATC simulations for various execution platforms which include SDRR and Target Generation Facility (TGF), which is used for controller-in-the-loop simulations. These exported products can be transferred over the network to the appropriate simulator. In addition, radar, non-radar, and merge tapes can be created directly from GSGT for use in the current NAS/En Route Simulation environment.

10.1.3.2 Simulation Inputs to GSGT

10.1.3.2.1 Pre-existing GSGT Scenarios

In addition to user sessions with the tool, scenarios can be created by directly importing other GSGT scenarios and then modifying them to create new scenarios.

10.1.3.2.2 Pre-existing GSGT Scenarios

GSGT can import existing NAS SIM source files. Once imported, the tool can then create executable SIM tapes or export to other simulation modes. In this way, existing NAS simulations can be converted for execution by other simulators such as SDRR and TGF. Note that GSGT has subsumed the functionality of the NAS SIM program which takes NAS SIM source files and creates a NAS SIM tape as an output.

10.1.3.2.3 Pre-existing TGF Scenarios

GSGT can import TGF formatted scenarios and export simulation products as above. In this way, existing TGF scenarios can be modified and new TGF scenarios can easily be generated. The efficiency of this capability is very important because when controllers request a change to a scenario it can be modified in a manner of minutes versus the typical one day effort required with the current NAS/simulation environment.

10.1.3.2.4 System Analysis Recordings (SAR Tapes)

GSGT has the capability to import NAS SAR tapes via the Small Computer System Interface (SCSI) attached tape drive and derive ATC simulations. This is a powerful feature of the tool because it allows simulations to be derived from live, operational ATC situations such as severe weather or high traffic situations, as well as typical use situations and provides for high fidelity testing. This technique was employed successfully this year for an ERAM Metrics project that will be discussed later in another section. Also reference Technical Note entitled “Host Radar Tracking Simulation and Performance Analysis”, dated August 2005.

10.1.4 Simulation Execution /SDRR

The SDRR processor is a Linux-based PC which provides long-range and short-range radar, NAS interfacility and Display System keystroke message injection. NAS facility simulation includes en route, terminal, and central flow interfaces. SDRR provides significant enhancements to the fidelity of multi-site simulations because each interfacility interface responds dynamically in accordance with all the requirements for

that interface. This allows for automatic responses from interfacility interfaces and thus, for example, allows for high fidelity interfacility handoff processing. This is contrasted against scripted messages on a SIM tape as discussed in a previous section.

SDRR can inject radar and NAS interfacility messages serially or via Internet Protocol (IP) to the En Route Communications Gateway (ECG). Display System (DS) keystroke messages can be injected into the DS consoles via the Facility and Simulation Control Local Communications Network Interface Unit (F&SC LIU).

SDRR imports scenarios created by GSGT. The SDRR tool also has the capability to import NAS SIM tapes, including radar, non-radar and MERGE tapes, to create SDRR scenarios. SDRR also features a SIM data reduction and analysis tool. This tool logs data that can be filtered by aircraft ID, facility, or time stamp. Text files are used for flexibility in transferring data to more common media for retrieval, storage, and archiving.

As mentioned above, SDRR waits for the Host GI message to start the simulation. In practice the synchronization is not always precise. This can be compensated for by temporarily adjusting NAS time via the NAS 'SETT' command until the external simulator (SDRR) and the Host Sim tape messages are sufficiently synchronized.

10.1.5 Examples of Recent IIF Projects

Over the past year, the IIF has been involved in several projects which made use of the new capabilities in simulation, radar transmission and data recording and reduction. Among these projects were the Helicopter In-Transit System (HITS), the ASR-9 to ECG study, and ERAM Metrics.

10.1.5.1 HITS

The HITS project was a study to evaluate Automatic Dependent Surveillance-Broadcast (ADS-B) technology as a potential surveillance enhancement to extend coverage into the Gulf of Mexico beyond the range of Radar. The emphasis of this study was to use ADS-B data to seamlessly extend surveillance coverage while limiting changes to the NAS to minor adaptation changes. This proof-of-concept demonstration consisted of the integration of the ADS-B subsystem into the NAS, the processing of converted ADS-B sensor data by the Host Computer System (HCS), and the seamless presentation of the air traffic symbols on the Display System Replacement (DSR) system.

The HITS program implemented eight Ground Based Transceivers (GBTs) in the Gulf of Mexico. Flight tests were conducted using test aircraft from the FAA Tech Center and NASA Ames and commercial airliners equipped with ADS-B Mode S extended squitter transponder. During the flights, ADS-B and Radar data were recorded. To utilize ADS-B surveillance data, ADS-B messages had to be converted from All Purpose Structured Eurocontrol Radar Information Exchange (ASTERIX) protocol to the En Route Common Digitizer-2 (CD-2) protocol. A Gateway Computer was developed for this purpose. This gateway was integrated into the IIF lab systems so that it could communicate with the En Route Communications Gateway (ECG) over a network. ECG then sent the data to the

HCS for processing. In order to process and display the ADS-B data, three "virtual radars" were added to the ECG and Host adaptation. The Gateway Computer performed a coordinate conversion to translate the positions of the ASTERIX targets so they appeared relative to the virtual radars. The HITS configuration is depicted in Figure 10-4.

IIF personnel tested this system using two sources of data. First, ADS-B and Radar recordings of the flights over the Gulf were re-played into the system at the IIF and, where the coverage overlapped, compared for accuracy. Second, IIF personnel developed extensive radar simulations to test for ASTERIX to CD-2 conversion, ADS-B beacon message report, Accuracy of Coordinate Conversion, System data latency, Boundaries of each virtual radar, Target load capacity, and Host Mode C conformance monitoring.

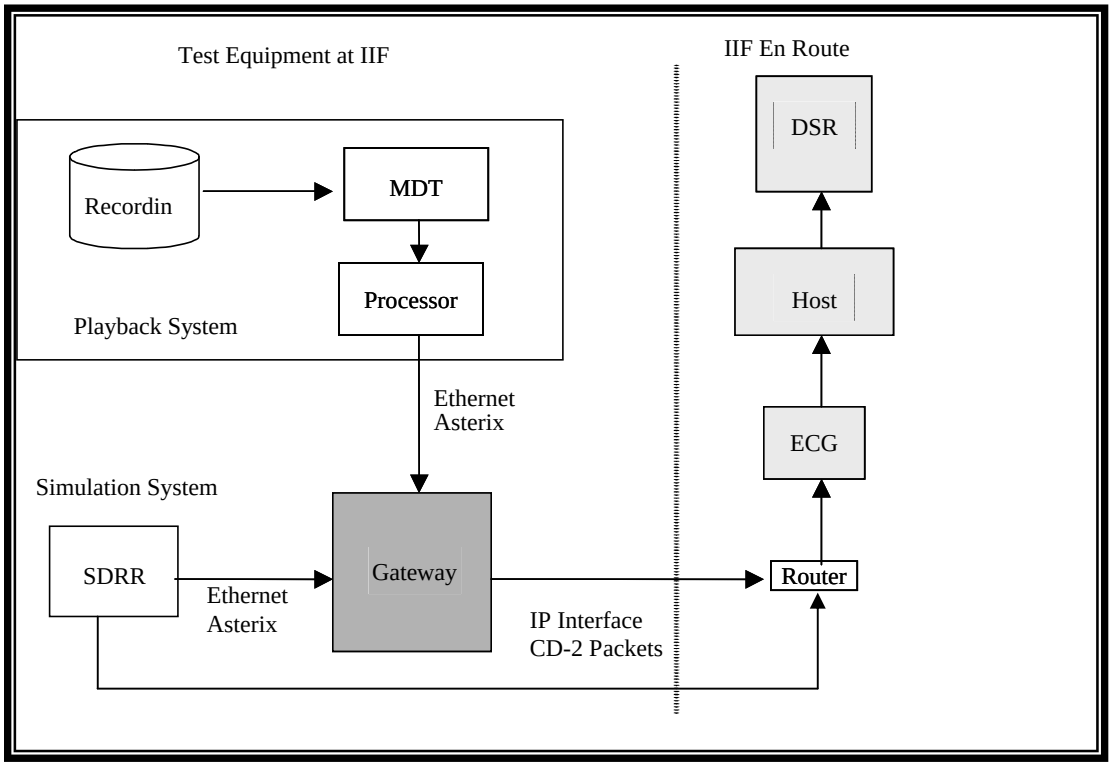

Figure 10-4 HITS Study Configuration

In both the Playback and the Simulation modes, radar data was sent to ECG via IP. The ASTERIX data was converted by the Gateway and sent via IP to ECG. Also, data from the other simulated radars was sent from SDRR to ECG over a LAN using IP. One important benefit of using this method for this study was that it overcame a limitation imposed by the IIF labs complement of serial connections to ECG. If serial connections had been used, only a handful of radars could have been simulated. With IP, the full set of Houston ARTCC's radar could be recreated.

The radar data for this study was created using a combination of GSGT and SDRR. Online Radar Recordings (ORRs) from Houston ARTCC were converted into Playback files using a tool included with SDRR. For the simulation scenarios, specialized flight

patterns were required to thoroughly test the functioning of the virtual radars. GSGT was used to create these scenarios. The scenarios were then exported from GSGT to SDRR for playback (Figure 10-5).

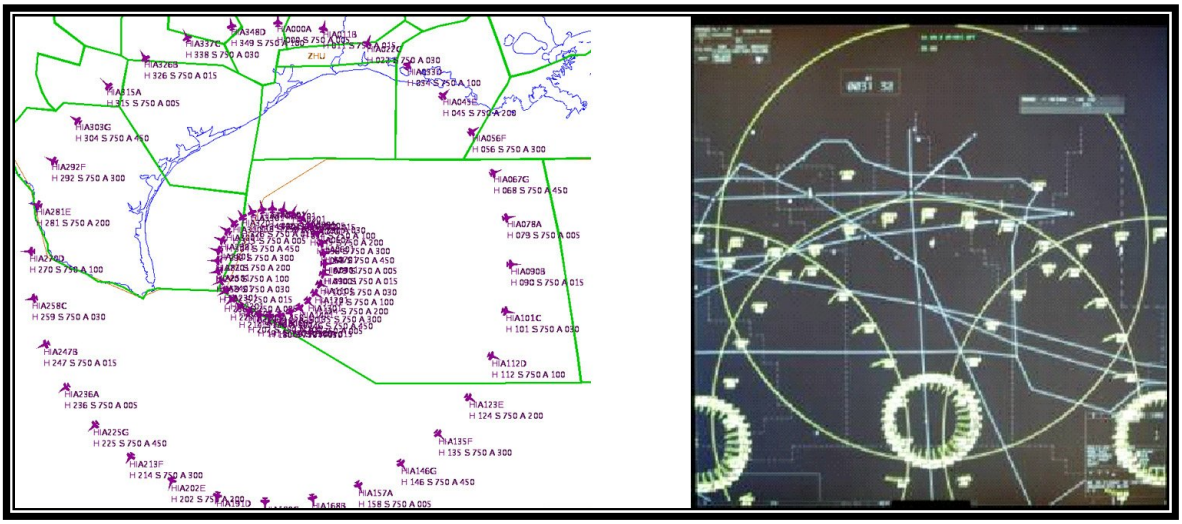

Figure 10-5 Example HITS Scenario under development on GSGT and during playback on DSR

10.1.5.2 ASR9 to ECG Study

This study focused on the ability of ECG to convert ASR-9 radar data into a format that the Host can process. ECG has a CV-4400 function built in for this purpose. For this demonstration, IIF staff connected live ASR-9 data from PHL and ACY through the WJHTC Bytex switch to the IIF SDRR. Because the Bytex connections at the IIF are all RS-232, SDRR was used to perform a real-time conversion of the data back to RS-422. The serial output of SDRR was connected to ECG.

The ECG Primary Interface Processors (PIPs) were configured to convert the ASR-9 CD data using the CV-4400 function parameters in the configuration file (Sepadapt.xml). The PIPs then sent the data to Host as CD2 Skip-Scan. No conversion was necessary on the ECG Backup Interface Processors (BIPs). The BIPs sent the unconverted ASR-9 CD data to EBUS which was adapted for ASR-9. Both the PIPs and BIPs also sent ASR-9 CD data to the RAPPI for display.

A Host Build (ZCY) was adapted to include two new radars (PHL and ACY). They were added as Long Range Radars and the Radar Sort Boxes were changed to emphasize the new Radars. The EBUS Build was also adapted to include PHL and ACY. But for EBUS they were added as Short Range Radars.

In addition to the live radar tests, simulations were developed using GSGT. GSGT was especially useful in creating a capacity test scenario, which included 240 targets each in PHL and ACY. This scenario was exported from GSGT to SDRR for playback. The

serial outputs of SDRR for both radars were connected to one ECG Serial Card Adapter to test for any hardware limitations. The ASR-9 configuration is depicted in Figure 10-6

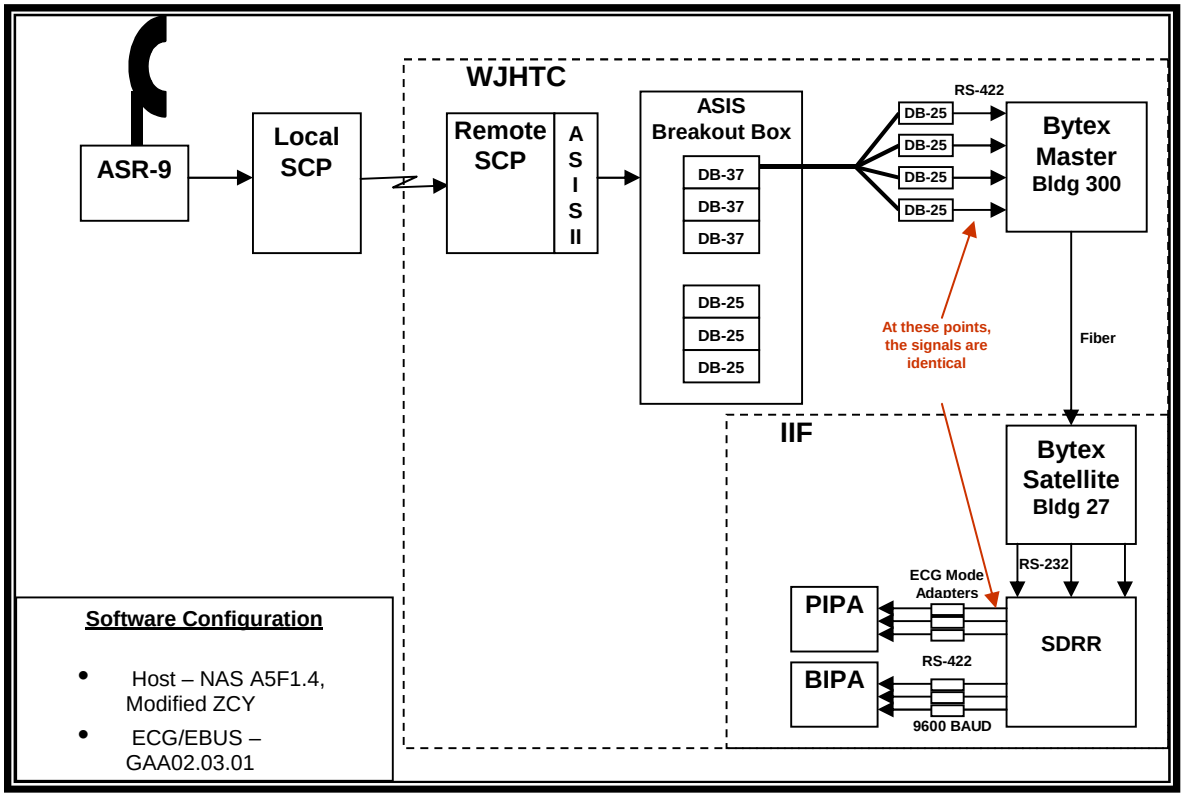

Figure 10-6 ASR-9 Test Configuration

10.1.5.3 ERAM Metrics

The ERAM Automation Metrics Test Working Group (AMTWG) has a charter to support developmental and operational testing of ERAM by developing a set of metrics that quantify the effectiveness of key system functions in ERAM. The metrics are designed to measure the performance of ERAM, but also of the current systems. The IIF staff worked with the AMTWG to develop scenarios for gathering metrics on Surveillance Data Processing, concentrating first on Tracking and Tactical Conflict Alert.

A recording of approximately 6 hours of air traffic data was collected from Washington ARTCC (ZDC) on March 17, 2005. IIF personnel used GSGT to convert this data into a simulation scenario (Figure 10-7). GSGT was then used to export three SDRR scenarios. One scenario had no radar noise, one had moderate radar noise and one had high radar noise. GSGT was also used to create a Track Report. The GSGT positions were considered the actual path of the aircraft for comparison with the Host track reports. Once again, Radar over IP was used to allow a more robust simulation configuration. In

this way, 11 radars (33 channels) and 21 Interfacility sessions were established using TCPI/IP over a single network connection to ECG.

For this study, a novel method of collecting system data was employed. The usual method involves the use of the Host System Analysis Recording (SAR) subsystem. This subsystem records various data types that represent raw input data, intermediate processed data and final processed products of the Host air traffic control subsystems. The SAR data is typically transferred to IBM Model 3480 cartridge tapes, which then become input for the off-line utility, Data Analysis and Reduction Tool (DART). The output of DART consists of many possible report types that represent the recorded data according to selected DART options. In this case the DART output had to undergo additional processing to extract the information needed for the ERAM Metrics processing tools.

To optimize this process, the IIF staff developed an alternative method of extracting the appropriate data. This alternative used the Host Common Message Set (CMS), which is a network stream that provides NAS processing information via a predefined set of messages. These messages are exported to various network clients via the Host Interface Device (HID). IIF personnel developed a HID client that receives the CMS messages and stores them in a format that the Metrics tools can use directly. For the Conflict Alert processing phase of ERAM Metrics, a Host patch was developed to add Conflict Alert messages to the CMS data stream. Additional detail regarding this Host patch is included as Appendix D of this document.

The results of this testing and analysis were incorporated into the AMTWG's Technical Note "Host Radar Tracking Simulation and Performance Analysis." Another Technical note on Conflict Alert processing is in development.

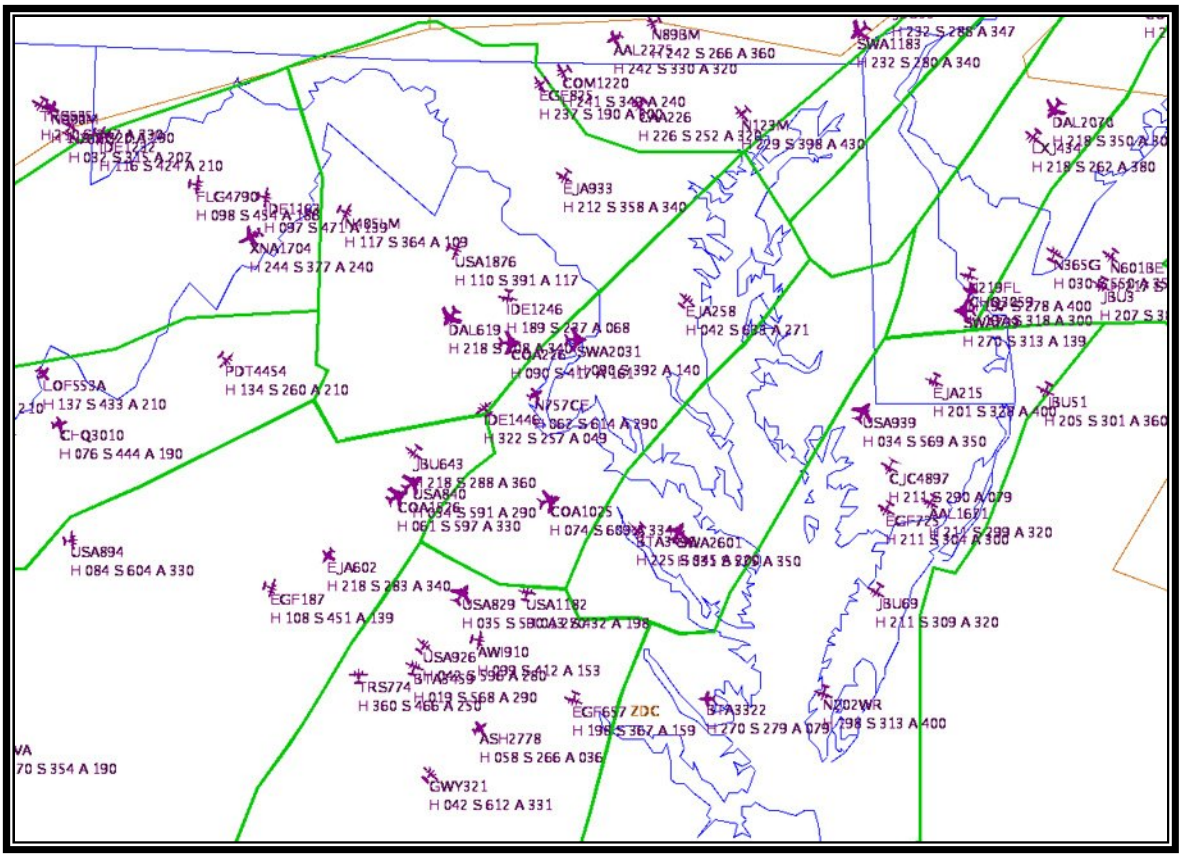

Figure 10-7 ERAM Metrics Scenario on GSGT

10.2 ERAM Simulation Environment

The ERAM simulation environment is analogous in form and function to the simulation environment at the IIF. ERAM Adaptation (EADP) will replace will replace ACES. EADP will provide nation wide airspace adaptation while Fixed Airspace Volumes (FAVs) and equipment will remain local. The ERAM simulation environment will provide for multi-site simulation but will need to support simulations involving existing Host adjacent facilities in addition to adjacent ERAM facilities.

10.2.1 Simulation Generation/SGET

Scenario Generation Tool (SGET) is a CSCI of the test and training subsystem of ERAM. It is, in fact, a version of GSGT that has been tailored for operation within the ERAM simulation environment. Specifically, the system will run on IBM's Advanced Interactive Executive (AIX) operating system versus LINUX and the system will use EADP adaptation versus ACES adaptation. The export capability has been enhanced to support an export to AT Coach, which provides the simulation execution capability for ERAM.

10.2.1.1 Simulation Execution/SIME

SIM Engine (SIME) is a CSCI of the test and training subsystem of ERAM. Its function is to inject radar, NAS interfacility, ERAM equipment and Display System (DS) keystroke messages. It is analogous to the function of SDRR. SIME has an ECG emulation mode which can be enabled and disabled. When it is disabled, radar and interfacility messages are injected via Internet Protocol (IP) to ECG. When it is enabled, all messages are sent directly to the operational local communications network for message injection, bypassing the ECG.

SIME also provides pilot workstations for human-in-the-loop simulations. These pilot workstations are fitted with air to ground communications equipment for communications with air traffic controllers working the ERAM situation display consoles. The pilot workstations simulate aircraft responses to heading, altitude, and speed changes as warranted by the simulation.

10.2.2 Simulation Transition Considerations

The SDRR tool will continue to provide value during the ERAM lifecycle. The following considerations represent some of the reasons that SDRR will continue to be a necessary tool well into the ERAM lifecycle.

10.2.2.1 Multi-site Testing

During the ERAM integration and test phase, and before the key-site ERAM system is deployed, interoperability with adjacent Host Systems will need to be tested. The SIME capability can only be utilized within the ERAM system boundary while SDRR is universal and is not restricted to any specific system boundary. This enables SDRR to drive all systems including Host and ERAM simultaneously. SIME would still need to run to supply ERAM oriented equipment keystroke messages and there is probably going to be some development work that is necessary to coordinate a simulation of this nature.

10.2.2.2 ERAM Automation Metrics Testing and General Baseline Measurements

For accurate comparisons of system performance between Host and ERAM, both systems must be driven by identical inputs to the maximum extent possible. For ERAM metrics, IIF personnel worked with ACB-330 to collect data on legacy systems using GSGT scenarios exported to SDRR. It follows then that the same surveillance inputs should be presented to the ERAM system. GSGT has its own SIM engine which is used to generate the surveillance radar data. In the ERAM environment, the SIM engine is actually part of SIME and therefore will have significant deltas in execution. Therefore, for accurate comparisons, input surveillance data to ERAM should originate from the same SIM engine and therefore, SDRR should be used to supply radar inputs for both systems.

10.2.2.3 Independent Government Test Tools

SIME is provided as part of the ERAM software. In general, it is not good practice to test ERAM exclusively with ERAM provided tools. SDRR is a certified FAA simulator that was government accepted in September of 2003 and has the capability to provide the

necessary inputs for any NAS En Route automation system. Additionally, TGF is a government accepted simulator that has been utilized extensively for controller-in-the-loop simulations. SDRR provides an independent government test tool for metrics, baseline and multi-site configuration as discussed above. TGF provides an independent government test tool for controller-in-the-loop simulations for government conducted operational test and evaluation purposes.

10.2.3 Simulated Radar via IP interfaces

In an effort to increase the lab's capabilities, IIF personnel have often made use of IP to transmit simulated Radar data over the lab's data network. Both SDRR and SIME have this capability. SDRR injects radar and NAS interfacility messages via IP and via serial cables. SIME only has the capability to inject via IP. IP offers very significant value in terms of hardware reduction (cables, distribution panels, serial card adapters, etc.) as well as floor space and power consumption. The legacy system uses leased telephone lines and employs three serial lines, one for each of three channels of radar. This will not change until the FTI program upgrades this wide area communication technique to IP. ECG has the capability to process both serial and IP inputs. Consider the radar simulation problem for an ARTCC with ten long-range radars. Thirty input serial cables to the ECG modem splitters would then fan out to 120 cables - 30 to each Primary Interface Processor (PIP) and Back-up Interface Processor (BIP). Tallying the hardware count, that makes 30 plus 120 = 150 serial cables with rack space for distribution panels, 120 serial ports which corresponds to 15 serial card adapters and their associated consumed power and generated heat. Now consider the same 10 radar simulation using IP. Tallying the hardware count we have one Category 5 cable for all ten radar, 30 channels. There are no serial card adapters and therefore no power consumed, no heat generated and no cable management space constraints. This raises the speculation that what's good for simulation might also be good for actual interfaces. The IIF has been using IP as a highly reliable means of radar transmission with since 1997.

10.2.4 Future Design Considerations

The following sections discuss some possible areas for innovation in En Route simulation.

10.2.4.1 Simulation Generation from CMS Data

As mentioned in the section on ERAM Metrics, CMS data can be used as an alternative to traditional Host SAR. Since the generation of simulation scenarios from SAR data is well established, it should also be possible to generate a scenario using CMS data. The IIF is exploring this possibility with the additional goal of tapping the Host Air Traffic Management Data Distribution System (HADDS) to create a near real-time conversion of CMS data into scenario data. This data would be converted and injected using SDRR and would be synchronized with live radar data.

10.2.4.2 Simulation Time Synchronization

As mentioned in the Section 10.1.4, the SDRR simulator uses a Host GI message to automatically start a simulation. Thus the time of the SDRR simulation and the

Host/DSR simulation time should be synchronized. In actual use, it has been found that a time lag of a few seconds can occur. While this lag may not be significant for all simulations, it can impact high load scenarios where timing is critical. Time synchronization problems are not unique to SDRR. Any external simulator could have the same problem.

One method to address this problem is for the simulator to receive time messages from DSR. IIF personnel have done some preliminary investigation into this method. By connecting into the DSR LCN (or BCN) the simulator can "sniff" for time-related DSR Service Requests (SRs) and decode the contents to update its own simulation time. This is non-intrusive to DSR and would provide an accurate time for beginning the simulation and counteract any time drift that might occur during the course of the run. This method should be extensible into the ERAM configuration. Additional detail regarding this method is included in Appendix E.

11 Proposed Follow-on Activities

The following items would be potential follow-on activities related to the activities that were reported out in this document:

Customized Remote Monitoring of COTS Equipment

Several COTS remote monitoring methods were identified in this report. A study could be conducted using these methods on COTS equipment that is reflective of the ERAM network environment. This study could lead to an optimal configuration/customization of these monitoring methods for implementation into the ERAM system. Remote monitoring methods for Cisco's Netflow data stream and RMON data types would be of particular interest.

Continued Centralized Maintenance Concepts with Extensibility into ERAM

Many of the concepts introduced and/or explored as a part of the Centralized Maintenance area of this task could be further expanded into the ERAM environment. Additional follow-on DSR maintenance research could also be pursued. These items could include further defining system notification events, exploration of data mining techniques for collected system health data, integrating the ISMT development environments with online/real time debugging tools, and automated problem resolution tools.

Continued Examination of New Simulation methods within the context of ERAM

The laboratory simulation methods described in this report will continue into the future as routine business practice at the IIF. As the ERAM acquisition proceeds, additional simulation techniques will undoubtedly be explored at the IIF.

Lab Operations in the ERAM Environment

As is indicated in this report, new ways of handling lab operations could save a significant amount of time and effort and optimize the laboratory operations environment. Lab operations within the context of the ERAM system would be an ideal area to begin implementing these new ways of performing lab management. Items that could be further explored in this area include defining ERAM MIB data types for lab management, further refining SNMP agent capabilities for ERAM-type test processors, examining lab resource sharing techniques and processes to address lab contention issues, and shared lab resource switching.

Appendix A SNMP Agent Configuration items for IBM SMUX/DPI

A.1 Snmpd.conf

```
# @(#)93        1.5  com/snmp/samples/snmpd.conf, snmp, tcpip320, 9143320 10/17/91
14:44:03

## COMPONENT_NAME: (SNMP) Simple Network Management Protocol Daemon

# ORIGINS: 27 60

# (C) COPYRIGHT International Business Machines Corp. 1991

# All Rights Reserved

# Licensed Material - Property of IBM

# US Government Users Restricted Rights - Use, duplication or

# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contributed by NYSERNet Inc.  This work was partially supported by the

# U.S. Defense Advanced Research Projects Agency and the Rome Air Development

# Center of the U.S. Air Force Systems Command under contract number

# F30602-88-C-0016.

# FILE:          /etc/snmpd.conf

#####################################################################

#####################################################################

# snmpd configuration information

#####################################################################
#####################################################################

# How to configure this file for your system:

# 1. If you want to direct your logging from the configuration file,

#    set the logging specifications as follows:

#         logging                 </path/filename>       enabled|disabled

#         logging                 size=<limit>           level=<debug level>

#    where </path/filename> specifies the complete path and filename of the
```

```
#    log file, enabled turns logging on, disabled turns logging off, <limit>
#    specifies the maximum size in bytes of the specified logfile, and
#    <debug level> specifies the logging level of 0, 1, 2, or 3.  There is no
#    default logging file.  The enablement default is disabled.  The size
#    default is 0, meaning unlimited, and the level default is 0.  There can
#    be no white spaces around the "=" in the size and level fields.  There
#    are no restrictions on the order in which the fields are entered in the
#    logging entries.  A logging entry can contain single or multiple fields.
# 2. Set the community names and access privileges for hosts that can make
#    requests of this snmpd agent.  Define these restrictions as follows:
#         community <name> <address> <netmask> <permissions> <view name>
#    where <name> is the community name, <address> is either a hostname or
#    an IP address in dotted notation, and <permissions> is one of:  none,
#    readOnly, writeOnly, readWrite.  The default permission is readOnly.
#    <netmask> specifies the network mask.  The default address and netmask
#    are 0.0.0.0.  If an address other than 0.0.0.0 is specified, the default
#    netmask is 255.255.255.255.  If a permission is specified, both the
#    address and netmask must also be specified.  <view name> defines a
#    portion of the MIB tree to which this community name allows access.
#    <view name> must be defined as a unique object identifier in dotted
#    numeric notation.  <view name> is further defined in the view
#    configuration entry.  If <view name> is not specified, the view for
#    this community defaults to the entire MIB tree.  Fields to the right
#    of <name> are optional, with the limitation that no fields to the
#    left of a specified field are omitted.
# 3. Set your MIB views as follows:
#         view  <view name>  <MIB subtree>...
#    where <view name> is a unique object identifier in dotted numeric
#    notation and <MIB subtree> is a list of the MIB subtrees in text or
```

```
#    dotted numeric notation that this view allows access.  The <view name>
#    is the same as that specified in the community configuration entry.  If
#    the MIB subtree list is not specified, the view defaults to the entire
#    MIB tree.
# 4. If your site has a management station that listens for traps, fill-in
#    the information for the trap destination as follows:
#         trap  <community>  <a.b.c.d>   <view name>  <trap mask>
#    where <community> is the community name that will be encoded in the
#    trap packet and <a.b.c.d> is the hostname or IP address in dotted
#    notation of the host where a trap monitor is listening on UDP port 162.
#    The <view name> is a unique object identifier in dotted notation. View
#    name is not implemented for traps in AIX3.2.  The snmpd agent only checks
#    the view name format and duplication.  The trap mask is in hexidecimal
#    format.  The bits from left to right stand for coldStart trap, warmStart
#    trap, linkDown trap, linkUp trap, authenticationFailure trap,
#    egpNeighborLoss trap, and enterpriseSpecific trap.  The right most bit
#    does not have any meaning.  The value "1" will enable the corresponding
#    trap to be sent. Otherwise, the trap is blocked.
#      ex.        fe       block no traps (1111 1110)
#                 7e       block coldStart trap (0111 1110)
#                 be       block warmStart trap (1011 1110)
#                 3e       block coldStart trap and warmStart trap (0011 1110)
# 5. Set your snmpd specific configuration parameters as follows:
#         snmpd  <variable>=<value>
#    where <variable> is one of maxpacket, querytimeout or smuxtimeout.
#    If <variable> is maxpacket, <value> is the maximum packet size, in
#    bytes, that the snmpd agent will transmit.  The minimum value to
#    which maxpacket can be set is 300 bytes.  If there is no snmpd entry
#    for maxpacket, the system socket default limits will be used.  If
```

```
#    <variable> is querytimeout, <value> is the time interval, in seconds,
#    at which the snmpd agent will query the interfaces to check for
#    interface status changes.  The minimum value to which querytimeout
#    can be set is 30 seconds.  If 0 (zero) is specified, snmpd will not
#    query the interfaces for status changes.  If no snmpd entry for
#    querytimeout is specified, the default value of 60 seconds is used.
#    If <variable> is smuxtimeout, <value> is the time interval, in
#    seconds, at which snmpd will timeout on a request to a smux peer.
#    If 0 (zero) is specified, snmpd will not timeout on smux requests.
#    If no snmpd entry for smuxtimeout is specified, the default value
#    of 15 seconds is used.  The "=" is absolutely required, and no white
#    spaces are allowed around the "=".  There are no restrictions on
#    the order in which the fields are entered in the snmpd  entry.  An
#    snmpd entry can contain single or multiple fields.
# 6. Set the smux peer configuration parameters as follows:
#       smux <client OIdentifier> <password> <address> <netmask>
#    where <client OIdentifier> is the unique object identifer in dotted
#    decimal notation of the SMUX peer client.  <password> specifies the
#    password that snmpd requires from the SMUX peer client to authenticate
#    the SMUX association.  <address> is either the hostname or IP address
#    in dotted notation of the host on which the SMUX peer client is
#    executing.  <netmask> specifies the network mask.  If no password is
#    specified, there is no authentication for the SMUX association. The
#    default address and netmask are 127.0.0.1 and 255.255.255.255.  If
#    neither the address nor netmask are specified, the SMUX association
#    is limited to the local host.  Fields to the right of
#    <client OIdentifier> are optional, with the limitation that no fields
#    to the left of a specified field are omitted.
# NOTE:  Comments are indicated by # and continue to the end of the line.
```

```
#       There are no restrictions on the order in which the configuration
#       entries are specified in this file.
##############################################################################s
nmpd            maxpacket=1024 querytimeout=120 smuxtimeout=60
logging         file=/usr/tmp/snmpd.log  enabled
logging         size=0    level=0
community       En Route_modern 10.156.189.3     0.0.0.0       readWrite
community       En Route_modern 127.0.0.1     255.255.255.0       readWrite
community       En Route_modern 172.26.150.72     255.255.255.0       readWrite
trap          En Route_modern 10.156.189.3 255.255.255.0   # nvs01
smux      1.3.6.1.4.1.2.3.1.2.1.2       gated_password            # gated
smux      1.3.6.1.4.1.2.6.4.1          nv6000      # nv6000 on nvs01
smux       1.3.6.1.4.1.2.3.1.2.2.1.1.2        dpid_password     #dpid
```

A.1 snmpd.peers

```
# @(#)94        1.3  com/snmp/samples/snmpd.peers, , tcpip320, 9141320 7/20/91
14:59:33
# COMPONENT_NAME: (SNMP) Simple Network Management Protocol Daemon
# ORIGINS: 27 60
# (C) COPYRIGHT International Business Machines Corp. 1991
# All Rights Reserved
# Licensed Material - Property of IBM
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contributed by NYSERNet Inc.  This work was partially supported by the
# U.S. Defense Advanced Research Projects Agency and the Rome Air Development
# Center of the U.S. Air Force Systems Command under contract number
# F30602-88-C-0016.
# FILE:          /etc/snmpd.peers
##############################################################################
```

```
#############################################################################
# snmpd configuration for SMUX peers
#############################################################################


#############################################################################
# Syntax:
#       <name>          <object id>     <password>    <priority>
#    where <name> is the name of the process acting as an SMUX peer and
#    <object id> is the unique object identifer in dotted decimal
#    notation of that SMUX peer.  <password> specifies the password that the
#    snmpd daemon requires from the SMUX peer client to authenticate
#    the SMUX association.  The highest priority is 0 (zero).  The lowest
#    priority is (2^31)-1.  The default password is the null string.  The
#    default priority is 0 (zero).  Fields to the right of <object id> are
#    optional, with the limitation that no fields to the left of a specified
#    field are omitted.
#    Each token is separated by white space, though double-quotes may be
#    used to prevent separation.
#############################################################################
"gated"     1.3.6.1.4.1.2.3.1.2.1.2       "gated_password"
"dpid"      1.3.6.1.4.1.2.3.1.2.2.1.1.2     "dpid_password"
"dpid"      1.3.6.1.4.1.2.3.1.2.2.1.1.2     "dpid_password"
```

A.3 Mib.defs

```
-- @(#)92  1.3.1.3  com/snmp/samples/mib.defs, snmp, tcpip325, 9503G325a 1/9/95 14:39:56
--
--
-- COMPONENT_NAME: (SNMP) Simple Network Management Protocol Daemon
--
```

```
-- FUNCTIONS:
--
-- ORIGINS: 27
--
-- (C) COPYRIGHT International Business Machines Corp. 1992, 1993
-- All Rights Reserved
-- US Government Users Restricted Rights - Use, duplication or
-- disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
--
--
-- FILE: /etc/mib.defs
--

-- automatically generated by mosy 6.8 of Mon Jun 24 10:26:06 CDT 1991, do not edit!

-- object definitions compiled from RFC1155-SMI { iso 3 6 1  }

internet            iso.3.6.1
directory           internet.1
mgmt                internet.2
experimental        internet.3
private             internet.4
enterprises         private.1

-- object definitions compiled from RFC1213-MIB { iso 3 6 1 2 1  }

mib-2               mgmt.1
system              mib-2.1
interfaces          mib-2.2
at                  mib-2.3
ip                  mib-2.4
icmp                mib-2.5
tcp                 mib-2.6
udp                 mib-2.7
egp                 mib-2.8
transmission        mib-2.10
snmp                mib-2.11
```

sysDescr	system.1	DisplayString	read-only	mandatory
sysObjectID	system.2	ObjectID	read-only	mandatory
sysUpTime	system.3	TimeTicks	read-only	mandatory
sysContact	system.4	DisplayString	read-write	mandatory
sysName	system.5	DisplayString	read-write	mandatory
sysLocation	system.6	DisplayString	read-write	mandatory
sysServices	system.7	INTEGER	read-only	mandatory
ifNumber	interfaces.1	INTEGER	read-only	mandatory
ifTable	interfaces.2	Aggregate	not-accessible	mandatory
ifEntry	ifTable.1	Aggregate	not-accessible	mandatory
ifIndex	ifEntry.1	INTEGER	read-only	mandatory
ifDescr	ifEntry.2	DisplayString	read-only	mandatory
ifType	ifEntry.3	INTEGER	read-only	mandatory
ifMtu	ifEntry.4	INTEGER	read-only	mandatory
ifSpeed	ifEntry.5	Gauge	read-only	mandatory
ifPhysAddress	ifEntry.6	PhysAddress	read-only	mandatory
ifAdminStatus	ifEntry.7	INTEGER	read-write	mandatory
ifOperStatus	ifEntry.8	INTEGER	read-only	mandatory
ifLastChange	ifEntry.9	TimeTicks	read-only	mandatory
ifInOctets	ifEntry.10	Counter	read-only	mandatory
ifInUcastPkts	ifEntry.11	Counter	read-only	mandatory
ifInNUcastPkts	ifEntry.12	Counter	read-only	mandatory
ifInDiscards	ifEntry.13	Counter	read-only	mandatory
ifInErrors	ifEntry.14	Counter	read-only	mandatory
ifInUnknownProtos	ifEntry.15	Counter	read-only	mandatory
ifOutOctets	ifEntry.16	Counter	read-only	mandatory
ifOutUcastPkts	ifEntry.17	Counter	read-only	mandatory
ifOutNUcastPkts	ifEntry.18	Counter	read-only	mandatory
ifOutDiscards	ifEntry.19	Counter	read-only	mandatory
ifOutErrors	ifEntry.20	Counter	read-only	mandatory
ifOutQLen	ifEntry.21	Gauge	read-only	mandatory
ifSpecific	ifEntry.22	ObjectID	read-only	mandatory
atTable	at.1	Aggregate	not-accessible	deprecated
atEntry	atTable.1	Aggregate	not-accessible	deprecated
atIfIndex	atEntry.1	INTEGER	read-write	deprecated
atPhysAddress	atEntry.2	PhysAddress	read-write	deprecated

atNetAddress atEntry.3 NetworkAddress read-write deprecated

ipForwarding ip.1 INTEGER read-write mandatory

ipDefaultTTL ip.2 INTEGER read-write mandatory

ipInReceives ip.3 Counter read-only mandatory

ipInHdrErrors ip.4 Counter read-only mandatory

ipInAddrErrors ip.5 Counter read-only mandatory

ipForwDatagrams ip.6 Counter read-only mandatory

ipInUnknownProtos ip.7 Counter read-only mandatory

ipInDiscards ip.8 Counter read-only mandatory

ipInDelivers ip.9 Counter read-only mandatory

ipOutRequests ip.10 Counter read-only mandatory

ipOutDiscards ip.11 Counter read-only mandatory

ipOutNoRoutes ip.12 Counter read-only mandatory

ipReasmTimeout ip.13 INTEGER read-only mandatory

ipReasmReqds ip.14 Counter read-only mandatory

ipReasmOKs ip.15 Counter read-only mandatory

ipReasmFails ip.16 Counter read-only mandatory

ipFragOKs ip.17 Counter read-only mandatory

ipFragFails ip.18 Counter read-only mandatory

ipFragCreates ip.19 Counter read-only mandatory

ipAddrTable ip.20 Aggregate not-accessible mandatory

ipAddrEntry ipAddrTable.1 Aggregate not-accessible mandatory

ipAdEntAddr ipAddrEntry.1 IpAddress read-only mandatory

ipAdEntIfIndex ipAddrEntry.2 INTEGER read-only mandatory

ipAdEntNetMask ipAddrEntry.3 IpAddress read-only mandatory

ipAdEntBcastAddr ipAddrEntry.4 INTEGER read-only mandatory

ipAdEntReasmMaxSize ipAddrEntry.5 INTEGER read-only mandatory

ipRouteTable ip.21 Aggregate not-accessible mandatory

ipRouteEntry ipRouteTable.1 Aggregate not-accessible mandatory

ipRouteDest ipRouteEntry.1 IpAddress read-write mandatory

ipRouteIfIndex ipRouteEntry.2 INTEGER read-write mandatory

ipRouteMetric1 ipRouteEntry.3 INTEGER read-write mandatory

ipRouteMetric2 ipRouteEntry.4 INTEGER read-write mandatory

ipRouteMetric3 ipRouteEntry.5 INTEGER read-write mandatory

ipRouteMetric4 ipRouteEntry.6 INTEGER read-write mandatory

ipRouteNextHop ipRouteEntry.7 IpAddress read-write mandatory

ipRouteType ipRouteEntry.8 INTEGER read-write mandatory

ipRouteProto ipRouteEntry.9 INTEGER read-only mandatory

ipRouteAge ipRouteEntry.10 INTEGER read-write mandatory

ipRouteMask ipRouteEntry.11 IpAddress read-write mandatory

ipRouteMetric5 ipRouteEntry.12 INTEGER read-write mandatory

ipRouteInfo ipRouteEntry.13 ObjectID read-only mandatory

ipNetToMediaTable ip.22 Aggregate not-accessible mandatory

ipNetToMediaEntry ipNetToMediaTable.1 Aggregate not-accessible mandatory

ipNetToMediaIfIndex ipNetToMediaEntry.1 INTEGER read-write mandatory

ipNetToMediaPhysAddress ipNetToMediaEntry.2 PhysAddress read-write mandatory

ipNetToMediaNetAddress ipNetToMediaEntry.3 IpAddress read-write mandatory

ipNetToMediaType ipNetToMediaEntry.4 INTEGER read-write mandatory

ipRoutingDiscards ip.23 Counter read-only mandatory

icmpInMsgs icmp.1 Counter read-only mandatory

icmpInErrors icmp.2 Counter read-only mandatory

icmpInDestUnreachs icmp.3 Counter read-only mandatory

icmpInTimeExcds icmp.4 Counter read-only mandatory

icmpInParmProbs icmp.5 Counter read-only mandatory

icmpInSrcQuenchs icmp.6 Counter read-only mandatory

icmpInRedirects icmp.7 Counter read-only mandatory

icmpInEchos icmp.8 Counter read-only mandatory

icmpInEchoReps icmp.9 Counter read-only mandatory

icmpInTimestamps icmp.10 Counter read-only mandatory

icmpInTimestampReps icmp.11 Counter read-only mandatory

icmpInAddrMasks icmp.12 Counter read-only mandatory

icmpInAddrMaskReps icmp.13 Counter read-only mandatory

icmpOutMsgs icmp.14 Counter read-only mandatory

icmpOutErrors icmp.15 Counter read-only mandatory

icmpOutDestUnreachs icmp.16 Counter read-only mandatory

icmpOutTimeExcds icmp.17 Counter read-only mandatory

icmpOutParmProbs icmp.18 Counter read-only mandatory

icmpOutSrcQuenchs icmp.19 Counter read-only mandatory

icmpOutRedirects icmp.20 Counter read-only mandatory

icmpOutEchos icmp.21 Counter read-only mandatory

icmpOutEchoReps icmp.22 Counter read-only mandatory

icmpOutTimestamps icmp.23 Counter read-only mandatory

icmpOutTimestampReps icmp.24 Counter read-only mandatory

icmpOutAddrMasks icmp.25 Counter read-only mandatory

icmpOutAddrMaskReps icmp.26 Counter read-only mandatory

tcpRtoAlgorithm tcp.1 INTEGER read-only mandatory

tcpRtoMin tcp.2 INTEGER read-only mandatory

tcpRtoMax tcp.3 INTEGER read-only mandatory

tcpMaxConn tcp.4 INTEGER read-only mandatory

tcpActiveOpens tcp.5 Counter read-only mandatory

tcpPassiveOpens tcp.6 Counter read-only mandatory

tcpAttemptFails tcp.7 Counter read-only mandatory

tcpEstabResets tcp.8 Counter read-only mandatory

tcpCurrEstab tcp.9 Gauge read-only mandatory

tcpInSegs tcp.10 Counter read-only mandatory

tcpOutSegs tcp.11 Counter read-only mandatory

tcpRetransSegs tcp.12 Counter read-only mandatory

tcpConnTable tcp.13 Aggregate not-accessible mandatory

tcpConnEntry tcpConnTable.1 Aggregate not-accessible mandatory

tcpConnState tcpConnEntry.1 INTEGER read-write mandatory

tcpConnLocalAddress tcpConnEntry.2 IpAddress read-only mandatory

tcpConnLocalPort tcpConnEntry.3 INTEGER read-only mandatory

tcpConnRemAddress tcpConnEntry.4 IpAddress read-only mandatory

tcpConnRemPort tcpConnEntry.5 INTEGER read-only mandatory

tcpInErrs tcp.14 Counter read-only mandatory

tcpOutRsts tcp.15 Counter read-only mandatory

udpInDatagrams udp.1 Counter read-only mandatory

udpNoPorts udp.2 Counter read-only mandatory

udpInErrors udp.3 Counter read-only mandatory

udpOutDatagrams udp.4 Counter read-only mandatory

udpTable udp.5 Aggregate not-accessible mandatory

udpEntry udpTable.1 Aggregate not-accessible mandatory

udpLocalAddress udpEntry.1 IpAddress read-only mandatory

udpLocalPort udpEntry.2 INTEGER read-only mandatory

egpInMsgs egp.1 Counter read-only mandatory

egpInErrors egp.2 Counter read-only mandatory

egpOutMsgs egp.3 Counter read-only mandatory

egpOutErrors egp.4 Counter read-only mandatory

egpNeighTable egp.5 Aggregate not-accessible mandatory

egpNeighEntry egpNeighTable.1 Aggregate not-accessible mandatory

egpNeighState egpNeighEntry.1 INTEGER read-only mandatory

egpNeighAddr egpNeighEntry.2 IpAddress read-only mandatory

egpNeighAs egpNeighEntry.3 INTEGER read-only mandatory

egpNeighInMsgs egpNeighEntry.4 Counter read-only mandatory

egpNeighInErrs egpNeighEntry.5 Counter read-only mandatory

egpNeighOutMsgs egpNeighEntry.6 Counter read-only mandatory

egpNeighOutErrs egpNeighEntry.7 Counter read-only mandatory

egpNeighInErrMsgs egpNeighEntry.8 Counter read-only mandatory

egpNeighOutErrMsgs egpNeighEntry.9 Counter read-only mandatory

egpNeighStateUps egpNeighEntry.10 Counter read-only mandatory

egpNeighStateDowns egpNeighEntry.11 Counter read-only mandatory

egpNeighIntervalHello egpNeighEntry.12 INTEGER read-only mandatory

egpNeighIntervalPoll egpNeighEntry.13 INTEGER read-only mandatory

egpNeighMode egpNeighEntry.14 INTEGER read-only mandatory

egpNeighEventTrigger egpNeighEntry.15 INTEGER read-write mandatory

egpAs egp.6 INTEGER read-only mandatory

snmpInPkts snmp.1 Counter read-only mandatory

snmpOutPkts snmp.2 Counter read-only mandatory

snmpInBadVersions snmp.3 Counter read-only mandatory

snmpInBadCommunityNames snmp.4 Counter read-only mandatory

snmpInBadCommunityUses snmp.5 Counter read-only mandatory

snmpInASNParseErrs snmp.6 Counter read-only mandatory

snmpInTooBigs snmp.8 Counter read-only mandatory

snmpInNoSuchNames snmp.9 Counter read-only mandatory

snmpInBadValues snmp.10 Counter read-only mandatory

snmpInReadOnlys snmp.11 Counter read-only mandatory

snmpInGenErrs snmp.12 Counter read-only mandatory

snmpInTotalReqVars snmp.13 Counter read-only mandatory

snmpInTotalSetVars snmp.14 Counter read-only mandatory

snmpInGetRequests snmp.15 Counter read-only mandatory

snmpInGetNexts snmp.16 Counter read-only mandatory

snmpInSetRequests snmp.17 Counter read-only mandatory

snmpInGetResponses snmp.18 Counter read-only mandatory

snmpInTraps snmp.19 Counter read-only mandatory

snmpOutTooBigs snmp.20 Counter read-only mandatory

snmpOutNoSuchNames snmp.21 Counter read-only mandatory

snmpOutBadValues snmp.22 Counter read-only mandatory

snmpOutGenErrs snmp.24 Counter read-only mandatory

snmpOutGetRequests snmp.25 Counter read-only mandatory

snmpOutGetNexts snmp.26 Counter read-only mandatory

snmpOutSetRequests snmp.27 Counter read-only mandatory

snmpOutGetResponses snmp.28 Counter read-only mandatory

snmpOutTraps snmp.29 Counter read-only mandatory

snmpEnableAuthenTraps snmp.30 INTEGER read-write mandatory

-- object definitions compiled from UNIX-MIB { iso 3 6 1 2 1 }

unix enterprises.4

agents unix.1

fourBSD-isode agents.2

mbuf unix.2

peers unix.3

unixd peers.1

smux unix.4

netstat unix.5

print unix.6

users unix.7

mbufs mbuf.1 Counter read-only mandatory

mbufClusters mbuf.2 Counter read-only mandatory

mbufFreeClusters mbuf.3 Counter read-only mandatory

mbufDrops mbuf.4 Counter read-only mandatory

mbufWaits mbuf.5 Counter read-only mandatory

mbufDrains mbuf.6 Counter read-only mandatory

mbufFrees mbuf.7 Counter read-only mandatory

mbufTable mbuf.8 Aggregate not-accessible mandatory

mbufEntry mbufTable.1 Aggregate not-accessible mandatory

mbufType mbufEntry.1 INTEGER read-only mandatory

mbufAllocates mbufEntry.2 Counter read-only mandatory

smuxPeerTable smux.1 Aggregate not-accessible mandatory

smuxPeerEntry smuxPeerTable.1 Aggregate not-accessible mandatory

smuxPindex smuxPeerEntry.1 INTEGER read-only mandatory

smuxPidentity smuxPeerEntry.2 ObjectID read-only mandatory

smuxPdescription smuxPeerEntry.3 DisplayString read-only mandatory

smuxPstatus smuxPeerEntry.4 INTEGER read-write mandatory

smuxTreeTable smux.2 Aggregate not-accessible mandatory

smuxTreeEntry smuxTreeTable.1 Aggregate not-accessible mandatory

smuxTsubtree smuxTreeEntry.1 ObjectID read-only mandatory

smuxTpriority smuxTreeEntry.2 INTEGER read-only mandatory

smuxTindex smuxTreeEntry.3 INTEGER read-only mandatory

smuxTstatus smuxTreeEntry.4 INTEGER read-write mandatory

unixNetstat netstat.1 INTEGER read-only mandatory

unixTcpConnTable netstat.2 Aggregate not-accessible mandatory

unixTcpConnEntry unixTcpConnTable.1 Aggregate not-accessible mandatory

unixTcpConnSendQ unixTcpConnEntry.1 INTEGER read-only mandatory

unixTcpConnRecvQ unixTcpConnEntry.2 INTEGER read-only mandatory

unixUdpTable netstat.3 Aggregate not-accessible mandatory

unixUdpEntry unixUdpTable.1 Aggregate not-accessible mandatory

unixUdpRemAddress unixUdpEntry.1 IpAddress read-only mandatory

unixUdpRemPort unixUdpEntry.2 INTEGER read-only mandatory

unixUdpSendQ unixUdpEntry.3 INTEGER read-only mandatory

unixUdpRecvQ unixUdpEntry.4 INTEGER read-only mandatory

unixIpRoutingTable netstat.4 Aggregate not-accessible mandatory

unixIpRouteEntry unixIpRoutingTable.1 Aggregate not-accessible mandatory

unixIpRouteFlags unixIpRouteEntry.1 INTEGER read-only mandatory

unixIpRouteRefCnt unixIpRouteEntry.2 INTEGER read-only mandatory

unixIpRouteUses unixIpRouteEntry.3 Counter read-only mandatory

unixRouteBadRedirects netstat.5 Counter read-only mandatory

unixRouteCreatedByRedirects netstat.6 Counter read-only mandatory

unixRouteModifiedByRedirects netstat.7 Counter read-only mandatory

unixRouteLookupFails netstat.8 Counter read-only mandatory

unixRouteWildcardUses netstat.9 Counter read-only mandatory

unixClnpRoutingTable netstat.10 Aggregate not-accessible mandatory

unixClnpRouteEntry unixClnpRoutingTable.1 Aggregate not-accessible mandatory

unixClnpRouteFlags unixClnpRouteEntry.1 INTEGER read-only mandatory

unixClnpRouteRefCnt unixClnpRouteEntry.2 INTEGER read-only mandatory

unixClnpRouteUses unixClnpRouteEntry.3 Counter read-only mandatory

printQTable print.1 Aggregate not-accessible mandatory

printQEntry printQTable.1 Aggregate not-accessible mandatory

printQName printQEntry.1 DisplayString read-only mandatory

printQStatus printQEntry.2 INTEGER read-only mandatory

printQDisplay printQEntry.3 DisplayString read-only mandatory

printQEntries printQEntry.4 INTEGER read-only mandatory

printQAction printQEntry.5 INTEGER read-write mandatory

printJTable print.2 Aggregate not-accessible mandatory

printJEntry printJTable.1 Aggregate not-accessible mandatory

printJRank printJEntry.1 INTEGER read-only mandatory

printJName printJEntry.2 DisplayString read-only mandatory

printJOwner printJEntry.3 DisplayString read-only mandatory

printJDescription printJEntry.4 DisplayString read-only mandatory

printJSize printJEntry.5 INTEGER read-only mandatory

printJAction printJEntry.6 INTEGER read-write mandatory

userTable users.1 Aggregate not-accessible mandatory

userEntry userTable.1 Aggregate not-accessible mandatory

userName userEntry.1 DisplayString read-write mandatory

userPasswd userEntry.2 DisplayString read-write mandatory

userID userEntry.3 INTEGER read-write mandatory

userGroup userEntry.4 DisplayString read-write mandatory

userQuota userEntry.5 INTEGER read-write mandatory

userComment userEntry.6 DisplayString read-write mandatory

userFullName userEntry.7 DisplayString read-write mandatory

userHome userEntry.8 DisplayString read-write mandatory

userShell userEntry.9 DisplayString read-write mandatory

userStatus userEntry.10 INTEGER read-write mandatory

groupTable users.2 Aggregate not-accessible mandatory

groupEntry groupTable.1 Aggregate not-accessible mandatory

groupName groupEntry.1 DisplayString read-write mandatory

groupPasswd groupEntry.2 DisplayString read-write mandatory

groupID groupEntry.3 INTEGER read-write mandatory

groupStatus groupEntry.4 INTEGER read-write mandatory

grUserTable users.3 Aggregate not-accessible mandatory

grUserEntry grUserTable.1 Aggregate not-accessible mandatory

grUserStatus grUserEntry.1 INTEGER read-write mandatory

-- object definitions compiled from RFCxxxx-MIB { iso 3 6 1 2 1 }

view experimental.11

defaultView view.2

defViewWholeRW defaultView.1

defViewWholeRO defaultView.2

defViewStandardRW defaultView.3

defViewStandardRO defaultView.4

defViewTrapDest defaultView.5

viewDomains view.3

localAgent 0.0

snmpDomain viewDomains.1

rfc1157Domain snmpDomain.1

cltsDomain snmpDomain.3

cotsNDomain snmpDomain.4

cotsXDomain snmpDomain.5

viewPrimTable view.1 Aggregate not-accessible mandatory

viewPrimEntry viewPrimTable.1 Aggregate not-accessible mandatory

viewPrimName viewPrimEntry.1 ObjectID read-write mandatory

viewPrimTDomain viewPrimEntry.2 ObjectID read-write mandatory

viewPrimTAddr viewPrimEntry.3 IpAddress read-write mandatory

viewPrimUser viewPrimEntry.4 DisplayString read-write mandatory

viewPrimCommunity viewPrimEntry.5 DisplayString read-write mandatory

viewPrimType viewPrimEntry.6 INTEGER read-write mandatory

viewAclTable view.4 Aggregate not-accessible mandatory

viewAclEntry viewAclTable.1 Aggregate not-accessible mandatory

viewAclView viewAclEntry.1 ObjectID read-write mandatory

viewAclCommunity viewAclEntry.2 DisplayString read-write mandatory

viewAclUser viewAclEntry.3 DisplayString read-write mandatory

viewAclPrivileges viewAclEntry.4 INTEGER read-write mandatory

viewAclType viewAclEntry.5 INTEGER read-write mandatory

viewTrapTable view.5 Aggregate not-accessible mandatory

viewTrapEntry viewTrapTable.1 Aggregate not-accessible mandatory

viewTrapView viewTrapEntry.1 ObjectID read-write mandatory

viewTrapGenerics viewTrapEntry.2 OctetString read-write mandatory

viewTrapSpecifics viewTrapEntry.3 OctetString read-write mandatory

viewTrapType viewTrapEntry.4 INTEGER read-write mandatory

viewTranTable view.6 Aggregate not-accessible mandatory

viewTranEntry viewTranTable.1 Aggregate not-accessible mandatory

viewSourceName viewTranEntry.1 ObjectID read-write mandatory

viewSourceSubtree viewTranEntry.2 ObjectID read-write mandatory

viewTargetName viewTranEntry.3 ObjectID read-write mandatory

viewTargetSubtree viewTranEntry.4 ObjectID read-write mandatory

viewTranType viewTranEntry.5 INTEGER read-write mandatory

-- object definitions compiled from IBM-MIB { iso 3 6 1 4 1 2 }

ibm enterprises.2

ibmAgents ibm.3

aix ibmAgents.1

mvs ibmAgents.2

vm ibmAgents.3

os2 ibmAgents.4

os400 ibmAgents.5

aixRT aix.1

aixRISC6000 aix.2

aix370 aix.3

aixPS2 aix.4

mvs370 mvs.1

vm370 vm.1

ps2PS2 os2.1

os400as400 os400.1

risc6000agents aixRISC6000.1

risc6000private aixRISC6000.2

risc6000public aixRISC6000.3

risc6000snmpd risc6000agents.1

risc6000gated risc6000agents.2

risc6000xmservd risc6000agents.3

risc6000ibm7318 risc6000agents.4

risc6000clsmuxpd risc6000agents.5

risc6000samples risc6000private.1

risc6000sampleAgents risc6000samples.1

risc6000perf risc6000private.2

-- automatically generated by mosy 6.8 of Sat Nov 23 12:30:27 CST 1991, do not edit!

-- object definitions compiled from RFC1317-MIB

rs232 transmission.33

rs232Number rs232.1 INTEGER read-only mandatory

rs232PortTable rs232.2 Aggregate not-accessible mandatory

rs232PortEntry rs232PortTable.1 Aggregate not-accessible mandatory

rs232PortIndex rs232PortEntry.1 INTEGER read-only mandatory

rs232PortType rs232PortEntry.2 INTEGER read-only mandatory

rs232PortInSigNumber rs232PortEntry.3 INTEGER read-only mandatory

rs232PortOutSigNumber rs232PortEntry.4 INTEGER read-only mandatory

rs232PortInSpeed rs232PortEntry.5 INTEGER read-write mandatory

rs232PortOutSpeed rs232PortEntry.6 INTEGER read-write mandatory

rs232AsyncPortTable rs232.3 Aggregate not-accessible mandatory

rs232AsyncPortEntry rs232AsyncPortTable.1 Aggregate not-accessible mandatory

rs232AsyncPortIndex rs232AsyncPortEntry.1 INTEGER read-only mandatory

rs232AsyncPortBits rs232AsyncPortEntry.2 INTEGER read-write mandatory

rs232AsyncPortStopBits rs232AsyncPortEntry.3 INTEGER read-write mandatory

rs232AsyncPortParity rs232AsyncPortEntry.4 INTEGER read-write mandatory

rs232AsyncPortAutobaud rs232AsyncPortEntry.5 INTEGER read-write mandatory

rs232AsyncPortParityErrs rs232AsyncPortEntry.6 Counter read-only mandatory

rs232AsyncPortFramingErrs rs232AsyncPortEntry.7 Counter read-only mandatory

rs232AsyncPortOverrunErrs rs232AsyncPortEntry.8 Counter read-only mandatory

rs232SyncPortTable rs232.4 Aggregate not-accessible mandatory

rs232SyncPortEntry rs232SyncPortTable.1 Aggregate not-accessible mandatory

rs232SyncPortIndex rs232SyncPortEntry.1 INTEGER read-only mandatory

rs232SyncPortClockSource rs232SyncPortEntry.2 INTEGER read-write mandatory

rs232SyncPortFrameCheckErrs rs232SyncPortEntry.3 Counter read-only mandatory

rs232SyncPortTransmitUnderrunErrs rs232SyncPortEntry.4 Counter read-only mandatory

rs232SyncPortReceiveOverrunErrs rs232SyncPortEntry.5 Counter read-only mandatory

rs232SyncPortInterruptedFrames rs232SyncPortEntry.6 Counter read-only mandatory

rs232SyncPortAbortedFrames rs232SyncPortEntry.7 Counter read-only mandatory

rs232InSigTable rs232.5 Aggregate not-accessible mandatory

rs232InSigEntry rs232InSigTable.1 Aggregate not-accessible mandatory

rs232InSigPortIndex rs232InSigEntry.1 INTEGER read-only mandatory

rs232InSigName rs232InSigEntry.2 INTEGER read-only mandatory

rs232InSigState rs232InSigEntry.3 INTEGER read-only mandatory

rs232InSigChanges rs232InSigEntry.4 Counter read-only mandatory

rs232OutSigTable rs232.6 Aggregate not-accessible mandatory

rs232OutSigEntry rs232OutSigTable.1 Aggregate not-accessible mandatory

rs232OutSigPortIndex rs232OutSigEntry.1 INTEGER read-only mandatory

rs232OutSigName rs232OutSigEntry.2 INTEGER read-only mandatory

rs232OutSigState rs232OutSigEntry.3 INTEGER read-only mandatory

rs232OutSigChanges rs232OutSigEntry.4 Counter read-only mandatory

-- automatically generated by mosy 6.8 of Sat Nov 23 12:30:27 CST 1991, do not edit!

-- object definitions compiled from RFC1318-MIB

para transmission.34

paraNumber para.1 INTEGER read-only mandatory

paraPortTable para.2 Aggregate not-accessible mandatory

paraPortEntry paraPortTable.1 Aggregate not-accessible mandatory

paraPortIndex paraPortEntry.1 INTEGER read-only mandatory

paraPortType paraPortEntry.2 INTEGER read-only mandatory

paraPortInSigNumber paraPortEntry.3 INTEGER read-only mandatory

paraPortOutSigNumber paraPortEntry.4 INTEGER read-only mandatory

paraInSigTable para.3 Aggregate not-accessible mandatory

paraInSigEntry paraInSigTable.1 Aggregate not-accessible mandatory

paraInSigPortIndex paraInSigEntry.1 INTEGER read-only mandatory

paraInSigName paraInSigEntry.2 INTEGER read-only mandatory

paraInSigState paraInSigEntry.3 INTEGER read-only mandatory

paraInSigChanges paraInSigEntry.4 Counter read-only mandatory

paraOutSigTable para.4 Aggregate not-accessible mandatory

paraOutSigEntry paraOutSigTable.1 Aggregate not-accessible mandatory

paraOutSigPortIndex paraOutSigEntry.1 INTEGER read-only mandatory

paraOutSigName paraOutSigEntry.2 INTEGER read-only mandatory

paraOutSigState paraOutSigEntry.3 INTEGER read-only mandatory

paraOutSigChanges paraOutSigEntry.4 Counter read-only mandatory

-- automatically generated by mosy 6.8 of Sat Nov 23 12:30:27 CST 1991, do not edit!

-- object definitions compiled from RFC1316-MIB

char mib-2.19

wellKnownProtocols char.4

protocolOther wellKnownProtocols.1

protocolTelnet wellKnownProtocols.2

protocolRlogin wellKnownProtocols.3

protocolLat wellKnownProtocols.4

protocolX29 wellKnownProtocols.5

protocolVtp wellKnownProtocols.6

charNumber char.1 INTEGER read-only mandatory

charPortTable char.2 Aggregate not-accessible mandatory

charPortEntry charPortTable.1 Aggregate not-accessible mandatory

charPortIndex charPortEntry.1 INTEGER read-only mandatory

charPortName charPortEntry.2 DisplayString read-write mandatory

charPortType charPortEntry.3 INTEGER read-only mandatory

charPortHardware charPortEntry.4 ObjectID read-only mandatory

charPortReset charPortEntry.5 INTEGER read-write mandatory

charPortAdminStatus charPortEntry.6 INTEGER read-write mandatory

charPortOperStatus charPortEntry.7 INTEGER read-only mandatory

charPortLastChange charPortEntry.8 TimeTicks read-only mandatory

charPortInFlowType charPortEntry.9 INTEGER read-write mandatory

charPortOutFlowType charPortEntry.10 INTEGER read-write mandatory

charPortInFlowState charPortEntry.11 INTEGER read-only mandatory

charPortOutFlowState charPortEntry.12 INTEGER read-only mandatory

charPortInCharacters charPortEntry.13 Counter read-only mandatory

charPortOutCharacters charPortEntry.14 Counter read-only mandatory

charPortAdminOrigin charPortEntry.15 INTEGER read-write mandatory

charPortSessionMaximum charPortEntry.16 INTEGER read-write mandatory

charPortSessionNumber charPortEntry.17 Gauge read-only mandatory

charPortSessionIndex charPortEntry.18 INTEGER read-only mandatory

charSessTable char.3 Aggregate not-accessible mandatory

charSessEntry charSessTable.1 Aggregate not-accessible mandatory

charSessPortIndex charSessEntry.1 INTEGER read-only mandatory

charSessIndex charSessEntry.2 INTEGER read-only mandatory

charSessKill charSessEntry.3 INTEGER read-write mandatory

charSessState charSessEntry.4 INTEGER read-only mandatory

charSessProtocol charSessEntry.5 ObjectID read-only mandatory

charSessOperOrigin charSessEntry.6 INTEGER read-only mandatory

charSessInCharacters charSessEntry.7 Counter read-only mandatory

charSessOutCharacters charSessEntry.8 Counter read-only mandatory

charSessConnectionId charSessEntry.9 ObjectID read-only mandatory

charSessStartTime charSessEntry.10 TimeTicks read-only mandatory

-- automatically generated by mosy 6.8 of Sat Nov 23 12:30:27 CST 1991, do not edit!

-- object definitions compiled from RFC1155-SMI { iso 3 6 1 }

internet iso.3.6.1

directory internet.1

mgmt internet.2

experimental internet.3

private internet.4

enterprises private.1

faa enterprises.2120

ibm enterprises.2

ibmAgents ibm.3

aix ibmAgents.1

aixRISC6000 aix.2

risc6000agents aixRISC6000.1

risc6000private aixRISC6000.2

risc6000public aixRISC6000.3

risc6000samples risc6000private.1

risc6000sampleAgents risc6000samples.1

dpid risc6000sampleAgents.2

york ibm.2

dpismux york.1

dpiSMUXport dpismux.1 INTEGER read-only mandatory

--faa dpismux.2120

labsupt faa.5

test_procs labsupt.1

system_health labsupt.2

sh_reports system_health.1 DisplayString read-only mandatory

lcnOp test_procs.1

bcnOp test_procs.2

lcnSar test_procs.3

bcnSar test_procs.4

esip test_procs.5

halp test_procs.6

warp test_procs.7

lcnOpFileSystems lcnOp.1

lcnOpInterfaces lcnOp.2

lcnOpDSRstate lcnOp.3

lcnOpSWrunning lcnOpDSRstate.1 DisplayString read-only mandatory

lcnOpRelCount lcnOpDSRstate.2 DisplayString read-only mandatory

lcnOpCurrentRel lcnOpDSRstate.3 DisplayString read-only mandatory

lcnOpFallbackRel lcnOpDSRstate.4 DisplayString read-only mandatory

DSRRelTable lcnOp.4 Aggregate not-accessible mandatory

DSRRelEntry DSRRelTable.1 Aggregate not-accessible mandatory

DSRRelName DSRRelEntry.1 DisplayString read-only mandatory

DSRRelAutoReboot DSRRelEntry.2 DisplayString read-only mandatory

DSRRelDisableLogin DSRRelEntry.3 DisplayString read-only mandatory

DSRRelDebugAction DSRRelEntry.4 DisplayString read-only mandatory

DSRRelLinkCheck DSRRelEntry.5 DisplayString read-only mandatory

lcnOpFSslot_a_r lcnOpFileSystems.1 DisplayString read-only mandatory

lcnOpFSslot_a_t lcnOpFileSystems.2 DisplayString read-only mandatory

lcnOpFSaasdebug lcnOpFileSystems.3 DisplayString read-only mandatory

lcnOpMaintLan lcnOpInterfaces.1 DisplayString read-only mandatory

lcnOpTokB lcnOpInterfaces.2 DisplayString read-only mandatory

lcnOpTokP lcnOpInterfaces.3 DisplayString read-only mandatory

lcnOpTokR lcnOpInterfaces.4 DisplayString read-only mandatory

lcnOpTokS lcnOpInterfaces.5 DisplayString read-only mandatory

bcnOpFileSystems bcnOp.1

bcnOpInterfaces bcnOp.2

bcnOpDSRstate bcnOp.3

bcnOpDSRRel bcnOp.4

bcnOpFSslot_a_r	bcnOpFileSystems.1	DisplayString	read-only	mandatory
bcnOpFSslot_a_t	bcnOpFileSystems.2	DisplayString	read-only	mandatory
bcnOpFSaasdebug	bcnOpFileSystems.3	DisplayString	read-only	mandatory
bcnOpMaintLan	bcnOpInterfaces.1	DisplayString	read-only	mandatory
bcnOpBCN	bcnOpInterfaces.2	DisplayString	read-only	mandatory
bcnOpRelChk	bcnOpDSRstate.1	DisplayString	read-only	mandatory
bcnOpSWrunning	bcnOpDSRstate.2	DisplayString	read-only	mandatory

lcnSarFileSystems	lcnSar.1			
lcnSarInterfaces	lcnSar.2			
lcnSarDSRstate	lcnSar.3			
lcnSarDSRRel	lcnSar.4			
lcSarFSslot_a_r	lcnSarFileSystems.1	DisplayString	read-only	mandatory
lcnSarFSslot_a_t	lcnSarFileSystems.2	DisplayString	read-only	mandatory
lcnSarFSaasdebug	lcnSarFileSystems.3	DisplayString	read-only	mandatory
lcnSarFSvolcat	lcnSarFileSystems.4	DisplayString	read-only	mandatory
lcnSarFSdebugop	lcnSarFileSystems.5	DisplayString	read-only	mandatory
lcnSarFScsrd	lcnSarFileSystems.6	DisplayString	read-only	mandatory
lcnSarMaintLan	lcnSarInterfaces.1	DisplayString	read-only	mandatory
lcnSarTokB	lcnSarInterfaces.2	DisplayString	read-only	mandatory
lcnSarTokP	lcnSarInterfaces.3	DisplayString	read-only	mandatory
lcnSarTokR	lcnSarInterfaces.4	DisplayString	read-only	mandatory
lcnSarTokS	lcnSarInterfaces.5	DisplayString	read-only	mandatory
lcnSarAuxLan	lcnSarInterfaces.6	DisplayString	read-only	mandatory
lcnSarRelChk	lcnSarDSRstate.1	DisplayString	read-only	mandatory
lcnSarSWrunning	lcnSarDSRstate.2	DisplayString	read-only	mandatory

Appendix B SWAC Configuration Items

B.1 Swac.mdx

```
-- CM_COPYRIGHT_EXPANSION_PLACEHOLDER DSR, URET

-----------------------------------

-- Health metric swac.mdx file --

--  @(#) ofpd/ofpd_tools/routines/system_metrics/nonop/nshrswac.mdx, 16.2, Tue Aug  2 14:10:02 2005

-----------------------------------


-- Commands and command feedback

-----------------------------------

-- DPE swac filters

filter  D_CMD       all

filter  D13_CMD     all

filter  D_CMDFB     all

filter  D_CMDRS     all

filter  A_CMD       all

filter  A13_CMD     all

filter  A_CMDFB     all

filter  A_CMDRS     all

filter  R_CMD       all

filter  R13_CMD     all

filter  MC_CMD      all

filter  R_CMDFB     all

filter  MC_CMDFB    all

filter  MC_CMDRS    all


-- EDARC lost RADAR

-------------------

format UF_A2133 12133 no_stdhdr

  "    RADAR  "

  if root = True

    "RECEIVED"

  else

    "LOST"
```

```
  endif
done
filter UF_A2133 all

-- HOST lost RADAR
--------------------
format UF_A2013 12013 no_stdhdr
  "    RADAR  "
  if root = True
    "RECEIVED"
  else
    "LOST"
  endif
done
filter UF_A2013 all

-- Session infomation
----------------------
-- Session status from Inter (message 9871)
filter  SESS        all
filter  SESSEVNT    all

-- Low level session aborts....
format SESSABRT 10091 no_stdhdr
  if rechdr.APPL_NAME.APPL_ID = 2
    if root.ERROR_ID = 1000
      "PMS recorded a session abort!  "
      field special root.CALL_CHAIN
    endif
  endif
  done
filter SESSABRT all

-- proc/as/hw status
--------------------
filter  PROCFAIL    all
filter  FAIL_AS     all
```

```
filter  LOAD_AS     all
filter  HW_STAT     all
filter  NMG_CSU     all

format GROUPERR   15  stdhdr
 loop root.PROC_LIST'RANGE
   if root.PROC_LIST(X).PROC_STATUS.PROCESSOR_ID /= 0
     if root.PROC_LIST(X).ERROR_INFO.ERROR_CODE  /= 0
       "PROC="
       field 4 root.PROC_LIST(X).PROC_STATUS.PROCESSOR_ID
       "ERR= "
       field 5 root.PROC_LIST(X).ERROR_INFO.ERROR_CODE
       field special 30 root.PROC_LIST(X).ERROR_INFO.ERROR_CODE
       newline
     endif
   endif
 endloop
 done

-- Error reporting heirarchy
----------------------------
filter  ERR_LOG     all
filter  EXCP        all
filter  WEER_AER    all
filter  AS_ERR      all
filter  GROUPERR    all
filter  FAL_ALT     all
filter  AAR_ERR     all
filter  AUD_ERR     all

-- SWAC formatter for flow control - ADEs
-----------------------------------------
-- Macro to print time (relies on a time field
-- being the first field in the tooled data structure -
-- at byte 0!).
format TIME 67 no_stdhdr
  field special root.COMMAND_ID.TIME
```

```
 done

-- Appl queue flow control (preceeded by WEER 1042)
format FLOWAPLQ 10091 no_stdhdr
 if root.ERROR_ID = -1807
   field special  5  rechdr.APPL_NAME.APPL_ID
   "Appl FC_Enabled= "
   if root.DATA7 = 0
     "F "
   else
     "T "
   endif
   if root.DATA4 = 0
     "RATE "
   else
     "Q    "
   endif
   "XCLS= "
   field root.DATA3
   "  FrmLim= "
   field root.DATA5
   "  Delay= "
   field root.DATA9
   "  TotalDelay= "
   field root.DATA10
 endif
 done
filter FLOWAPLQ all


-- Appl rate flow control (preceeded by WEER 1043)
format FLOWAPLR 10091 no_stdhdr
 if root.ERROR_ID = -1808
   field special  5  rechdr.APPL_NAME.APPL_ID
   "Appl FC_Enabled= "
   if root.DATA7 = 0
     "F "
```

```
   else
    "T "
   endif
   if root.DATA4 = 0
    "RATE "
   else
    "Q    "
   endif
   " SampSiz="
   field root.DATA5
   "  Delay= "
   field root.DATA9
   "  TotalDelay="
   field root.DATA10
  endif
  done
filter FLOWAPLR all


-- OSE frame rate (preceeded by WEER 1045) and OSE queue (preceeded by WEER 1044) flow control
format FLOW_OSE 10091 no_stdhdr
  if root.ERROR_ID = -1805
   field special  5  rechdr.APPL_NAME.APPL_ID
   "OSE  FC_Enabled= "
   if root.DATA7 = 0
    "F "
   else
    "T "
   endif
   if root.DATA4 = 0
    "RATE "
   else
    "Q    "
   endif
   " Total_Time_in_FC= "
   field root.DATA9
   "  Time_Delay_Started= "
```

```
   field root.DATA11
  endif
  done
filter FLOW_OSE all


-- RECOVERED FROM FRAME LIMIT APPLICATION FLOW CONTROL
format FLOWAPLQ 10052 no_stdhdr
  if root.ACTIVITY_ID = 19098
    field special  5  rechdr.APPL_NAME.APPL_ID
    "Appl FC_Enabled= "
    if root.DATA7 = 0
      "F "
    else
      "T "
    endif
    if root.DATA4 = 0
      "RATE "
    else
      "Q    "
    endif
    "XCLS= "
    field root.DATA3
    "  FrmLim= "
    field root.DATA5
    "  Delay= "
    field root.DATA9
    "  TotalDelay= "
    field root.DATA10
  endif
  done
filter FLOWAPLQ all


-- RECOVERED FROM RATE LIMIT APPLICATION FLOW CONTROL
format FLOWAPLR 10052 no_stdhdr
  if root.ACTIVITY_ID = 19096
```

```
   field special  5  rechdr.APPL_NAME.APPL_ID
   "Appl FC_Enabled= "
   if root.DATA7 = 0
    "F "
   else
    "T "
   endif
   if root.DATA4 = 0
    "RATE "
   else
    "Q    "
   endif
   " SampSiz="
   field root.DATA5
   "  Delay= "
   field root.DATA9
   "  TotalDelay="
   field root.DATA10
  endif
  done
filter FLOWAPLR all


-- RECOVERED FROM OSE FLOW CONTROL
format FLOW_OSE 10052 no_stdhdr
  if root.ACTIVITY_ID = 19093
   field special  5  rechdr.APPL_NAME.APPL_ID
   "OSE  FC_Enabled= "
   if root.DATA7 = 0
    "F "
   else
    "T "
   endif
   if root.DATA4 = 0
    "RATE "
   else
    "Q    "
```

```
   endif
   " Total_Time_in_FC= "
   field root.DATA9
   "  Time_Delay_Started= "
   field root.DATA11
  endif
  done
filter FLOW_OSE all

-- Route failures
-----------------
filter  RT_FAIL     all

-- Receiver congestion
----------------------
format RCVRCONG 11054 no_stdhdr
  if root.Congested_Component = PROCESSOR_ADAPTER
    "PROC "
  elsif root.Congested_Component = BRIDGE_ADAPTER
    "BRDG "
  else
    "UNKN "
  endif
  field 3 root.Component_Id
  "  AD= "
  field hex 4 trunc right root.Adapter_Addr(1)
  field hex 4 trunc right root.Adapter_Addr(2)
  field hex 4 trunc right root.Adapter_Addr(3)
  "   Rng= "
  field 2 root.Ring_Id
  "  "
  field 8 root.Transmission_Class
  "   "
  if root.Recovered = TRUE
    "RECOVERY"
  else
    "ERROR"
```

```
  endif
done
filter RCVRCONG all

-- BCN Dropped messages
-----------------------
filter  BCN_ERRS    all

-- PERFORMANCE STUFF
--------------------
format DCXUTIL 1633 stdhdr
  "CPU_Idle= "
  field 9 root.Util.Cpu_Idle
  "  MEM_Init= "
  field 7 root.Util.Memory_Init
  "  MEM_Free= "
  field 7 root.Util.Memory_Free
  "  RGL_Mem= "
  field 7 root.Util.Rgl_Memory
  "  CDG_Mem= "
  field 8 root.Util.Cdg_Memory
  "  Interval= "
  field 5 root.Interval
  "   "
  field 4 root.Oper_Role
done
filter DCXUTIL all

format LCN_AD 108 stdhdr
  loop root.NETWORK_DATA.ADAPTER_DATA'RANGE
    if root.NETWORK_DATA.ADAPTER_DATA(X).RING_ID /= 0
      "RNG= "
      field 3 root.NETWORK_DATA.ADAPTER_DATA(X).RING_ID
      "SENT_DG= "
      field 11 root.NETWORK_DATA.ADAPTER_DATA(X).NUM_BYTES_SENT_DG
      "SENT_CNCT= "
      field 11 root.NETWORK_DATA.ADAPTER_DATA(X).NUM_BYTES_SENT_CNCT
```

```
    "THRESHOLD= 0"
    newline
  endif
 endloop
 done

format BRDG_AD 1693 stdhdr
 loop 1 root.BRIDGE_COUNT
  "BRDG= "
  field 3 root.BRIDGE_TABLE(X).BRIDGE_ID
  "RING= "
  field 3 root.BRIDGE_TABLE(X).ADAPTER_INFO(1).RING_ID
  " AD= 1 BCAST= "
  field 11 root.BRIDGE_TABLE(X).ADAPTER_INFO(1).NUM_OF_BROADCAST_BYTES
  " NBCAST= "
  field 11 root.BRIDGE_TABLE(X).ADAPTER_INFO(1).NUM_OF_NON_BROADCAST_BYTES
  if root.BRIDGE_TABLE(X).ADAPTER_INFO(1).BRIDGE_RESET_COUNTERS = TRUE
    " RESET= TRUE "
  else
    " RESET= FALSE "
  endif
  " THRESHLD= 0"
  newline
  "BRDG= "
  field 3 root.BRIDGE_TABLE(X).BRIDGE_ID
  "RING= "
  field 3 root.BRIDGE_TABLE(X).ADAPTER_INFO(2).RING_ID
  " AD= 2 BCAST= "
  field 11 root.BRIDGE_TABLE(X).ADAPTER_INFO(2).NUM_OF_BROADCAST_BYTES
  " NBCAST= "
  field 11 root.BRIDGE_TABLE(X).ADAPTER_INFO(2).NUM_OF_NON_BROADCAST_BYTES
  if root.BRIDGE_TABLE(X).ADAPTER_INFO(2).BRIDGE_RESET_COUNTERS = TRUE
   " RESET= TRUE "
  else
    " RESET= FALSE "
  endif
  " THRESHLD= 0"
```

```
   newline
  endloop
  done
filter  LCN_AD      all
filter  BRDG_AD     all
filter  PERF_BCN    all

-- CPU/370/488 Util  &  Configuration info too
-- Note: Hw_Perf index "4" comes from smgt_common (Device_Index_Rec) in smbs000b.ada
format UF_M108 108 stdhdr
  "CPU%= "
  field 5 root.Proc_Perf.Cpu_Util
  "   Role= "
  lookup WSGRC.ROLE_ID_T root.Processor_Role
  "  HWTYPE= "
  field 25 root.Hw_Perf(4).Hw_Item.Hw_Type
  "%util= "
  field 3 root.Hw_Perf(4).Hw_Data.Pct_Hw_Util
  "       HWTYPE= "
  field 25 root.Hw_Perf(5).Hw_Item.Hw_Type
  "   %util= "
  field 3 root.Hw_Perf(5).Hw_Data.Pct_Hw_Util
  "   "
  field 5 root.Group_Id
  "   "
  field 8 root.Proc_Perf.Mem_Virtual
  "   "
  lookup WSPRC.PROC_CONFIG_T root.Processor_Config
done
filter UF_M108 all

-- Use only BS's SAR data, to avoid double counting SMDT's recording
format PERFFDDI 108   stdhdr
 if rechdr.APPL_NAME.APPL_ID = 6
  loop root.ADAPTER_DATA'RANGE
    if root.ADAPTER_DATA(X).HW_ITEM.HW_TYPE = FDDI_ADAPTER
      field special root.ADAPTER_DATA(X).PERF_DATA.ADAPTER_ADDRESS
```

```
    " FRM: R= "
    field  6 root.ADAPTER_DATA(X).PERF_DATA.NUM_FRAMES_RECV
    " S= "
    field  6 root.ADAPTER_DATA(X).PERF_DATA.NUM_FRAMES_SENT
    " KBYTE: R= "
    field  6 root.ADAPTER_DATA(X).PERF_DATA.NUM_BYTES_RECV
    " S= "
    field  6 root.ADAPTER_DATA(X).PERF_DATA.NUM_BYTES_SENT
    newline
  endif
 endloop
endif
done

filter PERFFDDI all

-- Process FDDI Status data
-- 10603
-- BS: whlum00b.ada:
--    Device_Info_T_Alignment : constant := 8;
--    for Device_Info_T'Alignment use Device_Info_T_Alignment;
--
--    type Adapter_Perf_Sar_Data_T is
--    record
--      Hw_Item     : Fdk.Com.Wshwd.Hw_Item_T; -- 8-bytes
--      Device_Info : Device_Info_T; -- see what wadev.c (called from wanad.c) puts here
--                                   -- based on type of device
--    end record;
-- U2: what follows is wanad.h's fd_wanad_device_info_t, which has a 4-byte device id first,
--     then adapter-specific data
-- U3: what follows is the adapter-specific data, as in wadev.h
--
-- hw_item at front of record, 27 => Wshwd.Fddi_Adapter

-- Exercise care in use: the values in the 10603 are cumulative
-- values from when the processor was last powered up. They
-- are NOT delta values since the last report.
```

```
-- 20000509 1959:00.000 FDDISTAT   63 <ierrors> <oerrors> <notxbufs> <norxbufs> <noxsmtbufs> <txsetuperr>
--                              <rxpfail> <rxpefail> <rxdupfail> <nocanputs> <badfc>


byte_format FDDISTAT 10603 \
  %if b3.0.8 = 27 %macro FDDISTAM,b8  %endif

filter FDDISTAT all

byte_format FDDISTAM macro_only \
                        %iu,b4  \
            \c49        %iu,b20  \
            \c60        %iu,b28  \
            \c71        %iu,b40  \
            \c82        %iu,b48  \
            \c93        %iu,b52  \
            \c104       %iu,b56  \
            \c115       %iu,b60  \
            \c126       %iu,b64  \
            \c137       %iu,b68  \
            \c148       %iu,b72

filter  TIMEMSTR    all

-- Other URET additions
filter AAR_NERR all
filter AUD_NERR all

-- Filters and formats for capturing Lost SAR information
-- Lost recording for errors, state, and msgs for both
-- the applications and the kernal are summed

-- NOTE on second loop:
-- we really want 0 .. TOTAL_RECODINGS_LOST - 1, but SWAC does not
-- support it, so we need to test the index to prevent running past the
-- end of the buffer.  Once the index goes past 90 and 180, we wants a
-- new line so that each line does not have more than 99 tokens.  That
```

```
-- is taken into account that SWAC puts a few more tokens at the beginning
-- of the line
--comment_block_stop
format LOSTSAR3 10095 no_stdhdr
  "Total lost:"
  field root.TOTAL_RECORDINGS_LOST
  " Total bytes lost:"
  field root.TOTAL_BYTES_LOST
  newline
  "Lost gates in order:"
  if root.TOTAL_RECORDINGS_LOST > 200
    loop 0 199
      if index = 90
        newline
      endif
      if index = 180
        newline
      endif
      field root.LOST_LIST(X).GATE_LOST
      " "
    endloop
  else
    loop 0 root.TOTAL_RECORDINGS_LOST
      if index = 90
        newline
      endif
      if index = 180
        newline
      endif
      if index /= root.TOTAL_RECORDINGS_LOST
        field root.LOST_LIST(X).GATE_LOST
        " "
      else
        "<<"
      endif
    endloop
  endif
```

```
done

-- NOTE on second loop:
-- we really want 0 .. TOTAL_RECODINGS_LOST - 1, but SWAC does not
-- support it, so we need to test the index to prevent running past the
-- end of the buffer.  Once the index goes past 90 and 180, we wants a
-- new line so that each line does not have more than 99 tokens.  That
-- is taken into account that SWAC puts a few more tokens at the beginning
-- of the line
--comment_block_stop
format LOSTSAR4 10097 no_stdhdr
 "Total lost:"
 field root.TOTAL_RECORDINGS_LOST
 " Total bytes lost:"
 field root.TOTAL_BYTES_LOST
 newline
 "Lost gates in order:"
 if root.TOTAL_RECORDINGS_LOST > 200
  loop 0 199
   if index = 90
    newline
   endif
   if index = 180
    newline
   endif
   field root.LOST_LIST(X).GATE_LOST
   " "
  endloop
 else
  loop 0 root.TOTAL_RECORDINGS_LOST
   if index = 90
    newline
   endif
   if index = 180
    newline
   endif
   if index /= root.TOTAL_RECORDINGS_LOST
```

```
      field root.LOST_LIST(X).GATE_LOST
      " "
    else
      "<<"
    endif
  endloop
 endif
done

filter LOSTSAR3 all
filter LOSTSAR4 all

-- RECON
filter WX_RECON all
-- Weather reconstitution requests

filter CPDRECON all
-- CPD reconstitution requests

filter CPDRCOMP all
-- CPD reconstitution complete messages



-- TRPSTAT
-- NOTE:
-- for the token-ring adapters on the DSGW and EDIS boxes.
-- note: 13,35,37 are hardware-ids as defined
--       in fdcom/fdcom_exports/wshwd00s.ada
--       for "Lan_8025", "Atc_Lcn_Adapter", "Spare_Lcn_Adapter"
--       in u3.1d3 baseline.
-- BS 10603 begins with a wshwd.hw_item_t object
-- which has a 4-byte hw_type field, then 4-byte hw_item.
-- This is followed by the device-specific data.
byte_format TRPSTAT 10603 \
 %if    b3.0.8 = 13  %macro RECHDRA,b0 "lan802.5  " %macro TRPSTATM,b8 \
                \n %macro RECHDRA,b0 \
 %elsif b3.0.8 = 35  %macro RECHDRA,b0 "lcn:atc    " %macro TRPSTATM,b8 \
```

```
                \n %macro RECHDRA,b0 \
%elsif b3.0.8 = 37  %macro RECHDRA,b0 "lcn:spare " %macro TRPSTATM,b8 \
                \n %macro RECHDRA,b0 \
%endif

filter TRPSTAT BS

-- comments say this field is unused  " link_status: " %iu,b32 \
byte_format TRPSTATM macro_only \
  " link_status/type: " %iu,b32 \
  "/" %iu,b36 \
  " int: " %iu,b56 \
  " speed: " %iu,b112 \
  " inits: " %iu,b108 \
  " nocanput: " %iu,b116 \
  " noallocb: " %iu,b120 \
  \n %macro RECHDRA,b0 \
  "in:  packets: " %iu,b0 \
     " multi: "   %iu,b8 \
     " brdcst: "  %iu,b16 \
     " errors: "  %iu,b24 \
     " bytes: "   %iu,b44 \
     " nobuf: "   %iu,b48 \
  \n %macro RECHDRA,b0 \
  "out: packets: " %iu,b4 \
     " multi: "   %iu,b12 \
     " brdcst: "  %iu,b20 \
     " errors: "  %iu,b28 \
     " bytes: "   %iu,b40 \
     " nobuf: "   %iu,b52 \
  \n %macro RECHDRA,b0 \
  "errors: line: "    %iu,b60 \
     " internal: "    %iu,b64 \
     " burst: "       %iu,b68 \
     " ari_fci: "     %iu,b72 \
     " abort: "       %iu,b76 \
     " lost_frame: " %iu,b80 \
```

```
      " congestion: " %iu,b84 \
      " frame_copied: " %iu,b88 \
      " freq: "       %iu,b92 \
      " token: "      %iu,b96 \
      " dma_bus: "    %iu,b100 \
      " dma_parity: " %iu,b104

-- QFESTAT
-- NOTE:
-- for the fast-ethernet interfaces on the FFP2 Sun processors
-- on the DSGW and on the EDIS box.
-- note: 28,29 are hardware ids as defined
--       in fdcom/fdcom_exports/wshwd00s.ada which based on
--       the ids for "Fe1_Lan" and "Fe2_Lan"
--       in u3.1d3 baseline.
-- BS 10603 begins with a wshwd.hw_item_t object
-- which has a 4-byte hw_type field, then 4-byte hw_item.
-- This is followed by the device-specific data.
byte_format QFESTAT 10603 \
 %if    b3.0.8 = 28  %macro RECHDRA,b0 "fe1_lan " %macro QFESTATM,b8 \
                \n %macro RECHDRA,b0 \
 %elsif b3.0.8 = 29  %macro RECHDRA,b0 "fe2_lan " %macro QFESTATM,b8 \
                \n %macro RECHDRA,b0 \
 %endif

filter QFESTAT BS

-- worker macro for QFESTATM to help print 64-bit values
-- Print "{0 yyy}" or "{xxx<<32 yyy}"
byte_format QFEST64M macro_only \
      "{" %iu,b0 \
      %if b0.0.32 /= 0 "<<32" %endif \
      " " %iu b4 "}"

-- see psvc/psvc_if/includes/src/wadev.h:xqfestats_t
-- for QFE... remember that int64's will be 8-byte aligned, so there are some gaps
byte_format QFESTATM macro_only \
```

```
    "link_up: "       %iu,b288 \
    " nocarrier: "    %iu,b84 \
    " collisions: "   %iu,b16 \
    " missed: "       %iu,b68 \
    " inits: "        %iu,b88 \
    " phy_inits: "    %iu,b208 \
    " norcvbuf: "     %iu,b200 \
    " nocanput: "     %iu,b92 \
    " noxmtbuf: "     %iu,b204 \
\n %macro RECHDRA,b0 \
"in:  ipackets: "            %iu,b0 \
    " rbytes: "              %iu,b176 \
    " ierrors: "             %iu,b4 \
    " multircv: "            %iu,b184 \
    " brdcstrcv: "           %iu,b192 \
\n %macro RECHDRA,b0 \
"out: opackets: "            %iu,b8 \
    " obytes: "              %iu,b180 \
    " oerrors: "             %iu,b12 \
    " multixmt: "            %iu,b188 \
    " brdcstxmt: "           %iu,b196 \
\n %macro RECHDRA,b0 \
"in+:  rx_late_error: "      %iu,b120 \
    " rx_parity_error: "     %iu,b132 \
    " rx_error_ack: "        %iu,b148 \
    " rx_tag_error: "        %iu,b152 \
    " no_rbufs: "            %iu,b168 \
    " rx_late_collisions: "  %iu,b172 \
    " rx_inits: "            %iu,b216 \
\n %macro RECHDRA,b0 \
"out+: tx_late_error: "      %iu,b116 \
    " tx_parity_error: "     %iu,b128 \
    " tx_error_ack: "        %iu,b140 \
    " tx_tag_error: "        %iu,b148 \
    " no_tbufs: "            %iu,b164 \
    " tx_late_collisions: "  %iu,b72 \
    " tx_inits: "            %iu,b212 \
```

```
\n %macro RECHDRA,b0 \
"perf64: ipackets64: " %macro QFEST64M b224 \
    " rbytes64: "      %macro QFEST64M b240 \
    " opackets64: "    %macro QFEST64M b232 \
    " obytes64: "      %macro QFEST64M b248 \
\n %macro RECHDRA,b0 \
"more1: ifspeed: " %macro QFEST64M,b48 \
    " defer: "     %iu,b20 \
    " framing: "   %iu,b24 \
    " crc: "       %iu,b28 \
    " sqe: "       %iu,b32 \
    " code_violations: "  %iu,b36 \
    " len_errors: "       %iu,b40 \
    " buff: "      %iu,b56 \
    " oflo: "      %iu,b60 \
    " uflo: "      %iu,b64 \
\n %macro RECHDRA,b0 \
"more2: allocbfail: " %iu,b96 \
    " runt: "         %iu,b100 \
    " jabber: "       %iu,b104 \
    " babble: "       %iu,b108 \
    " retry_error: "  %iu,b76 \
    " tmd_error: "    %iu,b112 \
    " slv_parity_error: "  %iu,b124 \
    " slv_error_ack: "  %iu,b136 \
    " eop_error: "    %iu,b156 \
    " no_tmds: "      %iu,b162 \
    " first_collisions: "    %iu,b80 \
    " ex_collisions: "  %iu,b272 \
\n %macro RECHDRA,b0 \
"more3: defer_xmts: " %iu,b268 \
    " align_errors: " %iu,b256 \
    " fcs_errors: "   %iu,b272 \
    " sqe_errors: "   %iu,b266 \
    " macxmt_errors: "  %iu,b276 \
    " carrier_errors: " %iu,b280 \
    " toolong_errors: " %iu,b284
```

```
format EBLA_UTL 108 stdhdr
 loop root.ADAPTER_DATA'RANGE
  if root.PROCESSOR_CONFIG = ESI
   if root.ADAPTER_DATA(X).HW_ITEM.HW_TYPE = EBL_A_ADAPTER
    "HW_TYPE= EBLA"
    " BYTES_IN= "
    field root.ADAPTER_DATA(X).PERF_DATA.NUM_BYTES_RECV
    " BYTES_OUT= "
    field root.ADAPTER_DATA(X).PERF_DATA.NUM_BYTES_SENT
   endif
  endif
 endloop
done

filter EBLA_UTL all

format EBLB_UTL 108 stdhdr
 loop root.ADAPTER_DATA'RANGE
  if root.PROCESSOR_CONFIG = ESI
   if root.ADAPTER_DATA(X).HW_ITEM.HW_TYPE = EBL_B_ADAPTER
    "HW_TYPE= EBLB"
    " BYTES_IN= "
    field root.ADAPTER_DATA(X).PERF_DATA.NUM_BYTES_RECV
    " BYTES_OUT= "
    field root.ADAPTER_DATA(X).PERF_DATA.NUM_BYTES_SENT
   endif
  endif
 endloop
done

filter EBLB_UTL all

format SHR_AAR 10053 no_stdhdr
  field rechdr.PROCESSOR_TYPE
  " "
  macro AAR_ERR root
```

```
  done

format SHR_WEER 10098 no_stdhdr
  field rechdr.PROCESSOR_TYPE
  " "
  macro WEER root
  done

format SHR_WEER 10099 no_stdhdr
  field rechdr.PROCESSOR_TYPE
  " "
  macro WEER root
  done

filter SHR_AAR all
filter SHR_WEER all
```

B.2 shr_threshhold.config

Detected processor failures: 0

Address space failures: 0

Risc Hardware failures: 0

Network Hardware failures: 0

WEER errors: 1000

CPU Util: 0

Appendix C Netview Configuration Items

C. 1 OVW

```
Application 'Tivoli NetView Windows'
{
   Description {
      "Main User Interface",
      "Submap Windows and Function Menus"
   }

   Version 'V6R0';

   Copyright {
    "Licensed Program Product:",
    "  Tivoli NetView",
    "(C) COPYRIGHT International Business Machines Corp. 1992,1994",
    "(C) COPYRIGHT Hewlett-Packard Co. 1992",
    "    All Rights Reserved",
    "US Government Users Restricted Rights - Use, duplication,",
    "or disclosure restricted by GSA ADP Schedule Contract with",
    "IBM Corp. and its licensors",
    ""
   }

   HelpDirectory 'OVW';

   /*
   **
   ** OVw Menu Bars
   **
   */

   /*
   **  File
   */
```

```
MenuBar <100> "File" _F
{
   <100>   "New Map..."            _N      f.new_map;
   <100>   "Open Map..."           _O      f.avail_maps;
   <100>   "Describe Map..."       _M      f.map_desc;
   <100>   "Refresh Map"           _R      f.refresh_map;
   <100>   "Save Map As..."        _A      f.save_map;
   <100>   "Delete Map..."         _D      f.avail_maps;
   <100>   "Map Snapshot"          _h      f.menu 'Map Snapshot';
   <0>     "Exit"                  _E      Ctrl<Key>E      f.exit;
}

Menu 'Map Snapshot'
{
   <100>   "Open..."               _O      f.avail_snaps;
   <100>   "Create..."             _r      f.create_snap;
   <100>   "Delete..."             _D      f.avail_snaps;
   <100>   "Describe..."           _s      f.snap_desc;
   <100>   "Close"                 _C      f.close_snapshot;
}


/*
**  Edit
*/

MenuBar <100> "Edit" _E
{
   <100>   "Add"                   _A      f.menu 'Add';
   <100>   "Create Submap..."              _r      f.new_submap;
   <100>   "Cut"                   _t      f.menu 'Cut';
   <100>   "Copy"                  _C      f.menu 'Copy';
   <100>   "Paste"                 _P      Ctrl<Key>P      f.paste;
   <100>   "Modify/Describe"               _M      f.menu 'Modify Description';
   <100>   "Hide Objects"                  _H      f.menu 'Hide';
   <100>   "Hidden Objects List..."        _L      f.list_hidden;
```

```
    <100>   "Show Hidden Objects"           _S      f.menu 'Show Hidden';
    <100>   "Delete Object"                 _D      f.menu 'Delete';
    <100>   "Delete Submap"                 _e      f.menu 'Delete Submap';
    <100>   "Select Background Picture..."  _B      f.submap_desc;
    <100>   "Show/Hide Labels"              _s      f.menu 'Label Menu';
}

Menu 'Delete Submap'
{
    <100>   "This Submap"           _T      f.delete_smap;
    <100>   "Any Submap..."         _A      f.list_submaps;
}

Menu 'Modify Description'
{
    <100>   "Object..."             _O Ctrl<Key>O  f.obj_desc;
    <100>   "Submap..."             _S      f.submap_desc;
    <100>   "Map..."                _M      f.map_desc;
    <100>   "Snapshot..."           _h      f.snap_desc;
}
Menu 'Add'
{
    <100>   "Object..."             _O      f.add_obj;
    <100>   "Connection..."         _C      f.add_conn;
}

Menu Copy
{
    <100>   "From This Submap"      _T Ctrl<Key>C  f.copy_obj_smap;
    <100>   "From All Submaps"      _A      f.copy_obj;
}

Menu Cut
{
    <100>   "From This Submap"      _T      f.cut_obj_smap;
    <100>   "From All Submaps"      _A      f.cut_obj;
}
```

```
Menu Delete
{
   <100>   "From This Submap"      _T      f.delete_obj_smap;
   <100>   "From All Submaps"      _A      f.delete_obj;
}

Menu Hide
{
   <100>   "From This Submap"      _T      f.hide_obj_smap;
   <100>   "From All Submaps"      _A      f.hide_obj;
}

Menu 'Show Hidden'
{
   <100>   "For This Submap"       _T      f.unhide;
   <100>   "For All Submaps"       _A      f.unhide_all;
}

Menu 'Label Menu'
{
   <100>   "For This Submap"      _T      f.submap_label;
   <100>   "For All Submaps"      _A      f.map_label;
}

/*
**  View
*/

MenuBar <100> "View" _V
{
   <100>   "Highlights"            _H      f.menu 'Highlights';
   <100>   "User Plane"            _U      f.menu 'User Plane';
   <100>   "Automatic Layout"      _A      f.menu 'Automatic Layout';
   <100>   "Redo Layout"           _R      f.redolayout;
   <100>   "Open Submap..."        _O      f.list_submaps;
   <100>   "Filter By Status"      _F      f.filter_submap;
```

```
    <100>   "Sort Submap By"        _B      f.sort_submap;
}

Menu 'User Plane'
{
    <100>   "For This Submap"       _T      f.submap_user_plane;
    <100>   "For All Submaps"       _A      f.map_user_plane;
}

Menu 'Automatic Layout'
{
    <100>   "For This Submap"       _T      f.submap_layout;
    <100>   "For All Submaps"       _A      f.map_layout;
}

Menu Highlights
{
    <100>   "Select Highlights"     _S Ctrl<Key>H    f.select_highlighted;
    <100>   "Undo Highlights"       _U      f.clear_highlights;
}



/*
**  Locate
*/

MenuBar <100> "Locate" _L
{
    <100>   "Selected Objects List..."      _L Ctrl<Key>L   f.sel_objs;
    <100>   "Objects"                       _O      f.menu 'Objects';
    <100>   "Submap..."                     _S Ctrl<Key>S   f.list_submaps;
}

Menu Objects
{
    <100>   "By Selection Name..."  _N Ctrl<Key>N   f.locate_name;
    <100>   "By Attribute..."       _A Ctrl<Key>A   f.locate_attr;
```

```
    <100>   "By Comment..."         _C      f.locate_comment;
    <100>   "By Symbol Status..."   _S      f.locate_status;
    <100>   "By Symbol Type..."     _T      f.locate_type;
    <100>   "By Symbol Label..."    _L      f.locate_label;
}

/*
**  Options
*/

MenuBar <100> "Options" _p
{
    <100>   "Manage Objects"        _M      f.manage_objects;
    <100>   "Unmanage Objects"      _U      f.unmanage_objects;
    <100>   "Acknowledge"           _A      f.acknowledge;
    <100>   "Unacknowledge"         _n      f.unacknowledge;
    <50>    "Set Default Map..."    _D      f.avail_maps;
    <50>    "Set Home Submap..."    _H      f.sh_submap;
}

MenuBar <100> "Monitor" _M
{
}

MenuBar <100> "Test" _T
{
}

MenuBar <100> "Tools" _o
{
  <100> "Submap Explorer"    _E      f.action explorer;
  <79>    "Display Object Information..."  _O  f.action displayoi;
}

MenuBar <100> "Administer" _A
{
  <100> "Server Setup"       _e      f.action nv_setup;
```

```
}

Menu 'Atools'
 {
    <100> "HMC"                  f.action hmc_manager;
 }
          MenuBar <100> "EI2F_Administrative_Functions"      _I
          {
                    <100>     "Access the EI2F Administrative Web Site"  f.action admin_site;
                    <100>     "Access the Cisco Easy ACS TACACS Application"     f.action easy_acs;
          }
MenuBar <100> "DSR Test Processor Functions"    _T
{
          }
MenuBar <100> "Help" _H
{
    <100>  "Indexes"                  _I     f.menu 'Help Index';
    <100>  "Help Topics"               _H     f.product_help;
    <100>  "Books Online"               _B     f.action nv_library;
    <100>  "Using Help"                _U     f.on_help;
    <100>  "Legend"                    _L     f.disp_legend;
    <100>  "On Version"                _V     f.on_version;
}

Menu 'Help Index'
{
   <100>   "Applications"  _A     f.app_index;
   <100>   "Tasks"        _T     <Key>F1        f.task_index;
   <100>   "Functions"    _F     f.function_index;
}

/******* Popups ***/

Objectmenu
{
   <100>   "Edit"         _E            f.menu 'P_Edit';
   <100>   "View"         _V            f.menu 'P_View';
```

```
   <100>   "Options"       _p              f.menu 'P_Options';
   <100>   "Monitor"       _M              f.menu 'P_Monitor';
   <100>   "Test"          _T              f.menu 'P_Test';
   <100>   "Tools"         _o              f.menu 'P_Tools';
   <100>   "Administer"    _A              f.menu 'P_Administer';
}

/*
**  Edit Popup
*/

Menu 'P_Edit'
{
   <100>   "Description..."        _n      f.obj_attributes;
   <100>   "Cut"                   _t      f.menu 'P_Cut';
   <100>   "Copy"                  _C      f.menu 'P_Copy';
   <100>   "Hide"                  _H      f.menu 'P_Menu_Hide';
   <100>   "Delete"                _D      f.menu 'P_Menu_Delete';
   <100>   "Set Star Center"       _S      f.star_center;
   <100>   "Change Symbol Type..." _y      f.change_symbol;
   <100>   "Modify/Describe"       _M      f.menu 'P_Modify_Description';
}

Menu P_Cut
{
   <100>   "From This Submap"      _T      f.cut_obj_smap;
   <100>   "From All Submaps"      _A      f.cut_obj;
}

Menu P_Copy
{
   <100>   "From This Submap"      _T Ctrl<Key>C   f.copy_obj_smap;
   <100>   "From All Submaps"      _A              f.copy_obj;
}

Menu P_Menu_Hide
{
```

```
   <100>   "Symbol..."            _y              f.hide_symbol;
   <100>   "Object..."            _O Ctrl<Key>O   f.menu 'P_Hide';
}

Menu P_Hide
{
   <100>   "From This Submap"     _T      f.hide_obj_smap;
   <100>   "From All Submaps"     _A      f.hide_obj;
}

Menu P_Menu_Delete
{
   <100>   "Symbol..."            _y              f.delete_symbol;
   <100>   "Object..."            _O Ctrl<Key>O   f.menu 'P_Delete';
}

Menu P_Delete
{
   <100>   "From This Submap"     _T      f.delete_obj_smap;
   <100>   "From All Submaps"     _A      f.delete_obj;
}

Menu P_Modify_Description
{
   <100>   "Symbol..."            _y              f.desc_symbol;
   <100>   "Object..."            _O Ctrl<Key>O   f.desc_object;
}

/*
**  View Popup
*/

Menu P_View
{
   <100>   "Open"                 _O      f.open_symbol;
}
```

```
   /*
   **  Options Popup
   */

   Menu 'P_Options'
   {
      <100>   "Manage Object"        _M      f.manage_objects;
      <100>   "Unmanage Object"      _U      f.unmanage_objects;
      <100>   "Acknowledge"          _A      f.acknowledge;
      <100>   "Unacknowledge"        _n      f.unacknowledge;
   }

   Menu 'P_Monitor'
   {
   }

   Menu 'P_Test'
   {
   }

   Menu 'P_Tools'
   {
     <79> "Display Object Information..."  _O  f.action displayoi;
   }

   Menu 'P_Administer'
   {
   }

/*******/
   /* action for Submap Explorer */
   Action explorer {
      Command  '/usr/OV/jre/bin/jre -mx64M -cp
/usr/OV/jars/i18n.jar:/usr/OV/jars/submapexplorer.jar:/usr/OV/jre/lib/swing.jar:/usr/OV/jars/motif.jar:/usr/OV/www/htdocs/netview.j
ar com.tivoli.netview.maptree.client.submapexplorer.SubmapExplorer';
   }

   Action displayoi    {
```

```
    MinSelected  1;
    MaxSelected  1;
    Command ' \
        /usr/OV/bin/xnmappmon \
        -commandTitle \" Object Information \" \
        -geometry 900x600          \
        -cmd /usr/OV/bin/ovobjprint -s $OVwSelection1';
  }
  Action admin_site    {
    Command ' \
        /usr/dt/bin/netscape \
        http://172.26.155.74/admin_site/index.htm';
  }
  Action easy_acs    {
    Command ' \
        /usr/dt/bin/netscape \
        http://192.168.27.1:2002';
  }
}
```

C.2 Test_Proc_OVW

```
 Application 'test_processor setup stuff'
{
  Description {
  }

        Version 'V6R0';

  Copyright {
    "Licensed Program Product:",
    "  Tivoli NetView",
    "(C) COPYRIGHT International Business Machines Corp. 1992,1994",
    "    All Rights Reserved",
    "US Government Users Restricted Rights - Use, duplication,",
    "or disclosure restricted by GSA ADP Schedule Contract with",
    "IBM Corp. and its licensors.",
    ""
```

```
}

        MenuBar <100> "DSR Test Processor Functions"        _T

        {

                <100>     "set_test_procs"
        f.action set_test_procs;

                <100>     "unset_test_procs"
        f.action unset_test_procs;


                <100>     "Show Current Release for DSR Test Processors"        f.action show_rel;

                <100>     "Hide Current Release for DSR Test Processors"        f.action hide_rel;

                <100>     "Show CHECKPROC report for DSR Test Processors"  f.action checkproc;

        }

        Action set_test_procs {
   Command ' \
        /usr/OV/bin/xnmappmon \
        -commandTitle \" set_test_procs \" \
        -geometry 900x600                \
        -cmd /utilities/bin/set_test_procs $OVwSelections';

        }

        Action unset_test_procs             {
   Command ' \
        /usr/OV/bin/xnmappmon \
        -commandTitle \" set_test_procs \" \
        -geometry 900x600                \
        -cmd /utilities/bin/unset_test_procs $OVwSelections';

        }

        Action show_rel       {
   Command ' \
                                        /utilities/bin/show_hide_current_release show';

        }

        Action hide_rel       {
   Command ' \
                                        /utilities/bin/show_hide_current_release hide';

        }

        Action checkproc      {
   Command ' \
        /usr/OV/bin/xnmappmon \
```

```
            -commandTitle \" checkproc \" \
            -geometry 900x600                   \
            -cmd /utilities/bin/snmp_checkproc.report $OVwSelections';
             }

}
```

C.3 Poll_Test_Processors

```
/* Registration for Test Processor Status

  Date 05/17/2005 - Souder
*/
 Application "Poll_Test_Procs" {

 Command -Initial -Shared "${Poll_Test_Procs:-/utilities/bin/poll_test_procs}";
}
```

C.4 Artcc_health_check (excerpt)

```
Application 'artcc health check setup stuff'
{
   Description {
   }

              Version 'V6R0';

   Copyright {
    "Licensed Program Product:",
    "  Tivoli NetView",
    "(C) COPYRIGHT International Business Machines Corp. 1992,1994",
    "    All Rights Reserved",
    "US Government Users Restricted Rights - Use, duplication,",
    "or disclosure restricted by GSA ADP Schedule Contract with",
    "IBM Corp. and its licensors.",
    ""
   }

            MenuBar <100> "ARTCC Health Check Reports"        _H
```

```
        {
                <100>   "ZAB Albuquerque  "                     f.menu   ZAB;
                <100>   "ZAU Chicago      "                     f.menu   ZAU;
                <100>   "ZBW Boston       "                     f.menu   ZBW;
                <100>   "ZDC Washington   "                     f.menu   ZDC;
                <100>   "ZDV Denver       "                     f.menu   ZDV;
                <100>   "ZFW Fort Worth   "                     f.menu   ZFW;
                <100>   "ZHU Houston      "                     f.menu   ZHU;
                <100>   "ZID Indianapolis "                     f.menu   ZID;
                <100>   "ZJX Jacksonville "                     f.menu   ZJX;
                <100>   "ZKC Kansas City  "                     f.menu   ZKC;
                <100>   "ZLA Los Angeles  "                     f.menu   ZLA;
                <100>   "ZLC Salt Lake    "                     f.menu   ZLC;
                <100>   "ZMA Miami        "                     f.menu   ZMA;
                <100>   "ZME Memphis      "                     f.menu   ZME;
                <100>   "ZMP Minneapolis  "                     f.menu   ZMP;
                <100>   "ZNY New York     "                     f.menu   ZNY;
                <100>   "ZOA Oakland      "                     f.menu   ZOA;
                <100>   "ZOB Cleveland    "                     f.menu   ZOB;
                <100>   "ZSE Seattle      "                     f.menu   ZSE;
                <100>   "ZTL Atlanta      "                     f.menu   ZTL;
                <100>   "ZCY WJH FAATC    "                         f.menu   ZCY;
                <100>   "ZIF WJH FAATC IIF"                         f.menu   ZIF;
        }

Menu   ZAB {
                "Get Health Reports"  _G f.action GET_ZAB;
                "View Reports"                          _V f.menu   View_ZAB;
        }

Menu   ZAU {
                "Get Health Reports"  _G f.action GET_ZAU;
                "View Reports"                          _V f.menu   View_ZAU;
        }

Menu   ZBW {
                "Get Health Reports"  _G f.action GET_ZBW;
```

```
                "View Reports"                          _V f.menu   View_ZBW;
        }

Menu   ZDC {
                "Get Health Reports"  _G f.action GET_ZDC;
                "View Reports"                          _V f.menu   View_ZDC;
        }

Menu   ZDV {
                "Get Health Reports"  _G f.action GET_ZDV;
                "View Reports"                          _V F.menu   View_ZDV;
        }

Menu   ZFW {
                "Get Health Reports"  _G f.action GET_ZFW;
                "View Reports"                          _V F.menu   View_ZFW;
        }

Menu   ZHU {
                "Get Health Reports"  _G f.action GET_ZHU;
                "View Reports"                          _V F.menu   View_ZHU;
        }

Menu   ZID {
                "Get Health Reports"  _G f.action GET_ZID;
                "View Reports"                          _V F.menu   View_ZID;
        }

Menu   ZJX {
                "Get Health Reports"  _G f.action GET_ZJX;
                "View Reports"                          _V F.menu   View_ZJX;
        }

Menu   ZKC {
                "Get Health Reports"  _G f.action GET_ZKC;
                "View Reports"                          _V F.menu   View_ZKC;
        }
```

```
Menu   ZLA {
                "Get Health Reports"  _G f.action GET_ZLA;
                "View Reports"                  _V F.menu   View_ZLA;
        }

Menu   ZLC {
                "Get Health Reports"  _G f.action GET_ZLC;
                "View Reports"                  _V F.menu   View_ZLC;
        }

Menu   ZMA {
                "Get Health Reports"  _G f.action GET_ZMA;
                "View Reports"                  _V F.menu   View_ZMA;
        }

Menu   ZME {
                "Get Health Reports"  _G f.action GET_ZME;
                "View Reports"                  _V F.menu   View_ZME;
        }

Menu   ZMP {
                "Get Health Reports"  _G f.action GET_ZMP;
                "View Reports"                  _V F.menu   View_ZMP;
        }

Menu   ZNY {
                "Get Health Reports"  _G f.action GET_ZNY;
                "View Reports"                  _V F.menu   View_ZNY;
        }

Menu   ZOA {
                "Get Health Reports"  _G f.action GET_ZOA;
                "View Reports"                  _V F.menu   View_ZOA;
        }

Menu   ZOB {
```

```
                "Get Health Reports"  _G f.action GET_ZOB;
                "View Reports"                          _V F.menu   View_ZOB;
        }

Menu   ZSE {
                "Get Health Reports"  _G f.action GET_ZSE;
                "View Reports"                          _V F.menu   View_ZSE;
        }

Menu   ZTL {
                "Get Health Reports"  _G f.action GET_ZTL;
                "View Reports"                          _V F.menu   View_ZTL;
        }

Menu   ZCY {
                "Get Health Reports"  _G f.action GET_ZCY;
                "View Reports"                          _V F.menu   View_ZCY;
        }

Menu   ZIF {
                "Get Health Reports"  _G f.action GET_ZIF;
                "View Reports"                          _V F.menu   View_ZIF;
        }

        Menu   View_ZAB {
                <23>    "0000"                  f.action View_ZAB_0000;
                <22>    "0100"                  f.action View_ZAB_0100;
                <21>    "0200"                  f.action View_ZAB_0200;
                <20>    "0300"                  f.action View_ZAB_0300;
                <19>    "0400"                  f.action View_ZAB_0400;
                <18>    "0500"                  f.action View_ZAB_0500;
                <17>    "0600"                  f.action View_ZAB_0600;
                <16>    "0700"                  f.action View_ZAB_0700;
                <15>    "0800"                  f.action View_ZAB_0800;
                <14>    "0900"                  f.action View_ZAB_0900;
                <13>    "1000"                  f.action View_ZAB_1000;
                <12>    "1100"                  f.action View_ZAB_1100;
```

```
        <11>    "1200"          f.action View_ZAB_1200;
        <10>    "1300"          f.action View_ZAB_1300;
        <09>    "1400"          f.action View_ZAB_1400;
        <08>    "1500"          f.action View_ZAB_1500;
        <07>    "1600"          f.action View_ZAB_1600;
        <06>    "1700"          f.action View_ZAB_1700;
        <05>    "1800"          f.action View_ZAB_1800;
        <04>    "1900"          f.action View_ZAB_1900;
        <03>    "2000"          f.action View_ZAB_2000;
        <02>    "2100"          f.action View_ZAB_2100;
        <01>    "2200"          f.action View_ZAB_2200;
        <00>    "2300"          f.action View_ZAB_2300;
}
```

Appendix D ERAM Metrics - CMS Host Patch Supplement

The ERAM Automation Metrics Development and Validation activity is to support the developmental and operational testing of ERAM by developing a set of metrics that quantify the effectiveness of key system functions in ERAM. In addition, the metrics are designed to measure the performance of legacy En Route automation systems in operation today. When appropriate, they will allow comparison of similar functionality in ERAM to legacy systems.

The initial method of collecting simulation generated, recorded output data is by the use of the legacy Systems Analysis Recording (SAR) subsystem. This subsystem records various data types that represent raw input data, intermediate processed data and final processed products of the air traffic control subsystems of the En Route National Airspace System (NAS). The use of this subsystem requires the selection of and enabling of the appropriate recorded data type category and codes. The resultant output is transferred to IBM model 3480 cartridge tapes, which become the data input source for the off-line processing utility of this data, the Data Analysis and Reduction Tool (DART). The output of DART consists of many possible report types that represent the recorded data according to selected DART processing options. For the purposes of this task, electronic copies of selected DART reports were subjected to additional processing to extract the information necessary for the metrics tasks processing tools.

To optimize the data recording and extraction process, the IIF staffs have integrated an alternative to the SAR/DART subsystem and utilities. This alternative is a Host Computer System (HCS) network stream called the Common Message Set (CMS) that provides a subset of NAS processing information via a predefined set of CMS messages. These messages are exported to various network clients via the Host Interface Device (HID).
The IIF staff has integrated a HID client specifically for this task to expedite the recorded NAS data collection step, and the forwarding of that data to the metrics processing tools step. The client receives the appropriate CMS messages and collects them in a file that the metrics processing tools can use directly. This CMS client alternative eliminates the time consuming legacy DART report generation and DART report parsing dependency, and takes advantage of the evolving NAS communications infrastructure.

A subtask of the metrics activity is entitled Host Radar Tracking Simulation and Performance Analysis. Various types of recorded NAS data are selected to conduct the analysis within this task and are received via the CMS client. However, this task requires additional data that is not represented in the CMS message set. The additional required data for this task is the set of Conflict Alert (CA) messages created by the NAS Conflict Detection subprogram (RCD). This problem was resolved by software patch that added

the Conflict Alert messages to the CMS stream. Subprogram RCD operates periodically and tests for conflict conditions. When the criteria are met that result in the output of Conflict Alert messages, the software patch in RCD imbeds a copy of each CA message into a CMS General Information (GH) message. The GH messages are then sent to the CMS client. The metrics processing tools have been modified to process imbedded messages in the GH messages.

This patch model can be extended to include any NAS data type into the CMS message stream. A sub-header that is pre-pended to imbedded data is used to uniquely identify the imbedded data type. Metrics processing tools can be quickly modified to accommodate any selected type of data.

Appendix E Simulation Time Synchronization Design Details

The Distributed System Testing and Maintenance Methodology Study is concerned with effective simulation processing amongst various NAS subsystems. For a successful simulation to occur the initial conditions of a set of parameters of the simulation environment must be set to specific values. Otherwise the simulation degrades into chaos.

Characteristic of the simulation input data, its injection into the simulation system and the synchrony of the various NAS subsystems, is sensitivity to time. All subsystems (NAS Host, SDR, SDRR, etc.) must experience the same time, within tolerance, to allow for the successful injection, synchronization and processing of the time-stamped simulation data. This tolerance varies with the composition of the simulation input data and is usually within the range of a couple of seconds to a sub second. If the tolerance is exceeded the simulation degrades and must be restarted, resulting in a waste of resources and possibly a loss of effective use of a Lab session.

A current means to synchronize the clocks of the subsystem components requires a visual observation of one subsystem's time, a vocal countdown and a manual entry of a start message to start the clock on another subsystem. This manual method is error prone and can result in numerous restarts due to exceeding the time tolerance for the simulation. The object subsystem of the manual start message is usually the NAS Host. A sometimes successful mitigating action to overcome a slightly out-of-tolerance time condition is to manually enter a sequence of NAS Set System Time Update Interval (SETT) commands to manipulate the passage of NAS Host time. Using this method the passage of time on the NAS host can be slowed down or sped up to allow synchronization to occur between the subsystems' displayed time.

A means to more accurately time synchronize the HOST/DSR subsystems with the SDRR subsystem for enhanced simulation fidelity is under research and uses an automated approach. This enhanced time synchronization process will eliminate the anomalous performance of the manual simulation start method. This automated method uses the JVN Token Ring sniffer to detect a DSR "time" System Request (SR) (SR# 1740) and forwards the information to the SDRR subsystem as the basis for a time adjustment. The JVN sniffer taps the DSR ring set and reads all the SRs that appear on the rings. The time SRs are identified and the time value is processed and forwarded to the SDRR subsystem for synchronization. This SR is sent during subsystem initialization and whenever a time drift is detected between the Host and DSR subsystems. This characteristic of the SR allows for time drift correction in the SDRR subsystem as well; this is a condition that is not currently corrected in simulation.