Monitoring an Integration Broker: A Case Study.

Bill Martorano
Hewlett Packard Company
100 Mayfield Avenue MS 36U3
Mountain View, California 94043

(650) 691-7099
(916) 785-9278 fax

bill_martorano@hp.com

HP World Conference
August 2001

Contents

1 Abstract

As Enterprise Application Integration (eAI) solutions begin to emerge as a predominant integration "backplane" solution in support of a "Services Oriented Architecture" they bring with them new challenges for software monitoring and high availability. In a mission critical environment, middleware forms the "glue" which integrates multiple applications and composite services together. Often this environment is enterprise-wide and has many distributed components. Middleware solutions may also consist of heterogeneous technologies, which must perform together even though they were not originally designed for this purpose.

Middleware based solutions may be purposely designed to be loosely coupled or latent (for example, publish/subscribe message-oriented solutions), or they may demand the highest levels of performance and availability in support of zero latent transactions (for example, web-portals). These solutions may co-exist, and rely on the same integration broker environment.

Business units often demand service-level guarantees for solutions constructed on top of a middleware environment. These service levels often define goals for end-to-end availability and response time, which are quite challenging for Information Technology organizations to deliver. Lack of availability and performance can cripple a business process, or endanger a customer relationship, especially as customers are becoming increasingly exposed to "internal" applications and business processes via the web.

The design for availability and performance in such a complex environment cannot be "bought off the shelf". Instead, its architecture must be defined up-front and integrally implemented into the integration broker (or middleware) architecture.

A case study of HP's Enterprise Information Architecture (EIA) will describe the approach taken to ensure end-to-end monitoring and high availability for critical components of one specific integration broker architecture. The study documents use of the Hewlett Packard software tools HP OpenView and HP ServiceGuard, for this purpose. These tools were used extensively to monitor and operationally maintain commercial off the shelf (COTS) software components, such as BEA/eLink, Tuxedo and CORBA, as well as custom-built software modules.

2 Business Case

The predominant business case for enabling a monitoring and high availability environment for the Enterprise Information Architecture (EIA) program, is to *enable a* ***stable, reliable and high performance middleware*** environment, for solutions which rely on this architecture as a "services backplane".

A primary focus for EIA is to provide a single point of interface and access to "back office" systems. As such reliability and performance are of paramount concern. If EIA were to become a "bottleneck" for "front office" solutions, the value proposition of using such middleware would be significantly reduced.

2.1 Support Model

A consistent monitoring architecture for support was required. Consistency in this sense refers to the commonality of support tools used to enable the support infrastructure across all of Hewlett Packard Information Technology. HP's global infrastructure is deployed worldwide and consists of multiple comprehensive applications and services that enable the enterprise; of which the Enterprise Information Architecture may be viewed (from a support perspective) as merely yet another component. *In this environment, fixed support technologies and tools were mandated*. It was not possible to install or deploy specific tools to fit the application, rather *the tools defined the monitoring solution* ***for*** *the application (*in this case the Enterprise Information Architecture*)*.

Without such a mandated model for support, each application could select customized monitoring tools. The results, when replicated company wide would extrapolate to an inconsistent and replicate monitoring environment and infrastructure; one which would be difficult if not impossible to maintain and support. Instead, a pre-defined suite of tools was specified, to be "built into application architectures " during design. A primary tool mandated in this suite was *HP OpenView IT Operations Console*, which is used by applications support as a primary user interface and single collection point for support personnel to monitor a multitude of regionally deployed software solutions.

2.2 High Availability Model

The EIA design model utilized three primary programming environments.

1. **BEA Tuxedo:** Used for asynchronous or loosely coupled message-oriented-middleware (MOM) solutions.

2. **CORBA (Common Request Broker Architecture):** Used for synchronous or tightly coupled message-oriented solutions.

3. **HP Process Manager:** Used to define and control business process automation. EIA uses HP Process Manager to direct messages from producer to consumer.

Within these environments, independent goals for high availability were described.

- 99.9% availability for the **synchronous** environment (CORBA).
- 85.0% availability for the **asynchronous** environment (Tuxedo, HP Process Manager).

These separate goals were in recognition that loosely coupled solutions require less availability. Tightly coupled solutions on the other hand, are primarily required for transactional solutions, i.e. solutions which interact with live customers.

3 Architectural Approach

Given the constraints of the support model and choice of monitoring tools, the architectural approach taken was to design the required monitoring and high-availability architecture from a distributed deployment perspective; then specify points of integration with the mandated monitoring tools. The challenge was to identify the primary solution components (as not all components of a complex software system can or should be monitored), then build a plan to integrate them into the mandated monitoring tools, using the following technologies:

- **SNMP : Simple Network Management Protocol** (an industry standard technology)

- **ARM : Application Response Measurement** (an API used to feed the HP MeasureWare product).

These technologies were then rationalized against the pre-defined list of approved monitoring products.

- **HP OpenView:** A product family consisting of management products dedicated to Application, Availability, Network, Performance, Service, Systems and Storage management. HP OpenView is a "framework" into which many applications may be built for an integrated network and system management solution.

- **HP OpenView IT Operations Console:** A graphical user interface which consolidates and displays the status of monitored components. HP IT uses the ITO Console as “single point of reference” mechanism.

- **HP (MC) Service Guard:** A software package that enables multiple computers to be defined into a highly available cluster. Nodes, networks and processes configured into these clusters are able to be monitored and automatically stopped, started or moved to a different “hot backup” node.

- **HP MeasureWare:** A resource and performance management collector. The MeasureWare architecture has an open interface which allows the collection of data from many sources, including:

  1. Application Response Measurement (ARM) a.k.a. Transaction Tracker is part of MeasureWare. It is a set of APIs which, when implemented into the application, provides end-to-end response time of a particular transaction.

  2. SNMP MIB (Management Information Base). MIB’s express a formal description of a set of network objects that can be managed using the Simple Network Management Protocol (SNMP).

4 Implementation

Implementation was defined as part of the design and development job of each development engineer. It was recognized early on in the project that comprehensive monitoring would not be possible if individual components were not designed “up front” with the monitoring architecture in mind.

"Off the shelf" components, were integrated based on one or more of the identified primal technologies (MIB, ARM). BEA Tuxedo already defined a comprehensive MIB monitoring architecture, which was closely integrated with the internal "bulletin board" transaction monitoring environment. The challenge here was to identify which, of thousands of possible monitoring conditions were appropriate for the EIA environment.

Additionally, it was specified that performance metrics were to be gathered from the Tuxedo environment. Performance was characterized as a metric model defined around the number of messages which passed through the EIA architecture. This task proved to be quite difficult, as BEA Tuxedo does a great job of recording quantum metrics, but does a poor job of reporting granular transactions. After some effort a group of MIB's was identified, which in composite could be used to *interpret* the number of messages passed through the information bus. Refer to appendix D and E for further detail about this approach.

For internally developed components, the process of integration to the standardized support tools was equally difficult. In some cases, internal designs were modified to include writing specified metrics to the Application Response Measurement application programming interface (ARM). This was done mainly for the synchronous CORBA architecture. Specifically, timing metrics and counting metrics were applied to transactions to enable recording of the number of messages passed through the interface and the latency factor associated with each message. In this way, it was possible to graphically display (on the ITO Console) the relative performance of the synchronous architecture.

Deployment of the monitoring architecture was fortunately already accomplished. However, an evolution of the deployment architecture was underway which also required developers to design their solutions to integrate with a future state monitoring architecture and process flow.

Deployment of EIA components into the monitoring architecture was required. This task also proved difficult to accomplish due to the number and complexity of the individual components. Overall, a period of approximately three months was required from code freeze to final deployment of the monitoring architecture.

5 Conclusions

Monitoring, from the point of view of the software developer is a necessary evil. Although in retrospect the monitoring challenge for EIA was indeed accomplished, there were a number of lessons learned.

One of the primary lessons learned was to design “up front” for monitoring. Most developers on the team did not have familiarity with either the SNMP or ARM technologies. These skills had to be developed before the monitoring designs became truly effective. In the case of BEA components, a specialized consultant was hired to build the monitoring infrastructure. This proved to be a problem, as the consultant was well versed in BEA products and technologies, yet unfamiliar with Hewlett Packard technologies in the same space (i.e. BEA Manager, HP OpenView). Additionally, the architectural goals for EIA with regard to performance metrics were not intuitive to the BEA consultant.

High availability, although only tangentially related to monitoring, proved to factor significantly into the design and successful deployment of the monitoring architecture. One specific example of this was a mismatch between the BEA Tuxedo failover architecture and HP Service Guards failover architecture. Since both products can be used to implement a failover model, it was sometimes difficult to choose which design to deploy.

Implementation was successful however, and with it came a new respect for the difficulty of integrating software components into a common monitoring architecture. It is hoped that in the future greater degree of standardization will occur around “industry standard technologies” in the monitoring space. In that regard, the Java Management Extension (JMX) specification, is looked at favorably as a potential reconciliation of the multiple technology implementations in this space.

6 Next Challenges

As the EIA architecture evolves from current to visionary state, it is expected that many more technologies will be deployed. Each of these technologies expresses a slightly different native monitoring design. Normalizing these technologies to enable a cohesive enterprise support model will be a continual challenge.

Two functional drivers emerged from EIA's first experience in providing a monitoring infrastructure:

1. The architecture must improve to become more pro-active and less reactive. Currently, pro-activity is somewhat limited in scope and needs to be expanded. Erratic use models and transactional volumes make this capability imperative as the capacity of EIA is more completely consumed.

2. Integration of J2EE monitoring technologies will be required. As EIA begins to deploy and support the E-Service Application Server (HP Bluestone) environment it will be necessary to monitor the components developed in this space. Specifically the J2EE container environments will be examined to ensure that load balancing, performance and other capabilities can be integrated to report into the overall ITO Console domain.

The impact of a successful monitoring architecture most closely impacts support. The main consideration and challenge to the monitoring of "middleware" is due to the lack of traditional support models. Because consistency of this environment is required across multiple applications, services, components and technologies, it will continue to be a challenging development challenge for the Enterprise Information Architecture.

7 Additional Reference

A significant amount of specific reference detail follows in the appendix section of this document. The material is represented to be technically accurate, however due to the volatility to the implementation design in this space there may be some duplication, inconsistencies and inaccuracy with regard to content. The purpose for sharing data in such a raw fashion is to provide the reader with a more granular view of the EIA monitoring solution.

8 Appendix A – EIA High Availability Architecture Principles.

8.1 Introduction

This document describes the architectural approach and principles used to ensure a high degree of availability (HA) for the EIA Wave 2 solution. The program goal to enable Request/Reply capability into EIA during Wave 2 has been a driving factor for many of the requirements for high availability. Prior to this phase, the only solution pattern engaged by EIA was "loosely coupled" asynchronous, and therefore by nature highly latent. Synchronous binding, by contrast is a "tightly coupled" solution, which demands a greater degree of high availability requirement.

8.2 Approach

Architecture, in and of itself is only part of the total HA picture. As presented to the executive sponsor at lab commit, the casual analysis of unplanned downtime (the outcome of insufficient availability) is the product of three distinct component pieces of the total solution.

- 20 % Technology: Hardware, Operating Systems, Environmental factors, Disasters.
- 40 % Application Failure: Bugs, Performance Issues, Design Patterns.
- 40 % Operator / Human Error: Not performing a task. Performing a task incorrectly.

Of these areas of concern, the focus for this document will only be the Application Architecture, including choice of technologies, implementation decisions and approach taken to enable a highly available solution.

8.3 Major Technology Additions (Wave 2)

The criteria for application of technologies in solution design, encompasses both the strategic goals for the EIA program (e.g. platform for e-services) as well as general tactical goals of the SET organization (e.g. follow established development practice).

With regard to Wave 2, additional technologies were added to the existing Tuxedo, eLink base. These major technology elements are;

- CORBA (Common Object Request Broker Architecture)
- HP Process Manager (Business Process Manager)

8.4 Software/Application Components

Complimentary to these major technologies are software sub-components which directly enable these technologies. Examples here include the Routing Server (CORBA), HP Process Manager Integrator (bolt on to eLink), HP ORBPlus (HP Process Manager ORB) etc.

A certain number of these sub-components will be required to maintain operation in order to ensure that a HA capability is maintained. All of these processes are based on the HPUX 11.0 platform. A formal list of these processes will be included in another document.

By selecting a critical set of components to keep up and running, one should not infer that other processes not listed in the critical set are unimportant. Instead, the purpose for this categorization is to enable management and supportability. In the event of application or platform failure, these categorizations could be used to determine the relative priority of process recovery.

8.5 Application Monitoring

Participant applications, both producers and consumers are part of the total end-to-end solution enabled by EIA. However, as presented in EIA's high availability policy, maintaining availability for such applications (e.g. SAP) is outside of the scope of the EIA program.

A separate effort, sponsored by HP Information Technology Services (Polly Yap) is responsible for identifying critical HP internal applications and ensuring their capability to support a highly available environment.

8.6 Process Monitoring

It is the goal of the EIA program, to enable the automated monitoring of all processes, which are part of the EIA technology suite. These processes include, but are not limited to the Tuxedo, eLink, CORBA and HP Process Manager bundles.

HP ITS has selected the ITO Agent as the recommended tool for process monitoring. ITO Agent is part of the HP Openview suite of monitoring solutions. This agent executes on each HPUX platform and can be configured to examine critical processes, notify the ITO Openview console of exceptions and (potentially) re-start processes. The major focus of ITO Agent from Wave 1, which will continue in Wave 2 is the monitor and alarm when processes fail. For example, all Tuxedo processes have been "hooked" to ITO Agent, by directing ITO Agent to read and interpret the Tuxedo log file. The process of engaging ITO Agent to critical processes will continue for Wave 2.

8.7 Process Re-Start/Recovery

It is the goal of the EIA program, to enable the automated re-start and recovery of all processes, which are part of the EIA technology suite, which may fail for any number of reasons. These processes include, but are not limited to the Tuxedo, eLink, CORBA and HP Process Manager bundles.

Tuxedo has a built-in transaction monitoring technology, which is implemented by direct integration of the Tuxedo bulletin board. This capability has been enabled for all Tuxedo processes. Tuxedo offers a large configurable option set for the monitoring, scaling and re-starting of identified processes. This is implemented by static configuration (UBBCONFIG) and by dynamic modifications to this configuration (SNMP, BEA manager MIB's).

BEA manager, an snmp-based bolt-on to eLink is also used as a monitoring agent. BEA manager is used to monitor Tuxedo internals and report anomalies to the NodeManager component of Openview.

Additional technologies introduced into the EIA suite for Wave 2 will include HP Process Manager and CORBA. Both of these technologies are based on HPUX implementations. Therefore the plan for integrating these technologies into the existing recovery design will be to identify critical components, at the process level, then configure these processes to be re-started according to the capability of the existing (or potentially additional) tools. CORBA offers a robust set of discovery and invocation options. The selected technology for Wave 2 (the ORBacus Naming Service) will allow the logical abstraction of invocation instance (IOR) to be presented to clients who wish to exercise EIA synchronous functionality. This abstraction allows some degree of platform and process recovery to take place independently of consumer knowledge. The exact configuration of this capability demands further refinement.

8.8 Performance Monitoring

It is the goal of the EIA program, to enable the automated performance monitoring for all processes, which are part of the EIA technology suite. These processes include, but are not limited to the Tuxedo, eLink, CORBA and HP Process Manager bundles

HP ITS has recommended the use of Openview (Perfview) as the designated tool of choice for the monitoring of application performance. This tool has limited interface capabilities. Currently, an "pipe" interface is available, and this interface is fed performance data from the Tuxedo environment by means of selective ping and interpretation of performance MIB's (management information base). These performance MIB's are analyzed by time-slice, then reported to the Perfview monitor.

Extensions the Perfview interface are in plan. Specifically, the Application Resource Management (ARM) interface has recently been provided by HP Openview. Use of this interface will allow more dynamic performance data to be populated to Perfview and this mechanism is being built into selected Wave 2 components (i.e. CORBA gateway).

In addition to Perfview, classic tools, such as HP GlancePlus may be used to monitor performance. Although not optimal, these optional tools may be considered for selected processes which are not deemed critical to monitor in a more automated fashion.

HP Process Manager does not have a product integration with HP Openview. This deficiency will force some degree of development to integrate critical HP Process Manager processes into the Perfview monitor (probably by use of the ARM interface).

8.9 Software Component Deployment

The primary criteria for deployment of software components in EIA are as follows.

- Maintaining to the best degree possible, a standard deployed software and hardware configuration across all EIA instances (AP, EU, Americas).
- Producer/Consumer application location.
- Data center location.
- WAN infrastructure costs.
- Capability of the software components to support distributed deployment.
- Isolation from single-point-of-failure where possible.
- Business model logical profiles.
- Complexity of the deployed solution.

This is not a complete list, as there are many factors external to the EIA program which factor into the deployment decision. For example, the Enterprise Integration program to focus implementation of integration broker BPM's (Business Process Managers) to a centralized instance, consolidation of HP data centers etc. These external variables may be independent from architectural design and optimal deployment, from the EIA perspective.

8.10 Infrastructure

The HP Services Information Technology organization within ESSO is responsible for scaling and infrastructure deployment for EIA.

8.11 Wave 2 (Lab Comment) Recommendations to Executive Sponsor

After analysis of Wave 1, the following recommendations were made to the EIA executive sponsor at the Wave 2 lab commit meeting. The purpose of these recommendations was to solicit high level support for a cross-organizational effort to ensure end-to-end high availability of the entire business process, from application producer, through EIA to application or service consumer.

RECOMMENDATIONS:

1. Define an owner at a higher level for EIA high availability and supportability.

2. Ratify approach and explicit limitations specified by HA policy.

3. Create disaster recovery plan. Will require investment in all areas (infrastructure, architecture, people and processes).

4. Immediately procure a highly available platform for HP Process Manager.

5. Invest in GBIT basic-level skills, by providing more focused training.

6. Develop and/or enhance cross-application support models to support application downtime notification, data recovery, application performance monitoring.

7. Develop cross-application recovery support model.

9 Appendix B – BEA Manager Architecture

HP EIA Project
BEA Manager: Systems Architecture

9.1 Management Requirements

The EIA project has two primary requirements for any management solution. These are

1. Alerts/Traps generated on the managed node (e.g. by the BEA Manager snmp agent) should be propagated to Openview ITO.
2. It should be possible to issue gets and sets from Openview ITO that will be propagated and implemented by the SNMP agents running on the managed node
3. A predefined list of system performance metrics should be collected against each of the managed nodes and passed centrally to Measureware.

9.2 The BEA Manager Components

BEA Manager 2.0 is essentially an SNMP based product. It is made up of 4 components. These are:

1. Agent Connection	This is the BEA snmp agent that monitors the TUXEDO application (tux_snmpd). There should be an agent running on each machine that runs tuxedo. If there are a number of tuxedo applications (domains) running on the same machine – there should be a separate tux_snmp agent for each domain.
2. Agent Integrator	This integration product (snmp_integrator) allows you to run more than one snmp agent on the same snmp port. Agent integrator is also supplied with a BEA agent for monitoring the UNIX operating system (unix_snmpd). It also provides the facilities to monitor any MIB attribute against a threshold and issue a trap when this threshold is exceeded.
3. Agent Development Kit:	This provides a SDK that helps clients to implement snmp agents that can be used to monitor their own proprietary applications.
4. Log Central	This consolidates log entries into one centralized location (RDBMS). It can also be configured to issue snmp traps (or execute a shell script or program) when particular message types are received.

When designing a management solution for the EIA project it became apparent that the solution should be based on agent integrator (which can use SNMP to manage TUXEDO) and agent integrator (which can be used to run BEA and HP agents on the same snmp port).

HP did not intend to develop any proprietary snmp agents and so had no need for the BEA agent development kit.

BEA Manager Log Central provides several useful capabilities.

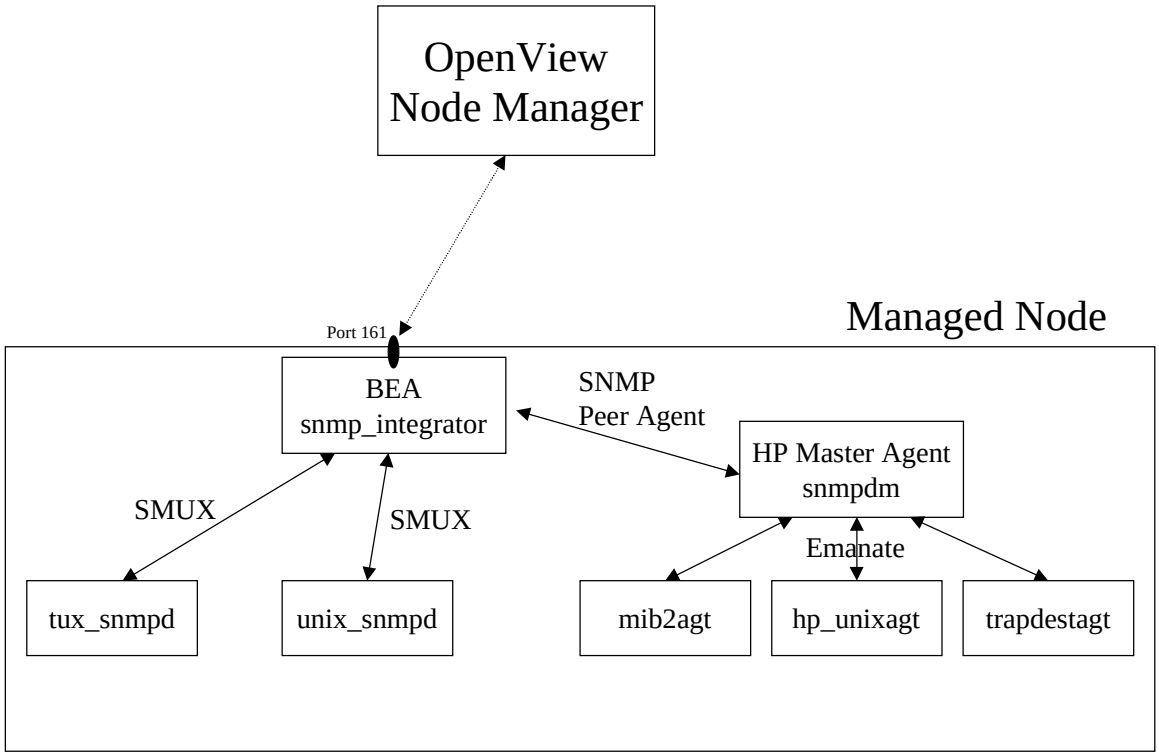
1. It allows logs from a number of disparate applications running on a number of platforms to be consolidated on one central data store (RDBMS).
2. Out of the box log central understand the format of TUXEDO and Oracle logs. It can be configured to understand the format of any other log file.
3. Log Central provides a WEB based GUI that can be used to query and view log entries.
4. Log Central can filter and only propagate and store certain log entry types
5. Log central can also be configured to generate either SNMP traps or execute a shell script or program) when particular message types are received. This allows us to extend the scope of the snmp traps raised by the TUXEDO application. Application developers could initiate ‘business ‘application traps by writing appropriate messages to the tuxedo userlogs.

Despite these advantages it was decided not to use Log Central for the HP EIA project. This decision was made because

1. Log Central requires either an Oracle (7.3.4 or 8.0.5) or MS SQLServer database. The EIA project team does not have access to either Oracle or SQLServer.
2. HP OpenView ITO already provides identical functionality to that of BEA Log Central. An ITO log collection agent can be installed on each managed node. This will read log files and propagate entries to the ITO central console, which would store these log messages in an Oracle database. ITO can be configured to generate alerts whenever a particular message type is received.

9.3 SNMP Agent Architecture

The diagram below represents the configuration of the snmp agents to be used on the HP EIA project



The management console (OpenView Network Node Manager) will communicate using SNMP with all of the snmp agents running on the managed node using the standard SNMP port (161). Traps will be generated by the various snmp agents on the managed node and propagate to port 162 on the machine running the OpenView Node Manager

This will be achieved by using the BEA agent integrator (snmp_integrator). The agent integrator will be configured to listen on port 161. In this configuration the snmp_integrator would be responsible for passing requests to the HP master agent (snmpdm) which would be configured as a peer SNMP agent. The HP master agent would then use EMANATE to communicate with the standard HP mib2agt, hp_unixagt and trapdestagt agents.

The agent integrator would also pass requests to the BEA tux_snmpd and unix_snmpd agents, which are configured as subagents of snmp_integrator using SMUX on port 199.

To implement this architecture the following must be done;

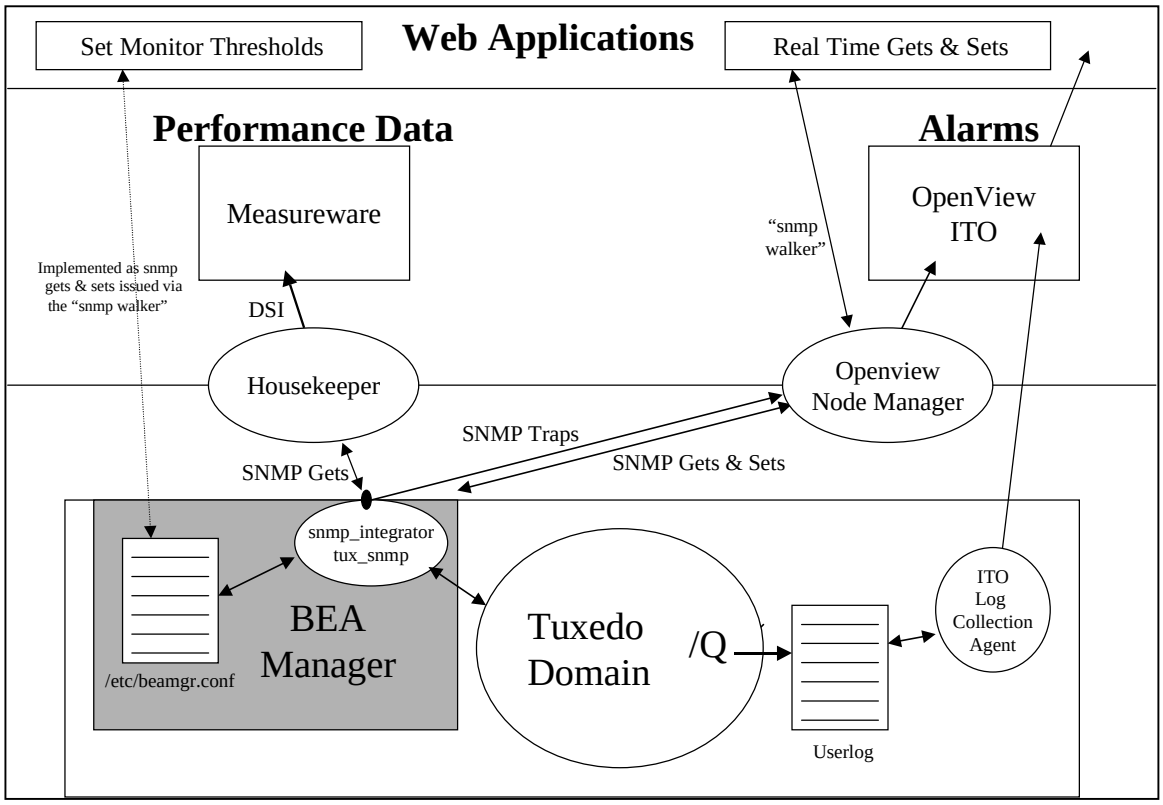
1. The HP agents (snmpdm, mib2agt, hp_unixagt, trapdestagt) should be already running on port 161(before the BE agents are started) and will be configured as a peer SNMP agent integrator by adding the following line to the /etc/beamgr.conf configuration file
   "NON_SMUX_PEER 161 * .1.3.6.1.4.1.11.2"
2. The BEA snmp_integrator, tux_snmpd and unix_snmpd agents can then be started (see Operational Procedures Document for details).

I should then be possible to issue both SNMP GETs and SETS to attributes in both HP and BEA MIB via port 161 from Openview Node Manager (ONM). Any traps generated by either the BEA agents or HP agents should result in the propagation of a SNMP trap to Openview Node Manager.

9.4 Management Architecture

This diagram below represents the management architecture that was developed for the HP EIA project.

It uses the snmp capabilities of BEA Manager 2.0 agent connect and agent integrator to achieve the Management requirements outlined in section 1.



9.5 Alerts

Alerts/Snmp traps will be generated by a number of components.

1. The BEA and HP snmp agents will generate a number of default SNMP traps. So for example the Tuxedo tux_snmpd agent will generate 32 standard traps (see the BEA Manager MIB review document for details).
2. The agent integrator (snmp_integrator) will be configured to generate a number of additional user defined traps. This will be done by the addition of a number of RULE_ACTIONS to the integrator /etc/beamgr.conf configuration file. These will instruct the agent to poll specified MIB attributes at a set frequency and generate a named trap if a threshold is exceeded.
3. Alerts will also be generated using the log collection facilities provided by Openview IPO

The snmp traps raised by the BEA & HP snmp agents and the BEA agent integrator will be passed to OpenView Node Manager. Dave Wilson will configure OpenView so that these traps are then passed to ITO.

An ITO log collection agent will run on each managed node. This agent will be incapable of generating SNMP traps or ITO alerts directly. Instead it will filter and pass certain userlog messages centrally to the ITO Oracle database. ITO will then be configured to raise alerts when particular message types are received **.**

9.6 Get and Sets

A web application will be implemented that will use the Openview Node Manager “snmp walker” facility to issue gets and sets against the BEA and HP agents.

This facility will also allow the user to set the frequency and thresholds for the MIB attributes monitored by agent_integrator and used to generate user defined snmp traps (RULE_ACTION – see above). This is done by issuing an SNMP set for the beaIntAgtRuleAction and beaIntAgtScanInvl attributes in beaIntAgt group of the BEA MIB.

9.7 Performance Metrics

Performance metrics will be gathered by a “housekeep” utility to be developed by Dave Wilson.
This utility will periodically issue an SNMP GET for a number of predefined attributes in the BEA and HP MIBs (see the document BEA Manager MIB Summary for details of the attributes to be monitored). This “housekeeper” utility will then use DSI to pass these values to Measureware.

10 Appendix C – BEA Manager High Availability Strategy.

HP EIA Project
BEA Manager: High Availability Strategy

10.1 Introduction

This document reviews the high availability solution implemented for the EIA project at HP. It also reviews the high availability strategy that should be used to ensure that the BEAM Manager agents are running even after failover occurs.

10.2 EIA HA Architecture

There has been a recent change to the HA architecture of the EIA project

The EIA architecture was always designed to use HP MC ServiceGuard to facilitate failover. However in the earlier design the TUXEDO domain (for each region) was configured to run in multiple machine mode. The EIA architecture has recently been updated and the TUXEDO domain is now configured to run in single machine mode

10.3 Multiple Machine Mode

In the original multiple machine mode most of the system components ran on the Master Machine. The Master machine held the application queues and transaction log on a mounted directory (/tuxq). Most of the TUXEDO servers ran on the Master machine. The backup machine ran a limited number of TUXEDO server s including a workstation listener (WLS), cr3fcin and the workflow agents (eg sapwfa).

10.3.1 Implications for BEA Manager

None. Even in a multiple machine configuration it is still only necessary to run the tux_snmpd agent on the Master machine. The /Q components are only installed on the Master machine. Furthermore all of aspects of the TUXEDO application on both the Master and also the Backup machines can be obtained by the tux_snmpd agent running on the master machine.

10.4 Single Machine Mode

In this architecture all components of the EIA application including the TUXEDO server and /Q run on the Master machine. The Backup machine is powered up and available as a hot standby machine.

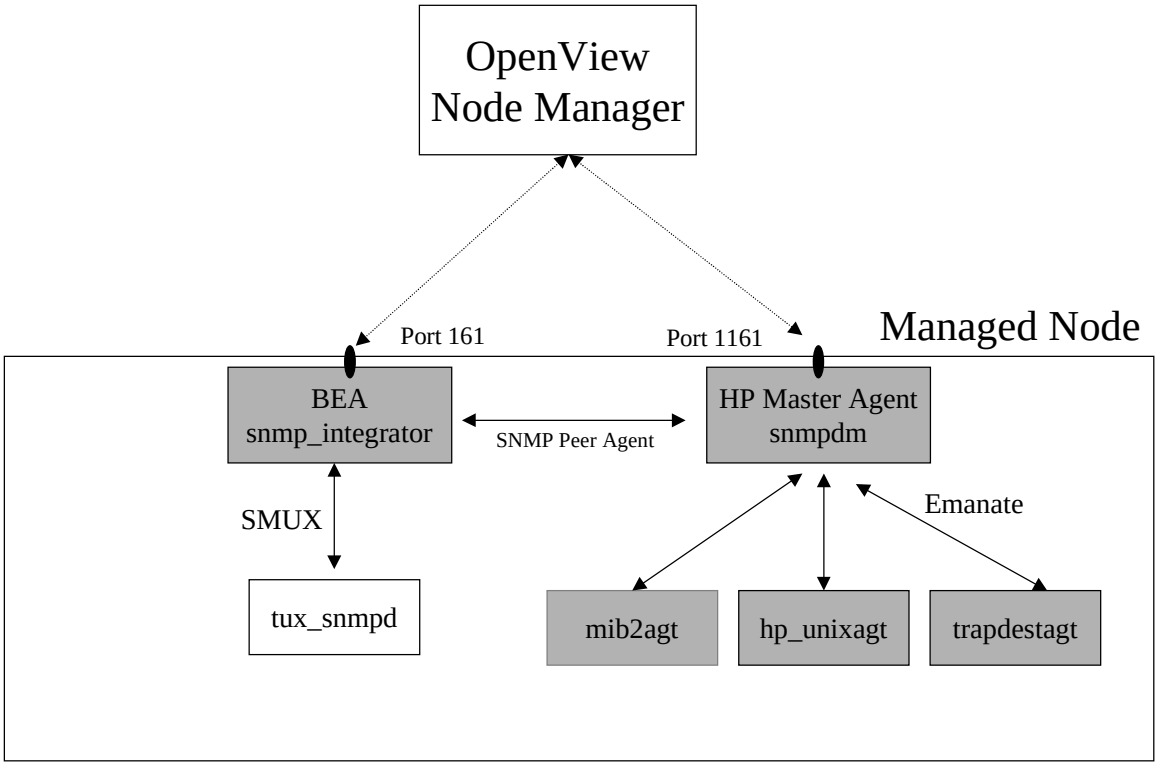
10.4.1 Implications for BEA Manager

All TUXEDO and /Q components run exclusively on the Master machine. As a result the tux_snmpd need only run on the Master machine. TUXEDO is not running on the Backup machines and consequently there is

nothing to monitor. However it should be noted that many of the traps defined for EIA monitor the health of inter-machine communications. These are no longer needed in a single machine configuration.

10.5 SNMP Agent Architecture

The SNMP agents running on each of the EIA managed node are essentially made up of two groups



1. The BEA Manager **snmp_integrator and HP SNMP agents** (snmp_integrator, snmpdm, mib2agt, hp_unixagt and trapdestagt) will run continuously on all EIA servers (Master and Backup). All of these agents can only be start and stopped by root. These agents will be started on every EIA machine whenever the UNIX operating system is started using the UNIX inetd/rc facilities (which executes as root). The agents will run continually and will not be stopped. This provides a baseline 'TUXEDO ready' configuration of SNMP agents on each EIA machine. Even when TUXEDO is not running the HP UNIX MIB (implemented by the hp_unixagt) and the BEA RULE_ACTIONS (implemented by the snmp_integrator) can be queried using SNMP.

2. The TUXEDO agent (**tux_snmpd**) will only run those machines that run the TUXEDO application (the active node). In contrast the HP agents and snmp_integrator this agent can be started and stopped by the TUXEDO administration account (eiaadm) This will be achieved by including the command to start and stop the tux_snmpd agent in the start_eia and stop_eia command (this is already the case).

   The tux_snmpd agent connects as a client the TUXEDO application. Consequently the order in which TUXEDO agents are started and stopped is significant.

The **start_eia** command should boot TUXEDO and then start the tux_snmpd agent.
The **stop_eia** command should stop the tux_snmpd agent first and then shutdown the TUXEDO application.

10.6 Failover and the BEA Manager TUXEDO Agent

- Under normal circumstances the EIA TUXEDO application and the TUXEDO snmp agent will run on only one of the two machines in each domain (the 'Master' machine).
- TUXEDO and the TUXEDO tux_snmpd agent will not be running on the 'backup' machine.
- MC ServiceGuard will detect a fatal system error.
- ServiceGuard will then 'failover' the EIA TUXEDO application and tux_snmpd agent to the Backup machine.
- This will be done by stopping TUXEDO and tux_snmpd on the master machine (using stop_eia) and then starting TUXEDO and tux_snmpd on the backup machine (using start_eia).

11 Appendix D – BEA Manager User Defined Trap List.

11.1 Background

This document gives details of each of the user defined traps that I will be implementing and testing over the next few days for the EIA project.

These traps when issued will be associated with;

1. A user defined trap number
2. The rule name and state change
3. Enterprise OID (where this OID is .1.3.6.1.4.1.140.1.1.0.'Trap number')

See the section trap output for further details

Two traps will be raised for each rule. Odd numbered traps (eg 101) indicate a change from OK to error state. Even numbered traps (eg 102) indicate a change from error to OK state.

11.2 Trap Table

Trap Number	State Change	Severity	Textual Description
100			TUXEDO Traps
101	Rule domState triggered from OK to ERR state	ERROR	Domain State is not active
102	Rule domState triggered from ERR to OK state	INFO	Domain State has returned to Active
103	Rule mcState triggered from OK to ERR state	ERROR	Machine State is Partitioned
104	Rule mcState triggered from ERR to OK state	INFO	Partitioned machine id now active
105	Rule grpState triggered from OK to ERR state	ERROR	Group State is no longer active
106	Rule grpState triggered from ERR to OK state	INFO	Group State returned to active
107	Rule svrState triggered from OK to ERR state	ERROR	Server state is not active
108	Rule svrState	INFO	Server state returned to active

	triggered from ERR to OK state		
109	Rule sysevtUp triggered from OK to ERR state	ERROR	The TMSYSEVT server status is not active
110	Rule sysevtUp triggered from ERR to OK state	INFO	The TMSYSEVT server status has returned to active
111	Rule sqState triggered from OK to ERR state	ERROR	The Server Queue status is not active
112	Rule sqState triggered from ERR to OK state	INFO	The Server Queue status has returned to active
113	Rule svcState triggered from OK to ERR state	ERROR	The service state is not active
114	Rule svcState triggered from ERR to OK state	INFO	The service state has returned to active
115`	Rule wshState triggered from OK to ERR state	ERROR	The Workstation Handler is not active
116	Rule wshState triggered from ERR to OK state	INFO	The Workstation Handler status has returned to active
117	Rule wslState triggered from OK to ERR state	ERROR	The Workstation Listener is not active
118	Rule wslState triggered from ERR to OK state	INFO	The Workstation Listener status has returned to active
119	Rule tlnState triggered from OK to ERR state	ERROR	The tlisten is not active
120	Rule tlnState triggered from ERR to OK state	INFO	The tlisten status has returned to active
121	Rule brdState triggered from OK to ERR state	ERROR	The bridge is not active
122	Rule brdState triggered from ERR to OK state	INFO	The bridge status has returned to active
123	Rule devState triggered from	ERROR	The device is not active

	OK to ERR state		
124	Rule devState triggered from ERR to OK state	INFO	The device status has returned to active
125	Rule tranState triggered from OK to ERR state	ERROR	The transaction status is aborted
126	Rule tranState triggered from ERR to OK state	INFO	The abort transaction has been committed
200			SAP Application Queue
201	Rule qss_sap triggered from OK to ERR state	ERROR	SAP Queue Space: Status is not active
202	Rule qss_sap triggered from ERR to OK state	INFO	SAP Queue Space: Status is active
203	Rule qsm_sap triggered from OK to ERR state	ERROR	SAP Queue Space: Number of messages on this queue space exceeds the maximum allowed
204	Rule qsm_sap triggered from ERR to OK state	INFO	SAP Queue Space: Number of messages on this queue space has fallen back below the maximum allowed
205	Rule qse_sap triggered from OK to ERR state	ERROR	SAP Queue Space: There is a message on the error queue
206	Rule qse_sap triggered from ERR to OK state	INFO	SAP Queue Space: There is a no longer a message on the error queue
220			ODY Application Queue
221	Rule qss_ody triggered from OK to ERR state	ERROR	ODY Queue Space: Status is not active
222	Rule qss_ody triggered from ERR to OK state	INFO	ODY Queue Space: Status is active
223	Rule qsm_ody triggered from OK to ERR state	ERROR	ODY Queue Space: Number of messages on this queue space exceeds the maximum allowed
224	Rule qsm_ody triggered from ERR to OK state	INFO	ODY Queue Space: Number of messages on this queue space has fallen back below the maximum allowed
225	Rule qse_ody triggered from OK to ERR state	ERROR	ODY Queue Space: There is a message on the error queue
226	Rule qse_ody	INFO	ODY Queue Space: There is a no longer a message

	triggered from ERR to OK state		on the error queue
240			ITRC Application Queue
241	Rule qss_itc triggered from OK to ERR state	ERROR	ITRC Queue Space: Status is not active
242	Rule qss_itc triggered from ERR to OK state	INFO	ITRC Queue Space: Status is active
243	Rule qsm_itc triggered from OK to ERR state	ERROR	ITRC Queue Space: Number of messages on this queue space exceeds the maximum allowed
244	Rule qsm_itc triggered from ERR to OK state	INFO	ITRC Queue Space: Number of messages on this queue space has fallen back below the maximum allowed
245	Rule qse_itc triggered from OK to ERR state	ERROR	ITRC Queue Space: There is a message on the error queue
246	Rule qse_itc triggered from ERR to OK state	INFO	ITRC Queue Space: There is a no longer a message on the error queue
247	Rule qqn_itc triggered from OK to ERR state	ERROR	ITRC Queue Space: Number of messages on the ERROR queue exceeds a threshold value
248	Rule qqn_err triggered from ERR to OK state	INFO	ITRC Queue Space: Number of messages on the ERROR queue has fallen back below the threshold value
260			ERROR Application Queue
261	Rule qss_err triggered from OK to ERR state	ERROR	ERROR Queue Space: Status is not active
262	Rule qss_err triggered from ERR to OK state	ERROR	ERROR Queue Space: Status is active
263	Rule qsm_err triggered from OK to ERR state	ERROR	ERROR Queue Space: Number of messages on this queue space exceeds the maximum allowed
264	Rule qsm_err triggered from ERR to OK state	ERROR	ERROR Queue Space: Number of messages on this queue space has fallen back below the maximum allowed
265	Rule qse_err triggered from OK to ERR state	ERROR	ERROR Queue Space: There is a message on the error queue
266	Rule qse_err triggered from ERR to OK state	ERROR	ERROR Queue Space: There is a no longer a message on the error queue

267	Rule qqn_err triggered from OK to ERR state	ERROR	ERROR Queue Space: Number of messages on the ERROR queue exceeds a threshold value
268	Rule qqn_err triggered from ERR to OK state	ERROR	ERROR Queue Space: Number of messages on the ERROR queue has fallen back below the threshold value

11.3 Trap Output

This is the example output printed by the BEA snmptrapd program. This is a utility (part of the agent development kit) that sits on the 162 port and prints the details of the any traps received to the standard output. This can then be redirected to a file.

This output was received after the TMSYSEVT sever was re-booted on the master machine (15.95.224.11) The first trap received was the standard trap 23 (server state change). This was followed after several seconds (representing the rule polling interval) by a user defined trap 109 (sysevtup).

11.4 Standard Trap (.1.3.6.1.4.1.140.300.0.23: server state trap)

15.95.224.11: Enterprise Specific Trap (23) Uptime: 0:00:00
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsName
OCTET STRING- (ascii): .SysServerState
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsSeverity
INTEGER: info(3)
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsLmid
OCTET STRING- (ascii): MMB_ctss121_EIA
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsTime
INTEGER: 955044512
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsUsec
INTEGER: 600492
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsDescription
OCTET STRING- (ascii): INFO: .SysServerState: TMSYSEVT, group EVENTGRP1, id 20 state change to ACTIVE
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsClass
OCTET STRING- (ascii): T_SERVER
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsUlogCat
OCTET STRING- (ascii): LIBTUX_CAT
Name: private.enterprises.bea.tuxedo.tuxEvents.tuxEventTrapVars.tuxEventsUlogMsgNum
INTEGER: 1518
Name: private.enterprises.bea.beaDomainList.beaDomainListEntry.beaDomainID
OCTET STRING- (ascii): AP_DOM
Name: private.enterprises.bea.beaDomainList.beaDomainListEntry.beaDomainKey
INTEGER: 113757
Name: private.enterprises.bea.beaDomainList.beaDomainListEntry.beaLogicalAgentName.0
OCTET STRING- (ascii): tux_snmpd

11.5 User Defined Trap (.1.3.6.1.4.1.140.1.1.0.110: user defined trap 110)

15.95.224.11:Enterprise Specific Trap (110) Uptime: 0:00:00
Name: private.enterprises.bea.beaSystem.beaTrapDescr.0
OCTET STRING- (ascii): Rule id <sysevtUp> has triggered from ERR to OK state

12 Appendix E – BEA Tuxedo MIB Definitions

HP EIA Project
BEA Manager: A Review of the BEA MIB Definitions

12.1 Introduction

This document reviews the structure of the SNMP MIB supplied as part of the BEA manager 2.0 product. This MIB enables the use of SNMP to manage BEA TUXEDO and some aspects of the underlying UNIX operating system.

In the architectural design of the management system for the EIA project, the BEA Manager Product (agent integrator together with its TUXEDO and Unix SNMP agents) will be used to generate default SNMP traps; user defined SNMP traps and will be used to enable the collection of performance metrics.

12.2 Default SNMP Traps

The BEA Manager TUXEDO SNMP agent generates 32 standard default SNMP traps 'out of the box'. This document reviews the nature of these default TUXEDO SNMP traps and give guidance on their relevance to the EIA project and mechanisms that can be used to trigger these traps for testing purposes.

12.3 User defined SNMP Traps

User defined traps will be specified and collected by the BEA Manager agent integrator. The agent integrator will be configured to periodically poll a selected number of the MIB attributes. If any of these exceed a specified threshold a user defined the agent integrator will generate trap. This is achieved by adding a RULE_ACTION to the BEA Manager Configuration file (/etc/beamgr.conf).

The following RULE generates a trap if the machines CPU is busier than 80% and another when it fall back below 80%.

```
RULE_ACTION cpu 600 if ( VAL(140.11.1.0) > 80 ) { TRAPID_ERR = 104  TRAPID_OK = 105 }
```

In this case the monitored attribute (.140.11.1.0) is the beaSysPerfCpu attribute in the beaSysPerf group of the BEA MIB which is managed by the unix_snmpd agent.

A RULE_ACTION can be specified for any MIB attribute associated with any SNMP agent managed by the agent integrator. Consequently any of the attributes specified by the BEA MIB (and also the HP MIB) can be used to generate user-specified traps.

This document reviews those attributes in the BEA MIB (that relate to both the TUXEDO and also the UNIX OS) that are of particular relevance to the EIA project and should therefore be monitored to generate user defined SNMP traps.

12.4 Performance Metrics

On the EIA project performance metrics will be gathered and logged to Measureware. This will be achieved by the development of a ‘housekeeping’ process. This housekeeping process will periodically issue SNMP gets against attributes in the SNMP MIBs that relate to systems performance.

This document therefore reviews those MIB attributes that can be used to provide a measurement of system performance.

13 Overview of the BEA MIB

The BEA MIB is defined in the documents bea.asn1 and bea_lc_trap.asn1. The document bea_lc_trap.asn1 defines attributes that are used by the BEA Manager Log Central product (which is not going to be used on the EIA project).

The bea.asn1 document defines a number of MIB's from the BEA enterprise root OID located at .1.3.6.1.4.1.140 (ISO.ORG.DOD.INTERNET.PRIVATE.ENTERPRISES.BEA).

The whole BEA MIB comprises a total of 790 attributes. Most of these are read only and can only support SNMP GET requests. However 290 MIB attributes are read write and can support both SNMP GETs and SETs.

13.1 BEA MIB GROUPS

The BEA Mib is divided into a number of sub MIB's. These are:

OID	MIB Name		Description
BEA 1	BeaSystem		This MIB contains general-purpose objects such as operating system name and version and some of the fields in the beamgr.conf configuration file. Fields in the configuration file can be changed through an SNMP SET on the corresponding MIB object. The unix_snmpd subagent provided with Agent Integrator supports this MIB.
BEA 2	BeaUnix		This MIB includes such objects as the process table, file system table, the ipcs tables, the performance measurements rstat, and the pstat –s command. The unix_snmpd subagent provided with Agent Integrator supports this MIB.
BEA 4	BeaPm		The process monitor MIB is found in the BEA Manager MIB file (bea.asn1) and defines objects that support the Log Central process monitor. This MIB is supported by the pm_snmpd subagent provided with the Agent Integrator.
BEA 5	BeaSmgr		The shared memory manager MIB specifies attributes for each block of shared memory used by TUXEDO. It is supported by the unix_snmpd subagent shipped with the Agent Integrator.
BEA 11	BeaSysPerf		BEA System Performance MIB supports attributes that are similar to those supported by the Sun pefmeter on Sun workstations. It is supported by the unix_snmpd subagent shipped with the Agent Integrator.
BEA 12	BeaNt		This MIB represents system performance attributes specific to the Windows NT system. This MIB group is supported by the nt_snmpd subagent shipped with the Agent Integrator.
BEA 21	BeaTrap		The Log Central Traps MIB can be found in the file `bea_lc_trap.asn1` and contains Log Central attributes that are used as variables in the traps.
BEA 200	BeaIntAgt		This MIB is supported directly by the Agent Integrator master agent. The beaIntAgtTable permits the addition or deletion or modification of RULE_ACTION entries that are used to control local polling by the Agent Integrator. Each entry

			(row) in the beaIntAgtTable supports a particular polling rule. The beaIntAgtStatus object is a read-write object that can be used to de-activate or activate polling execution by particular rules.
BEA 300	TUXEDO		This is the largest group and allows an SNMP management console to query and in some cases set TUXEDO system attributes. Its sub groups are described in detail below
BEA 305	DomainList		This is part of the TUXEDO MIB described in detail below

= Number of attributes specified in this MIB

The BEA Manager product is supplied with 4 SNMP agents (tux_snmpd, unix_snmpd, nt_snmpd, and pm_snmpd) and an agent integrator. Each of these is responsible for managing different sub-MIBs within the BEA MIB as represented below.

Mib file	Agent	MIB Group
Bea.asn1	Tux_snmpd	TUXEDO, DomainList
	Unix_snmpd	BeaSystem, BeaUnix, BeaSmgr, BeaSysPerf
	Snmp_integrator	BeaIntAgt
	Nt_snmpd	BeaNt
Bea_lc_trap.asn1	Pm_snmpd	BeaPm, BeaTrap

Note: The EIA project is not using either Log Central or NT and so the BeaNt, BeaPm and BeaTrap MIBs will not be examined.

13.2 The TUXEDO MIB

The TUXEDO MIB is the largest defined by BEA and is managed by the TUXEDO SNMP agent (tux_snmpd). This MIB is further divided into the 24 groups described below.

These groups are in turn consolidated into 7 TUXEDO Sub MIBs. The largest of these is the Core TUXEDO Mib. The TUXEDO system Core MIB defines the set of groups through which the fundamental aspects of an application may be configured and managed. This includes management of machines, servers, networking, and load balancing. The TUXEDO Core MIB defines the basic objects that form a TUXEDO application

Other TUXEDO sub MIBs include the Workstation, TUXEDO Domain, Application Queue, ACL, Event and M3 sub MIB.

OID	Group Name	Sub MIB		Description
Tux 1	TuxWsmib	Workstation		This MIB is an extension of the TUXEDO Core MIB and specifies the information required to control access to a TUXEDO application from multiple workstations. The TUXEDO Workstation subsystem consists of a workstation

				clients (WSC) library, the workstation listener (WSL) executable, and the workstation handler (WSH) executable. The Workstation MIB specifies information about workstation listeners and workstation handlers.
Tux 2	TuxEvents	Event		The Event Broker MIB defines the characteristics of an event subscription. You can use the Event Broker MIB to obtain the characteristics of current event subscriptions, define new subscriptions, or invalidate subscriptions. To enable both system event and application event notification, you need to define the system event broker and the application event broker in the TUXEDO Core MIB.
Tux 3	TuxTdomain	Core		This group represents the global application attributes for the domain to which the TUXEDO SNMP agent is currently connected. These object values serve to identify, customize, size, secure, and tune a TUXEDO System/T application. Many of the object values represented here serve as application defaults for other groups represented in this MIB.
Tux 4	TuxTgroup	Core		This group represents application attributes pertaining to a particular server group. These attribute values represent group identification, location, and DTP information
Tux 5	TuxTmachine	Core		This group represents application attributes pertaining to a particular machine. These attribute values represent machine characteristics, per-machine sizing, statistics, customization options, and UNIX system filenames
Tux 6	TuxTmsg	Core		This group represents runtime attributes of the TUXEDO System/T managed UNIX system messages
Tux 7	TuxTqueue	Core		This group represents the runtime attributes of the queues in an application. These attribute values identify and characterize allocated TUXEDO System/T request queues associated with servers in a running application. They also track statistics related to application workloads associated with each queue object
Tux 8	TuxTrouting	Core		The group represents configuration objects of routing specifications for an application. These object values identify and characterize application data dependent routing criteria with respect to field names, buffer types, and routing definitions.
Tux 9	TuxTulog	Core		This group represents runtime attributes of userlog files within an application.
Tux 10	TuxTsvc	Core		This represents configuration attributes of services within an application. These attribute values identify and characterize configured services. A TuxTsvc object provides activation time configuration attributes for services not specifically configured as part of the group
Tux 11	TuxTAcl	ACL		The ACL MIB enables a system manager to administer TUXEDO security through authenticating users, setting permissions, and controlling access. The ACL MIB defines the objects controlled by the ACL facility

Tux 12	TuxTAppQ	Application Queue			The TUXEDO Application Queue MIB provides the administrative environment required for managing and controlling access to application queues. The Application Queue MIB defines the structure of the application queues. In TUXEDO applications, messages are stored on a queue, and queues are defined within a particular queue space. Queuing and de-queuing is done within a transaction. The Application Queue MIB consists of five different groups for defining queue access, queues, messages, queues spaces, and queue transactions
Tux 14	BeaEventFilters	Core			This MIB group represents all the event filters defined for the Agent Connection. These are used to determine the collection of events to be forwarded as SNMP trap notifications.
Tux 16	TuxTbridge	Core			This group represents those runtime attributes pertaining to the connectivity between logical machines that make up an application. These attribute values represent connection status and statistics.
Tux 17	TuxTclient	Core			This group represents runtime attributes of active clients within an application. These attribute values identify and track the activity of clients within a running application.
Tux 18	TuxTconn	Core			This group represents runtime attributes of active conversational servers within an application.
Tux 19	TuxTdevice	Core			This represents configuration and runtime attributes of raw disk slices or UNIX system files being used to store TUXEDO System/T device lists. This class allows for the creation and deletion of device list entries within a raw disk slice or UNIX system file.
Tux 20	TuxTserver	Core			This group represents configuration and runtime attributes of servers within an application. These attribute values identify and characterize configured servers as well as provide runtime tracking of statistics and resources associated with each server object.
Tux 21	TuxTlisten	Core			This group represents runtime attributes of /T listener processes for a distributed application
Tux 23	TuxTranaction	Core			This table represents runtime attributes of active transactions within the application
Tux 28	TuxTnetGroup	Core			This table represents application attributes of network groups. Network groups are groups of logical machine IDs that can communicate over the network
Tux 33	TuxTnetMap	Core			The rows in this table identify which logical machines belong to which network groups.
Tux 48	M3	M3			This is the 5 MIB tables that are specific to the M3 product. To access these MIB objects, the M3 version of the Agent Connection should be running on the machine where M3 application resources are accessible
BEA 305	BeaD	TUX			This MIB group represents information about the TUXEDO domain

	omain List	EDO DOM AIN		the agent is monitoring, as specified at startup

= Number of attributes specified in this sub MIB

Note: The EIA project TUXEDO domain will not use publish and subscribe (events), data dependant routing, access control conversational servers or M3 (WLE). Consequently the Event MIB, TuxRouting group, ACL MIB, TuxTconn group and M3 MIB has not been reviewed.

13.3 The Standard TUXEDO SNMP traps

The TUXEDO SNMP agent (tux_snmpd) is capable of raising 32 standard SNMP trap notifications. These are provided 'out of the box' and cannot be configured. These standard TUXEDO traps are associated with 12 variables (attribute/value pairs) in the variable bindings of the trap packet. These are

1. BeaDomainId: The id of TUXEDO domain that generated the trap
2. BeaDomainKey: IPC key of TUXEDO domain
3. BeaLogicalAgentName Logical name of SNMP agent. Will be either tux_snmp/unix_snmpd or the name supplied as –l when agent started
4. TuxEventsLmid The logical Machine Identifier of the machine where trap originated
5. TuxEventsName String that uniquely identifies trap
6. TuxEventsSeverity Severity of trap (Error=1, Warn=2, Info=3)
7. TuxEventsTime Long that contains the event detection time in seconds. Taken from local system clock
8. TuxEventsUsec Long that contains the event detection time in milliseconds. Taken from local system clock
9. TuxEventsDescription A one line string summarizing the event
10. TuxEventsClass The MIB class of the object associated with this event
11. TuxEventsUlogCat TUXEDO catalog name from which message was derived
12. TuxEventsUlogMsgNum Catalog message number

13.4 List of Standard TUXEDO SNMP Traps

In the table presented below each of the 32 standard TUXEDO SNMP traps is categorized with a severity level. These levels are the same as those used by application programmers to categorize any application errors they encounter

These are:

1/FATL Fatal (1). This is an error condition. The system cannot function properly and this error must be immediately corrected and the system recycled.

2/FUNC Functional (2). This is an error condition. It should be immediately fixed, however some part of the system will continue to function while this error is being investigated

3/WARN Warning (3). This represents an event that should be investigated for relevance and impact on system functionality

4/INFM Information (4). This has been generated during the norm execution of the system. No action is required.

5/DEB *Debug (5). This is an error message category that should never result in a SNMP trap being generated*

Trap		Description	Remedy	Level
MachineMsgQTrap		A server posting a message encountered a blocking condition while putting a message on a message queue	Configure larger message queues and/or distribute the load equally on all the machines.	1/FATL
MachinePartionedTrap		The TUXEDO system (DBBL(has partitioned the machine either because BBL was slow in responding or network link to master has been broker	Check network BBL, BRIDGE processes are running tlisten The software is capable of un-partitioning the machine if things stabilize.	1/FATL
NetworkDroppedTrap		Network Link between machines has been dropped abnormally	Check Machine Bridge tlisten	1/FATL
NetworkFailureTrap		Connection failure between bridge processes	Machine Bridge tlisten	1/FATL
NetworkStateTrap		A server has died and the BBL has cleaned up the slot occupied by that server	Debug and fix server before it is restarted	1/FATL
ServerCleaningTrap		Server has died abnormally and BBL has cleaned up slot owned by server	Debug and fix server before it is restarted	1/FATL
ServerDiedTrap		Server has died abnormally and the BBL has detected this during its periodic scan of the BB	Debug server fix and restart	1/FATL
ServerInitTrap		Server tpsrvinit failed during startup and therefore couldn't be booted	Debug, fix and restart Could also be due to TUX resource limit	1/FATL
ServerMaxgenTrap		Re-startable Server	Debug server and	1/FATL

		has died Maxgen-1 time during the specified grace period	restart	
MachineBroadcastTrap		This trap implies that a broadcast delivery (tpbroadcast) has failed	Configure larger message queues and load balance clients and servers such that excessive load in not put on certain machines	2/FUNC
MachineFullMaxAccessorsTrap		Maximum number of assessors has been reached for this machine LMID	Increase the MAXACCESSERS value for the particular machine. Or, if the hardware/software limits have been reached for the maximum number of users on the machine, move additional users to other machines.	2/FUNC
MachineFullMaxConvTrap		The limit has been reached on the number of concurrent conversations supported by this machine LMID	Increase the value of MAXCONV for the particular machine to the point that this event is not generated.	2/FUNC
MachineFullMaxGttTrap		The limit has been reached on the number of concurrent transactions supported by this machine LMID	Increase the value of MAXGTT for the particular machine to the point that this event is not generated	2/FUNC
MachineFullMaxWsClientsTrap		The limit has been reached on the number of workstation clients that can be supported by this machine LMID	Increase the value of MAXWSCLIENTS for the particular machine to the point that this event is not generated.	2/FUNC
MachineSlowTrap		The BBL on a non master machine is slow in generating the	Check network traffic and performance BBL	3/WARN

		heartbeat sent to he Master	Bridge NB This problem could be intermittent	
ServerRestartingTrap		Server died abnormally and has being successfully restarted	Server should not have died in first place – debug, fix and recycle	3/WARN
ClientDiedTrap		Client exited without doing a tpterm	None – Warning However application developers should always endure that clients perform a tpterm before exiting	3/WARN
ClientSecurityTrap		Client failed security validation when trying to join TUXEDO	Warning – makes sure no unauthorized access is being attempted	3/WARN
TransHeurisic AbortTrap		Database performed heuristic abort for a particular transaction	Check that transaction manager is still running	3/WARN
TransHeurisic Commitrap		Database performed heuristic commit for a particular transaction	Check that transaction manager is still running	3/WARN
EventDeliveryTrap		Event server failed to perform at least one notification of a posted event	Make sure notifications are doable	3/WARN
EventFailureTrap		The event server has failed during a self-check. Cannot put message on own message queue	Configure larger message queues or distribute the load in the application equally among all the machines.	3/WARN
ResourceConfigTrap		This trap is generated whenever the system configuration changes	None – Informational	4/INFM
MachineConfigTrap		Change in machine configuration	None – Informational	4/INFM
MachineStateTrap		The Machine has changed its state	None – Informational	4/INFM
NetworkConfigTrap		Network Link between machines has changed state	None – Informational	4/INFM
NetworkFlowTrap		The virtual circuit	None – Informational	4/INFM

		between 2 machines has changed to a new state		
ServerConfigTrap		Configuration Parameters for server has been updated	None – Informational	4/INFM
ServerStateTrap		Server has changed state	None – Informational	4/INFM
ServerTpExitTrap		Server received a request but did a tpreturn while server was executing application specific code	None – Informational	4/INFM
ClientConfigTrap		Client has change configuration	None – Informational	4/INFM
ClientStateTrap		Client has changed state	None – Informational	4/INFM

13.5 Raising the standard SNMP Traps in a Test Environment

Trap		**Cause**	**How to trigger**
ResourceConfigTrap		This trap is generated whenever the system configuration changes	Use the tmconfig utility to make a dynamic change to a value in resources section of the bulletin board (e.g. increment the MAXSERVICES by 1)
MachineBroadcastTrap		This trap implies that a broadcast delivery (tpbroadcast) has failed	This trap should not be tested since we are using the tpbroadcast facility on the EIA project.
MachineConfigTrap		Change in machine configuration	Use the tmconfig utility to make a dynamic change to a value in machine section of the bulletin board (e.g. increment the MAXSERVICES by 1)
MachineFullMax AccessorsTrap		Maximum number of assessors has been reached for this machine LMID	Use the tmconfig utility to set the MAXACCESSORS value in the machine section of the bulletin board to the current number of accessors on that machine –1
MachineFullMax ConvTrap		The limit has been reached on the number of concurrent conversations supported by this machine LMID	This trap should not be tested since we are using the conversational servers on the EIA project.
MachineFullMax GttTrap		The limit has been reached on the number of concurrent	Use the tmconfig utility to set the MAXGTT value in the machine section of the bulletin board to a small number. Then start a larger number of testclients and

		transactions supported by this machine LMID	issues concurrent transactions
MachineFullMax WsClientsTrap		The limit has been reached on the number of workstation clients that can be supported by this machine LMID	Use the tmconfig utility to set the MAXCWSCLIENTS value in the machine section of the bulletin board to a small number (e.g. 2) and then start more than this number of workstation testclients.
MachineMsgQTrap		A server posting a message encountered a blocking condition while putting a message on a message queue	Cause the message queue to block. This will be fairly difficult to achieve. Can I suggest that you implement a service call LOOPSVC. This should either loop forever or sleep for a very long time before it returns. During the test make a large number of requests to LOOPSVC. Only the first of these requests will be processed. The others will queue on the services input queue. Eventually this queue will fill and block raising the MachineMsgQTrap.
MachinePartionedTrap		The TUXEDO system has been partitioned the machine either because BBL was slow in responding or network link to master has been broker	Kill the BRIDGE process on the one master machine. Kill the tlisten process. This will ensure that the machine remains partitioned
MachineSlowTrap		The BBL on a non master machine is slow in generating the heartbeat sent to he Master	Kill the BRIDGE process on the one master machine. Kill the tlisten process. This will ensure that the machine remains partitioned and no heartbeat will be sent.
MachineStateTrap		The Machine has changed its state	Kill the Bridge on the none master machine. This machine should be marked as partitioned until the Bridge process is restarted when it will change its state back to un-partitioned
NetworkConfigTrap		Network Link between machines has changed state	Kill the Bridge on the none master machine. This machine should be marked as partitioned until the Bridge process is restarted when it will change its state back to un-partitioned.
NetworkDroppedTrap		Network Link between machines has been dropped abnormally	Kill the BRIDGE process on the one master machine. Kill the tlisten process. This will ensure that the machine remains partitioned
NetworkFailureTrap		Connection failure between bridge processes	Kill the BRIDGE process on the one master machine. Kill the tlisten process. This will ensure that the machine remains partitioned
NetworkFlowTrap		The virtual circuit between 2 machines has changed to a new	Since we only have 2 machines in the domain there can only be 1 virtual circuit and therefore this trap should never be generated.

		state	
NetworkStateTrap		A server has died and the BBL has cleaned up the slot occupied by that server	Kill the testserver using the kill –9 command
ServerCleaningTrap		Server has died abnormally and BBL has cleaned up slot owned by server	Kill the testserver using the kill –9 command
ServerConfigTrap		Configuration Parameters for server has been updated	Use Tmconfig to set the MAXGEN of the testserver to 4.
ServerDiedTrap		Server has died abnormally and the BBL has detected this during its periodic scan of the BB	Kill the testserver using the kill –9 command
ServerInitTrap		Server tpsrvinit failed during startup and therefore couldn't be booted	Re-link the testserver with a tpsrvinit that returns an error when the servers starts.
ServerMaxgenTrap		Re-startable Server has died Maxgen-1 time during the specified grace period	Configure a the testserver to be restartable with the entry RESTART=Y MAXGEN=3 GRACE=3600 This will instruct TUXEDO to restart the sever 3 times within 60 minutes. Issue a kill against the testserver. Wait until TUXEDO restarts the testserver and issue another kill. At this point testserver will have died MAXGEN-1 times and the ServerMaxgenTrap should be generated.
ServerRestartingTrap		Server died abnormally and has being successfully restarted	Kill a restartable server (e.g. testserver). This trap should be generated when the server is started by the system
ServerStateTrap		Server has changed state	Change the state of the server. Perform a selective shutdown on this server (tmshutdown –s testserver).
ServerTpExitTrap		Server received a request but did a tpreturn while server was executing application specific code	Call a service that performs a tpreturn with TPEXIT (EXITSVC). This will cause the server to exit and will generate this trap.
ClientConfigTrap		Client has change configuration	Force a change in the client's configuration. This can be done by running the testclient, which will change state from Active to Dead when it terminates execution.

ClientDiedTrap		Client exited without doing a tpterm	Develop a test client application that makes a call to the TOUPPER test service but then exits without performing a tpterm. This should result in the generation of the ClientDiedTrap.
ClientSecurityTrap		Client failed security validation when trying to join TUXEDO	This trap should not be tested since we are using TUXEDO based authentication on the EIA project.
ClientStateTrap		Client has changed state	Force a change in the client's state. This can be done by running the testclient, which will change state from Active to Dead when it terminates execution.
TransHeurisic AbortTrap		Database performed heuristic abort for a particular transaction	This trap should not be tested since we are using this capability on the EIA project.
TransHeurisic Commitrap		Database performed heuristic commit for a transaction	This trap should not be tested since we are using this capability on the EIA project.
EventDeliveryTrap		Event server failed to perform at least one notification of a posted event	This trap should not be tested since we are using the publish and subscribe facility on the EIA project.
EventFailureTrap		The event server has failed during a self-check. Cannot put message on own message queue	This trap should not be tested since we are using the publish and subscribe facility on the EIA project.

13.5.1 Trap Test Software

Test Client: This client connects to TUXEDO (tpinit) successfully calls the TOUPPER service but then terminates execution without performing at tpterm to disconnect from TUXEDO.

TestServer: This server should have a tpsrvinit that can be replaced with one that returns failure on startup (so that ServerInitTrap can be raised). It should publish three services – TOUPPER LOOPSVC and EXITSVC.
The EXITSVC will simply call tpreturn with TPEXIT to test ServerTpExit trap.
The LOOPSVC will loop forever or sleep for a long period of time before it returns
The ubb entry for this server should specify that it is a restartable server (e.g. RESTART=Y MAXGEN=3 GRACE=3600)

13.6 A Review of the BEA MIB

It is clear that the BEA MIB defines a bewildering array of near 800 attributes across a large number of groups and functionality areas.

Which of these attributes should be used on the EIA project for the generation of user defined traps or should be monitored to provide performance metrics?

I would suggest it might clarify the situation if we were to derive further (functional) classification of these MIB attributes.

Functional Area	Mib Group
The Unix Operating System	BeaSystem, BeaUnix, BeaSmgr, BeaSysPerf, TuxTdevice
BEA Manager	BeaIntAgt, BeaEventFilter
TUXEDO Application	BeaDomainList, TuxTdomain, TuxTmachine, TuxTgroup TuxTserver, TuxTsvc, TuxTqueue, TuxTmsg, TuxTTulog,,
TUXEDO Communications	Workstation, TuxTclient, TuxTnetGroup, TuxTnetmap, TuxTlisten, TuxTbridge
Transactions	TuxTransaction,
Application Queues	TuxTAppQ
Not Applicable	BeaNt, BeaPm, BeaTrap, Tux Event, TuxTAcl, TuxTrouting, TuxTconn

The EIA project TUXEDO application is built on the UNIX operating system and makes extensive use of application queues (/Q).

The EIA operations team have expressed an interest in measuring the activity of transactions within the system and the number of messages held on application (/Q) queues.

Clearly the Functional groups that would be of prime interest are

1. Those relating to the underlying UNIX operating system (disk & CPU utilization etc).
2. The /Q group.
3. The BeaIntAgt group that allows RULE_ACTIONS to be maintained dynamically
4. The Tuxedo application and Tuxedo communications groups that allows the health of the TUXEDO application to be monitored.

13.7 The Unix Operating System

The unix_snmpd agent supplied with the agent integrator product uses can monitor a number of mib groups including beaSystem, BeaUnix, BeaSmgr and BeaSysPerf. All of the attributes associated with these groups are read only.

13.7.1 BeaSystem

This group provides access to basic Unix OS characteristics. All of these attributes are read only and would not be suitable either a basis for raising user defined traps or for generating performance metrics.

13.7.2 BeaUnix

All of the attributes in this group are read only
beaPsTable - Process table

beaPsPid	
beaPsCpu	Percentage of CPU capacity in use
beaPsMem	Percentage of real memory being used by process
beaPsSize	combined size of data and stack segments – kB
beaPsStatus	R,T,P,D,I,Z

beaDfTable – Filesystem

beaDfIndex	Index of file system entry
beaDfkbytes	Size of file system in Kb
beaDfUsed	The number Kb used
beaDfCapacity	Percentage of total capacity used
beaPstat –	
beaPstatSwAlloc	Amount swap space KB allocated to private pages
beaPstatSwapUser	Total swap space allocated or reserved (kB).
BeaPstatSwapAvail	Total swap space available (kB)

BeaMqTable – IPC message queues

BeaMqCbytes	Number of bytes outstanding on queue
BeaMqQnum	Number of messages on queue
BeaMqQBytes	Max number bytes allowed on queue

BeaShmTable – IPC shared memory

BeaShmSeqgsz	Size of shared memory segment
BeaSemTable	IPC semaphores

BeaLclDfTable – local File System

beaLclDFilesystem	name of file system entry
beaLclDfkbytes	Size of file system in Kb
beaLclDfUsed	The number Kb used
beaLclDfCapacity	Percentage of total capacity used

BeaSmgr – shared memory table

BeaShmgrIpcKey	IPC key of shared memory
BeaSmgrShmAllocated	Status of shared memory can be monitored to detect if shared memory is corrupted
BeaSmgrSubSystem	Name of subsystem using shared memory
BeaSmgrMaxShmEntries	Maximum number of entries configured for shred memory
BeaSmgrtCurrentShmEntries	Current number of entries in shared memory
BeaShmgrPctShmUsed	Percentage of entries in shared memory used

BeaSysPerf – workstation performance attributes – most of these are cumulative counters

BeaSysPerfCPu	Percentage of CPU capacity being utilized
BeaSysPerfSwap	cumulative counter of jobs swapped
BeaSysPerfPerfDisk	Disk Traffic in number of blocks
BeaSysPerfintr	Number of interrupts
BeaSysPerfPfLoad	Size of run queue
BeaSysPerfPage	Paging Activity
BeaSysPerfIfNumber	Number of network interfaces
BeaSysPerfDiskDelta	Number of blocks transferred since last poll

BeaSysPerfLoadDelta	Size of run queue since last poll
BeaSysPerfPageDelta	Paging activity since last poll

BeaSysPerfIf - Interface Performance Attributes table

BeaSysPerfIfDescr	Description of interface
BeaSysPerfIfOperStatus	Status of interface up(1) down(2) testing(3)
BeaSysPerfIfInPackets	Total number of packets received on this interface
BeaSysPerfIfOutPackets	Total number packets sent form this interface

13.8 TUXEDO Application /Q (TuxTAppQ)

The Application queue (/Q) MIB consists of five groups:

TuxTAppQctrl	Controls and limits range of access via the other TuxTAppQ mib classes.
TuxTQspaceTbl	Application **Queue Space.** This group can be used to modify or create a new queue space. To create a new row in this table, a SET request should be issued with an index (tuxTQspaceGrpNo) of 40000. This is a reserved value for row creation in the table.
TuxTAppQTbl	**Application Queues**. One or more application queues may exist in a single application queue space. Each row represents a separate queue. This mib group can be used to create a new queue (row on tab le) by issue a SET request that specifies at least the values for the tuxTAppQname, tuxTAppQspaceName, and tuxTAppQmConfig attributes
TuxTAppQmsgTbl	**Messages** within Application Queues. A message is not created by an administrator; instead, it comes into existence as a result of a call to tpenqueue. A message can be destroyed either by a call to tpdequeue or by an administrator. Certain attributes of a message can also be modified by an administrator. For example, an administrator can move a message from one queue to another queue within the same queue space or change its priority.
TuxTQtransTbl	**Transactions** associated with application queues.

TuxTAppCtrl - Control Table

These read-write attributes control access to queue spaces, queues and message managed by other groups within the TuxTAppQ MIB. Using this group you can limit your TuxTAppQ access by machine, queue space, queue, message priority (not used here at EIA) and time. If you don't set specific values the defaults are used

TuxTAppQctrlLmid	R/W	Machine. Local (1) or all (2).
TuxTAppQctrlQmConfig	R/W	Device. Default is '*' - all
TuxTAppQctrlSpaceName	R/W	Queue Space. Default is '*' - all
TuxTAppQctrlQname	R/W	Queue. Default is "*' - all
TuxTAppQctrlMsgLoPrio	R/W	Lowest priority message to return. Default is 0 - none
TuxTAppQctrlMsgHiPrio	R/W	Highest priority message to return. Default is 0 - none
StartTime	R/W	Start Date and time in format YYMMDDHHMMSS default is '*' - all
TuxTAppQctrlMsgEndTime	R/W	End Date and time in format YYMMDDHHMMSS default is '*' - all

TuxTappQSpaceTbl – Queue Space

This Table specifies each of the table spaces that have been created to hold the application queues. This MIB group can be used to add a new queue space or modify certain attributes of an existing queue space.

Creating a New Queue Space

To create a new row in this table, a SET request should be issued with an index (tuxTQspaceGrpNo) of 40000. This is a reserved value for row creation in the table. The SET request also needs to specify values for at least tuxTQspaceQmConfig, tuxTQspaceName, tuxTQspaceLmid, tuxTQspaceIpckey, tuxTQspaceMaxMsg, tuxTQspaceMaxPages, tuxTQspaceMaxProc, tuxTQspaceMaxQueues, and tuxTQspaceMaxTrans. The newly created instance (row) will not be visible until it is attached to some server group.

Deleting an Existing Queue Space

An existing queue can be deleted by setting the tuxTQspaceState attribute to invalid

Modifying an Existing Queue Space

The following attributes can be modified for an existing queue by issuing a SNMP SET.
State (tuxTQspaceState), Blocking factor (tuxTQspaceBlocking), Error Queue Name (tuxTQspaceErrQname), Initiation Mode (uxTQspaceForceInit), IPC Key (tuxTQspaceIpckey), Max Messages (tuxTQspaceMaxMsg), Max Pages (tuxTQspaceMaxPages), Max Procedures (tuxTQspaceMaxProc), Max Queues (tuxTQspaceMaxQueues), Max Transactions (tuxTQspaceMaxTrans).

Each TuxTappQEntry row has the following attributes:

tuxTQspaceName	R/W*	Name of the application queue space
tuxTQspaceQmConfig	R/W*	Absolute pathname of the file/device where queue space is located
tuxTQspaceLmid	R/W*	Local Machine where queue space is located
tuxTQspaceGrpNo	R/W*	Group No of any server group for which this group is resource manager
tuxTQspaceState	R/W	SNMP GETs can return inactive(1), initializing(2), open(3), or active(4). SETS can set to open(3), cleaning(5) or invalid(6). Setting to invalid will delete the queue space
tuxTQspaceBlocking	R/W	Blocking factor. Default is 16
tuxTQspaceErrQname	R/W	Name of error queue associated with this table space.
tuxTQspaceForceInit	R/W	Do we initialize disk pages on new extents (yes(1), no(2).
tuxTQspaceIpckey	R/W	IPC key used to access queue space in shared memory
tuxTQspaceMaxMsg	R/W	The max numbers of messages this queue space can contain
tuxTQspaceMaxPages	R/W	Max disk pages for all queues in this queue space
tuxTQspaceMaxProc	R/W	The max number of processes that can attach to queue space
tuxTQspaceMaxQueues	R/W	The max number of queues
tuxTQspaceMaxTrans	R/W	The max number of simultaneous active transactions
tuxTQspaceCurExtent	R/O	The current number of extents used by queue space (< 100)
tuxTQspaceCurMsg	R/O	The current number of messages on all queues in queue space
tuxTQspaceCurProc	R/O	The current process accessing the queue space
tuxTQspaceCurQueues	R/O	The current number of queues
tuxTQspaceCurTrans	R/O	The current number of outstanding transactions
tuxTQspaceHwMsg	R/O	The highest number of messages in queue space since it as last open or cleaned
tuxTQspaceHwProc	R/O	The highest number of processes simultaneously attached to the queue space since it as last open or cleaned
tuxTQspaceHwQueues	R/O	The highest number of queues in queue space since it as last open or cleaned
tuxTQspaceHwTrans	R/O	The highest number of transactions in queue space since it as last open or cleaned
tuxTQspacePercentInit	R/O	The percentage of disk space initialized for the queue space.

- Values that can only be set when a new row is created

TuxTappQTbl - Application Queue

The application queue table is made up of a sequence of TuxTappQEntry. This MIB group can be used to add a new queue or modify certain attributes of an existing queue.

Creating a New Queue

The attributes TuxTAppQname, Tux,TappQspaceName, TuxTAppQmConfig, tuxTAppQlmid and TuxTAppQgrpNo can only be modified when a new row (queue) is added and must then be specified.

Deleting an Existing Queue.

An existing queue can be deleted by setting the TuxTAppQstate attribute to invalid

Modifying an Existing Queue

The following attributes can be modified for an existing queue by issuing a SNMP SET.
State (TuxTAppQstate), Order (TuxTAppQorder, High Water Mark (TuxTAppQcmdHw), Low Water Mark (TuxTAppQcmdLw) High/Low Water Mark Command (TuxTAppQcmd), Max Number Retries (uxTAppQmaxRetries), Out of Order Processing (TuxTAppQoutOforder) and Retry Delay (TuxTAppQretryDelay

Each TuxTappQEntry row has the following attributes:

TuxTAppQname	R/W*	Name of application queue
TuxTAppQspaceName	R/W*	Queue Space
TuxTAppQmConfig	R/W*	Absolute path of Queue space Device
TuxTAppQlmid	R/W*	Identifier of the logical machine where the application queue space is located.
TuxTAppQgrpNo	R/W*	Group No of any server group for which this queue is resource manager
TuxTAppQstate	R/W	Valid(1), invalid(2). Issuing a SET with invalid deletes the specified queue
TuxTAppQorder	R/W	Order in which messages on the queue are to be processed. Valid values are PRIO, FIFO or time. Default is FIFO
TuxTAppQcmd	R/W	The command to be executed when high water mark is reached
TuxTAppQcmdHw	R/W	The queue high water mark
TuxTAppQcmdLw	R/W	The queue low water mark
TuxTAppQmaxRetries	R/W	Max number of retires for a failed queue message. When exhausted message is place on error queue. Default is 0.
TuxTAppQoutOforder	R/W	The way in which out of order message is to be handled (valid values are none(1) top(2), msgid. Default is none
TuxTAppQretryDelay	R/W	The delay in seconds between retries for a failed message. Default is 0.
TuxTAppQcurBlocks	R/O	The number of disk pages currently consumed by the queue
TuxTAppQcurMsg	R/O	The number of messages currently in the queue.

- Values that can only be set when a new row is created

13.8.1 TuxTAppQmsg – Messages

The TuxTAppQmsgTbl made up of on or more tuxTAppQmsgEntry. The values returned by this MIB are controlled by tuxTAppQctrl.

Creating a New Message

New Message rows cannot be created

Deleting an Existing Message

An existing message can be deleted by setting the tuxTAppQmsgState attribute to invalid

Modifying an Existing Message

A message can be moved to a new queue(tuxTAppQmsgNewQname), it priority changed (tuxTAppQmsgState) or order (tuxTAppQmsgPrior) or time (tuxTAppQmsgTime) of processing altered if applicable.

Each Row has the following attributes

TuxTAppQmsgId	R/O	Message identifier
tuxTAppQmsgSerNo	R/O	
tuxTAppQmsgGrpNo	R/O	Server Group No
tuxTAppQmsgQname	R/O	Application Queue Name
tuxTAppQmsgQmConfig	R/O	Device Holding Queue Space
tuxTAppQmsgQspaceName	R/O	Queue Space Name
tuxTAppQmsgLmid	R/O	Logical Machine ID holding Queue Space
tuxTAppQmsgState	R/W	Message Status Valid (1) invalid(2). Setting the value to will delete the message from the queue
tuxTAppQmsgNewQname	R/W	Name of Queue into which to move message
tuxTAppQmsgPrior	R/W	Priority for the message. Valid only for Priority based queues
tuxTAppQmsgTime	R/W	The time when the message will be processed. Valid only for TIME based queues
tuxTAppQmsgCorId	R/O	Correlation Identifier provided by tpenqueue
tuxTAppQmsgCurRetries	R/O	Number of Retries attempted for this message
tuxTAppQmsgSize	R/O	Size of the message in Bytes.

13.8.2 TuxTQtransTbl – Transactions

This is primarily a read only group. The only attributes of a queued transaction that can be set is Transaction state which can be set to either aborted or committed.

TuxTQtransXid	R/O	The transaction identifier
tuxTQtransIndx1	R/O	Transaction index
tuxTQtransIndx2	R/O	Transaction index
tuxTQtransIndx3	R/O	Transaction index
tuxTQtransIndx4	R/O	Transaction index
tuxTQtransIndx5	R/O	Transaction index
tuxTQtransGrpNo	R/O	The group number of server group for which queue space is resource manager
tuxTQtranSpaceName	R/O	Queue Space Name
tuxTQtransQmConfig	R/O	Path of device holding the Queue Space
tuxTQtransLmid	R/O	The Machine holding the Queue Space

tuxTQtransState	R/W	Transaction State. GETS can return actdive(1), abort-only(2), aborted(3), com-called(4), ready(5), decided(6), suspended(7). SETS can set habort(8), hcommit(9)

13.9 None Queued Transactions

The BEA MIB can be used to track transactions through the TUXEDO core components (servers, messages, queues etc). The TUXEDO MIB also provides a transaction table (tuxTranTbl) that is managed by the tux_snmpd agent and represents the runtime attributes of active transactions within the application.

The transaction table is made up of one or more rows. Each row in this table is identified and/or indexed using the attributes TuxTranCoordLmid, TuxTPTranId, tuxTranXid, tuxTranIndx1, tuxTranIndx2, tuxTranIndx3, tuxTranIndx4, or tuxTranIndx5.

TuxTranCoordLmid	R/O	Logical machine identifier of server group coordinating the transaction.
TuxTranState	R/W	This attribute can hold the values of active(1), abort-only(2), aborted(3), com-called(4), ready(5), decided(6) and suspended(7). SNMP sets can be used to set the value only to aborted.
TuxTranTimeOut	R/O	Time left in seconds before transaction times-out.
TuxTranGrpCnt	R/O	Number of groups that participate in transaction
TuxTranGrpNo	R/O	Group number of participating group
TuxTranGState	R/W	Group status can be active(1), aborted(2), rd-only(3), ready(4), hcommit(5), habort(6), done(7) . SNMP sets can be used to set the value to either hcommit(5) orhabort(6)

13.10 List of Attributes to be Polled for User Defined Traps Mechanism

User defined traps are specified by adding a RULE_ACTION to the BEA Manager configuration file (beamgr.conf). These rules can poll any attribute in any MIB accessible from the agent integrator. In the EIA architecture this means that it can poll any attribute in either the BEA or HP (peer agent) mib.

RULE syntax

RULE_ACTION <name> <polling frequency in seconds> <condition> <action>

Example (CPU > 80% busy)

RULE_ACTION cpu 600 if (VAL(140.11.1.0) > 80) { TRAPID_ERR = 104 TRAPID_OK = 105 }

Name

Must be unique and less than 8 characters

Condition

Can be made up of one or more Boolean operation. Each operation has the basic format (VAL(OID) relation value). Operations are combined using the OR (||), AND (&&) operator. Relationships may be expressed using any of the following operators (==, != , <, >, >=, <=).

Action

There are four action types;

1. TRAPID_ERR = specific trap number. This trap generated when state of rule transitions from OK to ERR
2. TRAPID_OK = specific trap number. This trap generated when state of rule transitions from ERR to OK

3. COMMAND_ERR = “command” Where command is the path for a executable or script (e.g. /usr/errscript.sh).
4. COMMAND_OK = “command”

Traps pass the following information.

- The user defined trap number
- Rule name and state change
- Enterprise ID (OID)

13.11 Changing Rules Dynamically using SNMP (beaIntAgt)

The agent integrator support the MIB group beaIntAgt – The BEA intelligent agent group. This group contains the beaIntagtTable.

Each row in this table specifies an individual RULE_ACTION. All of the attributes in this table are read write and SNMP (either direct from Openview Node Manager or using Snmpwalker) can be used to add, delete or modify RULE_ACTIONS.

Each Row is made up of the following attributes:

BeaIntAgtRuleId	The name of the rule
BEAIntAgtScanIntvl	Polling interval in seconds
BeaIntAgtRuleAction	The RULE, condition and action (e.g. “if (VAL(140.11.1.0) > 80) { TRAPID_ERR = 104 TRAPID_OK = 105”)
BeaIntAgtStatus	active, invalid or inactive. If set to invalid the rule is deleted. If set to inactive polling is disabled but row not deleted.

13.12 UNIX RULES

These allow us to raise a trap if any of the underlying UNIX system resource is over utilized or nearing exhaustion.

```
# If the CPU is more than 80% busy
RULE_ACTION cpu 600 if ( VAL(140.11.1.0) > 80 ) { TRAPID_ERR = 104  TRAPID_OK = 105 }

# If any disk capacity in use is greater than 90%,
RULE_ACTION df 600 if ( VAL(140.2.22.1.5.*) > 90 ) { TRAPID_ERR = 102   TRAPID_OK = 103 }
```

13.13 TUXEDO COMMUNICATION RULES

These detect errors encountered by TUXEDO when attempting to communicate between machines within the same domain. When the Master machine cannot reach a machine its status is set to partitioned and remains partitioned until communications are re-established.

Partitioning can occur because of machine and network failures or slow downs and also if the BBL or BRIDGE process are slow or inactive.

Note: Many of these partitioning conditions are already covered by the 32 standard SNMP traps. Consequently the value of generating additional TUXEDO communication user defined traps must be questioned.

```
# Check if any machine the domain got partitioned. (tuxTmachineState)
RULE_ACTION mcState 60 if ( VAL(140.300.5.1.1.6.*) == 3 ) { TRAPID_ERR = 302 TRAPID_OK = 303 }
```

13.14 TUXEDO APPLICATION RULES

These allow us to monitor the health of the components within the TUXEDO application.

```
# Alert if any server group is not active. (tuxTgroupState)
 RULE_ACTION grpState 60 if ( VAL(140.300.4.1.1.4.*) != 1 ) { TRAPID_ERR = 300 TRAPID_OK = 301 }

# Check if TMSYSEVT is active. (tuxTsrvrName & tuxTsrvrState)
RULE_ACTION sysevtUp 60 if ( (VAL(140.300.20.1.1.3.*) >= "TMSYSEVT") &&  (VAL(140.300.20.1.1.5.*) != 1) )  { TRAPID_ERR = 304 TRAPID_OK = 305 }

#  Monitor server queue size. (tuxTsvcSrvrNqueued)
RULE_ACTION srvrQsz 60 if (VAL(140.300.10.2.1.15.*) > <THRESHOLD>) { TRAPID_ERR = 308 TRAPID_OK = 309 }
```

13.15 TRANSACTION RULES

These rules allow us to monitor transactions as they flow through the TUXEDO application.

```
#  Monitor Transaction States e.g. aborted(3).  (tuxTranState)
RULE_ACTION tranState 60 if (VAL(140.300.23.1.1.9.*) == 3)  { TRAPID_ERR = 306 TRAPID_OK = 307 }
```

13.16 APPLICATION QUEUE RULES

Application queues (/Q) lies at the heart of the EIA architectural design.

```
# Monitor number of messages in  application queues. (tuxTAppQcurMsg)
RULE_ACTION appqMsgs 60 if (VAL(140.300.12.1.1.15.*) > <THRESHOLD>) { TRAPID_ERR = 310 TRAPID_OK = 311 }
```

13.17 List of Attributes to be Monitored to Provide Performance Metrics

Performance metrics are to be gathered by a housekeeping process and logged to Measureware using DSI. The housekeeper periodically issue a SNMP Get (via Simpwalk) against predefined MIB attributes.

Which BEA MIB attributers should be polled to provide performance metrics;

1. Candidate attributes should provide a value that represents some aspect of the performance of either the Operating System or the TUXEDO application
2. The DSI interface to Measureware requires a single integer value. Consequently tables of values cannot be used
3. Candidate attribute should have a stable predetermined identifier. The housekeeping process will be configured to request values for a finite list of attributes that must has a known unique identifier. So for example rows from the beaPsTable cannot be used since there identifier (process id) will vary and cannot be predetermined.

These requirements greatly restrict the number of candidates that can be identified from the nearly 800 attributes of the BEA Mib. It should be noted that additional performance metrics could also be raised against attributes in the HP MIB – but this is outside the scope of the current study.

13.18 UNIX Attributes

Most of the UNIX attributes in the BEA MIB are in the form of tables .

Process Table

The process table (beaPsTable) provides attributes for all of the processes running on the managed node. It provides 2 potentially useful measures of performance. These are

1 beaPsCpu - % of CPU being used by each process
2 beaPsMem % of real memory being used by each process

Unfortunately the identifier for this table is process id which cannot be predetermined.

File System Tables

Two Mib tables provide information about file system usage. The MIB table beaDfTable provides details of each of the file systems mounted on this host. The Table BeaLclDfTable provides details of the attributes of the local file systems. Both of these tables provide a measure of the percentage of the filesystem in use. These are

1 BeaLclDfTable(beaLclDfCapacity - %filesystem in use)
2 BeaDfTable.beaDfCapacity

Each of the entries in these tables can be identified using the name of the filesystem

IPC Utilization

A number of BEA MIB tables provide details about IPC resource usage on the managed node (BeaMqTable ()
BeaShmTable(), BeaSemTable()). None of these provide attributes that could be used to measure system performance, and furthermore each is indexed by a none predetermined identifier (eg BeaMqId).

The beaShmgrtable lists the attributes of each page in shared memory. While these are identified by a none determinate index (beaSmgrIndex) it does provide some attributes that might be of interest

1 BeaSmgrAllocated – provides an indication of shared memory status and will be set to no (1) if the share memory is corrupted or absent.
2 BeaShmgrPctShmUsed – provides the percentage o entries in shared memory that are in use
3

The System Performance Group (beaSysPerf)

This group provides a measure of system performance. Many of these are individual measures (rather than table entries). Some of these scalar values are cumulative and represent the total value since TUXEDO was last rebooted. However this group also provides a delta attribute for many of these that specifies change since the last poll (SNMP GET).

1. beaSysPerfCpu – Percentage of CPU capacity utilized between polls
2. BeaSysPerfDiskDelta – Disk traffic in number of transfers in blocks since the last poll
3. BeaSysPerfIntrDelta – No device interrupts since the last poll
4. BeaSysPerfLoadDelta – Size of run queue since the last poll
5. BeaSysPerfPageDelta – Paging Activity in number of pages since the last poll.

The BeaSysPerf group also provides a BeaSysPerfIfTable that lists the attributes of each of the physical interfaces to the system. Again each of these rows is identified by a none deterministic index, but provides counts of the number of packets received and sent form an interface that could be summed across all interface (table rows).

13.19 APPLICATION QUEUE Attributes

The Tuxedo Application Queues (/Q) lie at the heart of the EIA Tuxedo Domain. All Transaction pass through the various system /Q's and system performance can be readily monitored via the TuxTAppQ Mib Group.

The TuxTAppQ Mib group is made up of 5 sub groups. For our purposes the mose interesting of these is likely to be the Queue Space Table. For the each machine in the EIA project there will be only one queue space. In contrast there will at least 6 queues per queue space.

Therefore if we want to query scalar rather than tabular data the TuxTappQSpaceTbl is likely to provide the most useful information. This table is likely to contain only one row. Several attributes of the TuxTappQSpaceTbl rows are likely to be of interest when trying to gather performance metrics. These are:

1. tuxTQspaceCurMsg The current number of messages
2. tuxTQspaceCurProc The current number of processes
3. tuxTQspaceCurTrans The current number of outstanding transactions
4. tuxTQspaceHwMsg The highest number (high water mark) of messages in the queue space since it was last opened or cleaned.
5. tuxTQspaceHwTrans The highest number (high water mark) of transactions in the queue space since it was last opened or cleaned.