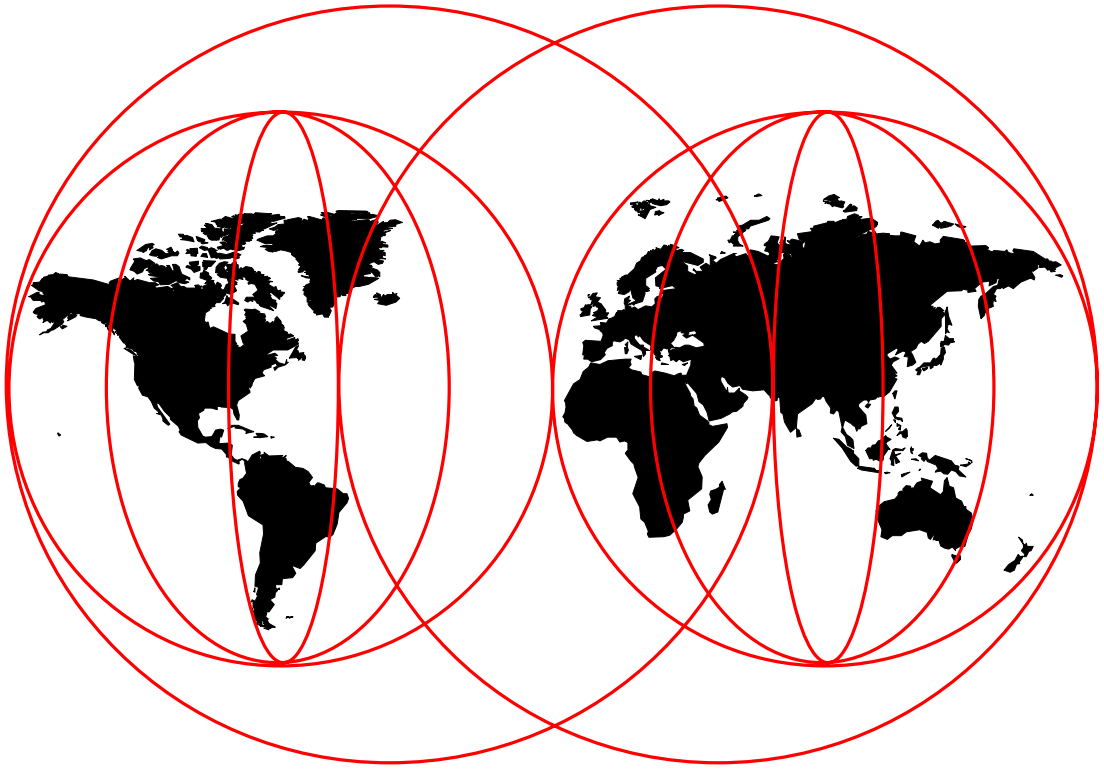
IBM

TEC Implementation Examples

Paul Fearn, Raj Chityal, Nancy Jarin, Elise Kushner, Gordon Lilly, Darren Pike



International Technical Support Organization

http://www.redbooks.ibm.com

Tivoli

SG24-5216-00

International Technical Support Organization

SG24-5216-00

TEC Implementation Examples

May 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 311.

First Edition (May 1998)

This edition applies to the Tivoli Framework Version 3.2, the TEC Version 3.1 and Distributed Monitoring Version 3.5 for use with the AIX Version 4 and upwards and NT Version 4.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

Preface

This redbook will help you install, tailor and configure Tivoli Enterprise Console (TEC) software and also show how to customize the Tivoli Management Environment applications such as tasks, Distributed Monitoring and TME 10 NetView showing how these applications can integrate with the TEC.

By using a real-life customer example we discuss the planning stages through to implementation showing practical examples based on a set of specific customer requirements.

This redbook will be especially helpful to those who need to implement the Tivoli Enterprise Console, and develop specific TEC rules for event correlation in order to provide an event management solution.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the Systems Management and Networking ITSO Center, Raleigh.

Paul Fearn is a Systems and Networking specialist at the Systems Management and Networking ITSO Center, Raleigh. He writes extensively and teaches classes worldwide on all areas of systems management. Before joining the ITSO two years ago, Paul worked in the IBM UK services department as a Systems Management Consultant.

Raj Chityal is an IBM and Tivoli Consultant working for the IBM UK Services department. He has several years' experience in the systems management consultancy area.

Nancy Jarin is a Systems Management Specialist for IBM Services in Canada working with the Tivoli applications.

Elise Kushner works for an IBM business partner based in Germany. She has several years' experience with the Tivoli applications and teaches Tivoli education in Europe.

Gordon Lilly is a Tivoli Systems Management consultant based in the UK working with UK customers implementing Tivoli solutions.

Darren Pike is an IT specialist working for Canadian Tire based in Toronto. He has a number of years' experience in the network management field.

Thanks to the following for their invaluable contributions to this project:

Canadian Tire

Dave Thoenen
IBM Consultant, Raleigh

Sean Starke
Tivoli US, Austin

Kathryn Casamento
The ITSO Editing Team

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 321 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com/`
  For IBM Intranet users `http://w3.itso.ibm.com/`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Chapter 1. The Tivoli Enterprise Console

The Tivoli Enterprise Console (TEC) is one of the Tivoli enterprise management applications. The TEC is mainly responsible for event management and automation. The events can be generated from various resources such as network devices servers and applications. The TEC also provides the console from which operations personnel can view the events and perform specific management tasks.

The TEC is a focal point for managing, correlating, and automating responses to events from a wide variety of sources in a distributed environment. TEC administrators and operators are defined using platform services, and the operators start their event consoles through the desktop provided by the platform. TEC delivers a default library of tasks that may be used to handle incoming events, making use of the scheduling services offered by the platform. In addition, you can add your own tasks to enhance the TEC environment.

One of the key features of the TEC is the ability to provide the filtering and event correlation facilities to reduce the number of events that the operators will see.

This redbook focuses on the installation, integration and customization of the TEC application. The following sections introduce the software applications referenced in this publication.

1.1 The Tivoli Management Environment (TME 10)

The Tivoli Management Environment is a collection of software products used for systems management in a heterogeneous client/server environment.

We briefly define the functional highlights and most important concepts. For a more detailed description see the *TME 10 Planning and Installation Guide*.

The Tivoli Management Platform (TMP) consists of a framework that provides administrative and desktop services, and a set of application services, such as scheduling, maintenance of configuration profiles and object database query facilities. The whole package is commonly referred to as the *platform*. The platform is a foundation for all of the other TME 10 products.

A description of all the TME components and latest releases are located on the Tivoli home page at //www.tivoli.com.

1.1.1 Tivoli Management Regions, Server and Clients

An installation can be divided logically into a set of regions, each with its own TME server for managing its TME clients.

Independent of the operating system platform, each TME server has a TMR database, and is started with the oserv daemon. These TMRs can then be connected to each other, enabling the coordination of activities across the network, while still maintaining local control over individual regions.

A TMR does not have to correlate to a particular physical or geographical region, but can be any collection of resources that makes business sense.

However, TMR boundaries are not represented in the desktop used by a TME administrator, but rather an administrator has responsibility for resources in one or more policy regions. Operations on these resources appear to the administrator as local operations, regardless of the number of TMRs or how they are connected to each other.

You can connect TMRs in one of two ways:

- In a one-way connection, one TMR knows about the resources on the other, but the other TMR has no information about the first TMR. One server can be the manager of the resources in both TMRs, useful in a centralized environment.
- In a two-way connection, the two TMRs share information with each other, so each can manage the other's resources. This is useful in a more decentralized configuration.

Resources in a TMR are given registered names when they are created and those names are stored in the TME Name Registry (TNR). There is a directory service that is responsible for registering and unregistering names and looking up referenced resources. One TNR oversees object names for one TMR. When two TMRs are connected, the TNRs exchange information. This is very efficient, because applications can then do lookups and make lists of resources without knowing in what TMR the resources are located and without having to make contact with TNRs in remote TMRs.

If you plan to use event consoles in a remote TMR, then you will need to establish a two-way connection to that TMR.

1.1.2 Resources

TME *resources* are the systems, devices, services, and facilities in a distributed environment that need managing. Examples of these would be nodes, administrators and software.

1.1.3 Profiles

A *profile* is common configuration information for a collection of resources. For example, you might divide your users into several collections, based on department or location or responsibilities. Then you would create a profile for each group, containing all of the user information in a platform-independent format. Profile managers are responsible for distributing any changes in this information to subscribers, or profile endpoints. In this way, you can centrally manage changes to a heterogeneous set of systems, by grouping those systems according to profiles.

1.1.4 Policy and Policy Region

A *policy* is a set of business rules and conventions governing your resources, for example, which users have access to which machines. TME 10 allows you to define multi-level policies for the different areas in your environment, and to enforce those policies.

A *policy region* is a collection of resources subject to a common set of policies, for example, all workstations in a given company department.

When you install TEC, it defines its own policy region to govern TEC resources. This allows you, for example, to assign permission for using the event server

specifically to certain administrators. Contained by default in this policy region is the resource called the TEC task library, containing predefined TEC tasks.

1.1.5 The TME Desktop

The TME desktop is a graphical user interface (GUI), which is customized specifically for each administrator. Each administrator sees only those TMRs, and only those resources, for which he or she has been authorized.

The TME desktop is the point from which TME applications are installed and configured. Each administrator who is defined to use an event console opens that console from an icon on the TME desktop.

1.1.6 Tasks, Jobs and Scheduling

An administrator creates a task by defining an executable (a shell script, compiled program, or any other type of valid executable) and providing a path to it. This task is then added to the task library and displayed as an icon. The proper authorizations also have to be set up so that tasks can have access to resources based upon an administrator's rights.

Jobs are simply tasks with a predefined set of managed node subscribers, meaning a group of nodes on which the jobs should be run. These jobs can be executed by double-clicking on their icon, or they can be dragged-and-dropped to the scheduler icon to be run at a later time.

The scheduler allows operations to be performed at a given time or with a given frequency. There is one scheduler per TMR, but it can initiate jobs on systems outside of its own TMR.

TEC comes with a ready-made set of tasks, useful in a variety of everyday situations, for example, clearing closed events from the TEC database and notifying an administrator by e-mail. You can automate these tasks so that they are triggered by specific events or you can use the TME scheduler to run them at regular intervals.

1.2 The Role of the TME 10 Enterprise Console

The TEC is a TME 10 application, running on the TME 10 Framework, which manages events from a wide variety of sources in a client/server environment. The TEC allows you to handle events according to priorities you have determined, correlate events from one source with those from another source, apply decisions about those events through the use of rules, take corrective actions, and inform the right people when required.

Every time the status of a resource changes in any way, an *event* occurs. If this event is important enough to warrant attention, or if it needs to be correlated with events from other sources and therefore cannot be fully processed at its local site, then the event can and should be forwarded to a central event server. Applications can be written to issue events when abnormal conditions occur and thus drive TEC to take some pre-defined action. Each event is identified by a class name defined by the respective event adapter.

1.3 The TEC Components

The foundation of TME 10 solution is the TME 10 Framework, an object-oriented, cross-platform software infrastructure designed to provide common services to the TME 10 system management applications.

The TME 10 Enterprise Console is part of TME 10, a comprehensive suite of interoperable software products for enterprise client/server management. To have a good understanding of any TME 10 application, a knowledge of the TME 10 Framework is essential.

The TME 10 Enterprise Console (T/EC) is a systems management application which provides the ability to manage the events coming from a wide variety of resources such as systems, network, database and application. It provides rule-based event correlation for integrating network, systems, database and applications management. It offers a centralized, global view of a distributed computing enterprise, while ensuring high availability of applications and computing resources.

The T/EC is comprised of the following components:

- Events or messages
- Event adapters
- An event server
- Event consoles
- Relational Database Management (RDBMS) RIM

1.3.1 Events

Events are the central unit of information within the T/EC. An event is any significant change in the state of a system resource, application or network resource that is detected by *event adapters*. The content of an event is referred to as an *event message*.

As an example from a UNIX system, if a user attempts to log on to a UNIX system to perform an action on the system and this action fails due to the security configuration, then an event will be generated providing the TEC with information pertaining to the specific incident.

1.3.2 Event Adapters

An event adapter is an application at or near the source of the managed entity that converts the event information to a standard format and sends it to the event server. The event adapter can also filter out insignificant events, so that network traffic is not needlessly generated. An event adapter can report to multiple event servers.

An event adapter can get its information from the source in several ways by:

- Receiving messages from a source that actively produces messages.
- Checking a file at regular intervals, where a source writes information to a file.
- Running commands or programs at regular intervals, then interpreting the information and forwarding it to the event server.

An event adapter is a software process that detects events, formats these events and sends the formatted events to the *event server*. Event adapters transform the information they receive by parsing and restructuring the information before sending it on the event server in the form of an event. The following event adapters are part of the TEC software:

- IBM AS/400 Alert Adapter
- IBM AS/400 Message Adapter
- Logfile Adapter
- IBM NetView/6000 for UNIX Adapter
- HP OpenView Adapter
- SNMP Adapter
- SunNet Manager Adapter
- Microsoft Windows NT Adapter

The event adapter is installed on the host machine that contains the system resource or application that will be monitored. It is possible to install more than one adapter on a host.

T/EC comes with several event adapters. Third parties also have written event adapters for their products. The customer can develop custom adapter using the Event Integration Facility (EIF).

1.3.3 Event Server

The event server manages all the events received from event adapters and evaluates the events against a set of rules to determine if it requires actions to be performed, such as modifying the event or discarding the event. Any event that passes the criteria is then forwarded to the *event console*.

When an event occurs in the enterprise, it travels to the event adapter on a host that translate these events into syntax that the event server can understand. Event adapters can send events to the event server in two ways:

- *Secure* interfaces establish a connection using services provided by the TME 10 Framework.

  A secure event adapter sends events through the object request broker (oserv) on that managed node, to the oserv on the T/EC server and then to the server process (tec_server).

- *Unsecure* interfaces establish connections using standard interprocess communication mechanisms. You choose the type of connection for your environment when you install the adapter.

  An unsecure adapter will send events from the adapter directly to the reception process.

1.3.4 Event Console

Event consoles allow users to monitor incoming events. Which user sees which events is based on event sources (the types of applications originating the events) and event groups (administrator-defined criteria for logically grouping events).

By default, only one administrator can open an event console at a time, but you can create multiple event consoles for multiple administrators with the wcrtconsole command.

The event console is the graphical user interface (GUI) that displays event information appropriate for specific administrators. It also lets the administrator respond to events and to perform tasks from the event console. It is also configurable to allow easy separation and assignment of maintenance tasks to the appropriate administrators.

The event console appears as an icon on the TME desktop and must run from a managed node.

1.3.5 Relational Database Management System (RDBMS)

The TEC uses an RDBMS for storing information about events received and what tasks have completed.

TEC 2.6 contained a run-time version of Sybase for this function.

TEC 3.1 does not contain an RDBMS, but will run with both Sybase and Oracle.

Two of T/EC event server components, the *reception log* and *event repository* are RDBMS tables accessed through the Repository Interface Module (RIM). RIM gives the customer the ability to choose and connect to a number of RDBMS systems. The TEC will run on the same machine as the RDBMS application. However, if the the RDBMS is running with an RDBMS client, then the client software must be installed on the same host as the event server for the Enterprise Console to function correctly.

1.3.6 TEC Rules

The TEC uses rules to correlate and analyze events. The TEC rules engine, which is a rule-based event processor, uses a set of rules to determine if an action needs to be performed on an event. The rule engine is responsible for:

- Finding applicable rules for a given event
- Controlling the execution of applicable rules

Rule describes the action that should be performed when a particular event is received by the event server. Rules are executed based on their order of definition. Rules are used to assess the received event and determine the appropriate actions to perform and to pro-actively address situations before they become problems. Some of the actions available to rules are:

- Change the status of an event
- Change a message in an event
- Upgrade or downgrade the severity of an event
- Correlate events

- Execute commands, tasks or programs
- Discard events
- Filter duplicated events
- Generate other events
- Forward events
- Set timers on events

They are written using a high-level interpreter language that interfaces to the Prolog programming language and are grouped together into rule sets.

1.3.7 The TEC Rule Sets

Rule sets are a list of rules grouped together and stored in one file. Some key points about a rule set are:

- It is made up of one or more rules.
- It is independent of all other rule sets.
- It is found as a single file in the TEC_RULES directory and can be active or inactive. If designated as being active, it will be processed by the event server when its associated *rule base* is loaded into the event server.

1.3.8 The TEC Rule Base

The rule base holds two types of information: *the event class*, which defines the structure of incoming events and *the event rules*, which define which actions to take when specific events are received.

1.4 TME 10 Distributed Monitoring

The Distributed Monitoring application provides the ability to define resource monitors for specific managed systems and use the TME framework to distribute and activate these monitors. It is fully integratable with the TEC.

Distributed Monitoring is delivered with a pre-defined set of monitors including:

- UNIX server monitors
- Windows NT monitors
- Database monitors for Oracle and Sybase

TME 10 Distributed Monitoring can also be considered event adapters when they send events to T/EC. It provides a set of event class definitions that format the information so that T/EC can interpret events.

Distributed Monitoring can monitor UNIX, NT and NetWare systems. Although the TME 10 provides a consistent way to invoke monitoring, regardless of the system platform, the monitors themselves differ from one platform to another. Distributed Monitoring reflects these differences by placing monitors on groups called *monitoring collections*. Each collection contains a group of related monitor definitions.

Distributed Monitoring must be installed on the TMR server and all TMR clients. The Distributed Monitoring collections should only be installed on the TMR server.

1.5 T/EC Toolkit and Utilities

There are toolkits and utilities that allow you to extend the T/EC by configuring and adding your own event adapters to monitor and report on events in your enterprise.

1.5.1 Event Integration Facility (EIF)

The EIF is a toolkit that allows the customization of system information and events you can monitor and manage using the T/EC. It also allows the creation of new event adapters.

The EIF must be installed on the same machine as the T/EC console.

For more information on EIF, you may refer to *Tivoli TME 10 Event Integration Facility User's Guide*.

1.5.2 Adapter Configuration Facility (ACF)

The ACF provides a graphical user interface (GUI) for customizing and configuring event adapters more easily.

The ACF must be installed on the TMR server and all clients to which adapter configuration profiles will be distributed.

1.5.3 Logfile Format Editor

The Logfile Format Editor also called Logfile Configuration Facility provides a GUI for editing and adding definitions for the logfile event messages that will be sent to the TEC.

1.5.4 TME 10 NetView for AIX

TME 10 NetView Version 5 is the software that manages the TCP/IP network devices and is capable of managing a large number of nodes. Some of the key features include integration with the TME Framework, automatic discovery of TCP/IP networks, displaying dynamic status for all discovered devices and providing a building block for third-party management applications.

TME NetView for AIX was used mainly for SNMP management of our network devices. All SNMP traps are first sent to the NetView server and then forwarded to the TEC.

There is some confusion when referring to SNMP management using adapters. There are a number of adapters that will provide the same function, that is, forwarding SNMP events to the TEC. The SNMP, HPOV and NV6K adapters function in the same way. For Versions 4 and 5 of TME 10 NetView a more comprehensive adapter was developed to allow more customization. The specific information sent to the TEC can be modified prior to the event being sent.

We used the nvserverd adapter for all our SNMP management as we were using NetView as the SNMP/TCPIP management tool. Using the old adapters limits the amount of customization and facilities that the nvserverd adapter provides.

The NV6K and OpenView adapter will send events directly from the NetView trap file, before any NetView event formatting so that the event sent to the TEC is in raw SNMP format.

For more information on the NK6K and the OpenView adapter see the *Enterprise Console Adapters Guide, Version 3.1*.

Chapter 2. Planning for the TEC Installation

Due to the complexity of the TEC implementation it is highly recommended to perform a planning exercise before installing the software. This chapter explains the stages we performed prior to our TEC installation.

We use a real-life customer environment to show how the TEC software can be implemented, and provide examples on what TEC customization took place. We worked closely with the customer and document each stage of the implementation.

This chapter also provides guidance and pointers to assist with installing the TEC, providing some areas of consideration before implementing the TEC software. The following sequence highlights the stages we performed prior to our installation.

Requirements Here we briefly document the customer requirements and document any project assumptions. We also list what software the customer was already using for management purposes.

Design We define the platforms and management software that we install and list some of the pre-installation considerations.

Implementation We list each of the installation tasks we perform in order to install and customize the management solution.

The customers main requirement was to limit the number of events from being sent to the management application. Also the customer wanted to provide a suitable management solution that would provide certain management functionality. The sections below outline the processes we took in implementing the solution.

2.1 Requirements

This section deals with the specific customer requirements for implementing the TEC. This section also provides us with the information we need to plan what software components we require and where the TME software will be installed. The process to define the customer requirements are out of the scope of this redbook, however you can see the assumptions we made.

The customer identified the management resources that they wanted included for management. These are listed in Table 1. We also list the current tools used that manage the environment.

Table 1. Devices to Be Managed

Managed Element	**Existing Management Software**
AIX Servers	No current management
NT Servers	No current management
Cisco Routers	NetView for AIX Version 4
Caletron Hubs	NetView for AIX Version 4
NetWare Servers	Managewise
IBM Network Devices (8272, 8250)	NetView for AIX Version 4

The existing environment was managed using NetView and receives SNMP traps from all the devices. These devices can potentially generate a large number of events. The customer wanted to install an enterprise management system to manage all the devices with the additional management of the AIX and NT servers. With this information the management software was selected.

2.1.1 Management Software

We did not have to provide any additional management software for the SNMP-managed devices such as the routers and hubs. An upgrade to the NetView application was also requested. The following list shows which applications were selected:

- TEC for the Event Management System
- TME 10 NetView for AIX Version 5 for all SNMP Management
- Distributed Monitoring for the AIX and NT servers
- TME 10 core products such as tasks and scheduler

The current NetView management system did not provide the functionality to monitor resources on the AIX and NT server. For this reason we assume that the TEC logfile adapter and Distributed Monitoring will provide these functions. The types of resource we monitor include operating system processes, checking of file system sizes and availability of applications such as the Relational Database Management System (RDBMS).

The customer had also decided to use RISC System/6000 machines in order to install the management applications. These are shown below.

- RISC System/6000 for the TMR server
- RISC System/6000 for the TEC server
- RISC System/6000 for the Oracle server
- RISC System/6000 for the TME 10 NetView server

For the purpose of this redbook we also install Sybase on the TEC server and install TEC on an NT server as an alternative to the RISC System/6000 server.

The server configuration that will provide the layout is reflected in Figure 1 on page 13.

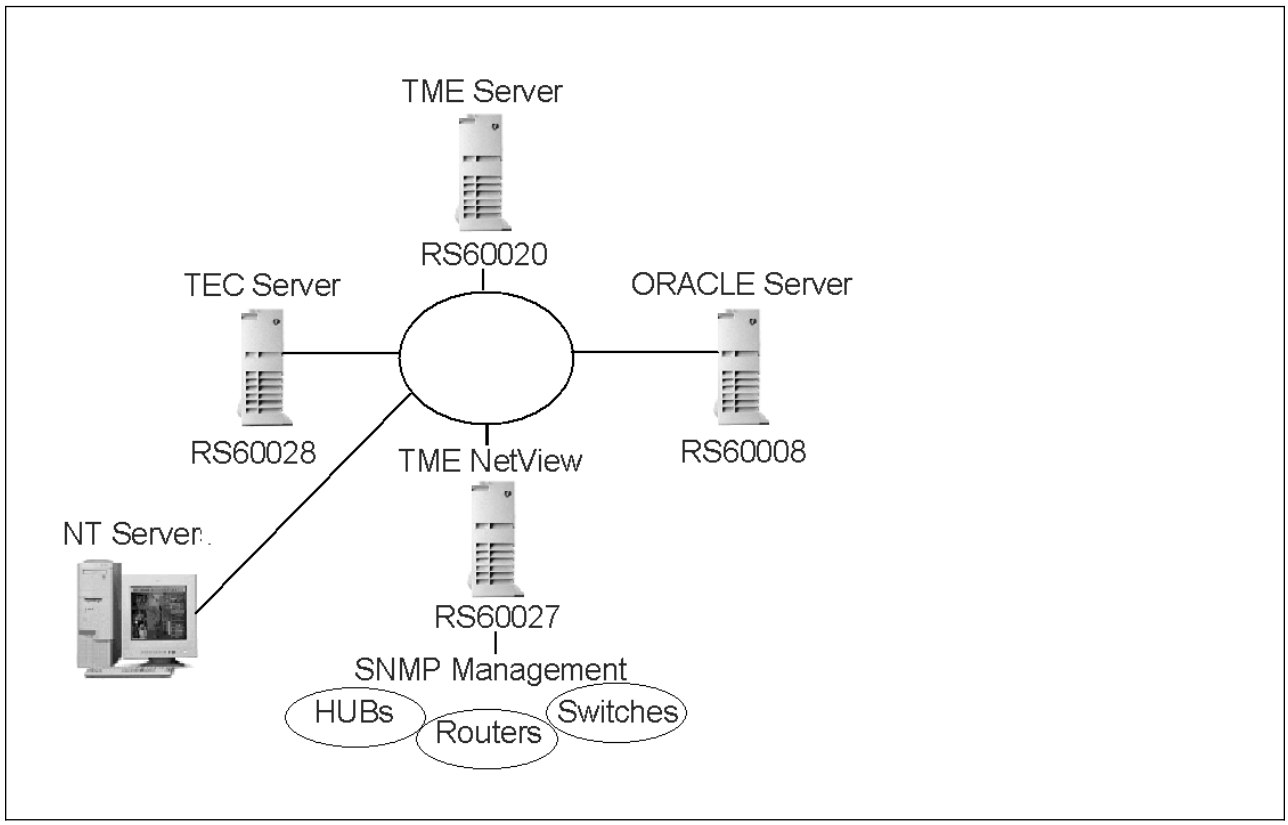


Figure 1. Physical Layout of the Devices

All the RISC System/6000 machines were running AIX 4.2. Each server was configured with 128 MB of RAM or more, and was installed with 2 or 4 GB hard disk drives.

The server sizing requirements are covered in the relevant release notes for each of the applications.

2.1.2 Managed Devices and Resources

The customer had identified a number of network and system devices that require managing. We compiled a list of these devices so that we could begin to look at what resources will be monitored and potentially generate events that will be sent to the TEC. The list of managed devices are:

- AIX NetView Server
- AIX Oracle Server
- NT Servers
- NT Workstations
- Cabletron Hubs
- Cicso Routers
- Novell Servers
- Compaq Insight Manager
- IBM 8250 Hub
- IBM 8272 Switches

We added additional management functions to manage certain resources on the AIX servers as follows:

- Distributed Monitoring
- UNIX logfile adapter for the AIX System Log
- Shell scripts for additional management of resources

Next we show each of the event sources.

2.1.3 The Event Sources

By building the list of managed devices and resources we now know what types of events we send to the TEC. We have listed each of the event sources below:

1. SNMP-Managed Devices via NetView:
   - Cabletron Hubs
   - Cicso Routers
   - Novell Servers
   - Compaq Insight Manager
   - IBM 8250 Hub
   - IBM 8272 Switches
2. RISC System/6000 AIX Servers
   - Distributed Monitoring for the RISC System/6000 AIX servers
   - UNIX logfile adapter for the AIX System Log on the RISC System/6000 servers
   - Specific shell scripts for additional management of the resources on the RISC System/6000 servers.
3. NT Servers
   - Distributed Monitoring for the NT servers

The main focus of this implementation is to provide the following functions listed below:

- Provide a stable management environment to provide the required management functions
- Customize the TEC and TME environment to provide the functions required
- Develop a set of TEC rules to provide the event correlation required

The amount of events that can potentially be generated from the event sources are very high. We calculated that with no event management the TEC could receive over 700 events, the majority of which will not be useful to an operator.

The event management requirements are highlighted below:

- Filter out insignificant events
- Correlate events that may be a cause of another event
- Only send events to the relevant operator

The next stage is to begin to analyze what events from all these sources are required to be sent to the TEC operator consoles. We used an event methodology developed by IBM that deals with each event source.

2.1.4 Analyzing the Events

Before any rules can be developed it is strongly recommended that you use a process to document what events can be generated from each source. This will then lead to deciding what rules need to be developed based on the event relationships. Some considerations for such a process are as follows:

- List all available sources that can generate events.
- What events will be generated from each source?
- How many potential events will the TEC receive for each source?
- What are the event relationships if any, for each source such as events that are caused by other events?
- What filtering is required and where can these events be filtered (either at the source or at the TEC)?

As part of our analysis of these events and event sources we used the IBM Event Methodology that has been developed by IBM consultants to provide event analysis. The event methodology uses a set of tools and a specific process to determine the following:

- All possible traps that can be generated from the sources.
- Events that are correlation candidates and can be linked to existing events. For example, a cause event can have any number of effect events.
- Events that are critical and must be passed directly to the TEC without any correlation.

In addition to the tables we also provided some additional information as follows:

- Diagrams showing event relationships
- The trap information, for instance the MIB definition file for each source
- The flowchart containing rule process flow and the pseudo code representing the rule process

From these documents we are able to develop the rules for each event source.

2.1.5 Rule Policies

We also investigated a set of rule policies. These policies govern the flow of each of the rules and are responsible for the actions performed when an event has been received. When an event arrives you can do one of the following:

- Change the status or severity of an event
- Check for duplicate events of the same type
- Check for related events and perform actions
- Wait a pre-defined time to check if a clear event has arrived
- Create a trouble ticket based on the information retrieved

The process we used was the IBM event methodology developed by IBM consultants based in Raleigh.

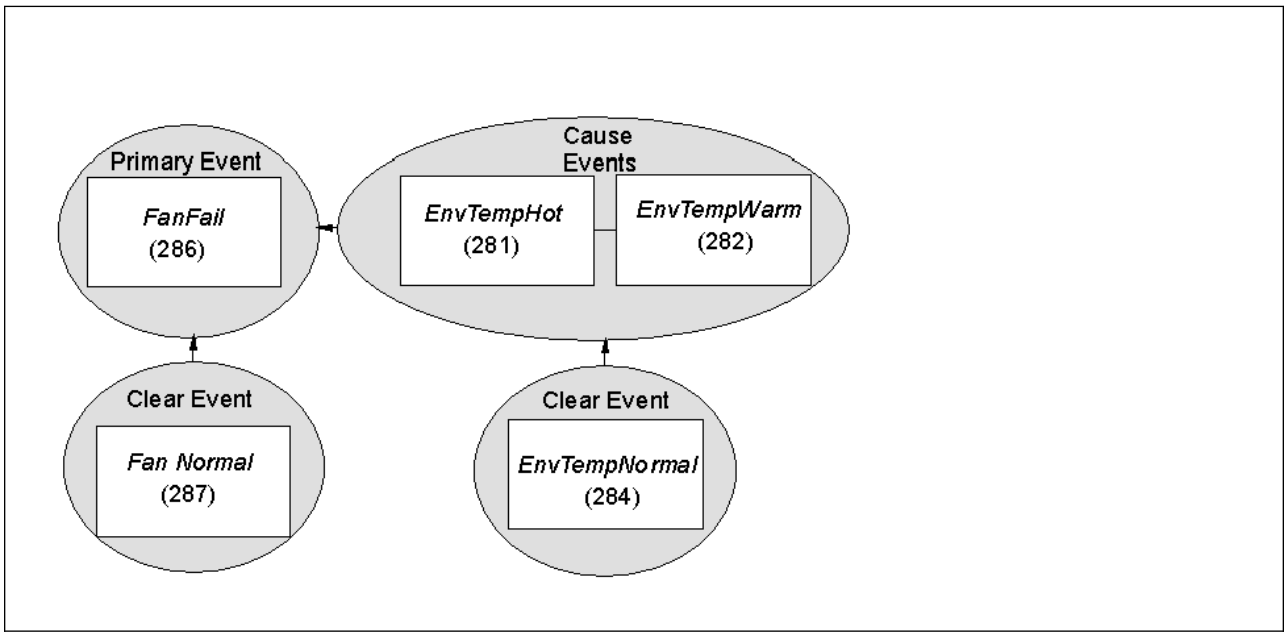


Figure 2. Event Relationships

Each source was analyzed. After listing all the events that could be generated from each source, these events were listed and for each event we began to create event relationship diagrams that graphically display the events, which are:

- Cause events
- Related events
- Resolution events
- Clear events

Given the event descriptions and relationship diagrams we can now begin to implement the solution. First we look at what design issues we considered.

2.2 Design

Using the list of assumptions already discussed we started to consider what configuration was required. Some key design considerations are listed below:

- What is the existing TMR environment if any?
- Where and what RDBMS is installed?
- Rule considerations.
- Where do we filter the unwanted events?
- What is the interface to the trouble ticket interface?
- How many TEC administrators do we need?
- What tasks do each of the administrators require?
- What automation is required?

Some other considerations that are of interest are as follows:

- What is the network layout?
- What is the required size of the TEC database?
- What is the size of the TMR and TEC environments?

In addition to the management of the specific resources we also need to provide a set of tasks that will perform specific management functions for our TEC environment. If the TEC fails for some reason, we need to know what is at fault. For this we provide a set of shell scripts that check for certain AIX processes and notify us of any problems using the console.

The rule base requirements are critical in our environment because each time we receive an important event a trouble ticket will be created. Without any rules being developed, this will become a huge issue due to the amount of tickets being generated.

Therefore, for each of the selected event sources we developed a set of rules based on specific documented customer requirements. By dealing with each specific source, for each source IBM event methodology was performed.

For each managed element an evaluation was performed to identify what events can be generated from the source. For each one there was a set of questions asked. The example below shows what we did for one specific event.

2.3 Implementation

Now that we have defined what we are to manage, we have to install the software components that will perform the required management function. The next stage is to develop an implementation plan. This is basically the flow the remaining chapters in this redbook.

The installation process is outlined below:

1. Install the RDBMS system (Oracle or Sybase)
2. Install the TMR environment
3. Install the TEC software
4. Install the TEC Console software
5. Configure the operators
6. Install and configure the Distributed Monitoring
7. Install and configure the TEC logfile and NT adapters
8. Install and configure the TME 10 NetView adapter
9. Develop and configure the TEC tasks
10. Develop and implement the required TEC rules
11. Test implementation

The process above is what we used as our implementation plan. Each stage is now covered in the remaining chapters of this redbook.

2.4 TEC Rule Development

As part of this project we develop a large amount of rules, class definition files, tasks and shell scripts. We create a directory structure on one of our machines, rs600028, that holds all these files. We refer to this directory in the remaining sections in the redbook. We did this in order to have an separate copy of all our configuration files.

Chapter 3. Installing the RDBMS

This chapter explains the RDBMS installation for our environment.

We installed the Oracle RDBMS on a separate server then the TEC to show how to create the RIM agent on the client machine rs60008 as the Oracle server and the SQL client on rs600028.

The Sybase RDBMS software was installed on rs600028, the same server as the TEC software to show how the TEC server can reside on the same platform as the RDBMS.

It is worth noting that the installation of the RDBMS software should be performed by an experienced RDBMS person, due to the fact that there are a number of issues when installing the software, such as where the databases will reside, if the database will be in raw format and what values need to be set for the tuning parameters. This chapter shows what was done to install the RDBMS without any tuning or customization.

The creation of the RIM object and the TEC database tables is covered in Chapter 4, “Software Installation” on page 49.

We also installed the Sybase database on an NT server. For these instructions refer to Appendix A, “TEC Installation on Windows NT” on page 285.

3.1 RDBMS Prerequisites

For installation on AIX, both the Sybase and Oracle RDBMSs use the asynchronous disk I/O, which means that control returns to the application as soon as an I/O request has been queued. To make sure there is an asynchronous I/O device available on your AIX system issue the command:

```
lsdev -C | grep aio0.
```

You should get a response similar to the following:

```
aio0       Available              Asynchronous I/O
```

If there is none defined, define one with the following command:

```
mkdev -l aio0
```

If preferred, you can use the smit panels by typing the command: `smitty aio`.

You will see the screen below:

```
                          Asynchronous I/O

Move cursor to desired item and press Enter.

  Change / Show Characteristics of Asynchronous I/O
  Remove Asynchronous I/O; Keep Definition
  Configure Defined Asynchronous I/O
  Generate Error Report
  Trace Asynchronous I/O
```

The default values for Asynchronous I/O must be modified. The initial values are shown in the panel below.

```
   Change / Show Characteristics of Asynchronous I/O


Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    Entry Fields
  MINIMUM number of servers                          [1]
  MAXIMUM number of servers                          [10]
  Maximum number of REQUESTS                         [4096]
  Server PRIORITY                                    [39]
  STATE to be configured at system restart available
```

We change the minimum number of servers to be 10, and insure that the state to be configured is available. All other values are left at the defaults.

Both Sybase and Oracle use the Portable Streams Environment (PSE) of AIX, which is an I/O subsystem, used for protocol-based inter-system and intrasystem communication. Certain PSE drivers have to be active on your AIX system. Use the following commands to check whether the necessary drivers are loaded:

```
strload -q -d stddev

strload -q -d spx

strload -q -m sc
```

If all three commands result in a `yes`, no further action is needed. If they don't, you can run the following command:

```
/usr/sbin/strload
```

This reads /etc/pse.conf to load and configure the PSE.

The portmap daemon is normally started at system startup time in /etc/rc.tcpip. Check that your portmap daemon is running with the command:

```
ps -ef | grep portmap
```

If the portmap daemon is not running, it can be started with the command:

```
startsrc -s portmap
```

You have to choose a port to be used by the SQL server. The default port for Sybase uses 6234, but the TEC documentation recommends 3344 as the default. We chose to use 3344 for the SQL server and 3346 for a backup SQL server, but you may use any unused port. First check in /etc/services to be sure that you don't already have that port defined for something else. If you do, you can select another one. One method to check the current ports in use is to issue the following command (to test for port 3344):

```
netstat -A | grep 3344
```

The port number is selected during the RDBMS installation process. The port you have chosen will not appear in /etc/services, but it is advisable that you add two lines of comments in /etc/services, documenting that these two ports have been assigned, to avoid any conflicts.

The following sections explain how we installed the RDBMS server. These examples only touch on the installation options that can be configured for both Oracle and Sybase. We do not cover such issues as backup management and RDBMS tuning. This information is contained in the relevant RDBMS documentation.

3.2 Installing Oracle

This chapter describes how to install Oracle Server V7.3.3 on our AIX machine rs60008. This installation also involved installing an Oracle client on AIX machine rs600028, which is our TEC event server and RIM host.

We selected a home directory for our Oracle installation on a separate filesystem with a mount point of /u01/app/oracle/product/7.3.3. In addition we had to create a temporary area for the installation process called /oracle_link, which is used as a temporary area to upload the Oracle files during the installation process.

For reference the amount of space required for our Oracle installation is shown using the dk -k command.

The output was as follows:

```
Filesystem 512-blocks  Free %Used Iused %Iused Mounted on
/dev/lv00   331776    115856  66%  2106     3% /u01
/dev/lv05   86016     14288   84%    22     1% /u02/oradata/tec
/dev/lv06   65536     62816    5%    18     1% /u03/oradata/tec
/dev/lv07   65536     62816    5%    18     1% /u04/oradata/tec
/dev/lv01  102400     41576   60%  7408    28% /oracle_link
```

The filesystem /oracle_link can be removed after the installation. This is a temporary working directory that contains installer and documentation files and symbolic links to the software on the CD-ROM.

3.2.1 Oracle Pre-Install Tasks

The following is a list of tasks that we performed prior to starting the oracle install program. First we created a UNIX group called dba:

```
mkgroup adms=root dba
```

Oracle requires a user oracle belonging to the group dba to which it assigns all membership of the Oracle files. To create an Oracle owner account issue the following command:

```
mkuser pgrp=dba home=/u01/app/oracle/product/7.3.3 \
       shell=/bin/ksh
```

As Oracle will be using the TCP/IP protocol adapter it is necessary to ensure that our underlying network is working properly. Also we needed to reserve a port in /etc/services, which the SQL*Net Version 2 listener will use. We edited the file /etc/services and added the following line:

```
lsnrctl 1521/tcp #Oracle Listener
```

Although the portmap daemon is normally started at system restart via rc.tcpip, ensure that it is running by issuing the command:

```
ps -ef | grep portmap
```

If the portmap daemon is not running, you can start it using the command:

```
startsrc -s portmap
```

The following environment variables have to be set prior to commencing the install. We included these variables in the login profile for the Oracle account. The defined variables defined in our .profile file are shown below:

```
export ORACLE_HOME=/u01/app/oracle/product/7.3.3
export ORACLE_SID=tec
export ORACLE_TERM=vt100
export OBK_HOME=$ORACLE_HOME/obackup

export PATH=$PATH:$ORACLE_HOME/bin:/opt/bin:/bin:/usr/bin:
   /usr/ccs/bin:/GNU/bin/make:$ORACLE_HOME/network:

$ORACLE_HOME/plsql:$ORACLE_HOME/rdbms:$ORACLE_HOME/lib
export ORACLE_BASE=/u01/app/oracle
export ORACLE_PATH=.:$ORACLE_HOME/bin:$ORACLE_HOME/obackup/bin:
   /opt/bin:/bin:/usr/bin:/GNU/bin/make:/usr/ccs/bin

export TMPDIR=/var/tmp
export ORA_CSM_MODE=line
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib:/lib
export ORA_NLS32=$ORACLE_HOME/ocommon/nls/admin/data
export EDITOR=vi
```

Next we made sure the the oracle_link and $ORACLE_HOME directories are owned by the user oracle and the group dba. The access permissions also need to be set to 774.

```
chown oracle.dba /oracle_link
chown -R oracle.dba /u01
chmod 774 /oracle_link
chmod -R 774 /u01
```

Mount the Oracle Server software CD as follows:

```
mount -rv cdrfs /dev/cd0 /CDROM
```

The following task is only required on the RDBMS server, not the client.

Create mount points for distributing control files. We used the Oracle recommended structure and actually created these as separate filesystems. This allows an element of resiliency to be built in to the installation as we spread the database across these to eliminate single point of failure. We did this via SMIT panels for creating filesystems. Filesystem sizes are dependant on the sizing that you will decide for your database. Our three mount points were as follows:

```
/u02/oradata/tec
/u03/oradata/tec
/u04/oradata/tec
```

These are initial sizes and can be extended via SMIT panels or from the command line using the chfs command.

Verify the oracle directories are owned by user oracle and group dba and that access permissions are set to 774:

```
chown oracle.dba /u02/oradata/tec
chmod 774 /u02/oradata/tec
```

It is possible to use raw logical volumes for these mount points, which offer dynamic performance tuning and also mirroring and online disk replacement, but for our install we chose not to use this method.

For additional information on raw filesystem configuration see the Oracle documentation.

We now step through the process of installing the software and configuring the RDBMS.

3.2.2 Oracle Server Installation Process

To start the Oracle RDBMS installation we need to run as the oracle user. Here we execute the command su - oracle and start the installation as follows:

```
cd /CDROM/orainst
./start.sh
```

Specify /oracle_link as the path to the oracle link directory when prompted. The linking of files takes several minutes to complete. Our output is shown below:

```
Please Enter your oracle_link directory:
/oracle_link
Linking files. This will take a few minutes
Linking /CDROM /oracle_link

You May now go to the orainst subdirectory of your oracle_link
directory and run your installer
```

Now log in as the root user to install the Oracle application:

```
cd /oracle_link/orainst
```

Run the following script to set up the environment:

```
 ./rootpre.sh
```

The command returned the following:

```
Installing kernel extensions...

An installed copy of the post/wait kernel extension was found
and has been replaced by a new version.
To enable the new version of the post/wait kernel extension,
the system should be restarted.



Configuring Asynchronous I/O...
Asynchronous I/O is already configured

 Loading AIX Portable Streams Environment...

Your current system already has stream facility running, we
 shall leave it as it is,  assuming it has been configured
 to be compatible with the /etc/pse.conf setup
```

Running this shell script as the root user checks for AIO configuration and availability, that the PSE drivers are active on your AIX system and installs any post-wait kernel extensions. Once this is complete switch back to the oracle user and execute the command:

```
cd /oracle_link/orainst
```

If any errors occur during installation, these are logged in the file $ORACLE_HOME/orainst/install.log. To start the installation execute the Oracle command ./orainst.

This will initiate the Oracle installer program interface. To move around within the dialog boxes, use the Tab key to move selections and the Spacebar to actually select. Generally this highlights the option or places an X or O next to the required option. Then press the return key to confirm.

At this point we selected the Custom Install option. This allowed us to choose the software components we wanted to install.

```
                              Install Type

Select the installation method,
either Default Install or Custom Install, and whether
you want to view the Readme files:

                     ( ) Default Install
                     (X) Custom Install
                     ( ) Display readme Files for this Release


  (Help...)        (Back)           (Cancel)         (OK)
```

Using the Spacebar and Tab keys select the **Custom install** option and then select **OK**.

```
                         Installation Activity Choice

Select the Installer activity:

(o) Install, Upgrade, or De-Install Software
( ) Create/Upgrade Database Objects
( ) Perform Administrative Tasks


(Help...)        (Back)           (Cancel)         (OK)
```

Choose the option **Install, Upgrade or De-install software** followed by **OK**.

```
                      Installation Options

       Select the Installer option:
       Note: there are two different Install New Product
       options.

       (o)     Install New Product - Create DB Objects
       ( )     Install New Product - Do Not Create DB Objects
       ( )     Add/Upgrade Software
       ( )     Build Oracle7 Staging Area
       ( )     Install Documentation Only
       ( )     De-Install Software

  (Help...)        (Back)           (Cancel)         (OK)
```

Select **Install New Product-Create DB Objects** followed by **OK**.

```
                    Environment Variables

Confirm, change, or enter values for the environment variables
listed:

    ORACLE_BASE:    /u01/app/oracle

    ORACLE_HOME:    /u01/app/oracle/product/7.3.3


   (Help...)          (Back)             (Cancel)           (OK)
```

Confirmation of the ORACLE_HOME directory and ORACLE_BASE variables is required here. If you wish to change these locations, do so at this point but the paths have to exist on your AIX system. Once complete click on **OK**.

```
                Confirm (or Change) Log File Location:

     If current versions of these log files exist in these locations,
     they will be archived and reinitialized.  Select Help for more
     details.


  Installer Log:  /u01/app/oracle/product/7.3.3/orainst/install.log

  SQL Log:        /u01/app/oracle/product/7.3.3/orainst/sql.log
  Makefile Log:   /u01/app/oracle/product/7.3.3/orainst/make.log

  OS Log:         /u01/app/oracle/product/7.3.3/orainst/os.log


(Help...)          (Back)             (Cancel)           (OK)
```

The logfile locations are defined here. The installer program appends to the ORACLE_HOME directory the designated locations. If you wish to change these, then do so at this point. We chose to accept the installer generated locations by selecting **OK**.

```
Pre-Installation OS Preparation

     Your platform requires the root user to perform certain pre-
     installation OS preparation.  The root user should run the shell
     script 'rootpre.sh' before you proceed with Oracle installation.
     rootpre.sh can be found in the newoi/ directory (if installing
     from tape) or the orainst/ directory (if installing from CDROM).

     Select (Yes) if root has run 'rootpre.sh' so you can proceed with
     Oracle  installation.  Select (No) to abort installation and then
     ask root to run 'rootpre.sh'.

(Help...)        (Back)          (Cancel)        (No)      (Yes)
```

We selected **Yes** as we had already run the rootpre.sh script. If you select **No**, then the installer program will exit and return you to the command prompt. At this point you can re-run the rootpre.sh and restart the installer program.

```
 Install Source

                 Select one of the following:


           (o) Install from CD-ROM
           ( ) Install from Staging Area

 (Help...)        (Back)           (Cancel)         (OK)
```

Specify the install source in the panel above. We selected CD-ROM.

```
                           $ORACLE_LINK Directory

 Enter the pathname of the $ORACLE_LINK directory:
/oracle_link


         (Help...)           (Back)              (Cancel)            (OK)
```

If this is not the first time the installer program has been run, then you will be prompted for the link directory. Enter `/oracle_link` and select **OK**.

```
ORACLE_SID

Enter Your ORACLE_SID:

tec

 (Help...)        (Back)           (Cancel)         (OK)
```

If you have specified the ORACLE_SID variable in the oracle profile, then this will be used. Otherwise specify it now. We used tec as our SID.

When prompted for NLS we selected American/English.

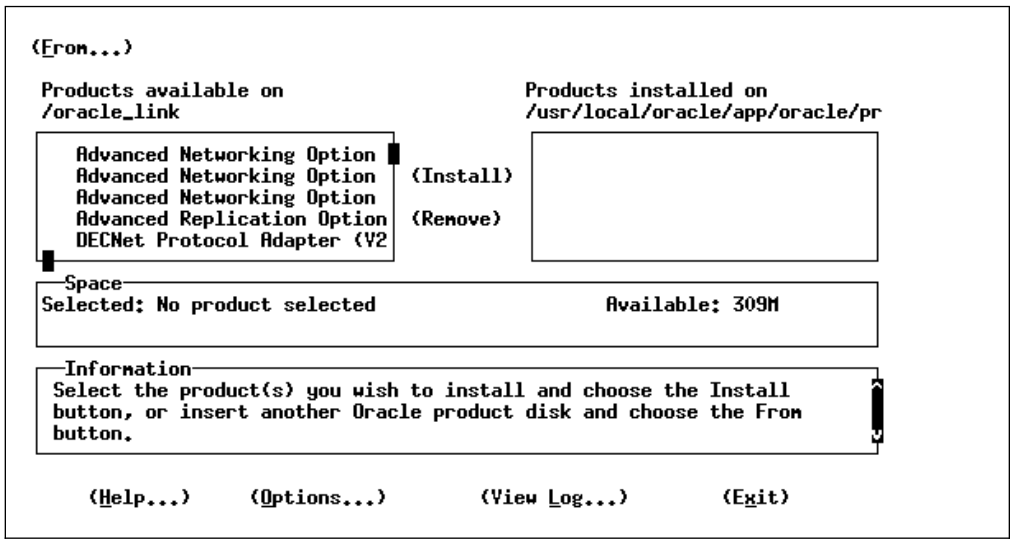
We did not choose to relink at this point, since even if you decline, the installer automatically relinks products that require re-linking.

```
Information Dialog

Post-installation steps that need to be run by root will
be written to /usr/local/oracle/7.3.3/orainst.root.sh

 (Help...)        (Back)           (Cancel)         (OK)
```

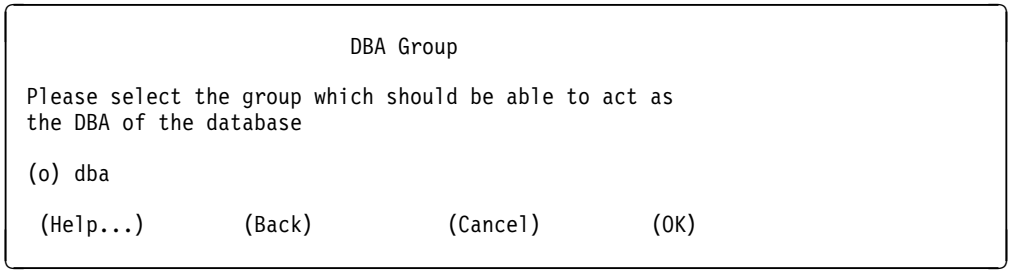
The above screen is informing us that the root.sh shell script, which should be run as a post install task, is being created in the specified location. This script sets the necessary file permissions for the Oracle products and performs some additional setup activities. Then click on **OK**.



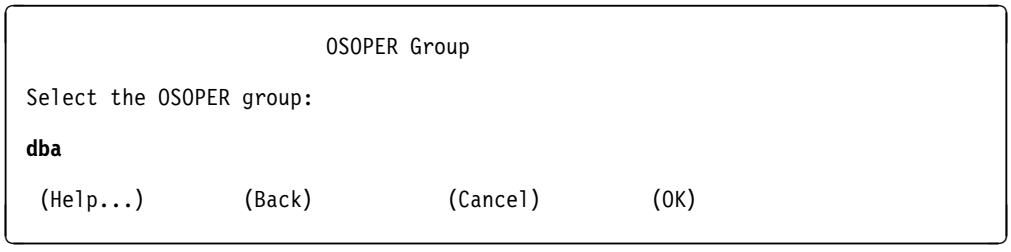
We selected to install the following products at this stage. From the installation screen we selected:

- Oracle7 Server (RDBMS) Version 7.3.3
- SQL*NET Version 2.3.3
- SQL*Plus Version 3.3.3
- Oracle TCP/IP Protocol Adapter Version 2.3.3

Once you have made your selection, tab to the Install option and press Return.



Specify the value for the database administrator's group. Here we used the group dba.



Confirm the group ID by clicking on **OK**.

```
                    Create DB Objects: Storage Type

                  Choose storage type for database:

                 (o) Filesystem-Based Database
                 ( ) Raw Device-Based Database

 (Help...)        (Back)           (Cancel)         (OK)
```

We used filesystem as our storage type.

```
              Create DB Objects (F/S): Mount Point Locator

Enter First Mount Point

 (Help...)        (Back)           (Cancel)         (OK)
```

The mount points we specified here (/u02/oradata/tec, /u03/oradata/tec and /u04/oradata/tec) were the mount points of the three filesystems we had created earlier in the pre-install steps.

The installer prompts for all three mount points to be specified in turn. When you have completed click on **OK**. When you are prompted for the character set select **US7ASCII**.

When we were prompted for the system password we entered `oracle` for all the required passwords.

At this point we encountered dialogs that prompted us to specify and confirm the passwords for the database IDs SYSTEM and SYS. We also were asked if we wished to set passwords for DBA and OPER, the internal users, but we declined as this was not essential at this point.

Finally we were prompted to specify and confirm the password for the TNS Listener. Here we also entered `oracle`.

```
Configure MTS and Start SQL*NET Listener

Would you like MTS (Multi-Threaded Server Configured and the
SQL*Net listener automatically Started

 (Help...)   (Back)        (Cancel)      (No)      (Yes)
```

We specified that we wanted to configure MTS and start the SQL Listener. MTS allows a single server process to service multiple client application process requests, increasing performance.

```
                          DB Domain Name

Please give the database domain name (db_domain)

tec.world

 (Help...)        (Back)           (Cancel)         (OK)
```

We specified tec.world as our domain. DB_DOMAIN is a text string that specifies the network domain where the database is created; this is typically the name of the organization that owns the database. If the database you are about to create will ever be part of a distributed database system, pay special attention to this initialization parameter before database creation. We chose to use world which is the default.

```
Create DB Objects(F/S): Control File Locator

Do you want to use the following
/u02/oradata/tec/control01.ctl
/u03/oradata/tec/control01.ctl
/u04/oradata/tec/control01.ctl

 (Help...)        (Back)           (Cancel)         (OK)
```

We accepted the locations where the installer was going to place the RDBMS control files.

Press Enter at the DB Defaults panel. We accepted the default database file sizes here but in reality you would wish to specify at this point how big you want your files to be. There were three screens to step through before being given the option to go back and edit the sizes. The default sizes are invariably too small and thought should be given to how much space is required prior to reaching this point.

The installer now analyzes any dependances for the software we had selected to install and if pre-requisite software was needed.

We were prompted to see if we required the SQL*Plus help and the SQL*Plus Demo tables; here we declined.

```
                           Oracle
Installing...

From: /oracle_link

To: /u01/app/oracle/product/7.3.3/.../doc/README.doc

                      TCP/IP Protocol Adapter (V2)


            Linking and Installing TNS Listener Executables
```

With the parameters specified, the Oracle install began.

Once the installation had finished we were prompted to run the root.sh script.

As the root user issue the commands:

```
cd /u01/app/oracle/product/7.3.3/orainst
./root.sh
```

Enter y when prompted as shown below:

```
Running ORACLE7 root.sh script...
The following environment variables are set as:
  ORACLE_OWNER=oracle
  ORACLE_HOME= /u01/app/oracle/7.3.3.
  ORACLE_SID= tec
Are these settings correct (Y/N)? (Y): y
```

The Oracle listener process needs to be started at this point. This can be done as the root user by issuing the following command:

```
lsnrctl start
```

The output is as follows:

```
LSNRCTL for IBM/AIX RISC System/6000: Version 2.3.3.0.0
Production on 24-MAR-98 09:09:42

Copyright (c) Oracle Corporation 1994.  All rights reserved.

Starting /u01/app/oracle/product/7.3.3/bin/tnslsnr: please wait...

TNSLSNR for IBM/AIX RISC System/6000: Version 2.3.3.0.0 - Production
System parameter file is /u01/app/oracle/product/7.3.3/network/admin/listener.ora
Log messages written to /u01/app/oracle/product/7.3.3/network/log/listener.log
Listening on: (ADDRESS=(PROTOCOL=ipc)(DEV=10)(KEY=tec.world))
Listening on: (ADDRESS=(PROTOCOL=ipc)(DEV=13)(KEY=tec))
Listening on: (ADDRESS=(PROTOCOL=tcp)(DEV=14)(HOST=9.24.104.30)(PORT=1521))

Connecting to (ADDRESS=(PROTOCOL=IPC)(KEY=tec.world))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for IBM/AIX RISC System/6000: Version 2.3.3.0.0
Start Date                24-MAR-98 09:09:53
Uptime                    0 days 0 hr. 0 min. 1 sec
Trace Level               off
Security                  ON
SNMP                      ON
Listener Parameter File /u01/app/oracle/product/7.3.3/network/admin/listener.ora
Listener Log File       /u01/app/oracle/product/7.3.3/network/log/listetner.log
Services Summary...
  tec           has 1 service handler(s)
The command completed successfully
```

We edited the /etc/oratab file to reflect that we wanted the tec database to start up and shut down by executing the oracle commands dbshut or dbstartup. These commands must be run as the oracle account. A sample /etc/oratab for achieving this is shown below. This made it easier to shut down the database rather than having to log in to the database using the commands sqplus or svrmgrl.

```
tec:/u01/app/oracle/product/7.3.3:Y
```

We also created an entry in /etc/inittab so that Oracle would start automatically each time the system was rebooted:

```
mkitab "oracle:2: :wait:/bin/su oracle -c /$ORACLE_HOME/bin/dbstart"
```

The Oracle server portion of the Oracle installation is now complete.

If there is a failure when executing this command, consult the listener.log file in the $ORACLE_HOME/network/log directory for detailed error messages.

3.2.3 Installing the Oracle SQL Client

If the TEC server is on the same machine as the Oracle server, then the SQL application can be loaded on the Oracle server to give access to the database. Otherwise there are fewer steps necessary to configure the SQL client.

We created the necessary SQL*Net configuration files via the Oracle Network Manager utility. The process involved:

- Installing the Oracle Network Manager software on a Windows-based PC (see Chapter 2, "Installing Oracle Network Manager", in the *Oracle Network Manager Administrator's Guide*).
- Using Network Manager to create the SQL*Net V2 configuration files for the network (see Chapter 4, "Quick Steps to Configure a Network", in the *Oracle Network Manager Administrator's Guide*).
- Placing the the SQL*Net Version 2 configuration files on the server and clients either by using FTP or transferring them on a 3.5 diskette, placing them in the /var/opt/oracle directory. We had to create this directory as it did not exist.

  By default these files should reside in the directory $ORACLE_HOME/network/admin on both the client and the server.

We followed the same installation instructions for the client, with the following exceptions.

```
                    Installation Options

       Select the Installer option:
       Note: there are two different Install New Product
       options.

       ( )     Install New Product - Create DB Objects
       (o)     Install New Product - Do Not Create DB Objects
       ( )     Add/Upgrade Software
       ( )     Build Oracle7 Staging Area
       ( )     Install Documentation Only
       ( )     De-Install Software

  (Help...)        (Back)           (Cancel)         (OK)
```

The client install process required all of the elements of the server install process except that when prompted for install type we selected the **Install Product - Do Not Create Database Objects** option. We only selected to install the Oracle tools shown below:

```
Oracle SQL*Net Version 2.3.3 SQL*Plus 3.3.3
```

Due to the fact that the client does not create a database the installation process is quicker to perform.

You will also see the additional screen below.

```
Your TWO_TASK is set (tec.world). The rest of the
     installation assumes you are performing a client-only
     install. If you do not want to perform a client-only
     install, select (Cancel) now and undefine TWO_TASK in
     your environment.
```

This is due to having the variable defined in our .profile for the user oracle.

Either using the PC-based configuration tool or manually editing the SQL configuration files, copy these files to the client machine in the directory $ORACLE_HOME/network/admin.

If you are copying from the PC application, you have to change the case of the files. If they have been created on a Windows-based PC, they must be uppercase and on the AIX client they must be in lowercase. Verify the two files are owned by oracle. We have included a copy of our configuration files for the network connections for the SQL client.

The files are as follows:

- tnsnames.ora
- listener.ora
- sqlnet.ora

The tnsnames.ora file contained the following:

```
tec.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (COMMUNITY = tcp.world)
          (PROTOCOL = TCP)
          (Host = rs60008)
          (Port = 1521)
        )
    )
    (CONNECT_DATA =
       (SID = tec)
       (GLOBAL_NAME = tec.world)
    )
```

The listener.ora file located on the Oracle server has the following configuration parameters defined.

```
################
# Filename......: listener.ora
# Name..........: rs60008.world
# Date..........: 11-FEB-98 16:11:37
################
SQLNET.AUTHENTICATION_SERVICES = (NONE)

USE_PLUG_AND_PLAY_LISTENER = OFF
USE_CKPFILE_LISTENER = OFF
LISTENER =
  (ADDRESS_LIST =
        (ADDRESS=
          (PROTOCOL=IPC)
          (KEY= tec.world)
        )
        (ADDRESS=
          (PROTOCOL=IPC)
          (KEY= tec)
        )
        (ADDRESS =
          (COMMUNITY = tcp.world)
          (PROTOCOL = TCP)
     (KEY= tec.world)
        )
        (ADDRESS=
          (PROTOCOL=IPC)
          (KEY= tec)
        )
        (ADDRESS =
          (COMMUNITY = tcp.world)
          (PROTOCOL = TCP)
          (Port = 1521)
        )
  )
STARTUP_WAIT_TIME_LISTENER = 0
CONNECT_TIMEOUT_LISTENER = 10
TRACE_LEVEL_LISTENER = OFF
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = tec)
      (ORACLE_HOME = /u01/app/oracle/product/7.3.3)
      (PRESPAWN_MAX = 10)
    )
  )
```

The sqlnet.ora file was as follows:

```
################
# Filename......: sqlnet.ora
# Name..........: TCP.world
# Date..........: 11-FEB-98 16:11:37
################
AUTOMATIC_IPC = ON
TRACE_LEVEL_CLIENT = OFF
SQLNET.EXPIRE_TIME = 0
NAMES.DEFAULT_DOMAIN = world
NAME.DEFAULT_ZONE = world
SQLNET.CRYPTO_SEED = "-1470094283-1470083480"
```

At this point we also logged into the client as the Oracle owner and placed the previously generated SQL*Net configuration files into the

$ORACLE_HOME/network/admin directory and tested the connection to the server via the sqlplus tool. Verify that the file has the same ownership and permissions set for the server files.

To test the client issue the following commands on the SQL client:

```
export TWO_TASK=tec.world
sqlplus sys/oracle@tec
```

Where sys is the database user ID, oracle is the password and tec is the service as defined in tnsnames.ora. This gave us a connection into the newly created tec database and an SQL> prompt, which indicated an active connection to the database server.

To verify that Oracle is up and running you can view the processes using ps -ef | grep ora. The processes running are:

```
ora_dbwr_tec
ora_lgwr_tec
ora_pmon_tec
ora_s000_tec
ora_d000_tec
ora_smon_tec
/usr/local/oracle/7.3.3/bin/tnslsnr LISTENER -inherit
```

The active processes we saw are the database and listener applications. This confirms that they were both successfully installed and started. For any startup errors check the log directory $ORACLE_HOME/rdbms/log.

3.2.4 Oracle Startup and Shutdown

To stop the oracle database use the command:

```
dbshut
```

The command output is as follows:

```
Oracle Server Manager Release 2.3.3.0.0 - Production
Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.
Oracle7 Server Release 7.3.3.0.0 - Production Release
With the distributed, replication, parallel query and Spatial Data options
PL/SQL Release 2.3.3.0.0 - Production
SVRMGR> Connected.
SVRMGR> Database closed.
Database dismounted.
ORACLE instance shut down.
SVRMGR>
Server Manager complete.
Database "tec" shut down.
```

To start the database use the command:

```
dbstart
```

The command output is as follows:

```
Oracle Server Manager Release 2.3.3.0.0 - Production

Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.

Oracle7 Server Release 7.3.3.0.0 - Production Release
With the distributed, replication, parallel query and Spatial Data options
PL/SQL Release 2.3.3.0.0 - Production

SVRMGR> Connected to an idle instance.
SVRMGR> ORACLE instance started.
Total System Global Area       5013176 bytes
Fixed Size                       38988 bytes
Variable Size                  4138604 bytes
Database Buffers                819200 bytes
Redo Buffers                     16384 bytes
Database mounted.
Database opened.
SVRMGR>
Server Manager complete.

Database "tec" warm started.
```

We got the following error message when using the sqlplus command from the client:

```
ERROR: ORA-12203: TNS:unable to connect to destination
```

This was caused by the the listener service failing to start.

3.3 Sybase Installation

We installed the Sybase RDBMS on rs600028. This server will eventually become our TEC server.

On rs600028 as root we created a home directory for the Sybase installation:

```
mkdir /sybase/code.
```

We set the $SYBASE environment variable in .profile to point to the directory /sybase/code.

We entered the following commands to install Sybase:

```
export SYBASE=/sybase/code export LANG=C cd $SYBASE cd install
```

Later we add other statements to enable us to query our TEC database outside of TEC.

Our install directory contained the files:

```
-rwxr-xr-x   1 254      60           445 Mar  7 1998  RUN_SYBASE
drwxr-xr-x   2 254      60           512 Mar  7 1998  SPR
-rw-r--r--   1 254      60          3063 Mar  7 1998  errorlog
-rwxr-xr-x   1 254      60          3307 Mar  7 1998  setperm_all
-rwxr-xr-x   1 254      60           630 Mar  7 1998  showserver
-rwxr-xr-x   1 254      60          9102 Mar  7 1998  startserver
-rwxr-xr-x   1 254      60       1643117 Mar  7 1998  sybinit
```

To start the process, enter `./sybinit`.

The first panels we show are concerned with defining the two ports that will be used for communication between TEC and your SQL servers. This is for the primary as well as the backup. After defining the ports, we configured the primary server itself, and subsequently the backup server.

```
The log file for this session is '/sybase/code/init/logs/log0219.006'.

SYBINIT

1.  Release directory:  /sybase/code

2.  Edit / View Interfaces File

3.  Configure a Server product
4.  Configure an Open Client/Server product


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 2
```

Choose option **2** to define the port (3344) over which TEC will communicate with the Sybase SQL server.

```
INTERFACES FILE TOP SCREEN

Interfaces File:

1.  Add a new entry
2.  Modify an existing entry
3.  View an existing entry
4.  Delete an existing entry


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.
Enter the number of your choice and press return: 1
```

Choose option **1** to add a new port definition.

```
CREATE NEW INTERFACES FILE ENTRY

1.  Server name:



Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
Enter the name of the server to add (default is ''):
TEC
```

Choose option **1** to define the SQL server name whose port you are about to define followed by pressing Ctrl-a. We used the name TEC for the SQL server.

Exit this panel by pressing Ctrl-a to confirm your choice of the server name.

```
SERVER INTERFACES FILE ENTRY SCREEN

    Server name:

1.  Retry Count:  0
2.  Retry Delay:  0

3.  Add a new listener service


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 3
```

Retry Count and Retry Delay can remain at 0. Choose option **3** to add a new listener service.

```
EDIT TCP SERVICE

1.  Hostname/Address: rs600028
2.  Port:
3.  Name Alias:

4.  Delete this service from the interfaces entry


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 2
Enter the port number to use for this entry (default is ''):
3344
```

Choose option **2** to define the port number. We entered 3344.

```
SERVER INTERFACES FILE ENTRY SCREEN

    Server name:  TEC

1.  Retry Count:  0
2.  Retry Delay:  0

3.  Add a new listener service

Modify or delete a service

Listener services available:

    Protocol  Address          Port          Name Alias
4.  tcp       rs600028         3344

Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 3
```

Enter the port number that you chose earlier for Sybase and TEC communication. Press Ctrl-a to confirm these choices.

Enter y when prompted. The file $SYBASE/interfaces will be created.

Should you need or want at a later time to change the port number over which Sybase and TEC communicate, you must:

- Edit the file $SYBASE/interfaces and change the port number.
- Stop the dataserver process with the kill command.
- Restart the dataserver process using the script $SYBASE/RUN_TEC.

Our script RUN_TEC is shown below:

```
#!/bin/sh
SYBASE=/usr/local/Tivoli/bin/aix4-r1/TME/TEC/sybase
export SYBASE
DSLISTEN=TEC
export DSLISTEN
cd $SYBASE
exec ./dataserver -dmaster.dat -eerrorlog_TEC > /dev/null 2>&1
```

If the Sybase server is stopped for database administration, then there will be a period of time when no incoming events will be processed by the TEC.

For Sybase we also created the port ID for the backup server. For this we followed the same procedure with the exception of the following parameters:

- SQL server name of TEC_BACKUP
- A value of 3346 for the port address

For the TEC and TEC_BACKUP servers the port definitions are entered in the file $SYBASE/interfaces as shown below.

```
## TEC on rs600028
##      Services:
## query tcp (3344)
## master tcp (3344)

TEC
    query tcp ether rs600028 3344
    master tcp ether rs600028 3344


## TEC_BACKUP on rs600028
##      Services:
## query tcp (3346)
## master tcp (3346)

TEC_BACKUP
    query tcp ether rs600028 3346
    master tcp ether rs600028 3346
```

Press Ctrl-a to return to the main menu.

Now you're ready to configure the primary SQL server itself.

```
SYBINIT

1.  Release directory:  /sybase/code

2.  Edit / View Interfaces File

3.  Configure a Server product
4.  Configure an Open Client/Server product


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 3
```

From the SYBINIT screen select option **3** to configure the server product.

```
CONFIGURE SERVER PRODUCTS

Products:

    Product                 Date Installed    Date Configured
1.  SQL Server              Feb 19 98 10:49   Feb 19 98 11:00
2.  Backup Server           Feb 19 98 10:50


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
```

Select option **1** to configure the SQL server.

```
NEW OR EXISTING SQL SERVER

1.  Configure a new SQL Server
2.  Configure an existing SQL Server
3.  Upgrade an existing SQL Server


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
```

Choose option **1** to configure the primary server.

```
ADD NEW SQL SERVER

1.  SQL Server name:  SYBASE


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
TEC
```

Change the current value of SYBASE to the database server name of TEC. Press Ctrl-a to confirm.

Next we customize the SQL server configuration.

```
SQL SERVER CONFIGURATION

1.  CONFIGURE SERVER'S INTERFACES FILE ENTRY               Incomplete

2.  MASTER DEVICE CONFIGURATION                           Incomplete
3.  SYBSYSTEMPROCS DATABASE CONFIGURATION                 Incomplete
4.  SET ERRORLOG LOCATION                                 Incomplete
5.  CONFIGURE DEFAULT BACKUP SERVER                       Incomplete

6.  CONFIGURE LANGUAGES                                   Incomplete
7.  CONFIGURE CHARACTER SETS                              Incomplete
8.  CONFIGURE SORT ORDER                                  Incomplete

9.  ACTIVATE AUDITING                                     Incomplete

Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
```

From the SQL SERVER CONFIGURATION screen enter option 2 to define the master device.

A database device is a storage location for the objects that constitute your databases. A device is not necessarily a distinct physical unit, although it can be, but it can also be a file in a file system. A device or file must be initialized and defined to the SQL server before it can be used for database storage. This

configuration process will initialize and define as a device the *master device* you choose.

It is recommended that you define a raw device for the master device, which will contain your master database with system database information. I/O access to a disk as a raw device means that all data transfers occur in multiples of the disk block size, over a character special file, rather than a block special file. This has performance benefits. As we had only one disk on our system, we defined an operating system file for master.dat.

Enter option 2 to change the size of the master device.

```
MASTER DEVICE CONFIGURATION

1.  Master Device:  /sybase/data/master.dat

2.  Size (Meg):  40

Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 2
Enter the size of the SQL Server's master device in megabytes:
40
```

We chose a size of 40 MB for our test environment. If you don't know from previous experience how large your database should be, start with 40 MB.

Press Ctrl-a to confirm your input.

If you chose to use a file rather than a raw device, you will see a warning about using a regular file. In our case we ignored the message and pressed Enter. You should follow the advice of your database administrator and performance analyst for your environment on whether to use a raw device or a regular file.

From the SQL Server configuration panel choose option **3**.

```
SYBSYSTEMPROCS DATABASE CONFIGURATION

1.  sybsystemprocs database size (Meg):  12

2.  sybsystemprocs logical device name:  sysprocsdev

3.  create new device for the sybsystemprocs database:  yes

4.  physical name of new device:  /sybase/code/sybprocs.dat

5.  size of the new device (Meg):  12

Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
```

Confirm the defaults by pressing Ctrl-a.

Next select option **4**.

```
SET ERRORLOG LOCATION

1.  SQL Server errorlog:  /sybase/code/install/errorlog

Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
```

Confirm the defaults by pressing Ctrl-a.

Choose option **5** from the SQL Server Configuration panel to name the backup SQL server. With this option you indicate the name of the database that will function as the backup server to the primary server you are configuring. Later we will configure the secondary server.

```
SET THE SQL SERVER'S BACKUP SERVER

1.  SQL Server Backup Server name:  TEC_BACKUP

Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
```

Enter 1 to change the name of the backup server.

We used the name TEC_BACKUP for the backup SQL server. Press Ctrl-a to confirm the name.

For the remaining options 6, 7, 8 and 9 we selected the default values by clicking on **OK**.

Sybase offers you the opportunity to audit database activities such as server logins and logouts, deletion of objects from the database, execution of stored procedures, actions performed by a specified user, and more. Those who are concerned about guarding against misuse of system resources, such as the security officer, will be interested in examining the Sybase auditing possibilities. For additional information refer to the manual *Sybase SQL Server System Administration Guide* for a detailed discussion of implementing auditing, what can be audited, how auditing works, and how to interpret the audit trail.

When all options show the status `Complete` you should confirm your input by pressing Ctrl-a from the SQL Server configuration window.

Enter `y` to execute the SQL server configuration. The install program shows the output similar to:

```
Execute the SQL Server Configuration now? ny

WARNING: '/sybase/data/master.dat' is a regular file which is not recommended
for a Server device.
Press <return> to continue.
Running task to create the master device.
Building the master device
Task to create the master device succeeded.
Running task to update the SQL Server runserver file.
Task to update the SQL Server runserver file succeeded.
Running task to boot the SQL Server.
waiting for server 'TEC' to boot...
Task to boot the SQL Server succeeded.
Running task to create the sybsystemprocs database.
sybsystemprocs database created.
Task to create the sybsystemprocs database succeeded.
Running task to install system stored procedures.
 ..................................................Done
Task to install system stored procedures succeeded.
Running task to set permissions for the 'model' database.
Done
Task to set permissions for the 'model' database succeeded.
Done
Task to set permissions for the 'model' database succeeded.
Running task to set the default character set and/or default sort order f
SQL Server.
Setting the default character set to iso_1
Sort order 'binary' has already been installed.
Character set 'iso_1' is already the default.
Sort order 'binary' is already the default.
Task to set the default character set and/or default sort order for the S
Server succeeded.
Running task to set the default language.
Setting the default language to us_english
Language 'us_english' is already the default.
Task to set the default language succeeded.

Configuration completed successfully.
Press <return> to continue.
```

Press Ctrl-x to go one step back in the panel sequence.

Now we can define and customize a backup server. The backup server is responsible for running backup and restore procedures, which are discussed in more detail in the Sybase documentation.

From the CONFIGURE SERVER PRODUCTS panel do the following:

```
CONFIGURE SERVER PRODUCTS

Products:

    Product                 Date Installed    Date Configured
1.  SQL Server              Feb 19 98 10:49   Feb 19 98 11:00
2.  Backup Server           Feb 19 98 10:50


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 2
```

Choose option **2** for Backup Server.

```
NEW OR EXISTING BACKUP SERVER

1.  Configure a new Backup Server
2.  Configure an existing Backup Server



Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
```

Choose option **1** to configure a new backup server.

```
ADD NEW BACKUP SERVER

1.  Backup Server name:  SYB_BACKUP



Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return: 1
Enter the name of the new Backup Server (default is 'SYB_BACKUP'):
TEC_BACKUP
```

Enter TEC_BACKUP as the new server name followed by pressing Ctrl-a.

```
BACKUP SERVER CONFIGURATION

1.  Backup Server errorlog:  /sybase/code/install/backup.log
2.  Enter / Modify Backup Server interfaces file information
3.  Backup Server language:  us_english
4.  Backup Server character set:  iso_1


Ctrl-a Accept and Continue, Ctrl-x Exit Screen, ? Help.

Enter the number of your choice and press return:
```

We left the rest of the configuration options as the defaults.

Press Ctrl-a to confirm your input. When prompted enter y to execute the backup server configuration. Our output is shown below:

```
Execute the Backup Server configuration now? y
Running task to update the Backup Server runserver file.
Task to update the Backup Server runserver file succeeded.
Running task to boot the Backup Server.
waiting for server 'TEC_BACKUP' to boot...
Task to boot the Backup Server succeeded.

Configuration completed successfully.
Press <return> to continue.
```

Press Return followed by Ctrl-a to exit the installation process.

3.3.1 Starting Sybase

At this point the processes dataserver and backupserver should be running. To verify enter the command:

```
$SYBASE/install/showserver
```

Our output looked like the following:

```
UID   PID  PPID   C    STIME    TTY  TIME CMD
root 23182 26252   4 10:27:42  pts/2  0:59 /sybase/code/bin/dataserver
    -d /sybase/code/master.dat -sTEC
    -e/sybase/code/install/errorlog
    -i/sybase/code

root 30598 28036   0 11:39:34  pts/0  0:00 /sybase/code/bin/backupsever
-ST EC_BACKUP -e/sybase/code/install/backup.log
     -I/sybase/code/interfaces
     -M/code/bin/sybmultbuf -Lus_english -Jiso_1
```

The scripts that started the two server processes were generated by the installation and placed in $SYBASE/install. If your SQL server and backup server are named TEC and TEC_BACKUP, then these scripts are named RUN_TEC and RUN_TEC_BACKUP. We edited both scripts to add environment variables needed by TEC.

Our RUN_TEC script looked like the following:

```
#!/bin/sh
export DSLISTEN=TEC
export DSQUERY=TEC

export SYBASE=/sybase/code
export PATH=$PATH:$SYBASE/bin

#
# SQL Server Information:
#  name:                        TEC
#  master device:               /sybase/code/master.dat
#  master device size:          20480
#  errorlog:                    /sybase/code/install/errorlog
#  interfaces:                  /sybase/code
#
/sybase/code/bin/dataserver -d/sybase/code/master.dat -sTEC \
-e/sybase/code/install/errorlog -i/sybase/code
```

Our RUN_TEC_BACKUP is also located in the same directory.

We also added these environment variables to our .profile to allow us, as an SQL client, to use the isql command to communicate with the SQL server:

```
export SYBASE=/code/sybase
export DSQUERY=TEC
export DSLISTEN=TEC
export PATH=$SYBASE/bin:$PATH
```

In order to reset the variables, you will need to re-execute your profile.

Next we stopped and restarted the data server using the edited version of RUN_TEC. Then we did the same for the backup server script RUN_TEC_BACKUP.

We then added two lines to /etc/inittab so that RUN_TEC and RUN_TEC_BACKUP would automatically be started at system boot:

```
:syb:2:wait:/usr/local/Tivoli/bin/aix4-r1
/TMF/TEC/sybase/install/RUN_TEC >/dev/console 2>&1
:sybk:2:wait:/usr/local/Tivoli/bin/aix4-r1
/TMF/TEC/sybase/install/RUN_TEC_BACKUP >/dev/console 2>&1
```

If you don't wish to edit /etc/inittab directly, you may define these lines with the AIX mkitab command:

```
mkitab "syb:2:wait:/usr/local/Tivoli/bin/aix4-r
1/TMF/TEC/sybase/install/RUN_TEC >/dev/console 2>&1"

mkitab -i syb "sybk:2:wait:/usr/local/Tivoli/bi
n/aix4-r1/TMF/TEC/sybase/install/RUN_TEC_BACKUP >/dev/console 2>&1"
```

You can test the communications with the SQL server by entering: `isql`, using a user ID of sa, and pressing Return when prompted for the password.

```
isql -Usa
Password:
1>
```

You are now in input mode, and can enter sql commands. For example, the command sp_helpdb shows you the known databases. As you will see in the following example, every isql command is followed by the word go to indicate that the statement is at an end. To leave isql mode, enter `quit`.

At this point you have the product Sybase installed and customized, but no user database for TEC has been defined, and the product TEC itself has not been installed. The next step is to install TEC, with all of its component parts. During this process, we enter the information that TEC will use to find and access the database. This information will be stored in a TME object named tec, of resource type RIM. Subsequently, we define the TEC user database to Sybase.

Chapter 4. Software Installation

The TEC is the central point for all the events that will be generated from the devices in our managed environment. This chapter covers the installation and customization of the TMR and TEC software. Using a step-by-step guide we create the required TEC consoles and define and activate a new rule base that will contain all the new classes and rules we define in the subsequent chapters of this redbook.

We install the TMR on server rs600020 and the TEC on server rs600028.

After the environment is set up we show how to access and verify that the RIM database is functioning correctly.

Based on the requirements discussed in Chapter 2, “Planning for the TEC Installation” on page 11 we devised a simple implementation plan. This is shown below:

- Install the TME Framework
- Customize the platform
- Install the TEC application
- Install the console application
- Verify the RIM connection has been created
- Create the TEC database tables
- Verify the TEC installation
- Create the TEC operators
- Create a new rule base
- Archive the TEC data

The final section provides additional information for the TEC database including information on the TEC tables.

To begin the installation procedure we performed each of these steps.

4.1 Installing the TME Framework

We followed the installation instructions contained in the TME manual *TME 10 Framework Planning and Installation Guide*. We installed the TME framework on our TME server rs600020.

The server was an AIX Version 4.2 with 128 MB of RAM and 2 GB of disk space. We created two filesystems where the TME applications will reside.

We used the the smit interface to create the following filesystems.

```
/usr/local/Tivoli of size 200 MBytes
/var/spool/Tivoli of size 100 MBytes
```

We mounted the Tivoli CD-ROM then proceeded to install the platform with the following commands:

```
mount /dev/rmt0 -r -v cdrfs /dev/cd0 /CDROM
mkdir /usr/local/Tivoli/install_dir
cd /usr/local/Tivoli/install_dir
/CDROM/wpreinst.sh
```

The next step was to execute the command wserver as prompted at the end of wpreinst.sh execution.

We left the defaults when prompted for the server install options for the directories where the Tivoli software will reside.

After the TMR software was installed we checked the permissions on the directory /etc/Tivoli. If the permissions are not rwxr-xr-x, modify these as follows:

```
chmod 755 /etc/Tivoli
```

We also checked the execute permissions on the Tivoli startup script, to set the correct execute permissions:

```
chmod 755 /etc/Tivoli/setup_env.sh
```

We added this script to our /etc/profile in order to set the TME variables when users log into the UNIX system. Add it to the .profile file for those users who will be using the product.

The next step was to check the TMR environment. We did this by checking to see whether the oserv daemon was running:

```
ps -ef | grep oserv
```

Also issue the commands to verify that the TMR has been set up:

```
odadmin
odadmin odlist
```

Next we installed the managed nodes using the desktop, as shown in Figure 3 on page 51. Alternatively we could have used the wclient command.

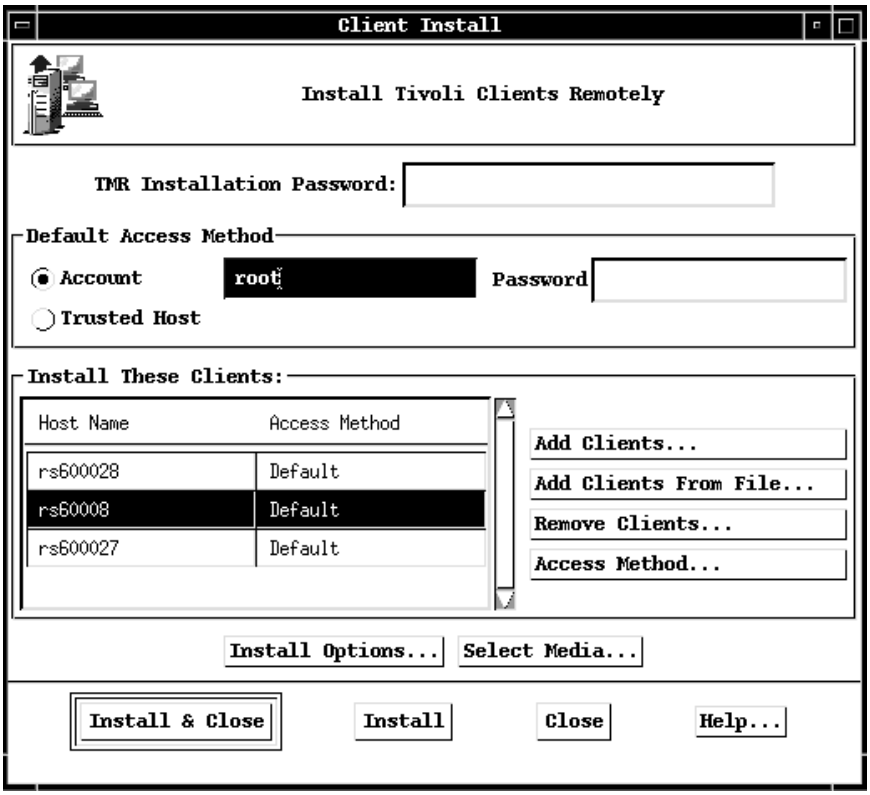


Figure 3. Creating the Managed Nodes

The managed nodes we installed are:

- rs600028 - TEC Server
- rs60008 - Oracle Server
- rs600027 - NetView Server

We also added an NT server called WTR05278. This NT server is used to show certain NT functions covered in this publication.

4.1.1 Setting Up the TMR

Next set the resource roles for the root administrator. Double-click on the **Administrator** icon from the desktop, then select **Set Resource Roles** as shown in Figure 4 on page 52.

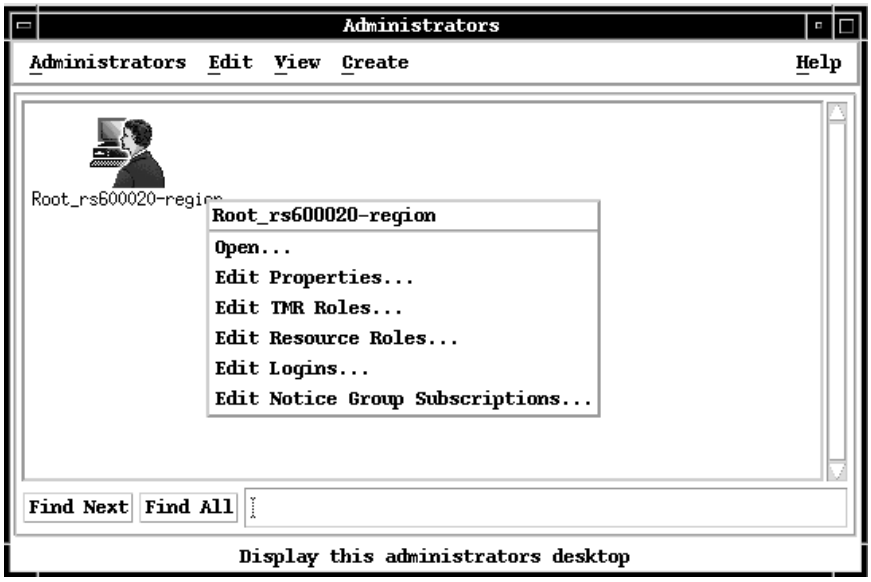


Figure 4. Selecting the Resource Roles

We moved all available roles to the Current Roles area in the window (see Figure 5).

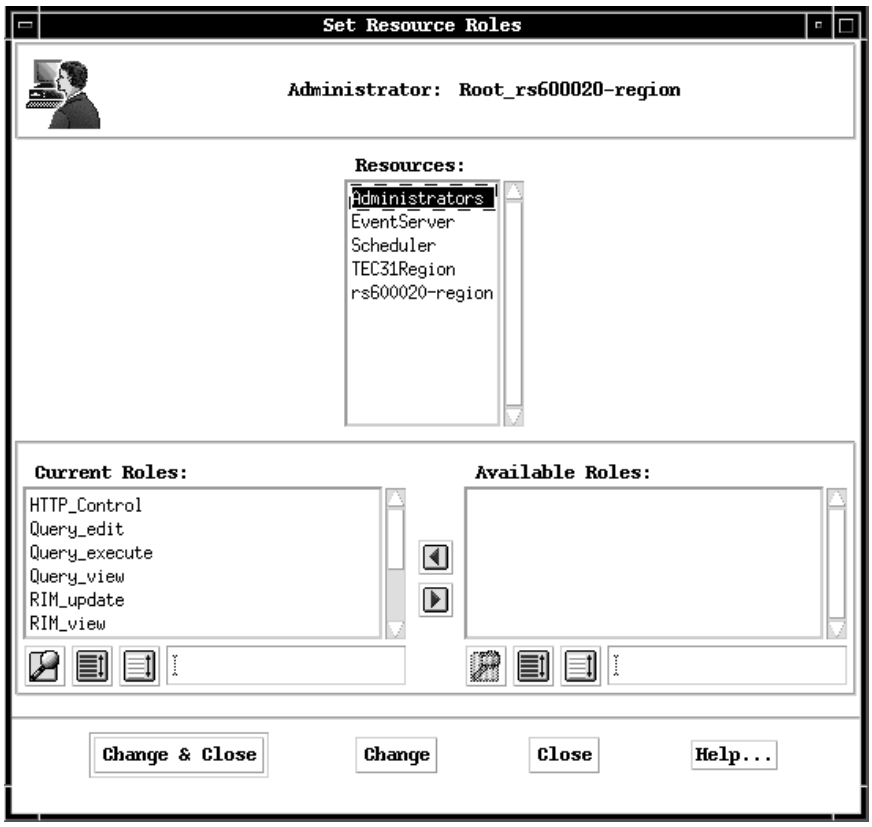


Figure 5. Setting the Resources for the Root Administrator

We then modified the login ID to also allow root users from the TEC server to access the Tivoli functions. This was done by adding the hostname for rs600028 in the field Add Login Name then pressing Return (see Figure 6 on page 53).

Figure 6. Setting the Login IDs for the TMR

This allowed users on our TEC server rs600028 to use to the Tivoli desktop.

Now we can modify our Tivoli environment. We create a policy region called TEC_MANAGEMENT that will contain all the new TE/C tasks. Distributed Monitoring configuration will use this new policy region.

The policy region was created using the command:

```
wcrtpr -a Root_rs600020-region TEC_MANAGEMENT
```

Define the following managed resources to the new policy region TEC_MANAGEMENT:

```
wsetpr ManagedNode @PolicyRegion:TEC_MANAGEMENT
wsetpr TaskLibrary @PolicyRegion:TEC_MANAGEMENT
wsetpr ProfileManager @PolicyRegion:TEC_MANAGEMENT
```

As we install the additional components throughout this redbook we show how to build on this configuration. The desktop has now been set up. The next stage is to install and customize the TEC.

Note

It is very good practice during the installation process that regular Tivoli database backups are performed.

The backup was performed using the command:

```
wbkupdb -d /usr/local/Tivoli/backups/TMR_Mar16
```

To check the TME database and update the database if any problems or inconsistencies are found, issue the command:

```
wchkdb -u
```

We checked the Tivoli database every time we installed new software.

4.2 Installing the TEC

While building the TEC environment we used a combination of command line interface (CLI) commands and the graphical user interface (GUI).

We started the install program for the TEC software as shown in Figure 7.

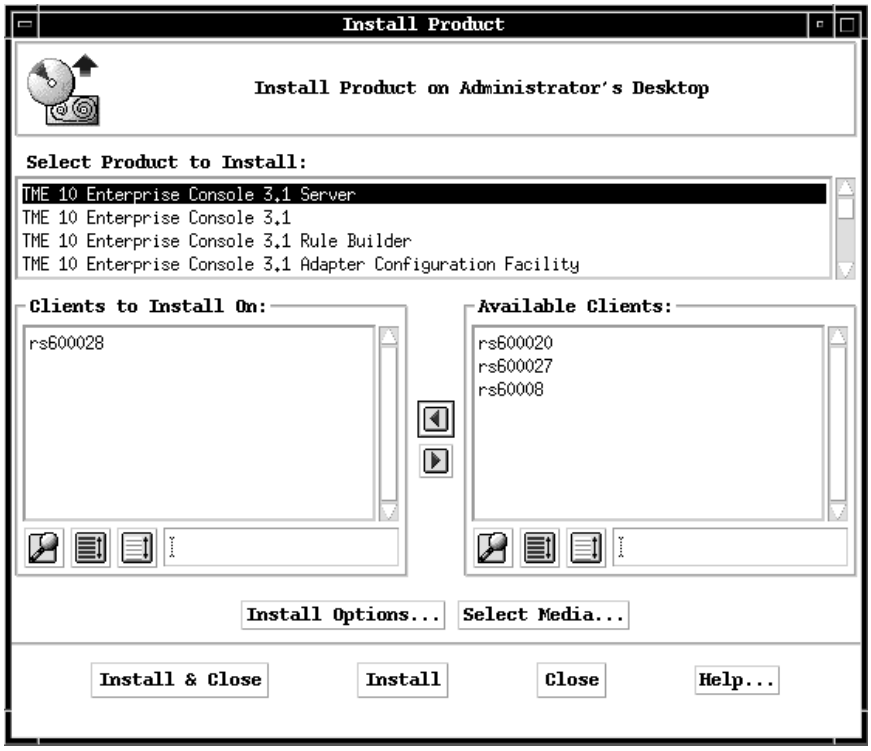


Figure 7. Install the TEC Server Application

We set the path to point to the TEC CD-ROM, then proceeded to select **TME 10 Enterprise Console 3.1 Server** and click on **Install**.

During the TEC server installation you will be prompted for database variables as installation options. With this information, the installation process will create the RIM definition automatically.

Using the variables from our Oracle configuration defined in Chapter 3, “Installing the RDBMS” on page 19 we entered the required values as shown in Figure 8 on page 55.

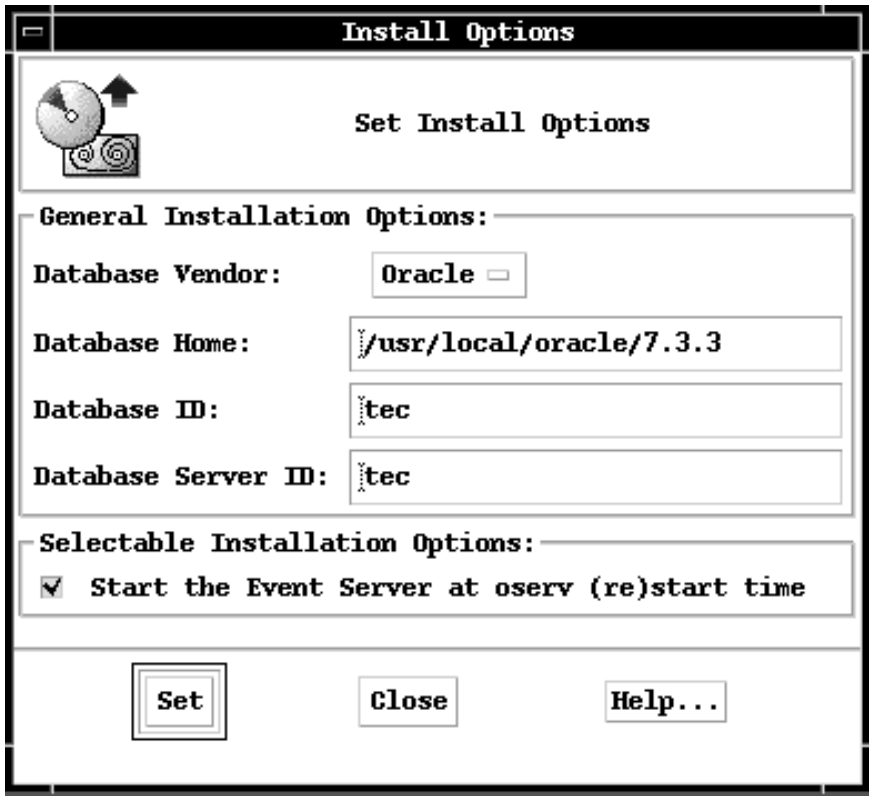


Figure 8. Setting the Database Options for Oracle

If you are using Sybase as your RDBMS, enter the values shown in Figure 9 on page 56.

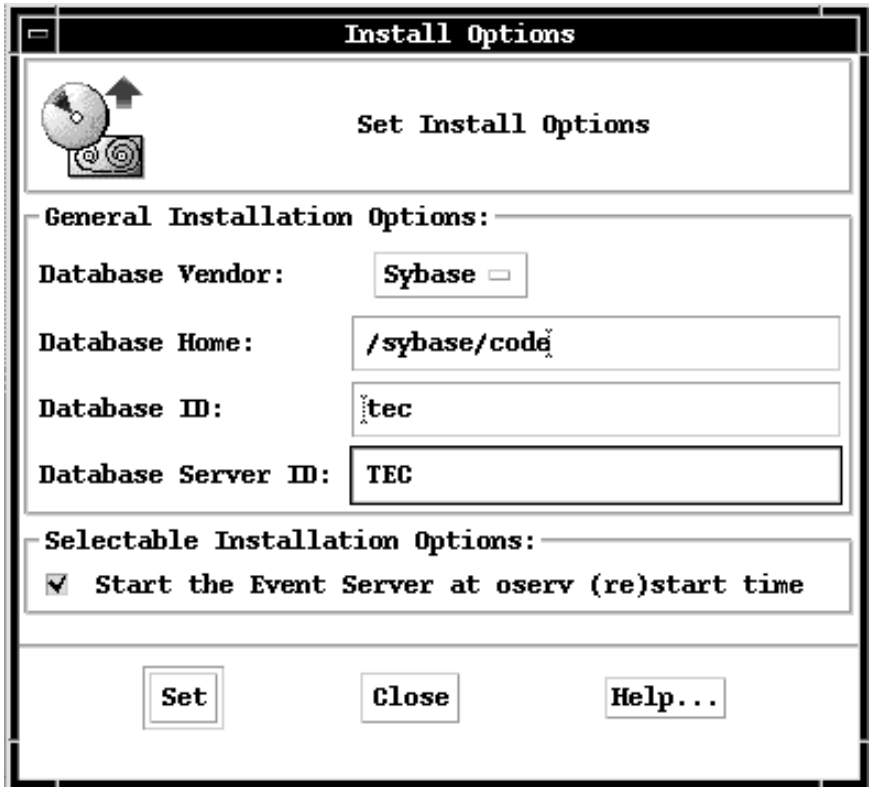


Figure 9. Setting the Database Options for Sybase

4.2.1 Installing the TEC Console

After the TEC server application was installed we then installed the TEC console software on rs600028. This is the application shown as TME 10 Console 3.1 in the Install product window. We installed this application on machine rs600028. Next we create the console.

To create the TEC console for the root user issue the command:

```
wcrtconsole -h @ManagedNode:rs600028 @Root_rs600020-region
```

The desktop now looks like the window shown in Figure 10 on page 57.

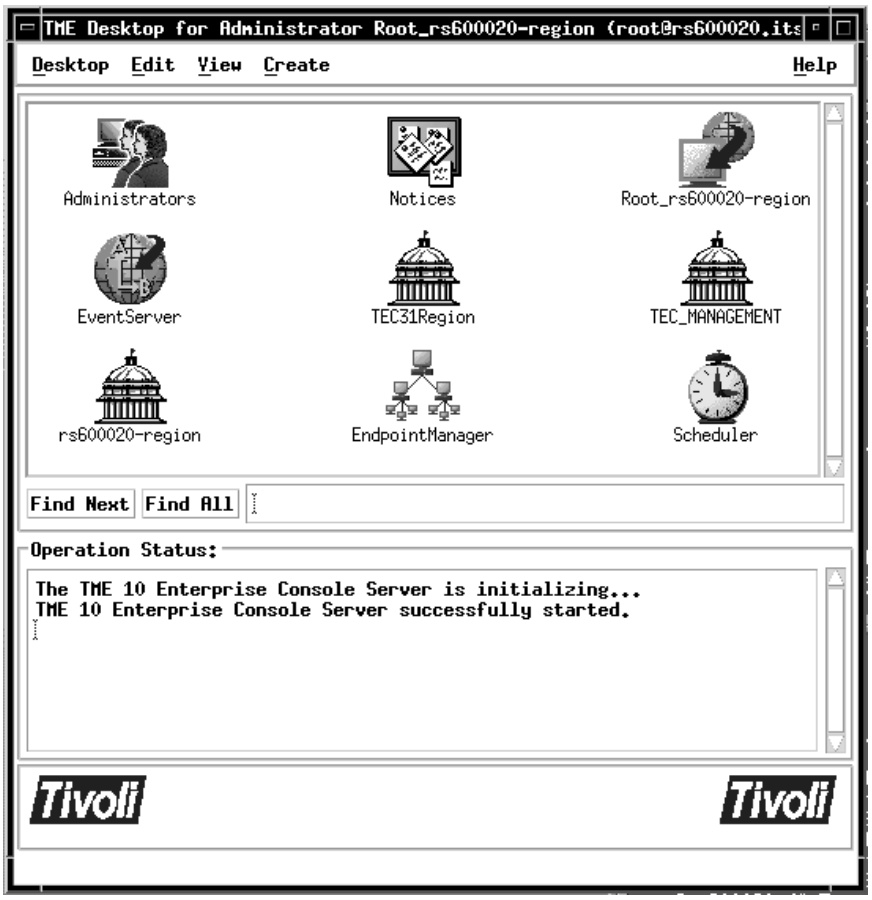


Figure 10. Tivoli Desktop after Creating the Console

We need to add the required resources for the root administrator in order for the Administrator to run and to customize the TEC application. The window is shown in Figure 11 on page 58.

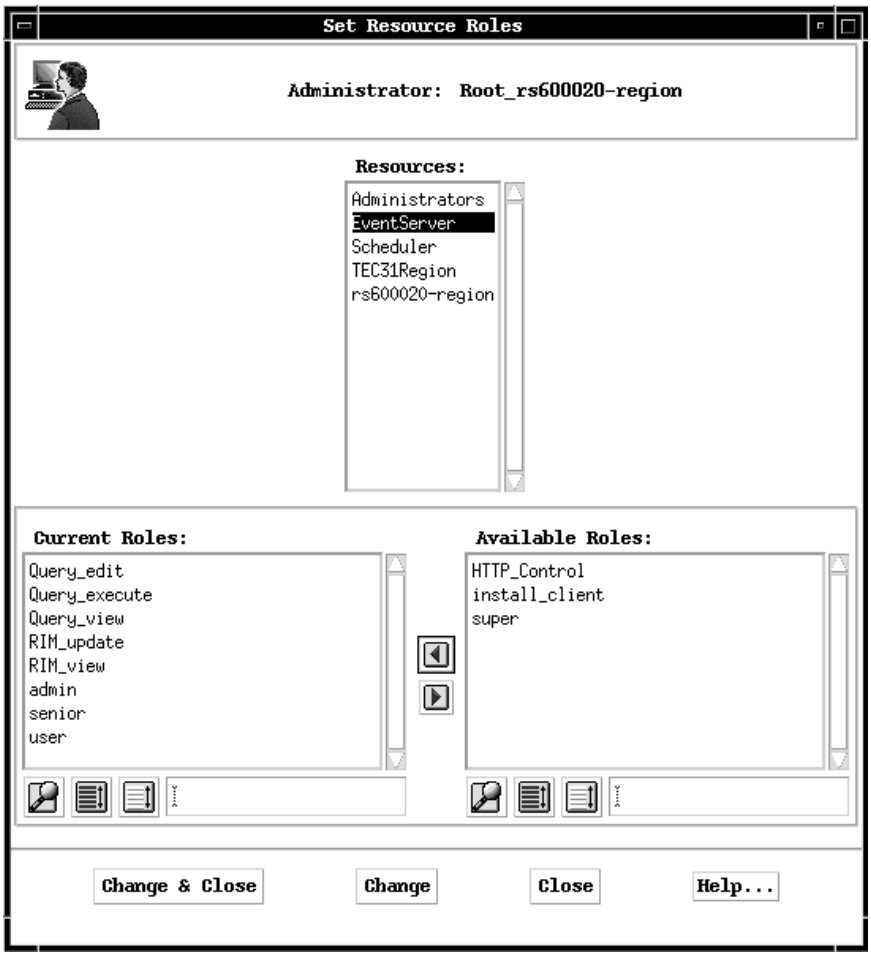


Figure 11. Setting the Resource Roles for the TEC

Select the **EventServer** resource followed by the available roles, then move these roles to the Current role window. When you have completed click on **Change & Close**. The Root_rs600020-region now has the required permissions for the TEC. Next we can verify the RIM installation.

4.3 Verifying the RDBMS Interface Module (RIM) Installation

The RIM object should have been defined as part of the TEC installation process. To verify this, issue the command:

```
wgetrim @RIM:tec
```

For Oracle the output should look like this:

```
RIM Host:       rs600028
RDBMS User:     tec
RDBMS Vendor:   Oracle
Database ID:    tec
Database Home:  /usr/local/oracle/7.3.3
Server ID:      tec
```

For Sybase the output should look like this:

```
RIM Host:       rs600028
RDBMS User:     tec
RDBMS Vendor:   Sybase
Database ID:    tec
Database Home:  /sybase/code
Server ID:      TEC
Instance Home:
```

The RDBMS database values are defined during the installation of the RDBMS covered in Chapter 3, “Installing the RDBMS” on page 19.

The RIM object has been successfully created. If however, incorrect information was entered at the time the TEC was installed, you can either change the current parameters or re-create the RIM object.

4.3.1 Re-Creating the RIM

To modify the RIM parameters use the command wsetrim.

To remove the RIM definition from the TEC, use the following command:

```
wdel @RIM:tec
```

Next, create the RIM object by issuing one of the the following commands:

- For Oracle:

  ```
  wcrtrim -v Oracle -h rs600028 -d tec -u tec\
          -H /usr/local/Oracle/7.3.3 -s tec tec
  ```

- For Sybase:

  ```
  wcrtrim -v Sybase -h rs600028 -d tec -u tec -H /sybase/code\
          -s TEC tec
  ```

To look up the current RIM definition use the command:

```
wlookup -r RIM tec
```

Our output was:

```
1194067385.2.21#RIM::RDBMS_Interface#
```

To view the parameters issue the command:

```
wgetrim tec
```

There is a useful command for debugging the RIM. To activate RIM tracing use the following command: `wrimtrace @RIM:tec ERROR`

You can also use the keyword INFORMATION instead of ERROR. The output is recorded in the file /tmp/rim_db_log.

To turn the rim trace off issue the command wrimtrace @RIM:tec TRACE_OFF.

On one occasion we had the following message occur because we had incorrectly entered the RIM parameters:

```
The TME 10 Enterprise Console Server is initializing...
Tue Mar 24 11:16:18 EST 1998 (17): system problem:
'The RIM Object tec was not found.'
```

Another message that we had during the project is shown in Figure 12. This occurred when we started the TEC.

Figure 12. RIM Error

On both instances we re-created the RIM object.

The process that is started for the Oracle RIM is called RIM_Oracle and is also started when the TEC server is initialized. The RIM is now defined. The database tables now need to be created for the RDBMS. The TEC application contains the required sql scripts to perform this action.

4.4 Creating the TEC Tables for Oracle

Once the RIM is active the TEC database tables can be created. This is performed using the shell script installed on the TEC server in the directory $BINDIR/TME/TEC/sql.

The same command is used independent of what RDBMS is installed. To execute the command do the following:

```
cd $BINDIR/TME/TEC/sql
./cr_tec_db.sh
```

The script will initially resolve the RIM parameters defined for the TEC. In our example the output was as follows:

```
Looking up TME 10 system information...
The RIM object within this TMR is configured to define the
following database:

   Database Vendor:    Oracle
   Database Server ID: tec
   Database Home:      /usr/local/oracle/7.3.3
   Database ID:        tec
   Database User ID:   tec

If this is correct, enter 'Y' to continue.  Otherwise, check the
configuration of the local RIM object or run this script from
the RIM host within the TMR you wish to configure.
Y

What size, in megabytes, do you want to make the
TME 10 Enterprise Console Server database? (default is 50)
40


At the Password: prompt, enter the DB 'sys' user ID's password...

+ sqlplus sys@tec @./cr_db.ora.sql

SQL*Plus: Release 3.3.3.0.0 - Production on Tue Mar 24 16:56:43 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

Enter password:

Connected to:
Oracle7 Server Release 7.3.3.0.0 - Production Release
With the distributed, replication, parallel query and Spatial Data options
PL/SQL Release 2.3.3.0.0 - Production

Tablespace created.

Tablespace altered.
Profile created.
CREATE USER tec
Grant succeeded.
Commit complete.

Disconnected from Oracle7 Server Release 7.3.3.0.0 - Production Release
With the distributed, replication, parallel query and Spatial Data options
PL/SQL Release 2.3.3.0.0 - Production
```

4.4.1 Removing the TEC Database Tables

If for some reason you may want to remove the TEC database (maybe due to the incorrect sizing of the initial database or being directed to do so from support), then you can execute the command:

```
./rm_tec_db.sh
```

The following output is then displayed:

```
+ sqlplus sys@tec @./rm_db.ora.sql

SQL*Plus: Release 3.3.3.0.0 - Production on Tue Mar 24 10:58:27 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

Enter password:

Connected to:
Oracle7 Server Release 7.3.3.0.0 - Production Release
With the distributed, replication, parallel query and Spatial Data options
PL/SQL Release 2.3.3.0.0 - Production

User dropped.
Profile dropped.
Tablespace altered.
Tablespace dropped.
Tablespace altered.
Tablespace dropped.
Commit complete.

Disconnected from Oracle7 Server Release 7.3.3.0.0 - Production Release
With the distributed, replication, parallel query and Spatial Data options
PL/SQL Release 2.3.3.0.0 - Production
```

4.4.2 Checking the Oracle Tables

To verify that the tables have been created, use the sqlplus command with a password of tectec:

```
sqlplus tec@tec
```

Execute the following SQL statement:

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
TEC_T_CLT_REQ_LOG              TABLE
TEC_T_EVT_REC_LOG              TABLE
TEC_T_EVT_REP                  TABLE
TEC_T_GEM_THRESHOLD            TABLE
TEC_T_ISA                      TABLE
TEC_T_OP_ASS_LOG               TABLE
TEC_T_ROLE                     TABLE
TEC_T_SEVERITY                 TABLE
TEC_T_SLOTS_EVT                TABLE
TEC_T_STATUS_EVENT             TABLE
TEC_T_STATUS_TASK              TABLE

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
TEC_T_TASK_REP                 TABLE
```

4.5 Creating the TEC Tables for Sybase

For Sybase we issued the command:

```
./cr_tec_db.sh
```

The output is shown below:

```
Looking up TME 10 system information...
The RIM object within this TMR is configured to define the
following database:

   Database Vendor:    Sybase
   Database Server ID: TEC
   Database Home:      /sybase/code
   Database ID:        tec
   Database User ID:   tec

If this is correct, enter 'Y' to continue.  Otherwise, check the
configuration of the local RIM object or run this script from
the RIM host within the TMR you wish to configure.
```

Confirm by entering `Y` followed by pressing Return. The script will now prompt for additional information:

```
What device do you want to put the TME 10 Enterprise Console Server
database in? (default is 'master')
press return

What size, in megabytes, do you want to make the
TME 10 Enterprise Console Server database? (default is 50)
40

At the Password: prompt, enter the DB 'sa' user ID's password...

+ isql -Usa -i ./cr_db.syb.sql
Password:
CREATE DATABASE: allocating 1024 pages on disk 'master'
Extending database by 15616 pages on disk master
Database option 'trunc log on chkpt' turned ON for database 'tec'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
New user added.
(return status = 0)
+ isql -Utec -Ptectec -i ./cr_tbl.syb
rs600028:/usr/local/Tivoli/bin/aix4-r1/TME/TEC/sql > exit
Script command is complete on Wed Mar 18 09:28:38 EST 1998.
```

4.5.1 Removing the TEC Database Tables for Sybase

To remove the Sybase tables issue the command ./rm_tec_db.sh.

```
+ isql -Usa -i ./rm_db.syb.sql
Password:
No user with the specified name exists in the current database.
(return status = 1)
Account locked.
Login dropped.
(return status = 0)
```

4.5.2 Checking That the Sybase Tables Have Been Created

In this section we show a number of ways to access the database tables for Sybase.

A transaction is a unit of work that modifies data. Each database is coupled with its own transaction log, in which every transaction of every user of a given database is automatically logged.

You can access and examine the database and its tables with the isql command.

Before executing Sybase set the variables as follows:

```
export DSLISTEN=TEC export DSQUERY=TEC export SYBASE=/sybase/code export
PATH=$PATH:$SYBASE/bin
```

From the Sybase server start the isql session:

```
isql -Usa
```

After you've entered your password, you should see the following prompt, and you're ready to enter isql commands:

```
1>
```

In order to query the TEC database, you must indicate which database you will be using:

```
1> use tec
2> go
1>
```

To display the database options in use for the TEC database, enter:

```
1> sp_dboption
2> go
```

To examine how much space you have left in your database, enter:

```
1> sp_spaceused
2> go
```

The output is shown as follows:

```
database_name                  database_size
 ------------------------------ -------------
 tec                            32.0 MB

 reserved        data            index_size      unused
 --------------- --------------- --------------- ---------------
 2486 KB         964 KB          616 KB          906 KB
```

This is just a brief sampling of the commands available for systems administration functions. For more information on these topics, read *Sybase SQL Server System Administration Guide*.

To list the tables, both user tables and system tables, defined in the database tec, enter:

```
1> sp_help
2> go
```

The output is as follows:

```
 Name                           Owner
 ------------------------------ ----
 tec_t_clt_req_log              tec
 tec_t_evt_rec_log              tec
 tec_t_evt_rep                  tec
 tec_t_gem_threshold            tec
 tec_t_isa                      tec
 tec_t_op_ass_log               tec
 tec_t_role                     tec
 tec_t_severity                 tec
 tec_t_slots_evt                tec
 tec_t_status_event             tec
 tec_t_status_task              tec
 tec_t_status_task              tec
```

The definition for the individual fields contained in each of the TEC tables can be seen in the file cr_tbl.syb.

A further discussion on the TEC database tables and how to view and archive these tables can be found in 4.7, “Additional TEC Database Information” on page 74.

With the TEC now functioning we can test whether the console is receiving events.

4.5.3 Verify the TEC Installation

The TEC software is now installed; however, we want to verify that the event console is receiving events correctly.

First we check that the console server is running by issuing the wstatesvr command. Also there should be a red arrow across the TEC server icon on the desktop. The command output is:

```
The TME 10 Console Server is Running
```

Use the command ps -ef | grep tec to verify that the TEC daemons are running. The list of daemons is shown below:

```
tec_server
tec_rule
tec_dispatch
tec_reception
tec_task
```

You can verify the installation of your TEC event server with the following command:

```
wlookup -ar EventServer
```

This should show you something like the following output:

```
EventServer     1276743309.2.19#Tec::Server#
```

Using the command wpostemsg we can send a dummy event to the console as follows:

```
wpostemsg -m "Test_Event" TEC_Notice rs600027
```

The event should be displayed on the TEC console, as shown in Figure 13.

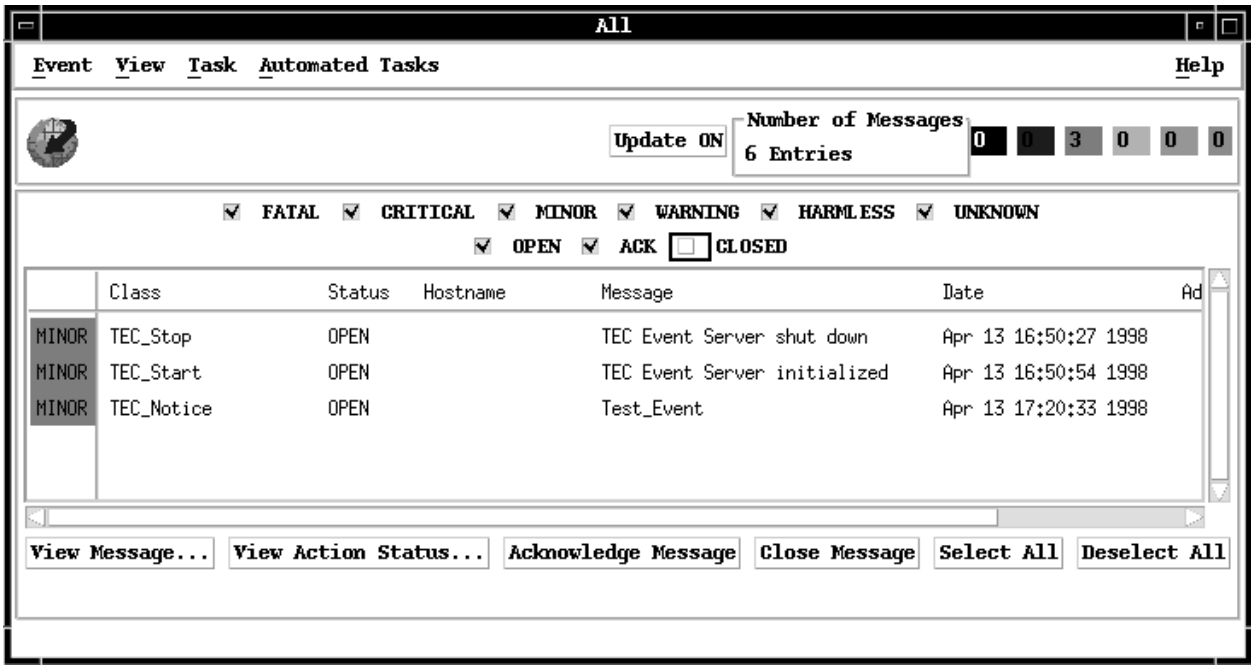


Figure 13. The Test Event Displayed on the TEC Console

Also you can check whether the event has been successfully sent to the TEC by issuing the command:

```
wtdumprl -o DESC | more:
```

The output should look something like:

```
1 5984 0 892488033(Apr 13 13:20:33 1998)
### EVENT ###
TEC_Notice;
source=rs600027;
msg=Test_Event;
origin=9.24.104.4;
END

### END EVENT ###
PROCESSED
```

Use the command wtdumper to see if the event has been fully processed by the TEC daemons. Our output is shown below:

```
rs600027  9.24.104.4    OPEN
admin  MINOR
Apr 13, 1998 17:20
Test_Event
```

The TEC processes write any error conditions to a number of files located on the TEC server for each of the TEC daemons:

- /tmp/tec_master
- /tmp/tec_reception
- /tmp/tec_task
- /tmp/tec_dispatch
- /tmp/tec_rule

The configuration file determining what messages are written to these log files is located in the directory $BINDIR/TME/TEC and is called .tec_diag_config. Some of the lines contained in this file are listed below:

```
# tec_master
#############

tec_master Highest_level        error
tec_master Master               error   /tmp/tec_master
tec_master Master_Msg           error   /tmp/tec_master
tec_master Master_Synchro       error   /tmp/tec_master
tec_master Master_Exec          error   /tmp/tec_master

# tec_reception
################

tec_reception Highest_level     error
tec_reception Tec_Reception     error   /tmp/tec_reception
tec_reception Recv_Msg          error   /tmp/tec_reception
tec_reception Recv_Proc         error   /tmp/tec_reception
tec_reception Recv_Synchro      error   /tmp/tec_reception

# to debug the DB modules
tec_reception DB_Utils          error   /tmp/tec_reception
tec_reception Enum              error   /tmp/tec_reception
tec_reception Evt_Rec_Log       error   /tmp/tec_reception
tec_reception Op_Ass_Log        error   /tmp/tec_reception
```

We did not have to amend this file.

Some of the errors that occur and are recorded in these files are shown below:

The tec_rule error log:

```
Mar 27 13:59:34 tec_rule 30624 ERR tec_rule_msg.c:129: Comm. with Master
Mar 27 13:59:34 tec_rule 30624 ERR tec_exit_msg.c:247: Process T/EC Rule
```

The tec_dispatch error log:

```
Mar 26 20:55:33 tec_dispatch 30684 ERR tec_ipc_accept.c:145: ipc_accept
Mar 27 13:59:34 tec_dispatch 36508 ERR tec_disp_msg.c:139: T/EC Dispatch
Mar 27 13:59:34 tec_dispatch 36508 ERR tec_exit_msg.c:247: Process T/EC
```

4.6 Creating the TEC Operators

We require two additional operators in addition to the root user, so that we can show different events on each of the user's consoles. To create the TEC operator environment we perform the tasks listed below:

- Create AIX users
- Create TME administrators
- Create the event consoles
- Define the required event groups
- Assign the relevant event groups to the administrators
- Customize the console for each administrator

The two additional TEC administrators will want to see different events. The network management operator wants to see just the SNMP events generated from NetView, whereas the system management operator wants to see the events generated from the AIX servers.

We started by defining two AIX users who will subsequently be assigned different responsibilities with regard to the TEC.

We added the AIX user IDs netop and sysop with these commands:

```
mkuser home=/home/netop shell=/usr/bin/ksh netop
mkuser home=/home/sysop shell=/usr/bin/ksh sysop
```

We added the following lines to each user's .profile:

```
. /etc/Tivoli/setup_env.sh
tivoli
```

Next we can create the administrators. Since we want to assign relevant notice groups to each user, we looked at the available notice groups with the command wlookup -ar TMF_Notice.

The user netop was defined with user and admin privileges, whereas sysop was defined with user, admin, and senior privileges. For the UNIX user sysop we assigned the name System_Operator, and for the user netop we assigned the name Network_Operator.

We used the following commands to define the operators:

```
wcrtadmin -l sysop -r global,user:admin:senior              \
          -n "TME Administration" -n "Enterprise Console"   \
          -u sysop System_Operator
wcrtadmin -l netop -r global,user:admin -n
        "Enterprise Console" -u netop Network_Operator
```

For the System operator we added the event server icon and the TEC regions to the desktop. To create the event server icon on the user's desktop, issue the commands:

```
wln /Administrators/Root_rs600020-region/EventServer         \
    /Administrators/System_Operator

wln /Administrators/Root_rs600020-region/TEC31Region         \
    /Administrators/System_Operator

wln /Administrators/Root_rs600020-region/TEC_MANAGEMENT      \
    /Administrators/System_Operator
```

For the Network_Operator we added only the rs600020-region icon:

```
wln /Administrators/Root_rs600020-region/rs600020-region     \
         /Administrators/Network_Operator
```

For both operators we created their associated event console:

```
wcrtconsole -h @ManagedNode:rs600028 @Network_Operator
wcrtconsole -h @ManagedNode:rs600028 @System_Operator
```

To verify the setup, issue the command:

```
wlookup -r EnterpriseClient -a
```

Our output was:

```
Root_rs600020-region  1276743309.2.23#TecClient::tec_client#
Network_Operator      1276743309.2.24#TecClient::tec_client#
System_Operator       1276743309.2.26#TecClient::tec_client#
```

4.6.1 Creating a Rule Base

Our next task was to create the a new rule base called CanTire. We did this as follows:

```
mkdir /usr/tec_rulebase
wcrtrb -d rs600028:/usr/tec_rulebase CanTire
```

This will create the following three subdirectories under /usr/tec_rule base:

```
TEC_CLASSES
TEC_RULES
TEC_TEMPLATES
```

To view the new rule base definition we issued the command:

```
wlsrb -d
```

We copied the root.baroc and the tec.baroc files from the default rule base directory to our development environment, so that we can always make reference to them, regardless of the state of customization of the TEC.

```
cp /usr/local/Tivoli/bin/aix4-r1/TME/TEC/default_rb/TEC_CLASSES/root.baroc \
   /usr/development/classes
cp /usr/local/Tivoli/bin/aix4-r1/TME/TEC/default_rb/TEC_CLASSES/tec.baroc \
   /usr/development/classes
```

It is not necessary to import root.baroc and tec.baroc into the new rule base CanTire, as this happens automatically when the rule base is created. When you are ready to import additional baroc files into your rule base, you will use the wimprbclass command.

Compile and load the new rule base. (We compiled with the -t option to activate the trace logic.)

```
wcomprules -t CanTire
wloadrb CanTire
```

After you load a new rule base for the first time, you must stop and re-start the event server, since this is a new set of class definitions.

```
wstopesvr
wstartesvr
```

Let's list the event groups predefined during the TEC installation. To do this, issue the wlseg command.

We decided to remove the default event groups with the exception of the All group, and create our own event groups.

To remove the four default event groups, use the command:

```
wdeleg Network Performance Security System
```

In order to plan meaningful event groups, you may want to list the classes known to your rule base. We did this with the following command:

```
wlsrbclass CanTire
```

We then defined two event groups, additionally to the All group, one with network events for our Network_Operator, and one with system events for our System_Operator:

```
wcrteg -b collection Network
wcrteg -b dog Servers
```

Event groups need filters in order to determine what events get included. The following are our filters for the two event groups:

```
waddegflt -s nvserverd Network
waddegflt -s LOGFILE Servers
waddegflt -s NT Servers
waddegflt -s SENTRY -u SYBASE Servers
waddegflt -s SENTRY -u ORACLE Servers
```

Each administrator who opens an event console must have at least one event group assigned. Here is what we assigned to our Network_Operator and our System_Operator:

```
wassigneg Network_Operator Network user admin
wassigneg System_Operator Servers user admin senior
wassigneg Root_rs600020-region All user admin senior
wassigneg Root_rs600020-region Network user admin senior
wassigneg Root_rs600020-region Servers user admin senior
```

The list of commands shown below will assist in viewing and modifying the rule base configuration:

```
wlsrb
wlsrbclass
wlsrbrules
wlscurrb
wlssrc
wlseg -f -a
```

4.6.2 Defining the Classes and Rule Structure

We developed a number of rules and class definition files. (See Chapter 11, “Rule Development” on page 235 for a detailed description.) The process below documents how we added the new definitions to the TEC.

Our entire development environment is in /usr/development, which in turn has the subdirectories:

- Rules containing all our new rule set files
- Classes containing all our new baroc files
- Tasks containing all our new tasks
- Tests containing our testing scripts

When new class definitions were to be added, we imported them into our rule base with the command:

```
wimprbclass /usr/development/classes/xxx.baroc CanTire
```

When new rule sets were to be added, we imported them into the rule base with the command:

```
wimprbrules /usr/development/rules/xxx.rls CanTire
```

4.6.3 Customizing the Desktop

We made the following changes to the console display.

From the desktop (see Figure 14) select **Sort Messages**.

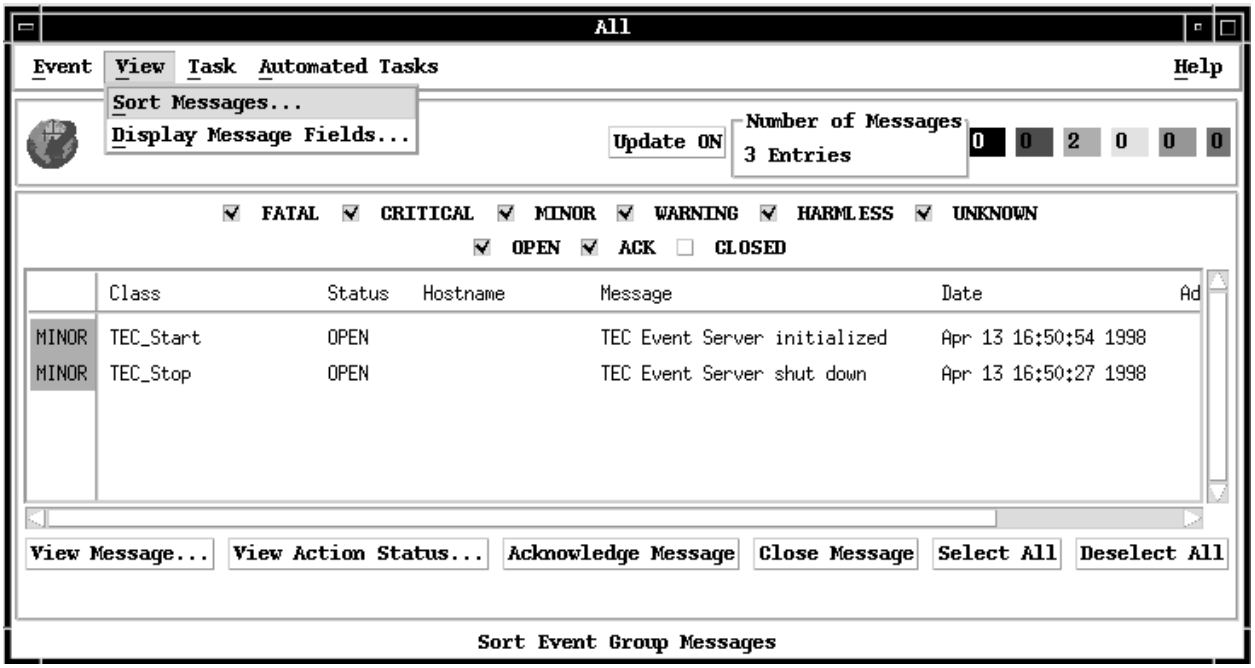


Figure 14. Sorting the Event Messages

We moved the Date field from the Message Fields window over to the Sort by window in order to display the most recent events first (see Figure 15 on page 73).

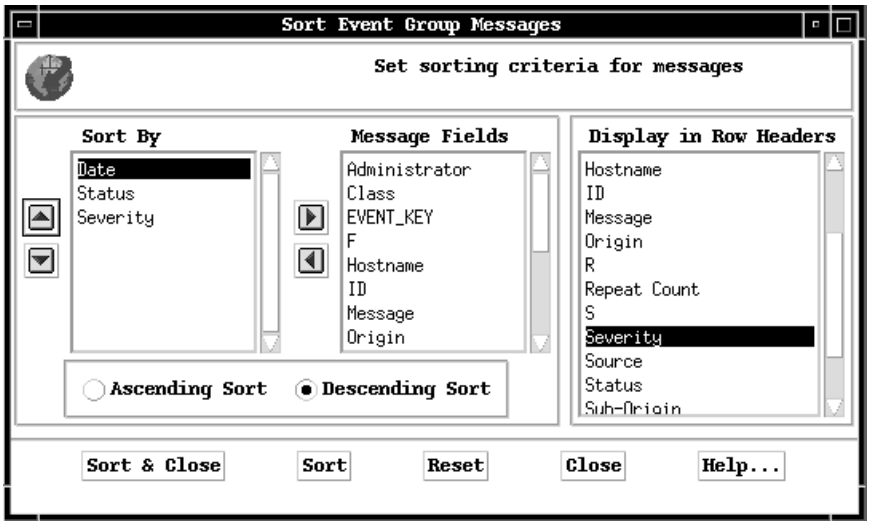


Figure 15. Sorting the Event Messages

We can examine the TEC event server environment using the command wlsesvrcfg.

```
Time allowed for server initialization: 300 seconds
Maximum number of events buffered in memory: 500 events
Time to keep logged events in reception log: 86400 seconds
Event cache size: 1000 events
Time to keep closed events in cache: 86400 seconds
Time to keep non-closed events: 15552000 seconds
Log client requests: NO
Time to keep logged client requests: 86400 seconds
Trace rule execution: NO
Rule trace output file: /tmp/rules.trace
```

We used most of the default values, but changed one parameter to keep closed events in cache for one week, rather than one day, using the command:

```
wsetesvrcfg -k 604800
```

The new settings look like this:

```
Time allowed for server initialization: 300 seconds
Maximum number of events buffered in memory: 500 events
Time to keep logged events in reception log: 86400 seconds
Event cache size: 1000 events
Time to keep closed events in cache: 604800 seconds
Time to keep non-closed events: 15552000 seconds
Log client requests: NO
Time to keep logged client requests: 86400 seconds
Trace rule execution: NO
Rule trace output file: /tmp/rules.trace
```

4.7 Additional TEC Database Information

This section details more information regarding the TEC database tables.

The TEC database contains several application tables. Here we describe the most important of the application tables in the TEC database.

- The event reception log (tec_t_evt_rec_log) holds all received events that are logged by the reception engine. Therefore, this table is the one place in which all events that arrive (as well as all information from each event adapter) are stored, before they are parsed, analyzed, or changed.

  Arriving events that do not match any event source defined to your event server are logged in the event reception log along with all others, but they are not available on your event console.

- What is referred to as the *event repository* consists of information in the event cache, as well as what is stored in two tables. The table tec_t_evt_rep is a collection of all events found to be valid and from known event sources, after they have been processed by the event server and subjected to rule processing. This information is updated when event information changes.

  Not all information from the event reception log is repeated in this table, and additional information is present.

  You can use the wtdumper command to retrieve information from the database files, for example, to show all events with a severity of fatal: `wtdumper -w severity=60`

  To show all the events that are acknowledged use `wtdumper -w status=20`.

  As long as an event remains in the OPEN state, it remains in the event repository.

  The information that you see on your enterprise console reflects the information that has not yet aged, according to how you configured your event server. However, older information may still be accessible in your database. You can cause events to be deleted from the database with the wtdbclear command.

  The table tec_t_slots_evt is the second table that makes up what is known as the event repository. It stores the additional slots and their values that do not get stored in tec_t_evt_rep, but which were received from an event adapter. An example would be a slot that you define in a self-written event adapter, which has no predefined position in the fixed fields of tec_t_evt_rep.

  If you stop and restart your event server, the event cache is rebuilt from these two tables tec_t_evt_rep and tec_t_slots_evt.

- The *client request log* has the table name tec_t_clt_req_log. This acts as a reception log for task execution requests.

- The *task repository* has the table name tec_t_task_rep. This contains all predefined TEC tasks, the commands they execute, their start time, current status, and exit status.

- The table tec_t_isa is a table that contains all known event classes and subclasses in the currently loaded rule base, stored in their hierarchical relationships.

  The table tec_t_severity contains a list of the six predefined severity levels for events and their internal codes:

  - UNKNOWN=10
  - HARMLESS=20
  - WARNING=30
  - MINOR=40
  - CRITICAL=50
  - FATAL=60

- The table tec_t_status_event contains a list of the four predefined status codes:
  - OPEN=0
  - RESPONSE=10
  - ACK=20
  - CLOSED=30
- The table tec_t_status_task contains a list of the four predefined internal codes for the TEC tasks. These are:
  - RUNNING=0
  - SUCCESS=1
  - FAILURE=2
  - UNKNOWN=3

The final section contains some pointers on how to archive the TEC database data.

4.8 Archiving the TEC Event Data Using Oracle

We did not investigate the Oracle archiving software. This is documented in the Oracle Administration manual. However we did manage to use the export function, which is sufficient to archive the event data.

The format of this command is:

```
exp sys/oracle owner=tec file=/tmp/tecdb.export
```

The output is shown below:

```
Export: Release 7.3.3.0.0 - Production on Wed Mar 25 15:44:36 1998
Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

Connected to: Oracle7 Server Release 7.3.3.0.0 - Production Release
With the distributed, replication, parallel query and
Spatial Data option
PL/SQL Release 2.3.3.0.0 - Production
Export done in US7ASCII character set

About to export specified users ...
About to export TEC's objects ...
exporting snapshots
exporting snapshot logs
exporting job queues
exporting refresh groups and children
exporting database links
exporting sequence numbers
exporting database links
exporting sequence numbers
exporting cluster definitions
about to export TEC's tables via Conventional Path ...
exporting table              TEC_T_CLT_REQ_LOG          0 rows export
exporting table              TEC_T_EVT_REC_LOG         19 rows export
exporting table                 TEC_T_EVT_REP          9 rows export
exporting table            TEC_T_GEM_THRESHOLD          0 rows export
exporting table                     TEC_T_ISA        740 rows export
exporting table               TEC_T_OP_ASS_LOG          9 rows export
exporting table                     TEC_T_ROLE          0 rows export
exporting table                TEC_T_SEVERITY          6 rows export
exporting table               TEC_T_SLOTS_EVT          7 rows export
exporting table             TEC_T_STATUS_EVENT          4 rows export
exporting table              TEC_T_STATUS_TASK          4 rows export
exporting table                 TEC_T_TASK_REP          0 rows export
exporting synonyms
exporting views
exporting stored procedures
exporting referential integrity constraints
exporting triggers
exporting posttables actions
Export terminated successfully without warnings.
```

To import the database tables:

```
imp tecdb.export
```

Another way to back up the Oracle data file is by using a filesystem file for the database. You can shut down the database using the command dbshut and then copy the tablespace file to a backup area.

For example, on our system we would issue the command:

```
cp /u01/app/oracle/product/7.3.3/dbs/TEC_DATA_TS ./rdbms/backup/
```

However we did not test this mechanism.

4.9 Archiving the TEC Data Using SYBASE

Regularly backing up your databases is important to insure your ability to recover in the event of a hardware or software crash.

We describe a basic procedure for backing up and recovering the Sybase system databases and the TEC user database. If you require more details, please read the appropriate chapters in the *Sybase SQL Server System Administration Guide*.

The dump database isql command backs up a database and its associated transaction log. It can be run dynamically, so users can continue to work during backup.

If you were to manually back up all the important databases (system as well as TEC) to a file on disk, you could enter:

```
isql -Usa -P (password)
1> dump database master to "/TEC_backups/master.Feb17"
2> go
1> dump database model to "/TEC_backups/model.Feb17"
2> go
1> dump database sybsystemprocs to "/TEC_backups/sybsys.Feb17"
2> go
1> dump database tec to "/TEC_backups/tec.Feb17"
2> go
```

4.9.1 Restoring the Sybase Database

The restore command is more complicated and can be reviewed in the Sybase manuals.

Chapter 5. Distributed Monitoring and Scripts

This chapter covers the TME 10 Distributed Monitoring configuration to provide monitoring support for our TEC management servers. The Distributed Monitoring application provides facilities for monitoring resources on managed nodes. This includes system resources such as CPU usage and application daemons. When a monitored resource exceeds a pre-defined threshold then an event is generated and sent to the TEC.

The initial set of examples show what we configured for the Distributed Monitoring monitors by creating monitors and distributing these monitors to our managed nodes. Next we show how to use a shell script to monitor a filesystem using wpostemsg to monitor resources on an AIX server that will generate a TEC event. This example does not use the Distributed Monitoring infrastructure.

5.1 Installing the Distributed Monitoring Application

To install the application using the desktop select **Desktop->Install->Install Product**.

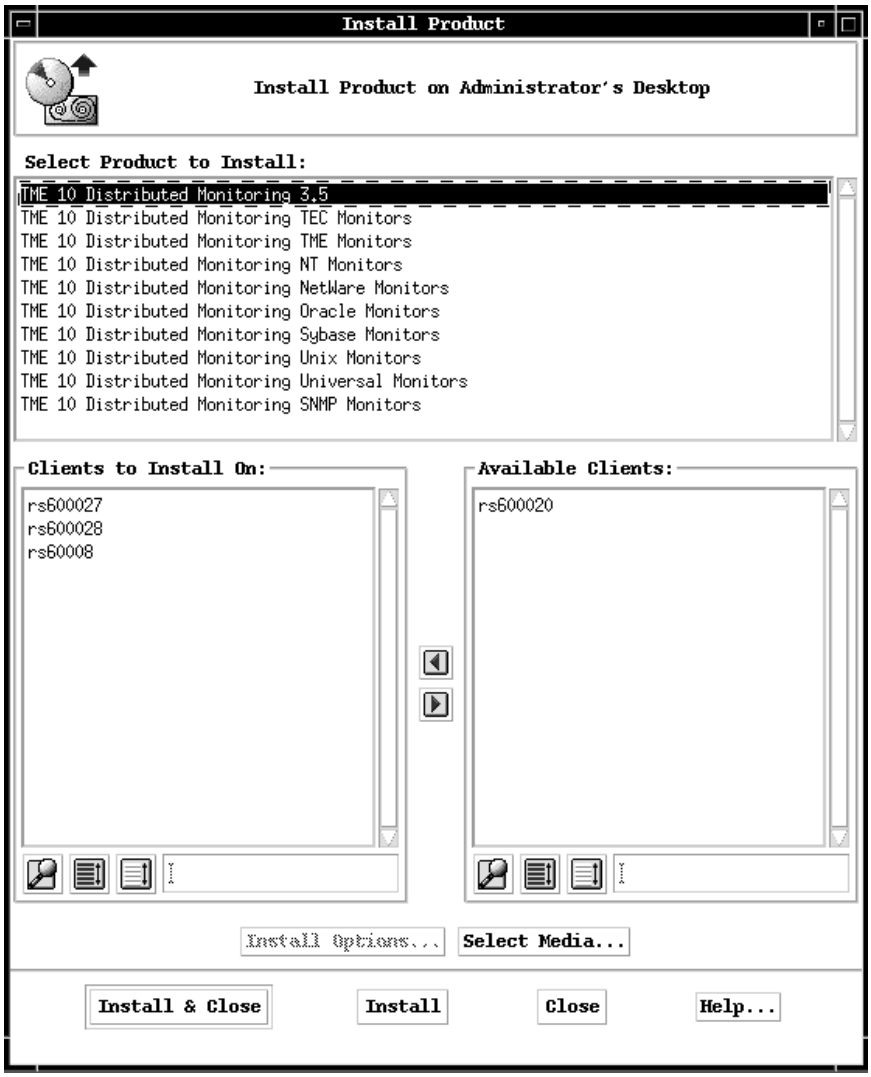


Figure 16. Installing Distributed Monitoring

We set the medium to the Distributed CD-ROM, then selected **Install** (see Figure 16).

In our environment we also have a copy of the CD-ROM on our filesystem to install the products from the command line. We installed the software on the following nodes:

- rs600020 - TMR server
- rs600028 - TEC server
- rs600027 - NetView server
- WTR05278 - NT server

To install from the command line issue the command:

```
winstall -c /:/tivoli32cd/distribmon.V35 -i SENT35.IND -y rs600020   \
rs600028 wtr05278 rs600027
```

Note: Remember that Distributed Monitoring 3.5 must be installed not only on the TMR server, but also on every managed node on which monitors will run. The monitoring collections themselves need to be installed only on the TMR server and

will be distributed to the managed nodes through profile managers and their subscribers.

Next we can install and configure the required monitors.

5.1.1 Installing the Monitors

The examples show how to install the monitors. We configure the monitors as shown in the list below:

- Oracle/Sybase for monitor rs600028
- TEC monitors for monitor rs600028
- UNIX monitors for monitor rs600027
- TMR monitors for rs600020
- NT monitors for WTR05278

We wanted to send alerts generated by Distributed Monitoring to the Tivoli Enterprise Console. In Distributed Monitoring 3.5 you can specify more than one event server as the destination for Sentry events. You can use this function to provide redundancy. When the primary event server goes down, the secondary event server will automatically start to receive the events from the Sentry monitor, which attempts to send events to an event server in the order in which the event servers are defined.

Before installing the monitoring collections from the command line we need to know what the package name is for each of the individual components. We found that out by looking at all *.IND files on our installation medium. You can examine the files on your CD in this way. The software packages we installed are named as follows:

ORA_MON Oracle Monitors

SYB_MON Sybase Monitors

NTMON35 Windows NT Monitors

TEC_MON TEC Monitors

TMR_MON TMR Monitors

UNX_MON UNIX Monitors

These monitoring collections are installed on the TMR server. We used the command sequence shown below:

```
winstall -c /:/tivoli32cd/distribmon.V35 -i ORA_MON.IND -y rs600020
winstall -c /:/tivoli32cd/distribmon.V35 -i SYB_MON.IND -y rs600020
winstall -c /:/tivoli32cd/distribmon.V35 -i NTMON35.IND -y rs600020
winstall -c /:/tivoli32cd/distribmon.V35 -i TEC_MON.IND -y rs600020
winstall -c /:/tivoli32cd/distribmon.V35 -i TMR_MON.IND -y rs600020
winstall -c /:/tivoli32cd/distribmon.V35 -i UNX_MON.IND -y rs600020
```

The required monitors are now defined in the TMR. Next we can define the profile manager.

5.1.2 Creating a New Profile Manager for Distributed Monitoring

Within the policy region TEC_MANAGEMENT we defined the resources ProfileManager and SentryProfile as managed resources:

```
wsetpr ProfileManager @PolicyRegion:TEC_MANAGEMENT
wsetpr SentryProfile @PolicyRegion:TEC_MANAGEMENT
```

Next we created the following profile manager to contain our monitors:

```
wcrtprfmgr TEC_MANAGEMENT TEC_MONITORS
```

Now we can create the Sentry profiles, a separate one for each group of monitors:

```
wcrtprf @ProfileManager:TEC_MONITORS SentryProfile ORACLE
wcrtprf @ProfileManager:TEC_MONITORS SentryProfile SYBASE
wcrtprf @ProfileManager:TEC_MONITORS SentryProfile TEC
wcrtprf @ProfileManager:TEC_MONITORS SentryProfile TMR
wcrtprf @ProfileManager:TEC_MONITORS SentryProfile NT
wcrtprf @ProfileManager:TEC_MONITORS SentryProfile UNIX
```

Finally, the subscribers to the profile manager must be defined:

```
wsub @ProfileManager:TEC_MONITORS @ManagedNode:rs600020 \
    @ManagedNode:rs600027 @ManagedNode:rs600028\
    @ManagedNode:wtr05278
```

The profile manager looks like Figure 17 on page 83.

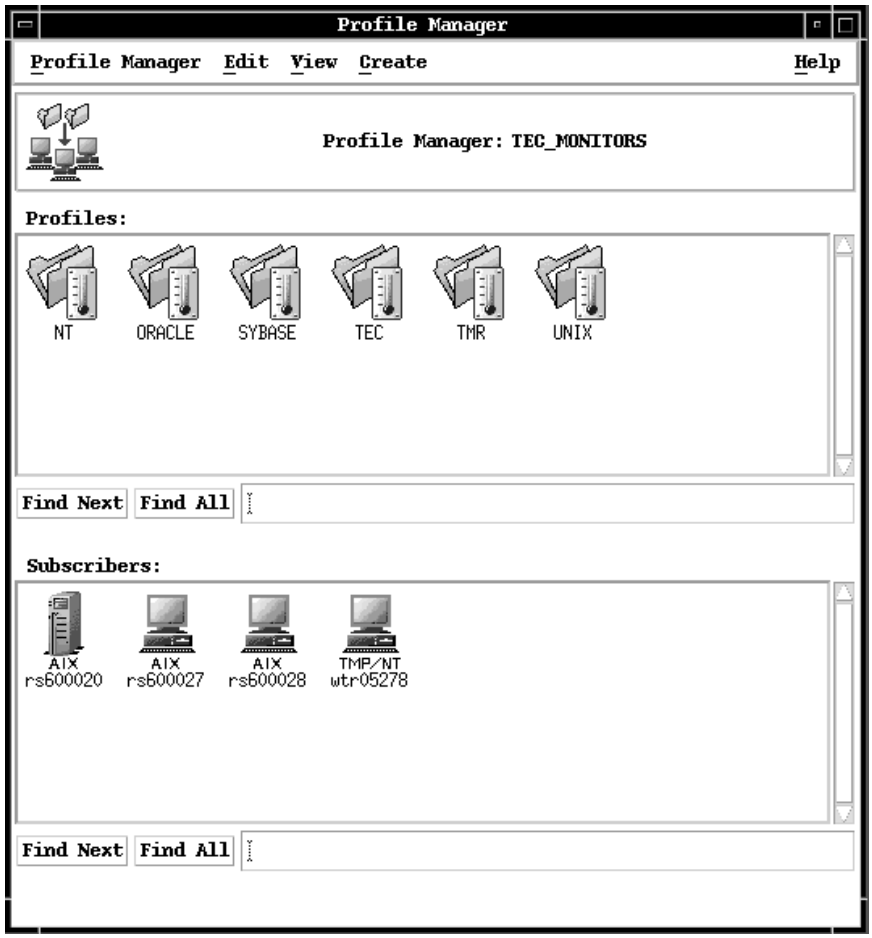


Figure 17. The TEC_MONITORS Profile Manager

The profile manager contains all the profiles and monitors used for managing our TEC environment.

5.1.3 The Distributed Monitoring Class Definitions

As with all event classes, the Sentry event classes have to be defined within the TEC event server. To do this, we imported the appropriate Sentry baroc files into our current and active rule base called CanTire.

All Sentry baroc files are shipped with the product and are located in the directory /usr/local/Tivoli/bin/generic/SentryMonitors.

We took the precaution of copying all Sentry baroc files to our development area as follows:

```
mkdir /usr/development/dist_mon
cp /usr/local/Tivoli/bin/generic/SentryMonitors/*
 /usr/development/dist_mon
```

A sample of the sentry.baroc file is shown below:

```
TEC_CLASS :
        Sentry2_0_Base ISA EVENT
        DEFINES {
                distrib_admin:  STRING, dup_detect= no ;
                response_level: STRING, dup_detect= no ;
                probe_arg:      STRING, dup_detect= no ;
                prev_value:     STRING, dup_detect= no ;
                value:          STRING, dup_detect= no ;
                effective_value:STRING, dup_detect= no ;
                collection:     STRING, dup_detect= no ;
                info:           STRING, dup_detect= no ;
                monitor:        STRING, dup_detect= no ;
                units:          STRING, dup_detect= no ;
                relation:       STRING, dup_detect= no ;
                relation_delta: STRING, dup_detect= no ;
                msg:            STRING, dup_detect= no ;
        };
END

TEC_CLASS :
        Sentry3_5_Base ISA Sentry2_0_Base
        DEFINES {
                probe:          STRING, dup_detect= no ;
        tmr:            STRING, dup_detect= no ;
                dispatcher:     STRING, dup_detect= no ;
        };
END
```

The Sentry class structure assigns a top Sentry event class called Sentry3_5_Base, which is defined in the file Sentry.baroc. Each Sentry monitor can send events of different classes, which are defined in additional baroc files as subclasses of the Sentry3_5_Base superclass.

Below is an example of the oracle.baroc file:

```
TEC_CLASS :
        Oracle ISA Sentry3_5_Base;
END
TEC_CLASS :
        Database_space_utilization ISA Oracle;
END
TEC_CLASS :
        Database_lock_utilization ISA Oracle;
END
TEC_CLASS :
        Database_user_connections ISA Oracle;
END
TEC_CLASS :
        Database_CPU_utilization ISA Oracle;
END
TEC_CLASS :
      Database_Performance ISA Oracle;
END
```

Note: The oracle.baroc and sybase.baroc files as delivered require a number of lines to be removed, as some event classes are duplicated.

To define the Sentry classes we needed, we imported them into our rule base named CanTire:

```
wimprbclass Sentry.baroc CanTire
wimprbclass TMEMonitors.baroc CanTire
wimprbclass TECMonitors.baroc CanTire
wimprbclass oracle.baroc CanTire
wimprbclass sybase.baroc CanTire
wimprbclass tivoli.baroc CanTire  (for the UNIX monitors)
```

Remember to import the Sentry.baroc file *before* the other Sentry monitor baroc files, since all other classes are derived from the general Sentry3_5_Base event class.

Then create or modify an event group to include the Sentry event classes, with filters if you need them, and then assign this event group to the appropriate event consoles. To create an event group, use the wcrteg command. To add filters to an event group, use the waddegflt command.

It may be that an existing event group already displays the Sentry events. Check what event groups are defined and which administrators see which event groups with the command wlseg -af.

Here is an example of creating an event group named DIST_MON defining a filter of source SENTRY and sub_source DIST_MON and assigning this event group to the administrators Server_Operator and Root_rs600020-region.

```
wcrteg DIST_MON
waddegflt -s SENTRY -u DIST_MON DIST_MON
wassigneg @Server_Operator DIST_MON user admin senior
```

When the TEC console is started the new event group will show on the event group desktop. We did not want our Network_Operator to view these events.

At this point the Distributed Monitoring application and the monitoring collections are installed on the appropriate nodes, and the event classes are known to the TEC event server.

5.1.4 Creating a Monitor for Oracle

This example shows the steps involved in creating and distributing an Oracle monitor to check for the Oracle table space available.

To define a new monitor for Oracle, double-click on the **TEC_MANAGEMENT** region, then double-click on the **TEC_MONITORS** profile manager.

To add a monitor from the desktop, move the cursor to the Oracle icon and select **Edit Properties** (see Figure 18 on page 86).

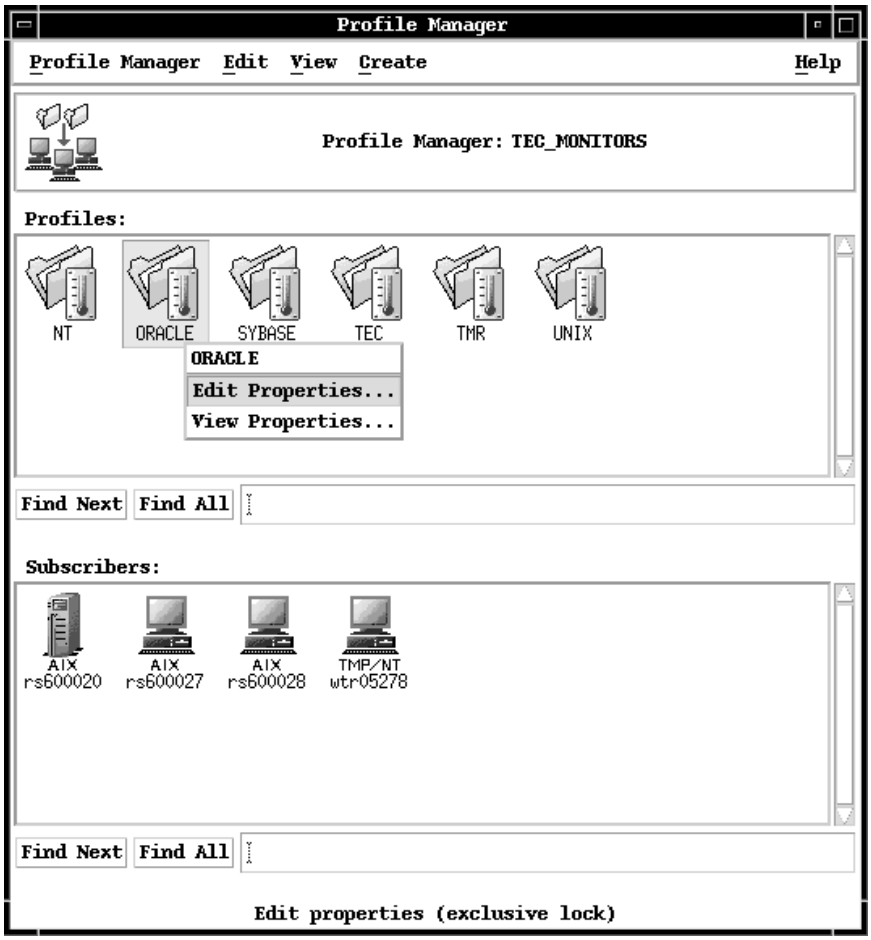


Figure 18. Edit Properties

The main monitor configuration window (Figure 19) will appear.

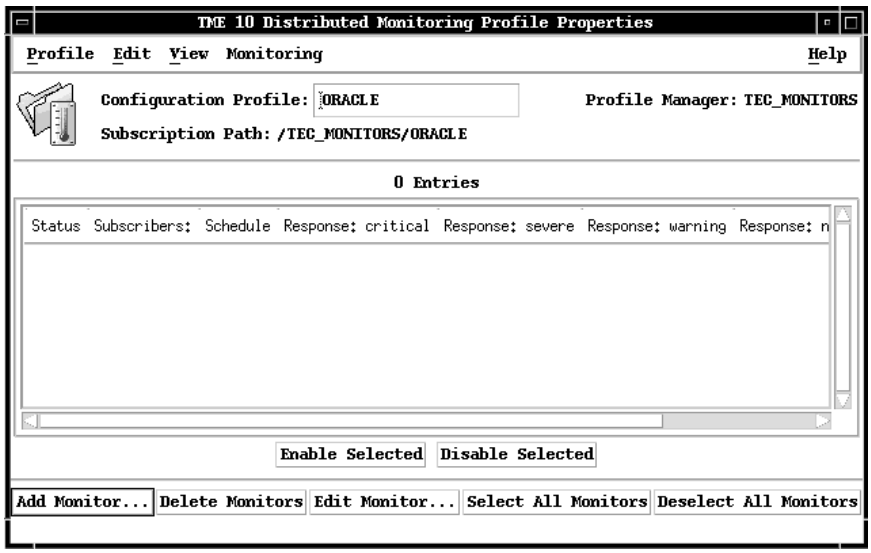


Figure 19. Profile Properties

From here select **Add Monitor** to enter the monitor configuration.

From Figure 20 on page 87 scroll down the Monitoring Collections window until you see Oracle. Double-click on **Oracle**. This will display all the available monitors for Oracle.

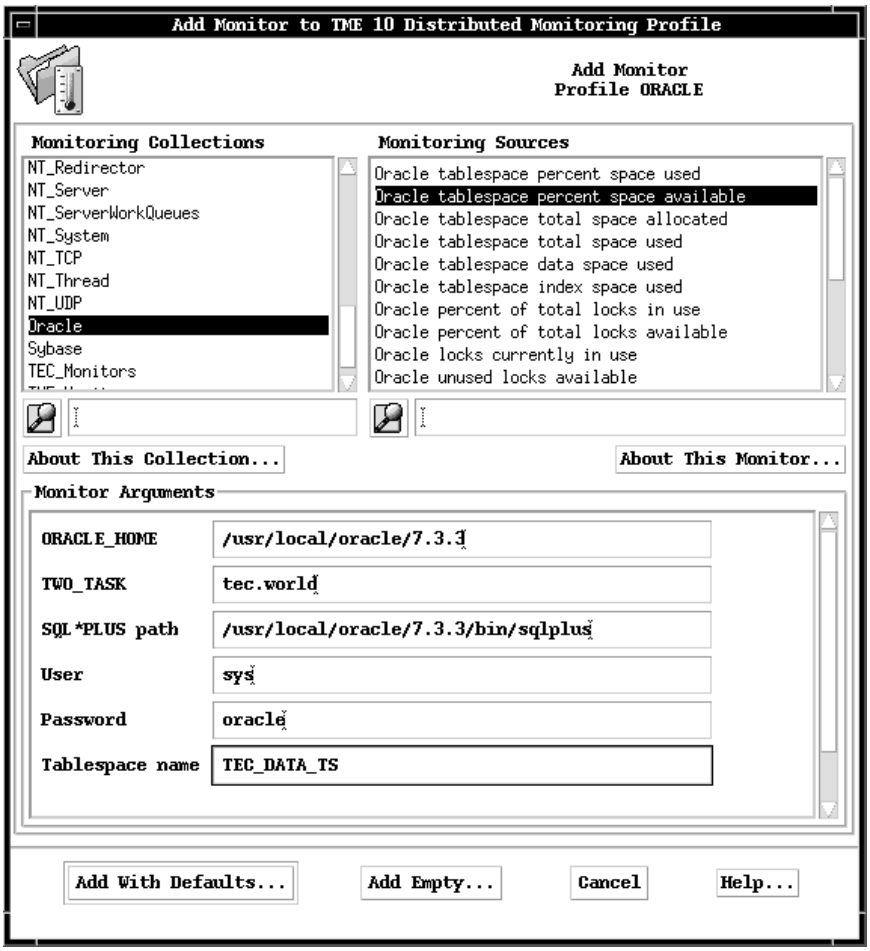


Figure 20. Creating A New Monitor

We selected **Oracle tablespace percent space used**.

When adding the Oracle monitor we were prompted for additional Oracle information. This information is the values we used when defining our Oracle database in Chapter 3, “Installing the RDBMS” on page 19. We entered the values for the Oracle database and then clicked on **Add Empty**.

From Figure 21 on page 88 we set the trigger level to be critical when the tablespace is greater than 90 percent. When this limit is exceeded we want to send a FATAL TEC event to our TEC server.

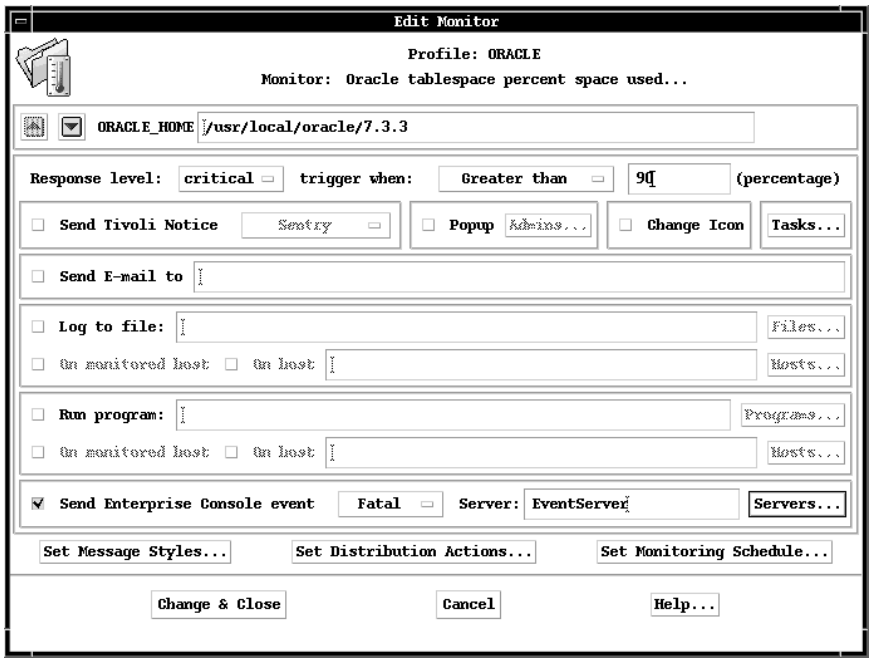


Figure 21. Edit Profile

To set the event server click on **Servers** from Figure 21.

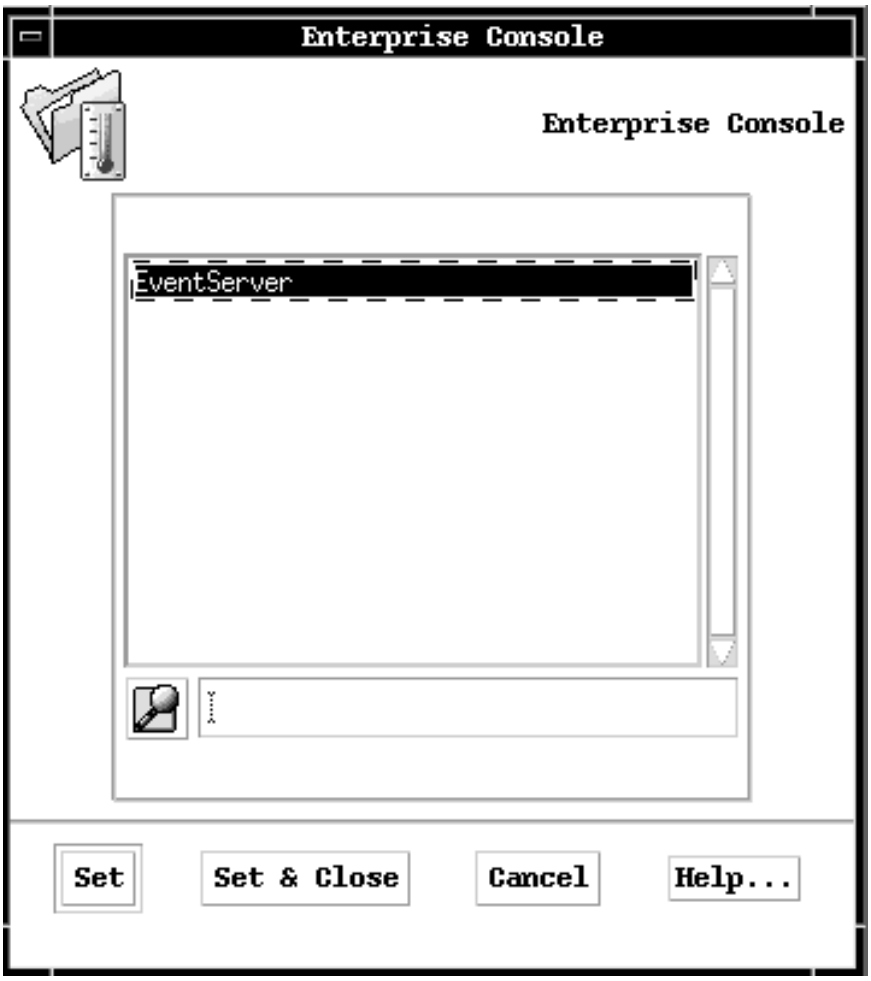


Figure 22. Selecting the Event Server

Select **Set & Close** to select the default TEC server as shown in Figure 22.

To set the frequency that the monitor will execute click on **Set Monitoring Schedule**. Figure 23 on page 90 shows the monitoring schedule.

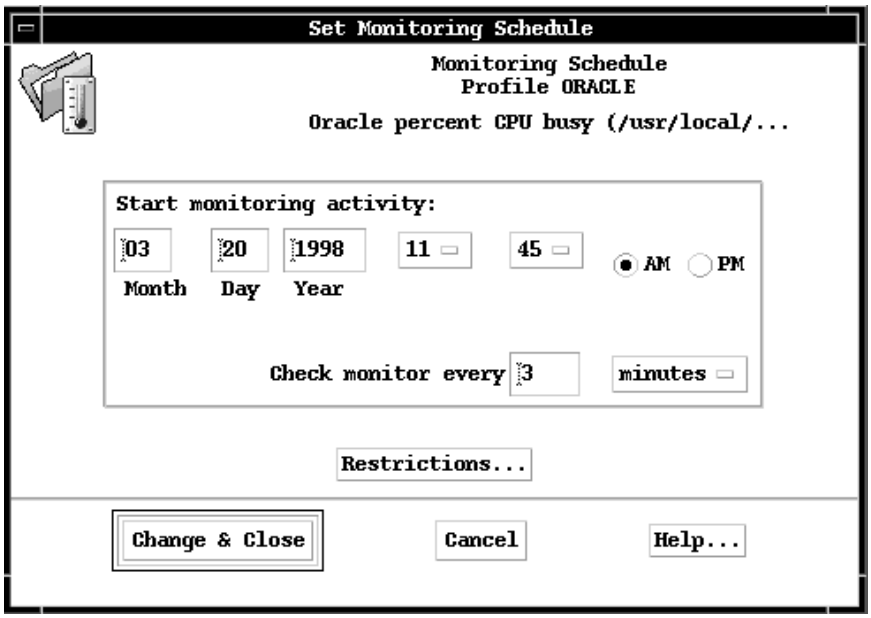


Figure 23. Setting the Timers

We also added a second Oracle monitor to check for the CPU usage.

To save the profile select the pull-down menus **Profile->Save** (see Figure 24).

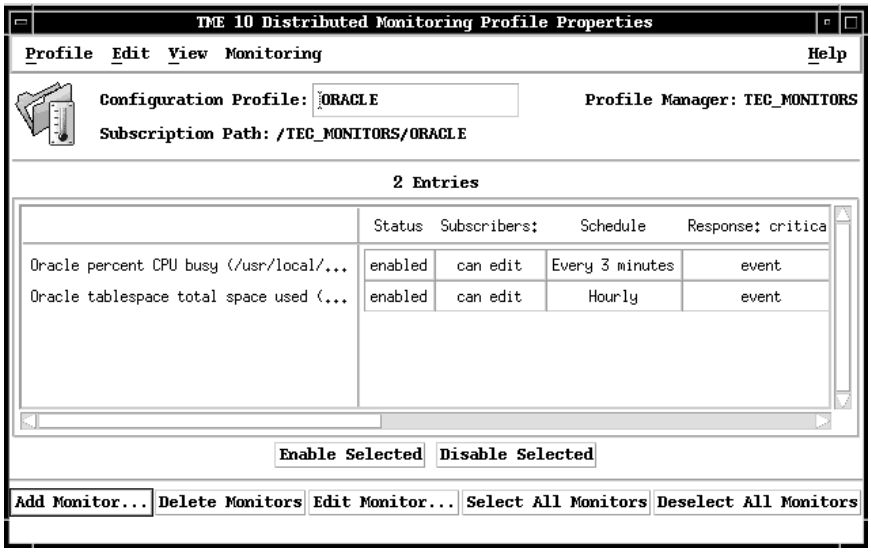


Figure 24. Saving the Profile

To distribute the profile select the pull-down menus **Profile->Distribute**.

Finally, we can distribute the profile to rs600028. This is the RIM server, not the actual Oracle database server.

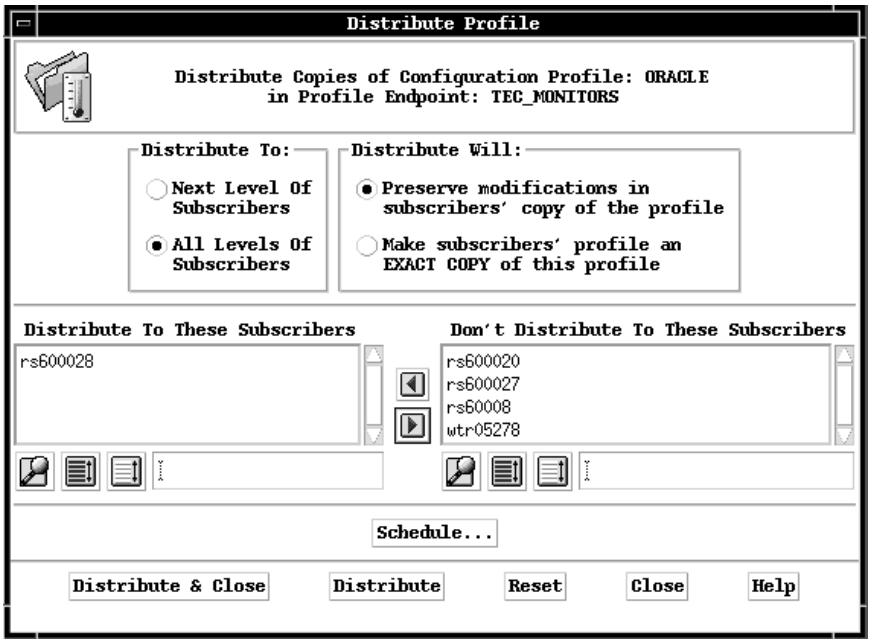


Figure 25. Distribute the Profile

5.1.5 Testing the Oracle Monitor

After a short period of time we started to receive events from the monitor. To test this monitor we reduced the value of the threshold from 90 to 1. Figure 26 shows the event arriving on the event window of the TEC.

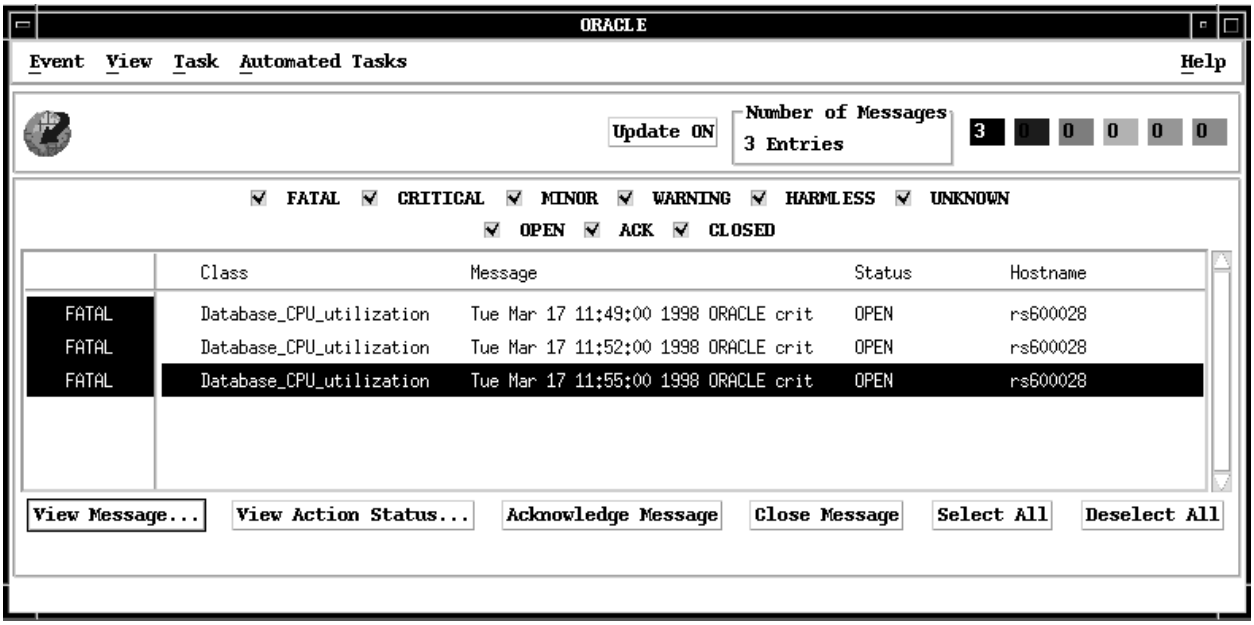


Figure 26. The Oracle Monitor Event

You can see the contents of events arriving at the event server at any time with the wtdumprl command. Output from this command showed us an example of an event sent by an Oracle monitor:

```
Database_CPU_utilization;
source='SENTRY';
sub_source='ORACLE';
severity='FATAL';
origin='127.0.0.1';
```

Seeing that the Oracle monitors fill the slot sub_source with the value ORACLE gave us the information we needed to create an event group filter to catch these events specifically.

This specific monitor creates temporary files in the directory /tmp on rs600028. These files are called:

```
oracleUsedSpace.in
oracleUsedSpace.out
```

These files contain the sql scripts that are executed by the Distributed Monitoring monitor.

```
Sql Statement;

SQL*Plus: Release 3.3.3.0.0 - Production on Mon Mar 16 22:20:01 1998

Copyright (c) Oracle Corporation 1979, 1996.  All rights reserved.

Connected to:
Oracle7 Server Release 7.3.3.0.0 - Production Release
With the distributed, replication, parallel query and Spatial Data option
PL/SQL Release 2.3.3.0.0 - Production

SQL> SQL>
TOTAL_SPACE
   31457280

SQL>
SEGMENT_TYPE      SPACE_USED
INDEX                2457600
TABLE                2457600

SQL> Disconnected from Oracle7 Server Release 7.3.3.0.0 - Production Rele
With the distributed, replication, parallel query and Spatial Data option
```

If the database you are using is Sybase, we have included the screen to show the parameters required (see Figure 27 on page 93).

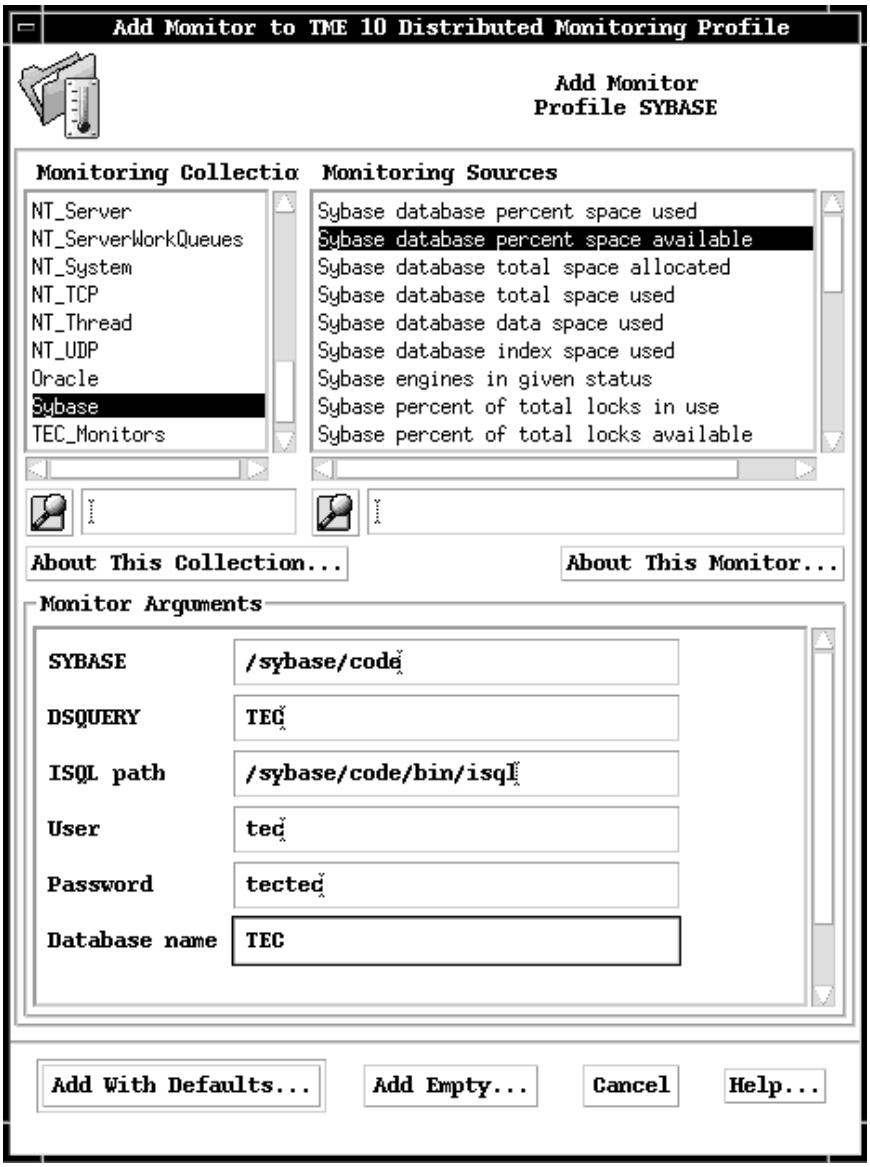


Figure 27. Sybase Monitors

Also the list of available monitors for Sybase is shown in Figure 27.

Next we continue with the monitor configuration.

5.2 Configuring the TME_MONITOR

The process is the same for the TME monitor, apart from the monitor screen.

From the TME_Monitors collection, one of the monitors we found important was for watching over the size of the TME database on our TMR server, as shown in Figure 28 on page 94.

Figure 28. Adding the TME Monitor

We added the TME monitor and distributed it to rs600020.

5.2.1 Using the UNIX Monitoring Collection

Our NetView server rs600027 is a critical resource in the flow of events, since traps that are received by the trapd daemon are then potentially forwarded by the nvserverd daemon to the TEC event server. We made the decision to run a Sentry monitor on our NetView server to alert us in the event that the trapd daemon should fail.

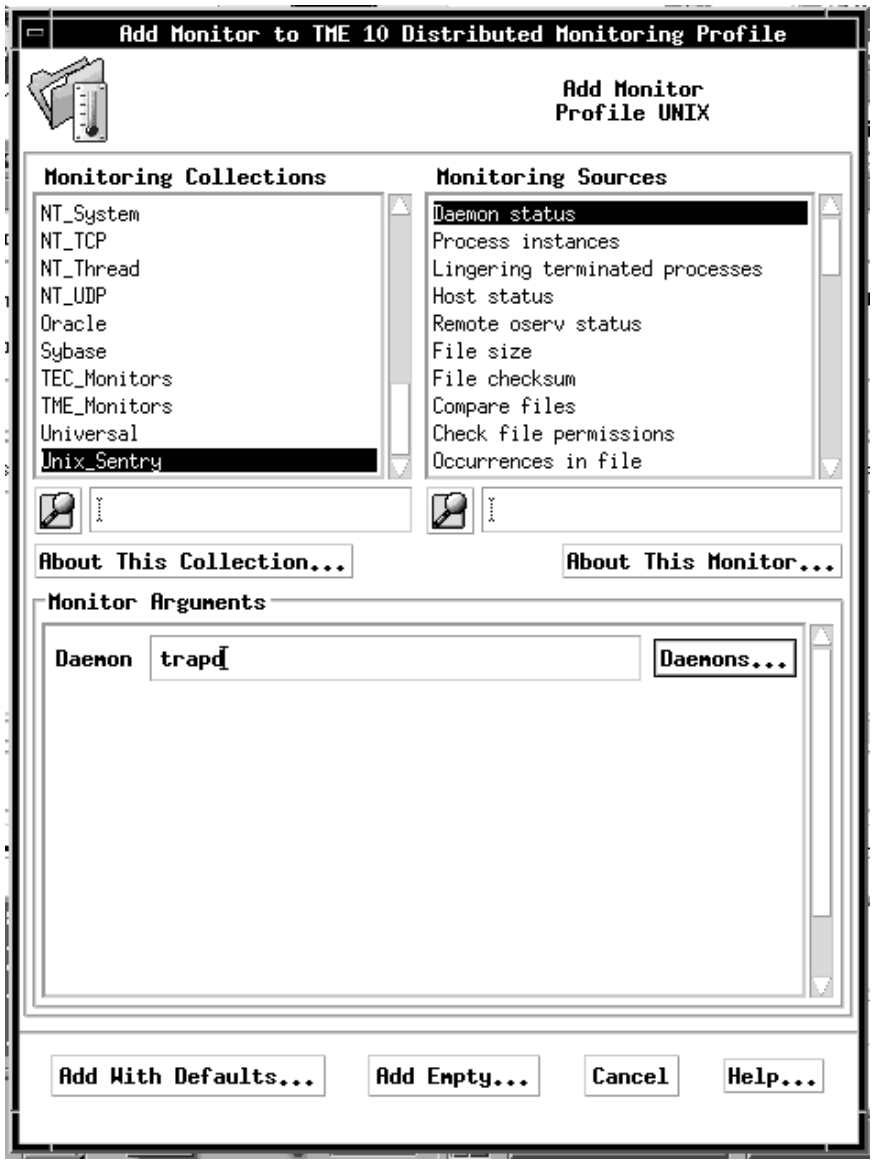


Figure 29. Unix_Sentry Monitoring Collection

In Figure 29 you see the Unix_Sentry monitoring collection and the monitor we chose for checking the status of the trapd daemon. We entered only the name `trapd` in the input field. If you click on the button marked **Daemons...**, you will see the list of choices available as in Figure 30 on page 96.

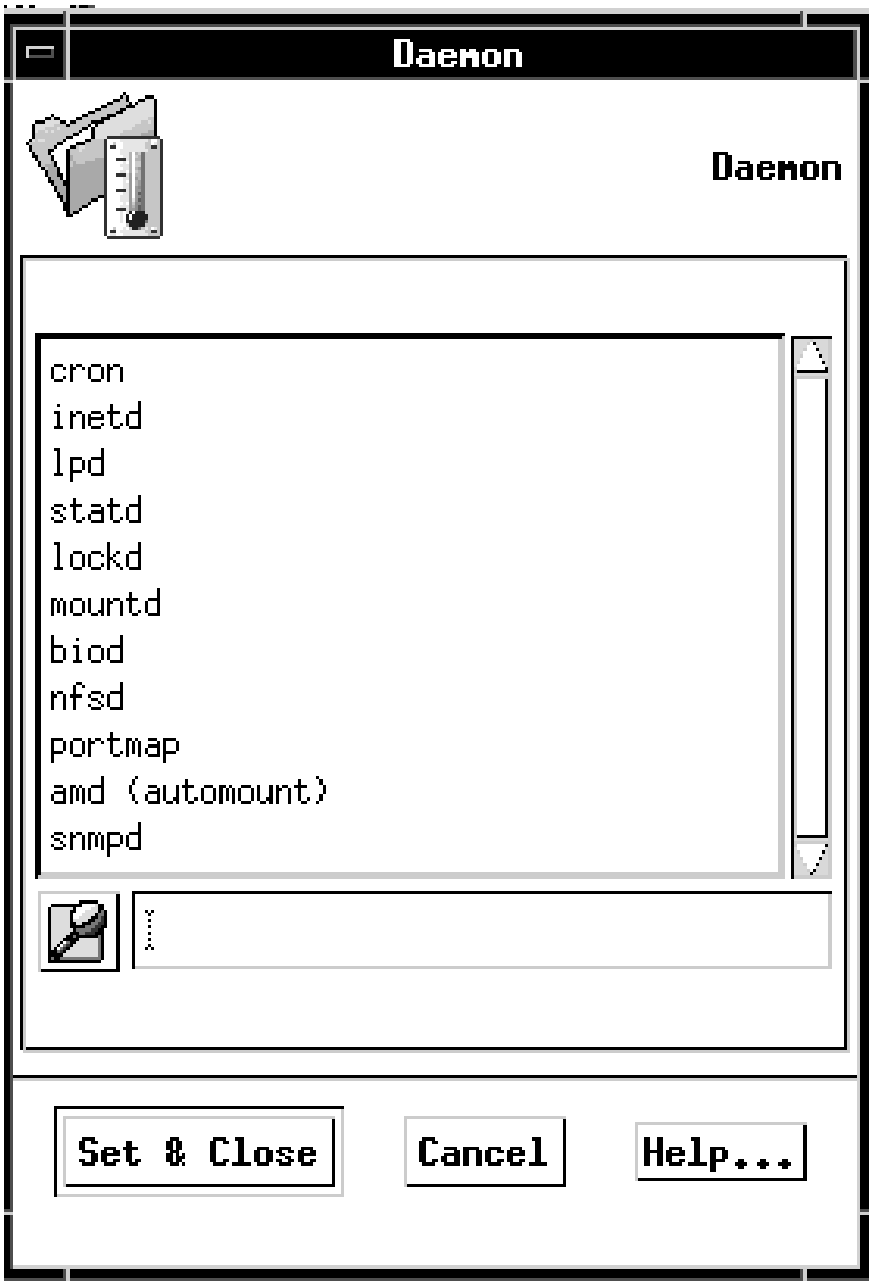


Figure 30. Choice of UNIX Daemons

The trapd daemon may not appear in the list of daemons. We found that this did not matter; we just entered it anyway and this worked fine.

Figure 31. Customizing the Daemon Monitor

In Figure 31, we opted to send a FATAL event to the TEC server when the trapd daemon is found to be down. We chose Is down/unavailable rather than Becomes down/unavailable, because we consider the absence of trapd to be so critical, that we wish to send an event to the event server *each* time the daemon is found to be unavailable, and not just the first time a change in status is detected.

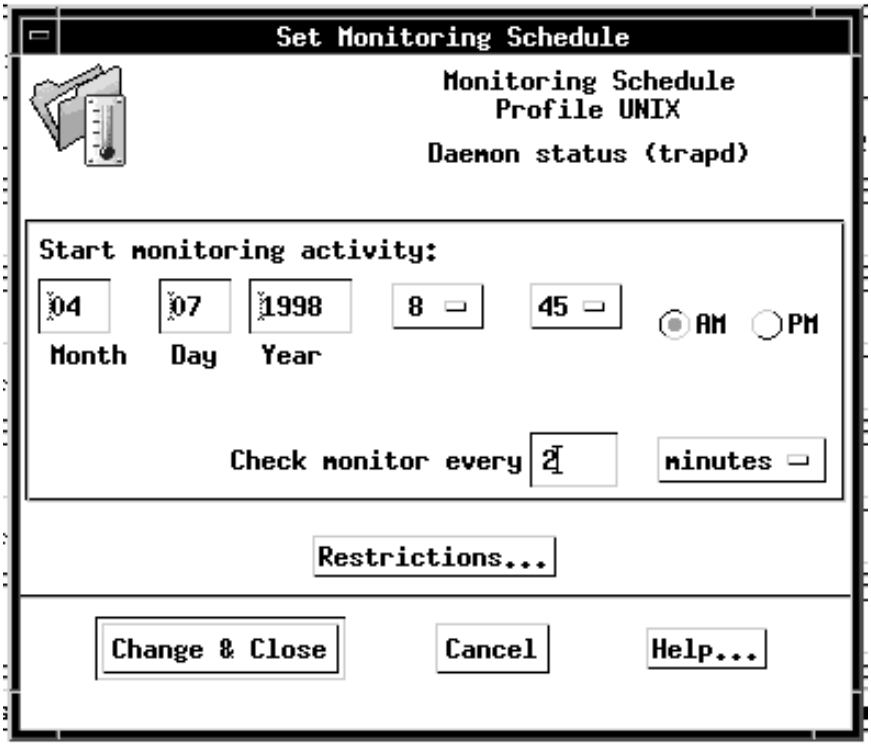


Figure 32. Monitoring Schedule

We chose a monitoring schedule of two minutes (see Figure 32).

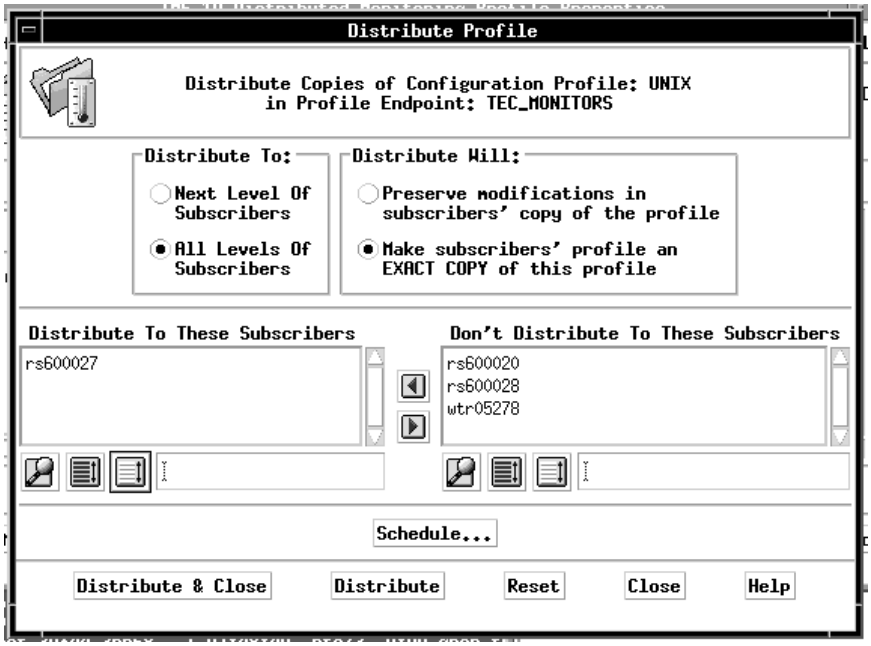


Figure 33. Distribution to NetView Server

The finished profile needs to be distributed only to rs600027, our NetView server.

When the monitor found the trapd daemon to be down, it delivered the following event to our event server, which we looked at in the TEC reception log with the wtdumprl command.

```
### EVENT ###
Sentry2_0_daemon;
source='SENTRY';
sub_source='UNIX';
severity='FATAL';
origin='127.0.0.1';
sub_origin='rs600027';
hostname='rs600027';
adapter_host='rs600027';
distrib_admin='Root_rs600020-region';
response_level='critical';
probe='daemon';
tmr='1194067385';
dispatcher='4';
prev_value='up';
value='down';
effective_value='down';
collection='Unix_Sentry';
info='Daemon: trapd';
monitor='Daemon status';
units='';
relation='Changes to';
relation_delta='';
msg='Sentry UNIX/Daemon status on host rs600027 Tue Apr 7 12:49:00 1998

Status: >>> critical <<<

Daemon status (trapd) Changes to down
(Previous: up  Current: down Effective: down)
Daemon: trapd';
END

### END EVENT ###
```

5.2.2 NT Monitors

Figure 34 on page 100 shows a list of the available NT monitors.

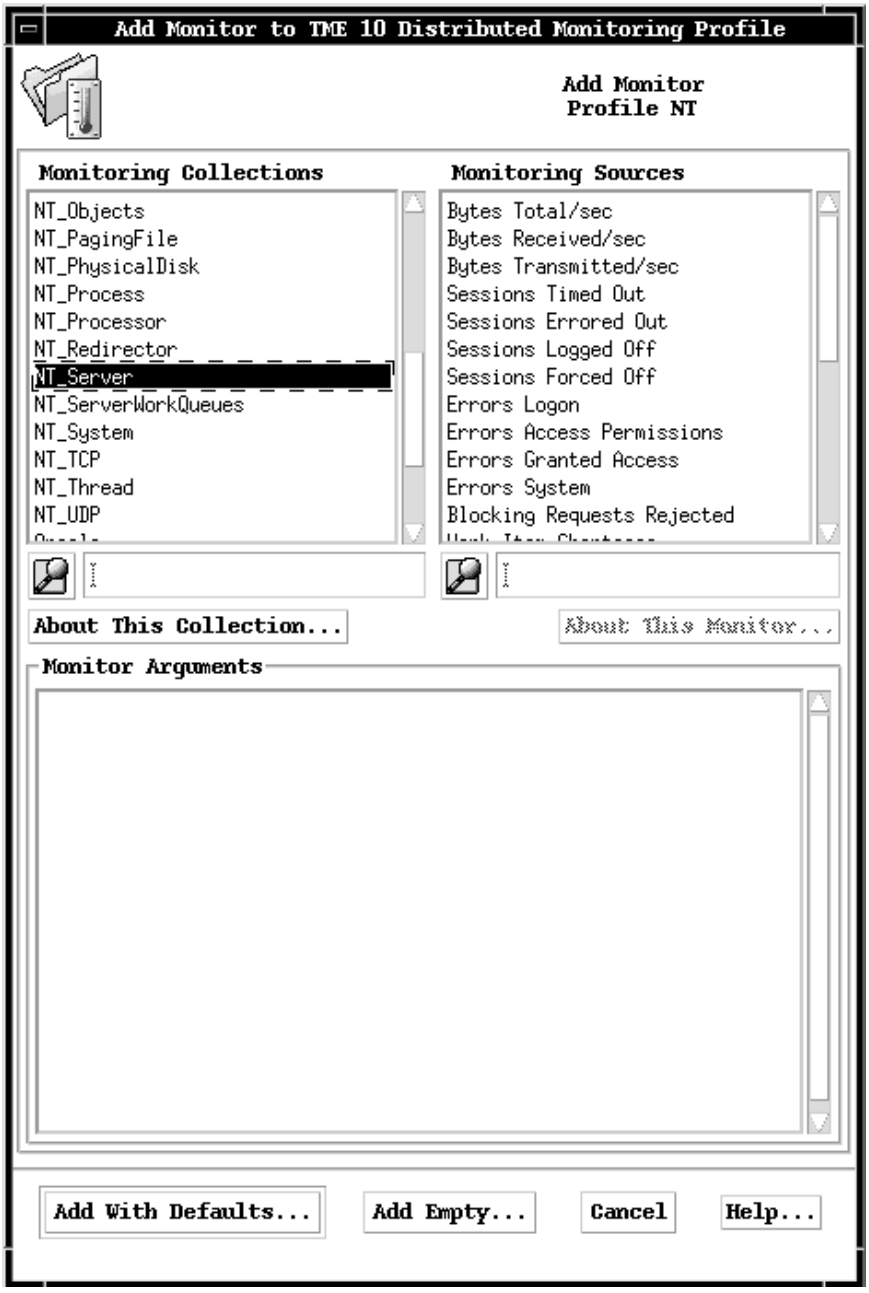


Figure 34. The NT Monitors

There is a large number of NT monitors available. You can browse each selection by clicking on each collection in turn and looking at the monitors listed in the Monitoring Sources window.

5.3 Example of Using a Script to Monitor Resources

If for some reason a monitor does not exist for a specific purpose, you can, of course, write your own monitors and bind them into the Distributed Monitoring structure. We had the requirement, however, to monitor the Distributed Monitoring daemon itself. In other words, we wanted to make sure that the TEC event server would receive an event if the process sentry_engine failed. We chose to implement this with a script, called regularly through the UNIX crontab mechanism, which uses

the command wpostemsg to send an event to the event server if the daemon is not found to be active:

```
#!/bin/sh
# This script checks for the status of the sentry_engine daemon

DAEMONNAME=sentry_engine
DAEMONTYPE="SENTRY"

DAEMON_STATUS= ps -ef | grep $DAEMONNAME | grep -v grep
if [ -z "$DAEMON_STATUS" ]
then
        wpostemsg -m Daemon_Error daemonname=$DAEMONNAME \
        daemontype=$DAEMONTYPE daemon_error STATUS
fi
exit 0
```

This monitor necessitated defining a baroc file with the new class daemon_error and its additional slots:

```
TEC_CLASS:
    daemon_error ISA EVENT
        DEFINES {
                source: default = "STATUS";
                severity: default = FATAL;
                daemontype: STRING;
                daemonname: STRING;
                };
END
```

We imported the new baroc file into the current rule base:

```
wimprbclass /usr/development/classes/scripts.baroc CanTire
```

We then compiled and loaded the changed rule base:

```
wcomprules CanTire
wloadrb CanTire
```

Whenever class definitions are changed, the event server needs to be stopped and restarted:

```
wstopesvr
wstartesvr
```

The output of wtdumprl shows the event in the TEC reception log:

```
### EVENT ###
daemon_error;
source=STATUS;
msg=Daemon_Error;
daemonname=sentry_engine;
daemontype='SENTRY';
origin=9.24.104.4;
END
```

We also coded a script to check for filesystems filled up more than a given percentage. There is a Sentry monitor in the UNIX monitoring collection to do just this for a single filesystem, but we wanted a simple solution that checks all filesystems with one action. This script serves, in addition, as an example of informing the TEC event server about the state of any resource where Distributed Monitoring is not installed.

The shell script to check for the filesystem is shown below:

```
#!/bin/sh
# Check each filesystem and send an event for any that
# is more than 90% full

THRESHOLD=90

df | grep -v Free | awk '{ print $4, $7}' | sed -e "s/%//"    \
     | while read SIZE FILESYS
do
        if [ "$SIZE" -gt "$THRESHOLD" ]
        then
             wpostemsg -m FileSystem_Threshold filesystem=$FILESYS  \
                size=$SIZE filesystem_threshold STATUS
        fi
done

exit 0
```

The baroc definition required for this new type of event is:

```
TEC_CLASS:
    filesystem_threshold ISA EVENT
        DEFINES {
                source: default = "STATUS";
                severity: default = WARNING;
                filesystem: STRING;
                size: STRING;
                };
END
```

Again, don't forget to import the baroc file with the class definition, compile and load your rule base, and stop and restart the event server.

Chapter 6. Deploying the Logfile and NT Adapters

The event adapters are responsible for generating, formatting and sending events to the T/EC server. The adapters are also responsible for discarding unwanted events using local filtering to help reduce network traffic and the load on the event server.

The event adapters can exist on any TCP/IP connected system, not only on TME 10 managed nodes.

In this chapter we concentrate on the following event adapters:

- T/EC 3.1 UNIX LOGFILE Adapter
- T/EC 3.1 Windows NT Event Log Adapter

The main requirements here are to deploy both the logfile adapter and the NT adapter to monitor the AIX and NT servers respectively. We show how to install and customize the adapter configuration files using examples.

The provided list of sources requires a number of distributed adapters to be configured. For each event source we have identified the required adapter.

Before we show how to install the adapters we describe the files associated with the logfile adapter and show what functions the logfile adapter provides.

6.1 Files Associated with Event Adapters

The following list explains the contents of the files associated with the event adapter. Every adapter has some of these files, but may not have all of them depending on the which function it performs.

- Global configuration file (.conf) - Tells the adapter where to find the event server, sets options such as maximum message size and also provides some filtering options.
- Format configuration file (.fmt) - Defines the format of system log messages and their mappings to classes and is converted to a .cds file by passing through a program called logfile_gencds.
- Adapter-specific configuration file (cds) - This file is generated from the tecad_logfile.fmt file. The selection of messages by the logfile adapter is based upon the specification of this file.
- Event class definition file (.baroc) - The class definitions required by the T/EC server to process the event. This file is imported into the T/EC server as a class definition and compiled. Each adapter comes with pre-defined baroc definitions.
- Error file (.err) - Contains error logging and tracing options for some adapters; the options are to send debugging information to a file or panel.
- Rules file (.rls) - Defines the default rules for the adapter. This file is imported into the T/EC server and compiled. Each adapter comes with a set of pre-defined rules.

- Start up configuration file (.cfg) - Used to configure the adapter to start up at boot time.

For additional information on the logfile adapter configuration files see the *Enterprise Console Adapter Guide*.

6.2 The UNIX Logfile Adapter

UNIX provides two applications where an error or warning message is logged. The first method provided is the function syslog; the second method is the error reporting program.

The syslog relies on a process called syslogd. This process will report operating system conditions to the system console or to a specific logfile.

Messages are tagged with codes indicating the classification of the message. A classification, known in UNIX as a priority, is encoded as a facility, which describes the component of the system generating the message, and as a level, which indicates the severity of the message.

The logger command, found in the /usr/bin directory, provides an easy-to-use interface to the syslog subroutine. A message variable can be specified on the command line, which is logged immediately, or a file variable is read and each line of the file variable is logged.

The logfile adapter connects to the syslogd daemon by adding an entry to the file /etc/syslog.conf file. This file controls the output of the syslog daemon. It supports subscriptions by other AIX processes to specific subsets of messages handled by the syslog daemon. Each entry in the file consists of two parts; a selector, which determines the messages of interest to the subscribing process, and a destination file to which the selected messages are directed. The selection of messages is based upon two criteria: the facility generating the message and the priority of the message. The facilities supported are:

- kern - Kernel
- user - User level
- mail - Mail subsystem
- daemon - System daemons
- auth - Security or authorization
- syslog - Syslogd daemon
- lpr - Line printer subsystem
- news - News subsystem
- uucp - UUCP subsystem
- * - (All facilities)

The priority levels that can be selected are:

- emerg - Emergency messages (LOG_EMERG) such as fixed-disk errors.
- alert - Important messages (LOG_ALERT) such as serious hardware errors.

- crit - Critical messages not classified as errors (LOG_CRIT), such as improper login attempts.
- err - Messages that represent error conditions (LOG_ERR) such as an unsuccessful disk write.
- warning - Messages for abnormal, but recoverable, conditions (LOG_WARNING).
- notice - Informational messages. All messages without a priority designation are mapped into this priority messages (LOG_NOTICE).
- info - Less significant informational messages (LOG_INFO).
- debug - Debugging messages (LOG_DEBUG).
- none - Excludes the selected facility.

Destinations may be specified as follows:

- /Filename - Full path name of file opened in append mode.
- @Host - Host name.
- User - User name(s).

The default entries created by the logfile adapter install process are shown below:

```
# Tivoli logfile adapter entry, ID=default
*.emerg;*.alert;*.crit;*.err;*.warning;*.notice;*.info
              /tmp/.tivoli/.tecad_logfile.fifo.rs600028.default
# End of logfile adapter entry, ID=default
```

This entry directs the syslogd daemon to pass all messages to the specific file for the logfile adapter in /tmp.

The refresh command would normally be used to stop and restart syslogd after updating of the syslog.conf file. However, when the logfile adapter process is running the refresh command should not be used. Use the init.tecad_logfile script instead. In our case, it was located in:

```
/usr/local/Tivoli/bin/aix4-r1/TME/TEC/adapters/bin
```

There may be a performance benefit in tailoring the logfile adapter entry to indicate that only messages from specific facilities or message-specific priority levels get passed to the adapter. In addition, these entries should be reviewed to make sure that all of the messages you are expecting to be passed by the event adapter actually appear at the event server.

6.2.1 Adapter Configuration File Considerations

The selection of messages by the logfile adapter is based upon specifications provided in the tecad_logile.cds file. This file is generated from the FORMAT statements defined by the user in the tecad_logfile.fmt file. The logfile_gencds utility is used to generate the tecad_logfile.cds file from the input statements in the tecad_logfile.fmt file.

The fmt file was located in /usr/local/Tivoli/bin/aix4-r1/TME/TEC/adapters/etc and the gencds utility was in /usr/local/Tivoli/bin/aix4-r1/TME/TEC/adapters/bin.

A brief example of the mapping of logfile TEC_CLASSES in the tecad_logfile.baroc file, the FORMAT statements in the tecad_logfile.fmt file, and the selection and mapping instructions used by the adapter from the tecad_logfile.cds file follows. The example message is:

```
Mar 18 17:14:57 tivoli syslogd: /var/adm/messages: No space left on device.
```

This message is located in the format file as shown below:

```
FORMAT Logfile_Base  1
%t%s%s*
hostname $2
date $1
origin DEFAULT
msg $3
END

FORMAT Logfile_Syslogd FOLLOWS Logfile_Base  2
%t%s syslog: %s*
sub_source syslogd
END

FORMAT Syslogd_Nospace FOLLOWS Logfile_Syslogd  3
%t%s syslogd: %s: No space left on device
logfile $3
msg PRINTF( "syslogd: %s: No space left ondevice", logfile)
END
```

The format statement for Logfile_Base **1** defines the message as consisting of an arbitrary number of strings, with the date and time stamp in the first position followed by hostname and an arbitrary number of strings in the message body. The event origin is assigned as adapter DEFAULT.

The format statement for Logfile_Syslogd **2** then further specializes the mapping to search for the string syslog in the third position of the message. If found, it assigns the value of syslogd to the subsource event slot.

Finally, the format statement for Syslogd_Nospace **3** refines the search criteria to include the string No space left on device beginning in string position five. The contents of string position three, syslogd, is assigned to the logfile event slot and the slot msg is assigned the values specified in the PRINTF statement.

```
CLASS Syslogd_Nospace
  SELECT
    1: ATTR(=,"_event_id"), VALUE(+,"42") :
    2: ATTR(=,"hostname") ;  1
    3: ATTR(=,"date") ;  2
    4: ATTR(=,"origin") ;  3
    5: ATTR(=,"logfile") ;
  MAP
    hostname = $V2 ;
    date = $V3 ;
    origin = $V4 ;
    msg=PRINTF('syslogd: %s: No space left on device", $V5) ;  4
    sub_source = "syslogd";  5
    logfile = $V5:  6
END
```

The CLASS block mapping statements generated by the logfile_gencds utility from the format definitions for the example event class, Syslogd_Nospace. This data built the following:

1 hostname - From message string 2, as defined in FORMAT Logfile_Base.

2 date - From message string 1, as defined in FORMAT Logfile_Base.

3 origin - Assigned the value of DEFAULT in FORMAT Logfile_Base.

4 msg - Generated from the PRINTF statement specified using the value of logfile as defined in FORMAT Syslogd_Nospace.

5 subsource - As defined in FORMAT Logfile_Syslogd.

6 logfile - As defined in FORMAT Syslogd_Nospace.

To summarize the stages undergone by the three files whose contents define message selection criteria, message-to-event mapping instructions, and event class definitions:

- The FORMAT statements in tecad_logfile.fmt provide the criteria used in message selection.
- These are passed to the logfile_gencds utility which creates the real-time instructions for the adapter for mapping of event defaults and message strings into the appropriate slots in the event message.
- The contents of these slots are then defined to the TEC using the definitions provided in the tecad_logfile.baroc file.

In order to test the newly installed adapter we commented out the filter statement for the Logfile_Base event class.

Filter statements will be added to the tecad_logfile.conf file to block the sending of event classes that appear in the tecad_logfile.cds file. For example, commenting out of the statement for the Logfile_Base event class enabled the forwarding of all log messages processed by the syslogd with the exceptions of the Logfile_SendMail, Amd_Unmounted, and Amd_Mounted event classes.

This change to the .conf file enabled testing of the adapter using practically any message that is logged by the system. Obviously, while useful for an initial adapter

test, uncommenting the filter for Logfile_Base events would be unwise in a production environment.

An AIX system does not process all error messages through the syslog daemon. Many of the software and hardware components of the system report errors using the errdemon daemon, a process totally separate from the syslog facility. Therefore, in order to use the logfile event adapter to forward to the event server all critical systems messages, the user must consider how to copy all relevant messages handled by the errdemon in the syslog message processing flow.

The error logging process begins when the error detecting module detects an error. The error-detecting segment of code then sends error information to either the errsave kernel or the errlog application subroutine, where the information is in turn written to the /dev/error special file. The file then adds a time stamp to the collected data. The errdemon daemon constantly checks the /dev/error file for new entries, and when new data is written, the daemon conducts a series of operations.

Before any entry can be written to the error log, the errdemon compares the label sent by the kernel or application code to the contents of the error record template repository. If the label matches an item in the repository, the daemon collects additional information from other parts of the system.

To write an entry to the error log, the errdemon retrieves:

- The appropriate template from the repository
- The resource name for the unit that caused the error
- Detail data
- Information from the message catalog

Also, if the error signifies a hardware-related problem and hardware vital product data (VPD) exists, the daemon retrieves the VPD from the object data manager. Most entries in the error log are attributed to hardware and software problems, but informational messages can also be logged.

The error notification object class allows applications to be notified when particular errors are recorded in the system error log. The application describes the set of errors to which notice is given of in an error notification object.

Each time an error is logged, the error notification daemon determines if the error log entry matches the selection criteria of any of the error notification objects. The notify method of each matched object is run.

6.2.2 Configuring the AIX Error Daemon

In order to provide a copy of messages logged by errdemon to the syslogd daemon for forwarding to the logfile adapter, we added a notification object that utilizes the logger within its method to write the error message to syslogd.

We created a file called /tmp/syslog.add. This file contained the following statements.

```
errnotify:
en_name = "syslog1"
en_persistenceflg = 1
en_method = "logger Msg from Error Log:
 errpt -l $1 | grep -v 'IDENTIFIER TIMESTAMP' "
```

We then entered the following command:

```
odmadd /tmp/syslog.add
```

This command added our new object, syslog1 to the ODM object class errnotify.

The meaning of the parameters are:

- en_name - A name assigned by the creator of the error notification object to uniquely identify the object.
- en_persistenceflg - Designates whether or not the error notification object should be automatically removed when the system is restarted. A non-zero value indicates that this object should not be automatically removed at system restart.
- en_method - This points to the notify method, a shell script of a command, that is to be run when an error matching the selection criteria of this error notification object is logged. The error notification daemon uses the sh -c command to execute the notify method.

Therefore, our error notification object, syslog1, is a persistent object whose existence survives system restart. The error class that is targeted is not specified as an object attribute. By not specifying an error class the default of all error messages is enabled. The method that we specified simply uses the logger command to pass a formatted entry from the error log to the syslogd.

6.3 The Logfile Adapter Configuration Files

This section looks at the logfile adapter files in detail.

6.3.1 The tecad_logfile.conf File

The logfile adapter configuration file (tecad_logfile.conf) is shown below:

```
ServerLocation=@EventServer  1
BufEvtPath=/etc/Tivoli/tec/logfile.cache  2
# ServerPort=9999  3
EventMaxSize=4096  4
Filter:Class=Logfile_Base  5
Filter:Class=Logfile_Sendmail  5
Filter:Class=Amd_Unmounted  5
Filter:Class=Amd_Mounted  5
```

- 1 The ServerLocation specifies the name of the host on which the server is installed. In our case, the adapter refers to the EventServer object (rs600028) and uses the TME 10 communications services to send events to it.

To send events to a T/EC server in a connected TMR, use the following format. (The remote TMR is called pluto.)

```
ServerLocation=@EventServer#pluto-region
```

To specify a list of alternate T/EC servers, you can specify a list of destinations. In this example the event adapter will try to send an event to the T/EC server in the local TMR. If it cannot send the events to the T/EC server in the local TMR, it will send it to the T/EC server in rs600026 region. Use the following format:

```
ServerLocation=@EventServer,@EventServer#rs600026-region
```

To send events to a T/EC server if the event adapter is installed on systems without the TME 10 environment installed, use the following format:

```
ServerLocation=rs60008
```

- **2** The BufEvtPath specifies the full path name of the event adapter's buffer file (/etc/Tivoli/tec/logfile.cache). The file is used to store events if the logfile adapter cannot communicate with the event server. When the communication is established, it will then send events from this file to the event server.
- **3** # ServerPort=9999 is commented out. This keyword specifies the port number on which the event server listens. Since it is commented out, the port number is retrieved by calling the portmapper.

  If the AIX portmapper is not running, the T/EC server will not start.
- **4** The EventMaxsize specifies the maximum length of event messages. We are using the default value of 4096.
- **5** The Filter specifies how events are filtered. When the logfile adapter is installed, the configuration file already has four filter statements. These filter statements for Logfile_Base, Logfile_Sendmail, Amd_Unmounted and Amd_Mounted would block any events that only match the base class for these event classes (messages only for which there is not a specific event class in the .baroc file).

  You can be more specific in your filtering by specifying the slot values of the event class:

  Filter:class_name,slot=value,;slot=value;....

  There are other keywords used for the .conf file (please refer to *TME 10 Enterprise Console Adapters Guide*).

6.3.2 The tecad_logfile.fmt File

The logfile adapter format configuration file (tecad_logfile.fmt) is shown below.

```
FORMAT Logfile_Base
%t %s %s* 1
hostname $2 2
date $1 3
origin DEFAULT
msg $3 4
END
```

- **1** %t - Matches a time stamp with a format (month date time) which refers to **3** date $1 as the first position.

- 1 %s - Matches one constant in the input message which refers to 2 hostname $2 as the second position.
- 1 %s* - Matches zero or more constants in the input message which refers to 4 msg $3 as the third position.

The logfile adapter specific configuration file (tecad_logfile.cds) is shown below.

```
CLASS Logfile_Base
  SELECT 1
    1: ATTR(=,"_event_id"), VALUE(=,"0"); 2
    2: ATTR(=,"hostname"); 2
    3: ATTR(=,"date"); 2
    4: ATTR(=,"origin"); 2
    5: ATTR(=,"msg"); 2
  MAP 3
    hostname = $V2; 4
    date = $V3; 4
    origin = $V4; 4
    msg = $V5; 4
END
```

- 1 SELECT - Consists of one or more statements (ATTR) that an incoming event must satisfy to match the corresponding class.
- The incoming event must contain the attributes 2 _event_id, hostname, date, origin, and msg. The _event_id must start with "0".
- 3 MAP - Assign the values to the slots of the T/EC event class instance.
- The slot names 4 :hostname, date, origin, msg have variable values.
- The other statement in the .cds file, FETCH - Retrieves attribute names, keys and values from the incoming event, but does not allow you to modify the selected pieces of information in any way.

For more information on the .cds file, please refer to *TME 10 Enterprise Console Adapters Guide.*

6.3.3 The tecad_logfile.baroc File

The logfile adapter event class definition file (tecad_logfile.baroc) is shown below.

```
TEC_CLASS : 1
    Logfile_Base 2  ISA 3  EVENT  4
    DEFINES 5  {
          source: 6  default= "LOGFILE";  7
          sub_source: 6  default= "LOGFILE";  7
          sub_origin: 6  default= "N/A";  7.
          adapter_host: 6  default= "N/A";  7
          msg_catalog: 6  default= "none";  7
          msg_index: 6  default= 0;  7
          repeat_count: 6  default= 0;  7
          pid: 6  STRING, 8  default="N/A";  7
          severity: 6  default = WARNING;  7
    };
END

TEC_CLASS :
    No_Resources ISA Logfile_Base  8
    DEFINES {
          severity: default = CRITICAL;  9
    };
END
```

- 1 TEC_CLASS is the label used to define a new event followed by the new event class name 2 Logfile_Base.
- 2 The Logfile_Base is the event class name which is really a subclass of the TEC root class event (the base event class ($BINDIR/TME/TEC/default_rb/TEC_CLASSES/root.baroc).
- 3 ISA defines a superclass from which the new events inherits
- 4 base class EVENT (from root.baroc).
- 5 DEFINES - Keyword use to define new slot values for the event and default values.
- 6 The slot names source, sub_source, sub_origin, etc. are the slots defined in the event class.
- 7 This is the slot facet that is used to define the type of information that is contained in the slot. The default value is used when the slot value is not assigned by the adapter. In this example, the slot names have default values assigned.
- 8 This is the slot value data type. STRING means a slot value may contain up to 255 characters.
- 8 The No_Resources event class is defined under the Logfile_Base event class.
- 9 The slot name severity value for the No_Resources event class has a default value of SEVERITY. If this was not defined, the severity has a value of WARNING taken from the Logfile_Base event class.

6.4 Installing and Customizing the Logfile Adapter

We installed the logfile adapter on all our managed nodes (see Figure 35).

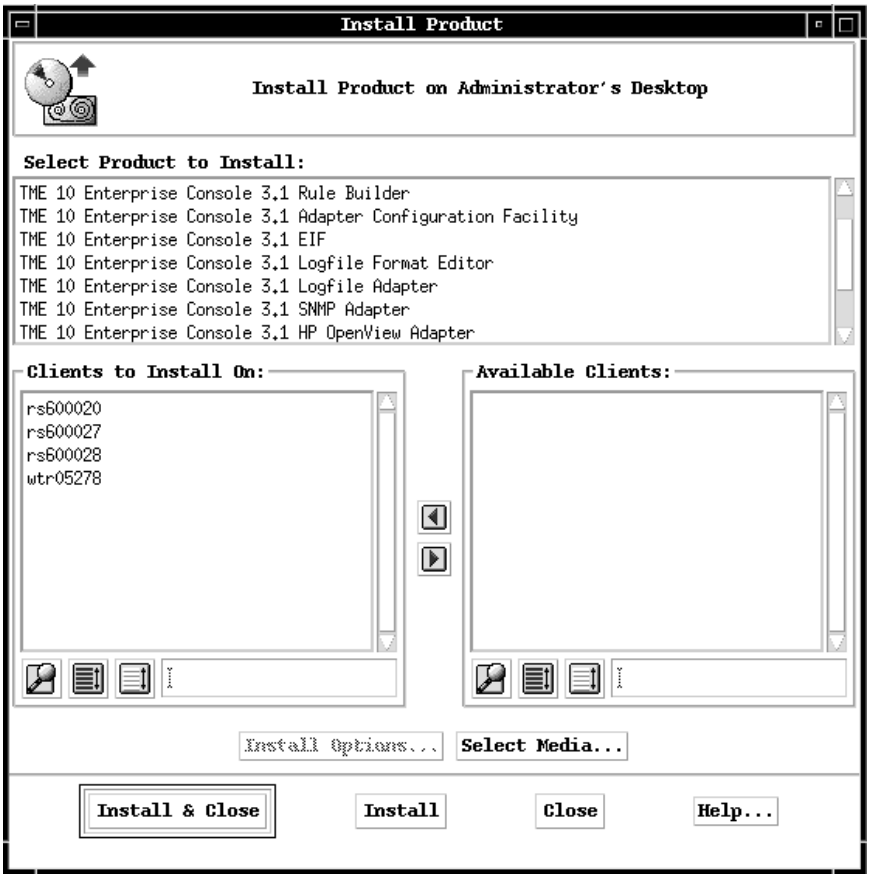


Figure 35. Installing the Logfile Adapter

The example below shows the logfile adapter working on our TME 10 NetView server, rs600027.

The following are the changes that occurred during the installation of the logfile adapter:

- The /etc/syslog.conf file controls the output of the syslog daemon. The following lines were added to the file by the logfile adapter install so that the logfile adapter can connect to the syslogd daemon:

```
# Tivoli logfile adapter entry, ID=default
*.emerg;*.alert;*.crit;*.err;*.warning;*.notice;*.info
      /tmp/.tivoli/.tecad_logfile.fifo.rs600027.default
# End of logfile adapter entry, ID=default
```

- The adapter files created after the installation of the logfile adapter on machine rs600027 that are used by the T/EC logfile adapter are shown below:

```
/etc/Tivoli/tecad/etc/tecad_logfile.baroc
/etc/Tivoli/tecad/etc/tecad_logfile.cds
/etc/Tivoli/tecad/etc/tecad_logfile.conf
/etc/Tivoli/tecad/etc/tecad_logfile.err
/etc/Tivoli/tecad/etc/tecad_logfile.fmt
```

The utilities used by the T/EC logfile adapter are shown below:

```
/etc/Tivoli/tecad/bin/init.tecad_logfile
/etc/Tivoli/tecad/bin/logfile_gencds
/etc/Tivoli/tecad/bin/tecad_logfile
/etc/Tivoli/tecad/bin/tecad_logfile.cfg
/etc/Tivoli/tecad/bin/update_conf
```

Next we can test the logfile adapter.

6.5 Testing the Logfile Event Adapter

The logfile adapter will work as soon as it has been installed. To verify whether it is running on rs600027:

```
ps -ef | grep logfile
```

This showed the output listed below:

```
root 20226 35444   0 13:53:34  pts/2  0:00 grep logfile
root 26052     1   0 12:38:53      -  0:01 bin/tecad_logfile -c /etc/    Tivoli/
 tecad/etc/tecad_logfile.conf
root 31236 35444   1 14:02:56  pts/2  0:00 grep logfile
```

As soon as the logfile adapter was installed, wtdumprl was issued and we found out that there were some Logfile_Snmpd alert events in the T/EC reception.

```
### EVENT ###
Logfile_Snmpd;hostname=rs600027;date='Mar 2 21:58:53';origin=9.24.104.188;msg='E
XCEPTIONS: authentication error: invalid community name: ITSO';pid=18584;sub_sou
rce=snmpd;END

### END EVENT ###
PARSING_FAILED 'Line 1: Class Logfile_Snmpd undefined'

1 36430 0 888894027(Mar 02 22:00:27 1998)

### EVENT ###
Logfile_Snmpd;hostname=rs600027;date='Mar 2 22:00:29';origin=9.24.104.188;msg='E
XCEPTIONS: authentication error: invalid community name: ITSO';pid=18584;sub_sou
rce=snmpd;END

### END EVENT ###
PARSING_FAILED 'Line 1: Class Logfile_Snmpd undefined'
```

The PARSING_FAILED.. error is caused by the tecad_logfile.baroc file not being imported into the current rule base.

The file was copied from rs600027, the logfile baroc file from /etc/Tivoli/tecad/tecad_logfile.baroc to our development area /usr/development/classes on rs600028.

This was a precaution as we are building a repository for all our baroc and rule configuration files.

Next we need to import the baroc file as follows:

```
wimprbclass /usr/development/classes/tecad_logfile.baroc CanTire
```

After importing the baroc file, it has to be compiled as:

```
wcomprules CanTire
```

We had the result shown below:

```
Loading CLASSES...
Parsing BAROC file /usr/canadian_tire/TEC_CLASSES/root.baroc
Parsing BAROC file /usr/canadian_tire/TEC_CLASSES/tec.baroc
Parsing BAROC file /usr/canadian_tire/TEC_CLASSES/tecad_logfile.baroc
Compiling Rules...
Final Compilation Stage...
```

We load the rules, stop the server, restart it again and display the reception log.

```
wloadrb CanTire
wstopesvr
wstartesvr
wtdumprl
```

The reception now shows:

```
### EVENT ###
Logfile_Snmpd;hostname=rs600027;date='Mar 2 22:18:53';origin=9.24.104.188;msg='E
XCEPTIONS: authentication error: invalid community name: ITSO';pid=18584;sub_sou
rce=snmpd;END

### END EVENT ###
PROCESSED
```

The PARSING FAILED error was corrected by compiling the baroc file into the current rule base.

The following are the other commands you can use to test the logfile adapter:

- To send an event using wpostemsg use the following command:

  ```
  wpostemsg -m "test using wpostemsg" NIS_OK LOGFILE
  ```

  The message is shown in the reception log:

```
1 1958 0 889336241(Mar 08 00:50:41 1998)
### EVENT ###
NIS_OK;
source=LOGFILE;
msg='test using wpostemsg';
origin=9.24.104.188;
END

### END EVENT ###
PROCESSED
```

- The following logger command found in /usr/bin/ provides an interface to the syslog subroutine:

  ```
  logger -t oserv -i execve failed
  ```

  The reception log displays:

```
### EVENT ###
Logfile_Oserv;hostname=rs600027;date='Mar 8 01:02:44';origin=9.24.104.188;msg='e
xecve failed';pid=17226;sub_source=oserv;END

### END EVENT ###
PROCESSED
```

The currently used rule base was loaded from the rule base named Canadian_Tire.

We issued the command shown below to show the current list of rule bases.

```
wlsrb -d
```

The output is shown below.

```
Rule Base Name Directory
-------------- -----------------------------------
Default        rs600028:/usr/local/Tivoli/bin/aix4-r1/TME/TEC/default_rb
CanTire        rs600028:/usr/canadian_tire
```

Here we continue to use the CanTire rule base.

All operating systems and applications system activity is recorded through some kind of system log facility. An example of this is the UNIX syslogd daemon. With the T/EC logfile adapter it is possible to generate T/EC events based on messages from syslogd, as well as any other logfiles on the system. The logfile provides a set of baroc classes to represent these messages.

The logfile adapter checks the messages in the logfile using pattern matching. If the message matches a specific pattern entry, the adapter can the assign parts of the message to the fields of a particular event class.

6.5.1 Customizing AIX Syslog Messages

An AIX system does not process all error messages through the syslog daemon. Many of the software and hardware components of the system report errors are using the errdemon daemon, a process totally separate from the syslog facility. Therefore, in order to use the logfile event adapter to forward to the event server all critical systems messages, the user must consider how to copy all relevant messages handled by the errdemon in the syslog message processing flow.

Since the AIX error daemon allows the user to define error notification methods that will be executed whenever the error occurs, we added an entry in the Object Data Manager (ODM) under the errnotify class and use the command odmadd to add the entry in the ODM.

We created a file called /tmp/syslog.add. This file contained the following statements:

```
errnotify:
en_pid = 0
en_name = "syslog"
en_persistenceflg = 1
en_crcid = 0
en_method = "errpt -l $1|tail -1|logger -t errpt -p daemon.notice"
```

We then issued the command odmadd /tmp/syslog.add.

The above AIX ODM stanza causes the error daemon to forward the most recent line of the error report to the system log where the T/EC logfile adapter is used to parse the text.

6.5.2 Using the Logfile Event Adapter Utilities

The logfile adapter contains the following utilities: tecad_logfile.cfg, which is the program to configure the event adapter to start up at boot time and init.tecad_logfile, which allows the startup and shutdown of the event adapter.

To stop the logfile adapter use the following command:

```
/etc/Tivoli/tecad/bin/init.tecad_logfile stop
```

The output is as follows:

```
Refreshing syslogd...
0513-095 The request for subsystem refresh was completed successfully.
```

To start the logfile adapter issue the following command:

```
/etc/Tivoli/tecad/bin/init.tecad_logfile start
```

The output from this command is shown below.

```
Starting Tivoli/Enterprise Console Logfile Adapter ...
cd /etc/Tivoli/tecad
exec bin/tecad_logfile  -c /etc/Tivoli/tecad/etc/tecad_log
Running bin/tecad_logfile
Tivoli/Enterprise Console Adapter Starting
Refreshing syslogd...
0513-095 The request for subsystem refresh was completed successfully.
```

To start the logfile adapter to monitor all syslog messages use the following command:

```
./etc/Tivoli/tecad/bin/init.tecad_logfile
start syslog
```

The output is as follows:

```
Starting Tivoli/Enterprise Console Logfile Adapter ...
cd /etc/Tivoli/tecad
exec bin/tecad_logfile  -c /etc/Tivoli/tecad/etc/tecad_log
Running bin/tecad_logfile
Tivoli/Enterprise Console Adapter Starting
Refreshing syslogd...
0513-095 The request for subsystem refresh was completed successful
```

6.5.3 Running the Adapter in Debug Mode

The adapter can also be started in debug mode.

To start the logfile adapter in debug mode, if there is a problem with it, you need to issue the kill command. If you don't use the kill command, you will get an error:

```
Running /etc/Tivoli/tecad/bin/tecad_logfile : Tivoli/Enterprise Console Adapter
warning: in Printer_Error_Cleared dropping inherited map  msg $3'
warning: in Printer_Powerup dropping inherited map  msg $3'
warning: in Printer_Toner_Low dropping inherited map  msg $3'
warning: in Printer_Page_Punt dropping inherited map  msg $3'
warning: in Printer_Offline dropping inherited map  msg $3'
warning: in Printer_Output_Full dropping inherited map  msg $3'
warning: in Printer_Paper_Out dropping inherited map  msg $3'
warning: in Printer_Paper_Jam dropping inherited map  msg $3'
warning: in Printer_Door_Open dropping inherited map  msg $3'
warning: in Oserv_IPC_Dispatch_Failed dropping inherited map  msg $5'
warning: in Oserv_Graceful_Exit dropping inherited map  msg $5'
error: FIFO open(read) failed, errno=2
```

The following are the steps to run the adapter in debug mode:

1. Issue the following command to stop the logfile adapter before running it in debug mode.

   ```
   ps -ef | grep logfile
   ```

   ```
   root 31854     1   0 19:28:28      -  0:00 bin/tecad_logfile -c /etc/    Tivoli/
   tecad_logfile.conf
   root 32978 35396   1 19:38:51  pts/3  0:00 grep logfile:
   ```

2. Stop or kill the process for the logfile adapter with the following command:

   ```
   kill 31854
   ```

3. Check if the process is no longer running with the following command:

   ```
   ps -ef | grep logfile
   ```

   ```
   root 27350 35396   1 19:42:19  pts/3  0:00 grep logfile
   ```

4. Now that the logfile is not running, we can start it in debug mode.

   ```
   /etc/Tivoli/tecad/bin/tecad_logfile -d \
         -c /etc/Tivoli/tecad/etc/tecad_logfile.conf
   ```

   The logfile adapter is now running in debug mode. Part of the log is shown below.

   ```
           msg 0 $3
           sub_source 0 "login"
   Logfile_Date_Set numVars 3
   %t %s date: set by %s
           hostname 0 $2
           date 0 $1
           origin 0 DEFAULT
           msg 0 PRINTF("date set by %s", set_by)
           sub_source 0 "date"
           set_by 0 $3
   Logfile_Date numVars 3
   %t %s date: %s*
           hostname 0 $2
           date 0 $1
           origin 0 DEFAULT
           msg 0 $3
           sub_source 0 "date"
   Logfile_Base numVars 3
   %t %s %s*
           hostname 0 $2
           date 0 $1
           origin 0 DEFAULT
           msg 0 $3
   ```

5. To test if the logfile is receiving messages, the su command was issued from another window on the machine rs600027.

   ```
   su ralph
   ```

   The screen showed the adapter was receiving messages. If you don't see any message, it means that the adapter is not receiving message from the logfile. You might have to refresh the syslog.

   ```
   matched Su_Success
           $1 is  Mar 8 19:53:00'
           $2 is  rs600027'
           $3 is  root'
           $4 is  ralph'
           $5 is  /dev/pts/4'
   ```

6. Issue the logger command. This command is useful for testing the logfile adapters.

   ```
   logger -t test -i exec test-adapter
   ```

   The command output is as follows:

   ```
   matched Logfile_Base
           $1 is  Mar 8 20:01:33'
           $2 is  rs600027'
           $3 is  test‰31758‘: exec test-adapter'
   ```

7. The logfile adapter running in debug mode was stopped by pressing Ctrl-c.

6.6 The NT Event Adapter System Interfaces

Here we show how to install the NT adapter on our NT server WTR06278.

NT provides standard systems support for three event logs: the system log, the application log, and the security log. The NT event adapter is configured to interrogate these logs and select events for processing that meet the specific selection criteria defined by the user.

Therefore, the implementation of the NT adapter is totally consistent with the management practices of the NT administrator. The NT adapter extends rather than replaces the functions of NT event management by providing the means to integrate the correlation and processing of significant events logged at the NT systems level with the handling of other significant events from non-NT sources.

6.6.1 Event Class Structure

The event class structure for the NT event adapter is a straight forward specialization of the EVENT base class, as illustrated in Figure 36 on page 121. These subclass definitions are abstracted from the tecad_nt.brc BAROC event class definition file for the NT event adapter.

```
TEC_CLASS :
 NT_Base ISA EVENT
        DEFINES {
                source: default= "NT";   1
                sub_source: default= "NT";   2
                sub_origin: default= "N/A";      3
                adapter_host: default= "N/A";         4
                msg_catalog: default= "none";              5
                msg_index: default= 0;   6
                repeat_count: default= 0;    7
                category:STRING;   8
                EventType:STRING;      9
                SID:STRING;     10
                ID:INT32;   11
                };
END

TEC_CLASS :
 NT_Diskfull ISA NT_Base   12
 DEFINES {
  disk: STRING;   13
  };
END
```

Figure 36. Example NT Event Subclasses

The NT_Diskfull event subclass is an example of a specialization recognizing the requirements imposed in the context of a specific NT event, the disk full system event. The class definition below adds a single slot to the parent class of NT_Base, the disk slot. This slot is added in order to carry the identity of the disk with the capacity problem within the event.

The tecad_NT.brc file, while distributed with the NT event adapter as an NT file, is not utilized directly by the event adapter itself. The class definitions contained in the file are loaded on the TEC event server. There the definitions exist to allow the TEC rules to recognize and properly process the slot information carried in the event message.

6.6.2 The NT Adapter Configuration Files

The NT event adapter configuration files are listed below:

- tecad_NT.con
- tecad_NT.fmt
- tecad_NT.cds
- tecad_NT.err

As indicated in the event adapter manual, the purpose of the tecad_NT.fmt file is to describe, through the use of patterns, the events that are to be selected from the NT event logs for further processing. The format definitions are related hierarchically in a manner very similar to the relationships between TEC event classes and subclasses. The format NT_Base, seen in the figure below, provides a general pattern for all NT events. Note in the definition the relationship between

the names that appear in the format definition and the names previously seen in the definition of the NT event classes.

```
FORMAT NT_Base
%t %s %s %s %s %s %s %s*
hostname DEFAULT
origin DEFAULT
category $3
EventType $4
SID $5
sub_source $6
ID $7
msg $8
END

FORMAT NT_Diskfull FOLLOWS NT_Base
%t %s %s %s %s %s %s The %s disk is at or near capacity.  You may
need to delete some files.
disk $8
END
```

Figure 37. Example Format Statements from tecad_nt.fmt

NT_Diskfull, is a refinement of the base NT format, while the NT_Base format describes a generic pattern for all NT events represented by the tecad_NT.cds file.

```
CLASS NT_Base
  SELECT
    1: ATTR(=,"_event_id"), VALUE(=,"0");
    2: ATTR(=,"hostname");
    3: ATTR(=,"origin");
    4: ATTR(=,"category");
    5: ATTR(=,"EventType");
    6: ATTR(=,"SID");
    7: ATTR(=,"sub_source");
    8: ATTR(=,"ID");
    9: ATTR(=,"msg");
  MAP
    hostname = $V2;
    origin = $V3;
    category = $V4;
    EventType = $V5;
    SID = $V6;
    sub_source = $V7;
    ID = $V8;
    msg = $V9;
END

CLASS NT_Diskfull
  SELECT
    1: ATTR(=,"_event_id"), VALUE(=,"16");
    2: ATTR(=,"hostname");
    3: ATTR(=,"origin");
    4: ATTR(=,"category");
    5: ATTR(=,"EventType");
    6: ATTR(=,"SID");
    7: ATTR(=,"sub_source");
    8: ATTR(=,"ID");
    9: ATTR(=,"msg");
    10: ATTR(=,"disk");
  MAP
    hostname = $V2;
    origin = $V3;
    category = $V4;
    EventType = $V5;
    SID = $V6;
    sub_source = $V7;
    ID = $V8;
    msg = $V9;
    disk = $V10;
END
```

Figure 38. NT tecad_NT.cds File

The installation of the NT adapter includes a tecad_NT.cds file similar to the tecad_logfile.cds file.

6.7 Windows NT Adapter Configuration

The adapter was installed using the TME desktop as shown in Figure 39.

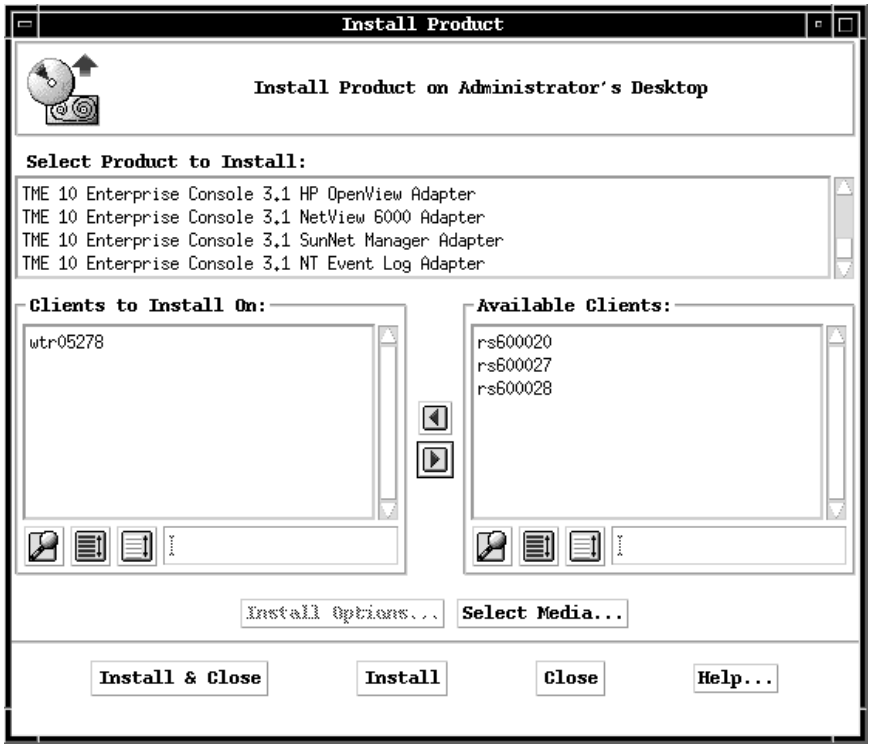


Figure 39. Installing the NT Adapter

The Windows NT adapter runs as an NT service. The adapter will send a T/EC event from information contained in an NT system, application and security logs. These logs are updated by the NT operating system, and by applications running on the system.

The adapter files used by the T/EC logfile adapter are located in the directory of \Tivoli\bin\w32-ix86\TME\TEC\adapters\etc and are listed below:

```
02/07/97  10:21p                8,641 tecad_nt.baroc
02/26/98  10:05a               47,184 tecad_nt.cds
02/26/98  10:05a                1,139 tecad_nt.conf
02/07/97  10:21p                2,301 tecad_nt.err
02/07/97  10:21p               10,990 tecad_nt.fmt
```

The utilities used by the NT adapter are in the directory \Tivoli\bin\w32-ix86\TME\TEC\adapters\bin and are shown below:

```
03/31/97  04:09p                5,632 instlsrv.exe
02/07/97  10:16p               48,128 nt_gencds.exe
03/31/97  04:09p              178,176 tecad_nt.exe
```

Also in the directory \Tivoli\bin\w32-ix86\bin the file:

```
03/31/97  04:08p              180,224 tecadnts.exe
```

For the description of the files (.baroc, .cds, conf, .err and .fmt), please refer to, 6.1, “Files Associated with Event Adapters” on page 103.

For a complete list of the NT adapter event class structure, please refer to *TME 10 Enterprise Console Adapter's Guide*.

6.7.1 Importing the NT Class Definitions

Due to a problem with the NT baroc file we had to modify the ID field setting before importing the baroc file.

The class definition file is shown below:

```
TEC_CLASS :
     NT_Base ISA EVENT
        DEFINES {
                source: default= "NT";
                sub_source: default= "NT";
                sub_origin: default= "N/A";
                adapter_host: default= "N/A";
                msg_catalog: default= "none";
                msg_index: default= 0;
                repeat_count: default= 0;
                category:STRING;
                eventType:STRING;
                sid:STRING;
                id:INT32; 1
                };
END
```

The event class is loaded. We look at the tecad_nt.baroc file and check the error again from the reception log. We found the error message Value does not match type INTEGER. We check the data type integer (INT32) under the NT_Base event class and find out that the slot id:INT32 **1** is expecting an integer value but the NT adapter passes a character value of id=Security **2** as shown in the reception log above.

We change the slot id:INT32 in the E:\test\rules\tecad_nt.baroc file to id:STRING and do the following:

1. Delete the .baroc file from the rule base with the following command:

   ```
   wdelrbclass tecad_nt.baroc CanTire
   ```

2. Validate the class with the following command before compiling:

   ```
   wchkclass tecad_nt.baroc CanTire
   ```

3. Copy the baroc file to our TEC server on rs600028 with the following command:

   ```
   wimprbclass teccad_nt.baroc CanTire
   ```

4. Compile the rules with the following command:

   ```
   wcomprules CanTire
   ```

5. Load the rule base with the following command:

```
wloadrb CanTire
```

6. Stop the event server with the following command:

   ```
   wstopesvr
   ```

7. Start the event server with the following command:

   ```
   wstartesvr
   ```

```
The TME 10 Enterprise Console Server is initializing...
The TME 10 Enterprise Console Server is running.
```

This example event is caused when there is an invalid logon attempt for NT.

```
### EVENT ###
NT_Base;hostname=wtr05095;origin=127.0.0.1;category=2;eventType=AuditFailure;sid
=NT;sub_source=AUTHORITY\SYSTEM;id=Security;msg='529 Logon Failure: Reason: Unkn
own user name or bad password User Name: Administrator Domain: DOMAIN Logon Type
: 2 Logon Process: Advapi Authentication Package: MICROSOFT_AUTHENTICATION_PACKA
GE_V1_0 Workstation Name: WTR05095';date='Mar 11 03:04:30 1998';END

### END EVENT ###
PROCESSED
```

6.7.2 Testing the NT Event Adapter

After the NT adapter installation, the TECNTAdapter was started by the system. To check if the TECNTAdapter is running from the NT desktop select **Settings->Control Panel**, then double-click on the **Services** icon.

You can check if the adapter is running from the dialog below.

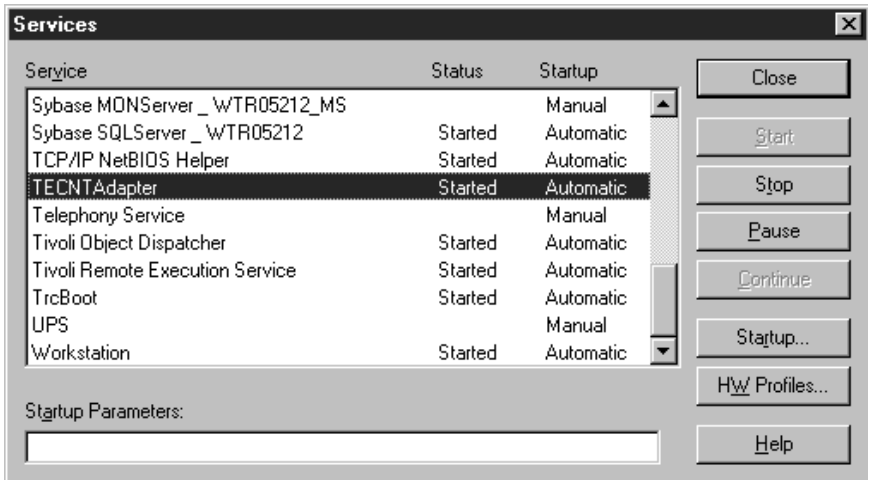


Figure 40. Services

The TECNTAdapter must have the status Started. If not, click the **Start** button.

To start the NT adapter from the command line issue the command:

```
net start TECNTAdapter
```

The command output is shown below:

```
The TECNTAdapter service was started successfully.
```

If you need to stop the NT logfile adapter, click the **Stop** button.

To stop the adapter from the command line issue the following command:

```
net stop TECNTAdapter
```

The output is shown below:

```
The TECNTAdapter service is stopping.
The TECNTAdapter service was stopped successfully.
```

We checked the reception log after installing the adapter to see if the NT adapter is sending events to the server.

There are no events coming from the NT adapter because the logging/auditing policy in NT is not enabled yet. First, let's talk about some of the events that are related to the system, security and application, that NT adapter can send to the T/EC server.

The events will only be logged or audited in the system by enabling the NT logging facility:

1. From the NT desktop select **Programs->Administrative Tools->User Manager Domain**.

   This shows the User Manager window (see Figure 41).

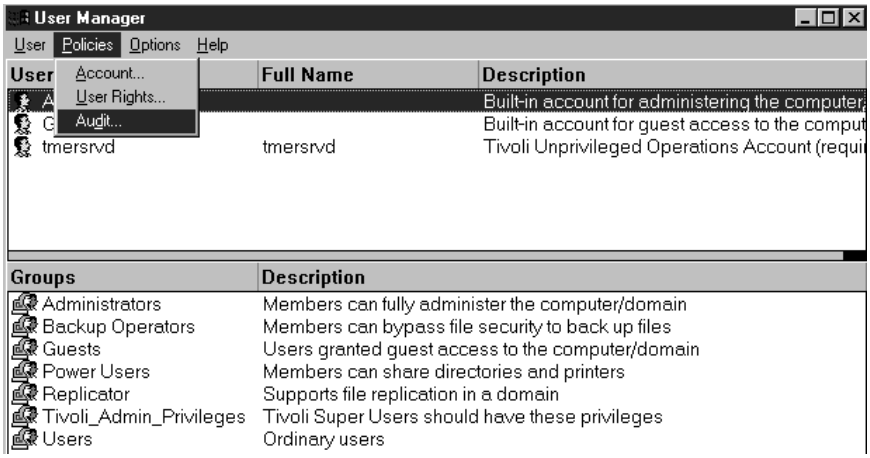


Figure 41. User Manager

2. Click on **Policies->Audit** from the menu bar.

   The system gives us the Audit Policy screen which has Do Not Audit checked.

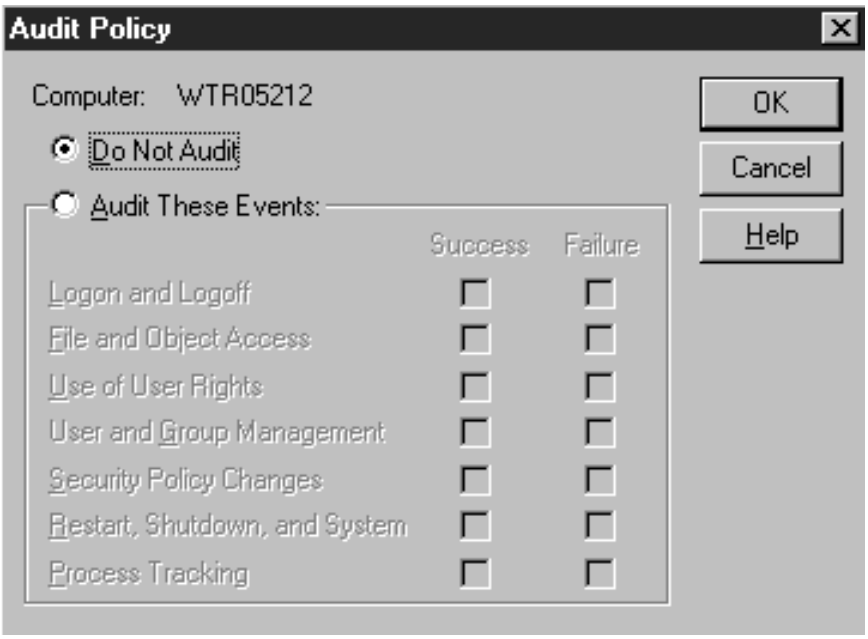


Figure 42. Audit Policy

3. Click on **Audit These Events**.

   Now we can enable these events under Success or Failure so that the system can log/audit these events. We are only interested with Success logging for all events.

4. Select Failure for all event types.

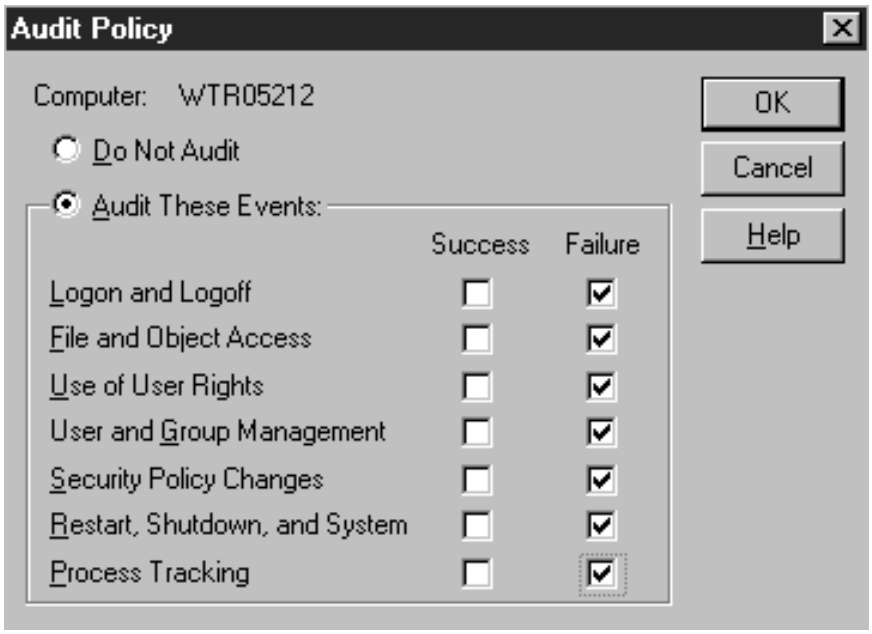


Figure 43. Audit Policy

5. Click **OK** and close the User Manager window.

The NT adapter can start sending messages to the T/EC server. We started a Tivoli desktop and entered the wrong password to see if the security failure would be detected.

The wtdumprl command shows the following:

```
1 102 0 889594430(Mar 11  1998)
### EVENT ###
NT_Base;hostname=wtr05095;origin=127.0.0.1;category=2;eventType=AuditFailure;sid
=NT;sub_source=AUTHORITY\SYSTEM;id=Security 2 ;
msg='529 Logon Failure: Reason:
Unknown user name or bad password User Name: Administrator Domain:
DOMAIN Logon Type: 2 Logon Process: Advapi Authentication Package:
MICROSOFT_AUTHENTICATION_PACKAGE_V1_0 Workstation Name: WTR05095';
date='Mar 10 23:33:45 1998';END
```

6.7.3 Example Using the NT Performance Monitor

This example uses the NT Performance Monitor to generate TEC events.

1. Select **Start->Programs->Administrative Tools->Performance Monitor**.

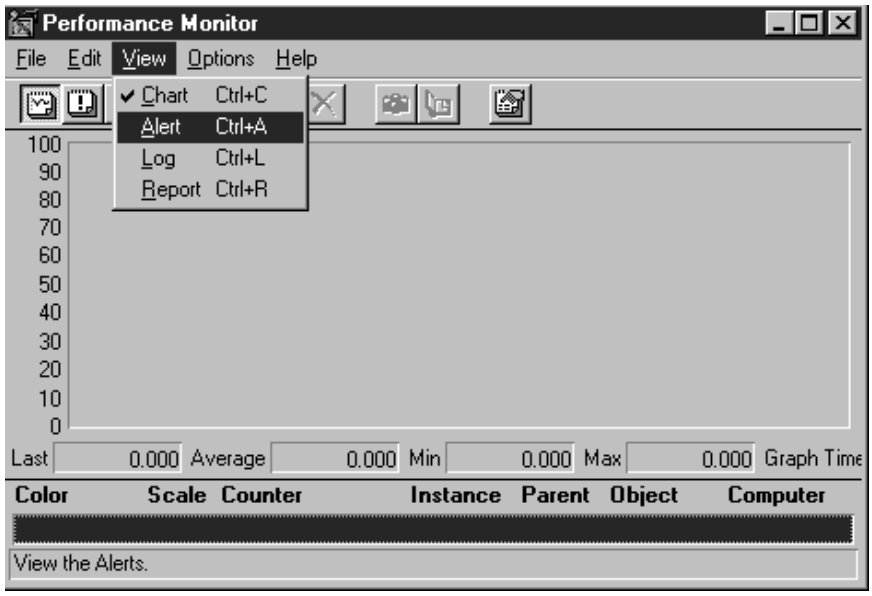


Figure 44. Performance Monitor

2. Select **View->Alert** to view the alerts.

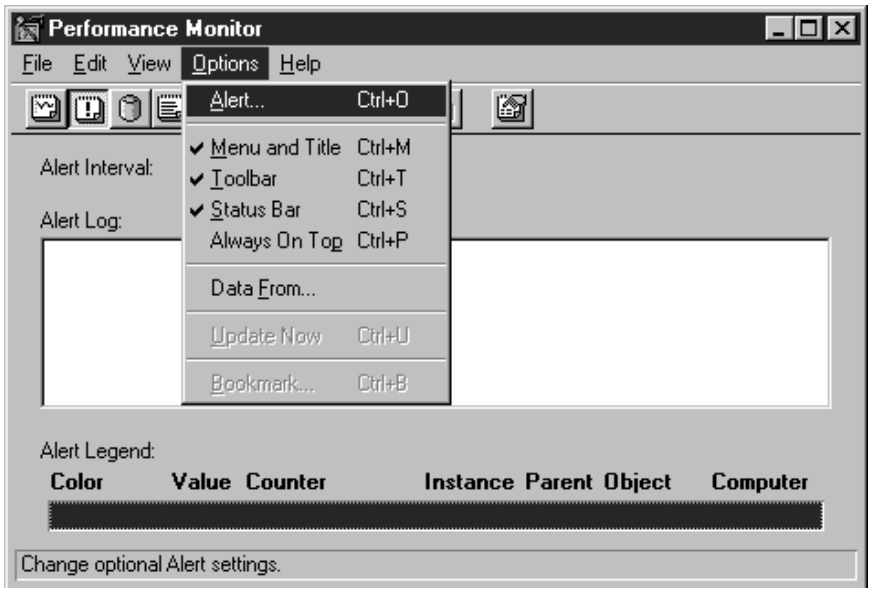


Figure 45. Performance Monitor

3. Select **Options->Alert**.

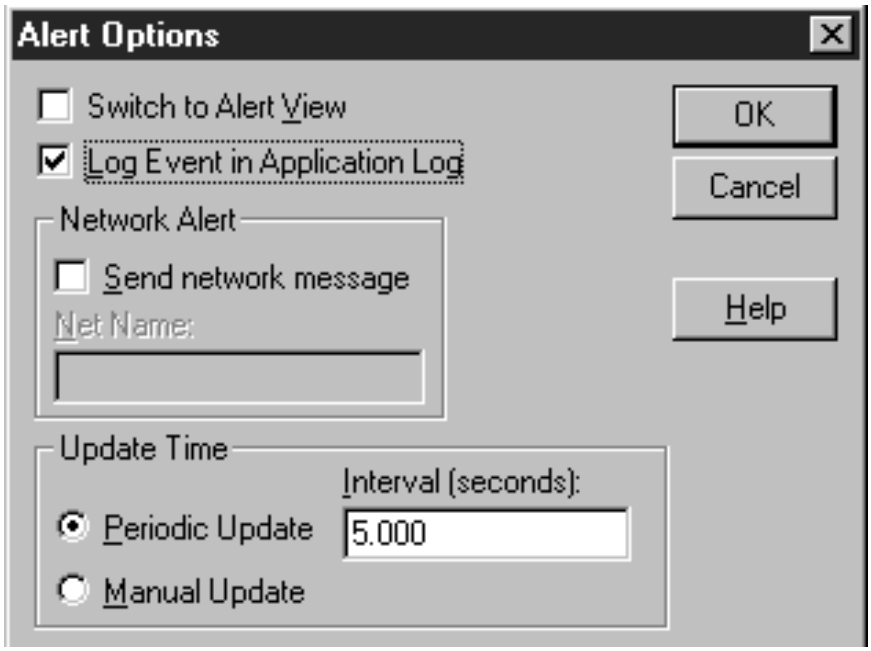


Figure 46. Performance Monitor

4. Click on **Log Event in Application Log** and click **OK** to change the status in order to log events.

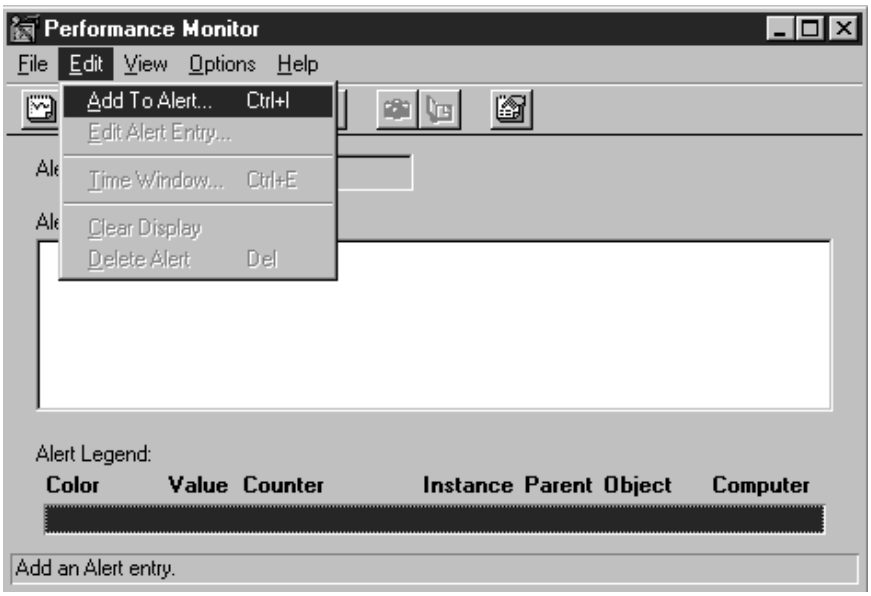


Figure 47. Performance Monitor

5. Select **Edit->Add To Alert** to add a new alert.

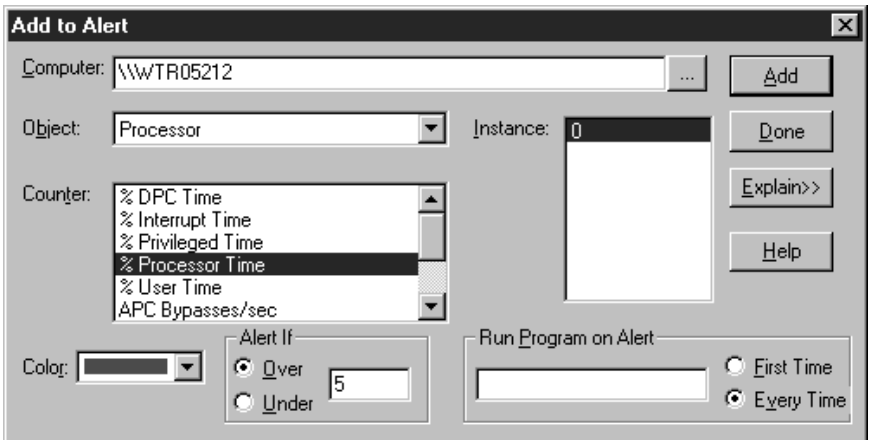


Figure 48. Add to Alert

6. Select the options shown in Figure 48. We chose Processor Time and set the threshold to send an event when the value is greater than 5. Press the Enter key followed by selecting **Add** when complete.

   At this point you can see the alert log being updated since we ask the system to report for every 5% of CPU processor time. You can save your alert settings, create others, etc. But for now, we leave the window open since we have not started the NT adapter yet.

To run the adapter in debug mode we executed the following command:

```
tecad_nt -d -c E:\Tivoli\bin\w32-ix86\TME\TEC\adapters\etc\tecad_nt.conf
```

To test this we signed onto NT with an incorrect password. The message generated is shown below:

```
matched NT_Performance_Alert
        $1 is  Mar 11 05:33:44'
        $2 is  1998'
        $3 is  0'
        $4 is  Information'
        $5 is  N/A'
        $6 is  PerfMon'
        $7 is  2000'
        $8 is  \\WTR05095 ; Object: Processor ; Counter: % Processor Time ; Inst
ance: 0 ; Parent: ; Value: 12.403 ; Trigger: > 5.000'
Mar 11 05:33:49 1998 0 Information N/A PerfMon 2000 An Alert condition has occur
red on Computer: \\WTR05095 ; Object:   Processor ;  Counter:  % Processor Time
; Instance: 0 ; Parent:     ; Value:          86.600 ;  Trigger:  >        5.000

matched NT_Performance_Alert
        $1 is  Mar 11 05:33:49'
        $2 is  1998'
        $3 is  0'
        $4 is  Information'
        $5 is  N/A'
        $6 is  PerfMon'
        $7 is  2000'
        $8 is  \\WTR05095 ; Object: Processor ; Counter: % Processor Time ; Inst
ance: 0 ; Parent: ; Value: 86.600 ; Trigger: > 5.000'

Mar 11 05:39:59 1998 2 AuditFailure NT AUTHORITY\SYSTEM Security 529 Logon Failu
re:
        Reason:         Unknown user name or bad password
        User Name:      Administrator
        Domain:         DOMAIN
        Logon Type:     2
        Logon Process:  Advapi
        Authentication Package: MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
        Workstation Name:       WTR05095
matched NT_Base
        $1 is  Mar 11 05:39:59'
        $2 is  1998'
        $3 is  2'
        $4 is  AuditFailure'
        $5 is  NT'
        $6 is  AUTHORITY\SYSTEM'
        $7 is  Security'
        $8 is  529 Logon Failure: Reason: Unknown user name or bad password User
 Name: Administrator Domain: DOMAIN Logon Type: 2 Logon Process: Advapi Authenti
cation Package: MICROSOFT_AUTHENTICATION_PACKAGE_V1_0 Workstation Name: WTR05095
'
```

To stop running the adapter in debug mode press Ctrl-c.

Chapter 7. Logfile Adapter Utilities

This chapter shows examples of using the following tools to configure and distribute the logfile adapter configuration files:

- Logfile Adapter Configuration Facility (ACF)
- Manually adding a new logfile to be monitored by the logfile adapter
- Using the Logfile Format Editor

We show simple examples of how to use these tools. These utilities are useful for large installations and allow a level of control to be applied when deploying the logfile adapter to large numbers of UNIX servers.

7.1 Using the T/EC Adapter Configuration Facility

The T/EC Adapter Configuration Facility (ACF) provides a GUI interface that makes it easier to configure and customize the event adapters. This eliminates the need to manually edit the files and allows you to update multiple adapters centrally. The adapter configuration files (as profiles) can be distributed using the TME 10 Framework.

We installed the ACF on our TMR server rs600020 and the managed node rs600027 since we will be distributing the adapter configuration record profile onto rs600027. The installation follows the standard TME 10 product installation.

We installed the ACF as shown in Figure 49.

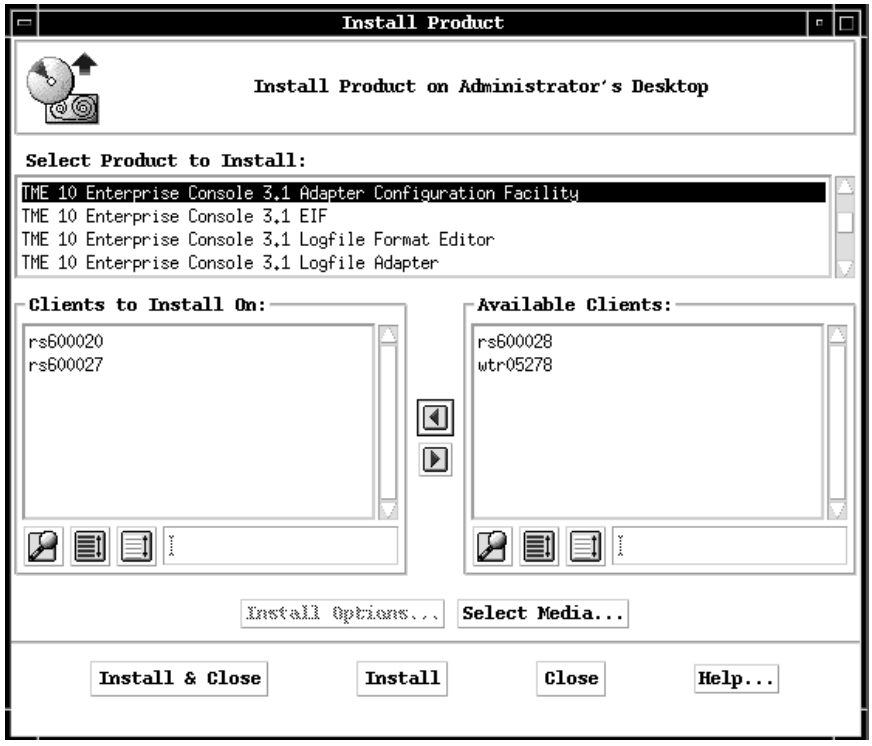


Figure 49. Installation of the ACF

Now that the software is installed we can use the ACF.

7.1.1 Updating the Event Adapter Configuration Files

The following steps were done to add the configuration file (tecad_logfile.conf) and distribute the file onto rs600027.

We used the CLI to set up the ACP profile. To begin we set the resource ACP as the managed resource, which was created when ACF was installed, as root on rs600020:

```
wsetpr ACP @PolicyRegion:TEC_MANAGEMENT
```

Create a profile manager ACFPrfMgr for the ACP profiles with the following command:

```
wcrtprfmgr @PolicyRegion:TEC_MANAGEMENT ACFPrfMgr
```

The command output is shown below.

```
1724816353.1.926#TMF_CCMS::ProfileManager#      ACFPrfMgr
```

Create an adapter configuration profile record ACFPrf with the following command:

```
wcrtprf @ProfileManager:ACFPrfMgr ACP ACFPrf
```

The command output is shown below.

```
1724816353.1.927#ACP::Prof#     ACFPrf
```

Next we added the subscribers with the following command:

```
wsub @ProfileManager:ACFPrfMgr @ManagedNode:rs600027
```

The next steps are all done using the GUI interface.

From the policy region rs60020-region double-click on the **TEC_MANAGEMENT** region (see Figure 50 on page 135).

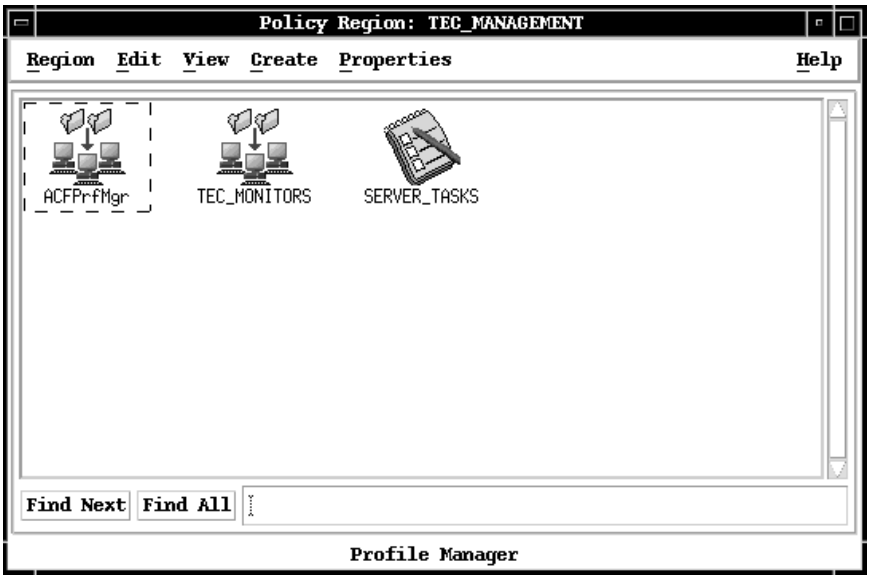


Figure 50. The TEC_MANAGEMENT Policy Region

You will be presented with the Profile Manager screen (see Figure 51 on page 136).

Double-click on the profile manager icon labeled **ACFPrfMgr**.

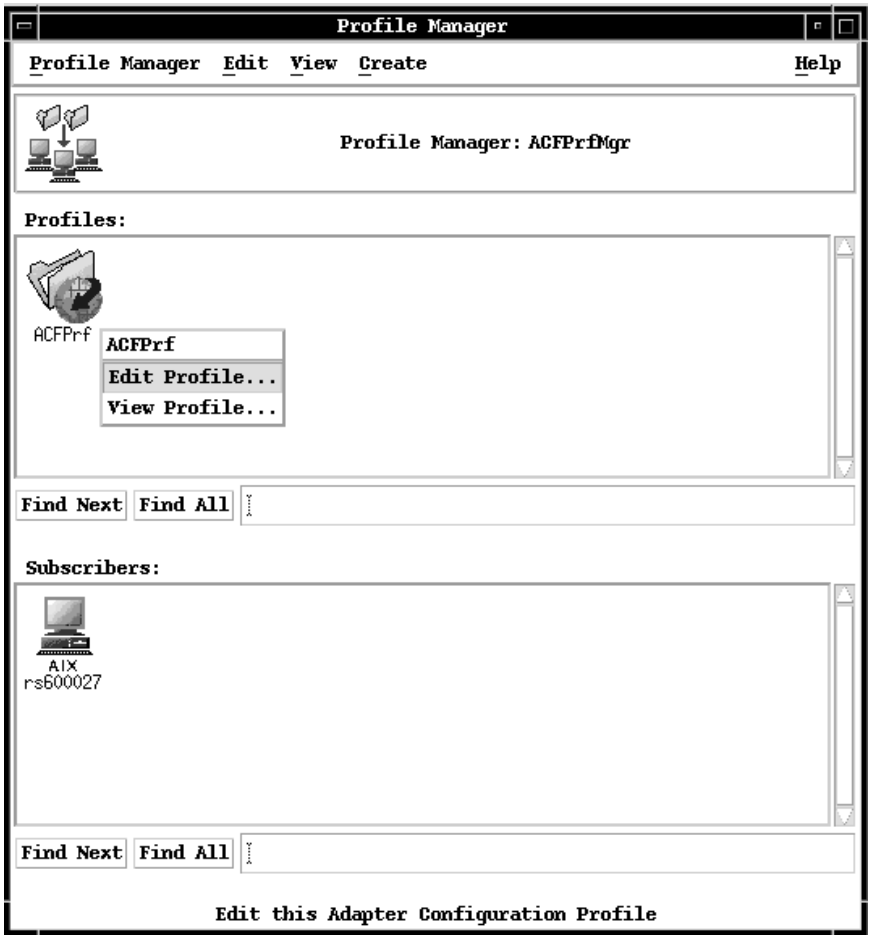


Figure 51. Profile Manager

From the Adapter Configuration Profile window double-click on the profile **ACFPrf**.

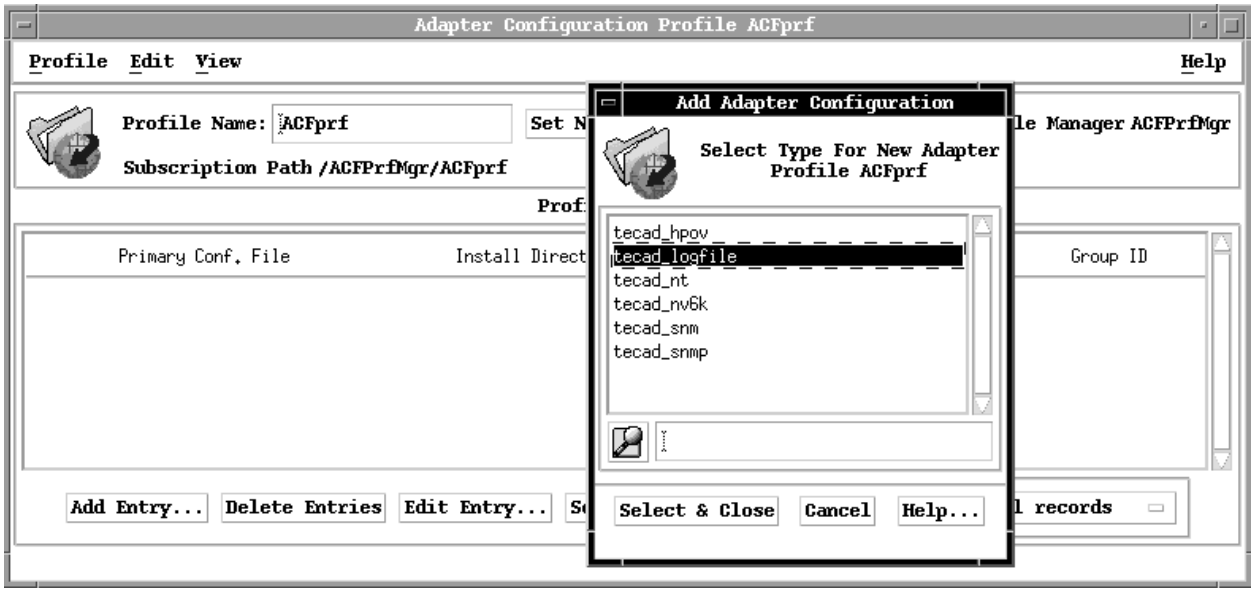


Figure 52. Selecting the Logfile Adapter File

You will be presented with the Add Adapter Configuration screen.

Click on **Add Entry..** and select the **tecad_logfile** file for the new adapter profile ACFPrf.

Click on **Select & Close**.

You will be presented with the Edit Adapter Configuration window (see Figure 53).

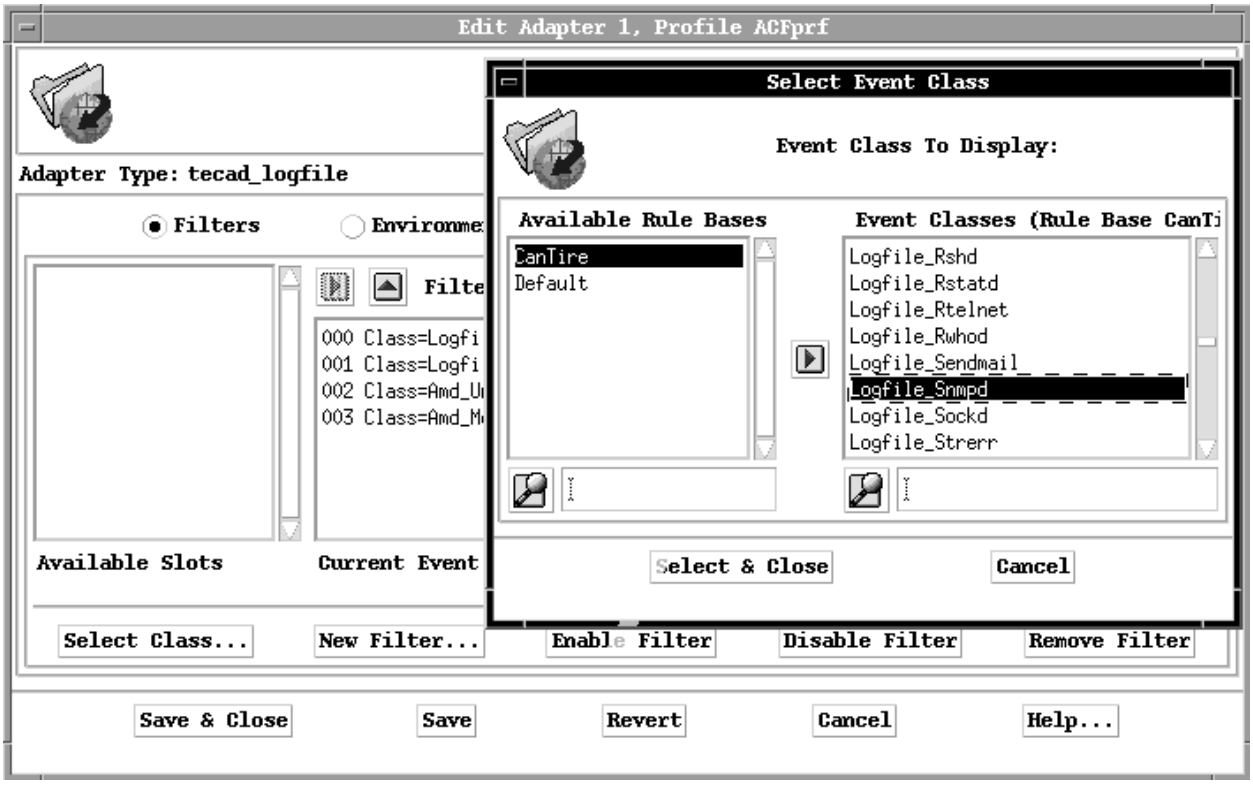


Figure 53. Adapter Configuration Screen

There are five selections where you can make adapter modifications: Filter, Environment, Actions, Distribution and General.

7.1.2 Filter

You can add a new filter, remove a filter, disable a filter (indicated by # after the sequence number) and enable the filter.

Since we are at the Filter dialog screen, we will use this screen to a add filter definition. There are already four filter statements added as defaults.

We will add a filter statement for the class Logfile_Snmpd, which we kept receiving after the logfile adapter was installed. The messages we do not want to send to the TEC are as follows:

```
1 1999 0 889409000(Mar 08 21:03:20 1998)
### EVENT ###
Logfile_Snmpd;hostname=rs600027;date='Mar 8 21:03:20';origin=9.24.104.188;msg='E
XCEPTIONS: authentication error: invalid community name: ITSO';pid=18584;sub_sou
rce=snmpd;END

### END EVENT ###
PROCESSED

1 2000 0 889409097(Mar 08 21:04:57 1998)
### EVENT ###
Logfile_Snmpd;hostname=rs600027;date='Mar 8 21:04:57';origin=9.24.104.188;msg='E
XCEPTIONS: authentication error: invalid community name: ITSO';pid=18584;sub_sou
rce=snmpd;END

### END EVENT ###
PROCESSED
```

From the Edit Adapter Profile ACFPrf screen, click on the **New Filter...** button.

You will be presented with the Select Event Class window.

We know that our default rule base is CanTire. You can check by using the command wlscurrb.

Select **CanTire**, and then select the right arrow.

The dialog lists the event classes defined in the rule base CanTire. Next select **Logfile_Snmpd** from the list of the event classes.

Click on **Select & Close**.

This takes us back to the Edit Adapter Configuration dialog with the list of available slots for class Logfile_Snmpd. You may choose to narrow down your filtering by selecting the available slot (**Select Class** button). But in our example we are just interested for the event class Logfile_Snmpd.

The class Logfile_Snmpd was added onto Current Event Filters.

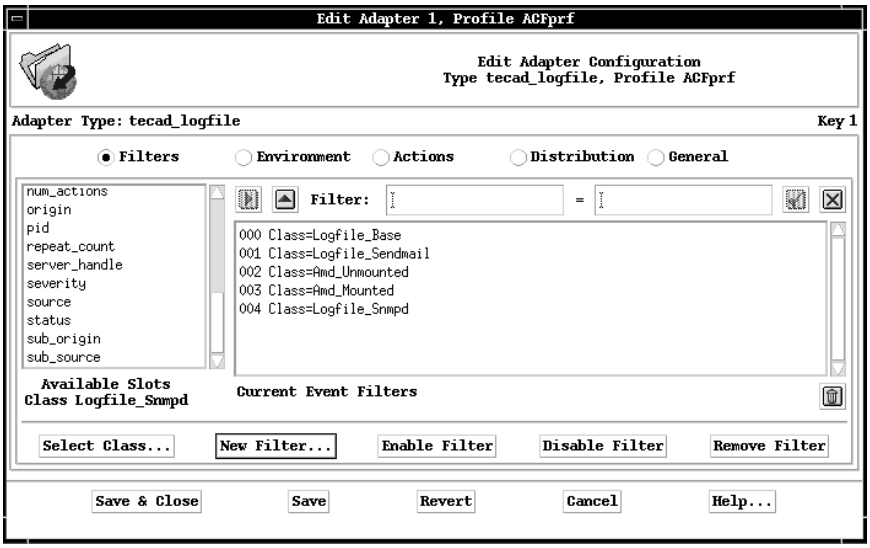


Figure 54. Adding a New Filter

7.1.3 Environment

The next step is to change the environment. Click on the **Environment** button.

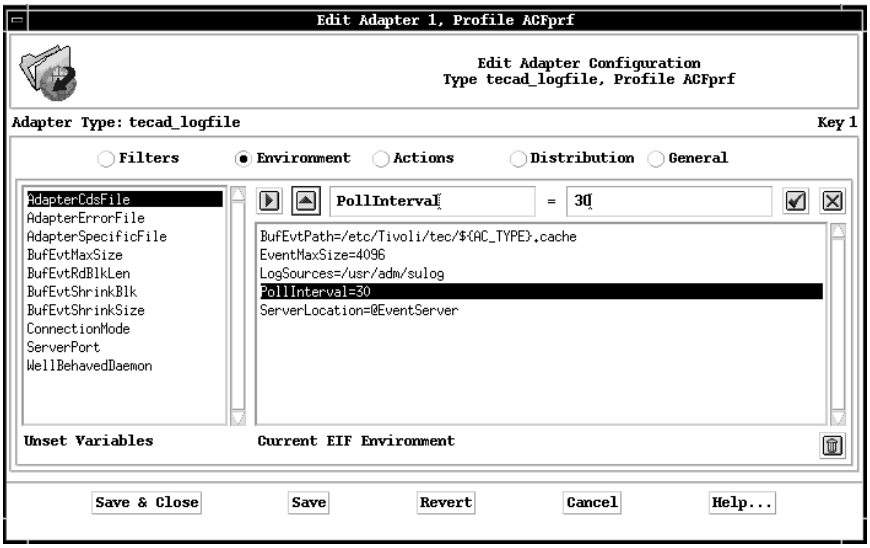


Figure 55. Changing the Environment Variables

We are presented with the environment definitions and all the keywords used in the .conf file.

The environment variables that are used by the adapter to control its behavior can be defined on this screen. For example, LogSources=/usr/adm/sulog specifies the location of the logfiles to poll and PollInterval=30 specifies to poll the /usr/adm/sulog for new messages every 30 seconds.

To show the configuration change we will change the polling interval from 30 to 60 seconds. To do this click on **PollInterval=30** to select it, then click on the up arrow button, modify the value 30 to be 60 and then click on the check button.

PollInterval=30 was changed to PollInterval=60. Select the **Action** button.

7.1.4 Distribution

The next step is to change the actions to take before and after distribution. You can configure the adapter configuration record to perform actions on the subscribers upon distribution.

You can also add your own scripts from this dialog by entering the script in the Before file distribution or After file distribution list.

In our example, for the before distribution, we selected **Abort distribution** from the When action fails selection. For the after distribution, we selected to disable the action from the Actions selection dialog as shown in the next window.

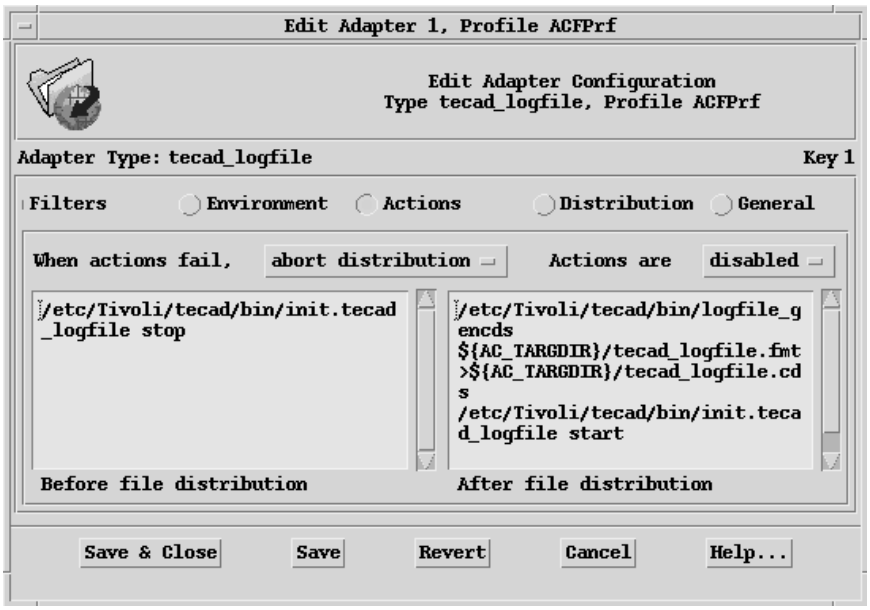


Figure 56. Distribution Options

The next step is to change the distribution. Click the **Distribution** button.

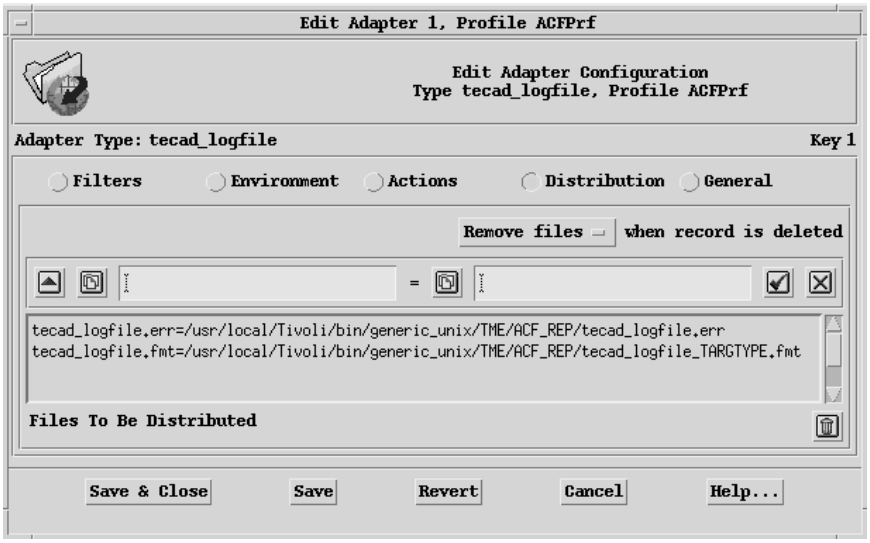


Figure 57. Distribution Options

You can also add other distribution files from this dialog by entering the destination file name on the left-hand box and the source file name on the right-hand box.

When an adapter configuration record profile is deleted, the ACP can also remove all distributed files. The control at the top, When record is deleted, allows you to set this option.

In our example we are not interested in distributing the .fmt and .err files, so we just removed them from Files To Be Distributed. Select the files and click on the trashcan button.

The files were removed from the Files To Be Distributed list.

The next step is to change the the General options.

7.1.5 General

Click on **General** to see the General options.

This dialog is where we set how the adapter configuration record will be installed at the endpoint or target node (rs600027).

The dialog gave us the install directory ./etc/Tivoli/tecad/etc and the .config file. We don't need to change the install directory since this is where the (.conf) file resides.

Also each record is given a user and group ID that determines the ownership of distributed files and which user ID is allowed to perform the action.

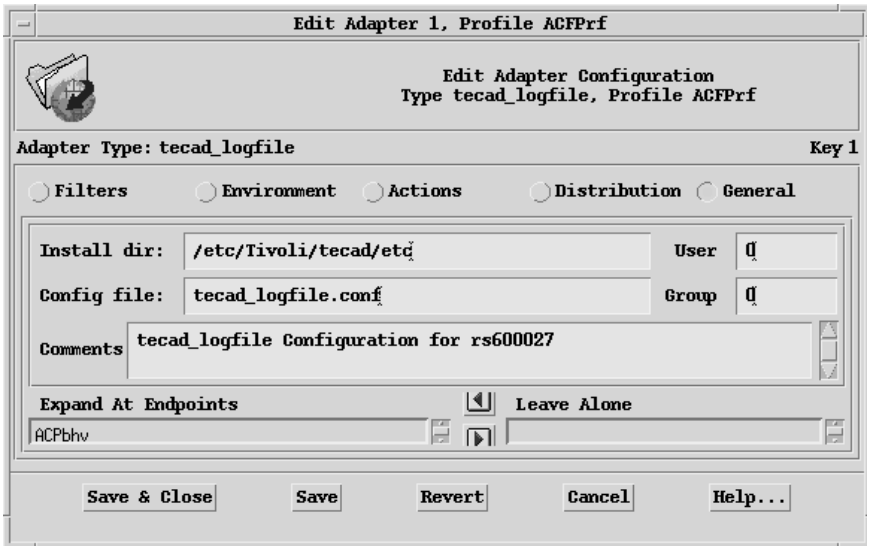


Figure 58. General Options

We only change the comments to tecad_logfile Configuration for rs6020027. Now we can click **Save & Close**.

The next step is to add rs600027 as a subscriber to the profile manager ACFPrfMgr.

The next steps will be done using the CLI commands. You can use the GUI desktop to do these steps.

7.1.6 Distributing the Files

We set up the subscriber for this profile as shown below:

```
wsub @ProfileManager:ACFPrfMgr @ManagedNode:rs600027
```

To distribute these files click on **Distribute** (see Figure 59).

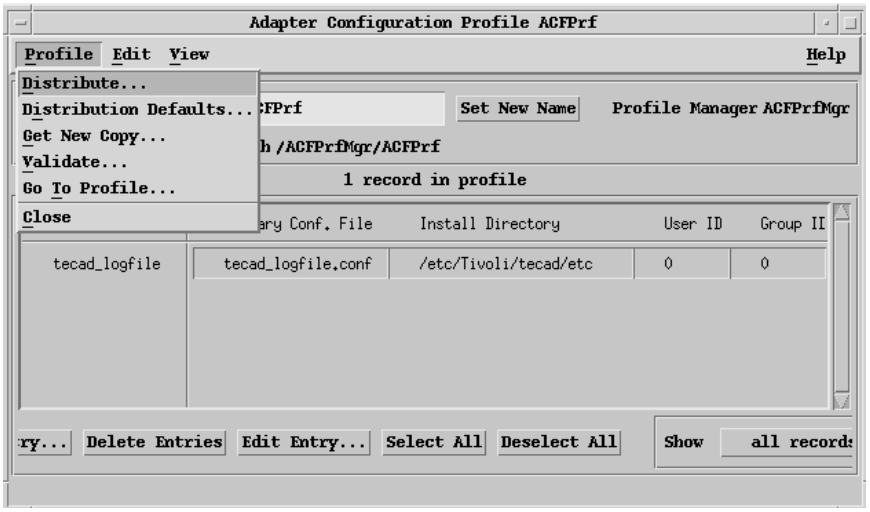


Figure 59. Distributing the Files

Select **Distribute**.

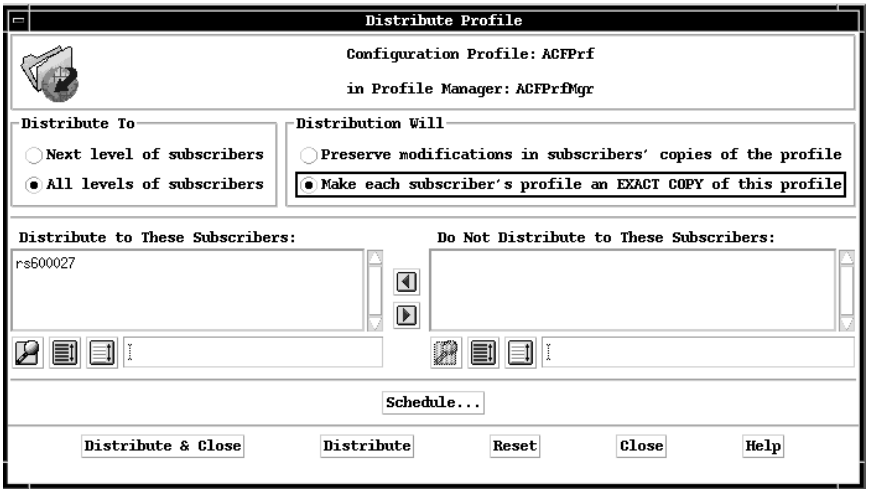


Figure 60. Distribute to Profile

Select **Distribute & Close**. This distributes the file to rs600027.

The final step is to check whether the adapter configuration record profile was distributed to rs600027. On rs600027 we looked at the file:

```
cat /etc/Tivoli/tecad/etc/tecad_logfile.conf
```

The command output is as follows.

```
# Mon Mar  9 06:03:09 1998
#
# tecad_logfile Configuration for rs600027  1
#
ServerLocation=@EventServer
EventMaxSize=4096
LogSources=/usr/adm/sulog
PollInterval=60
#
Filter:Class=Logfile_Base
Filter:Class=Logfile_Sendmail
Filter:Class=Amd_Unmounted
Filter:Class=Amd_Mounted
Filter:Class=Logfile_Snmpd; 2
```

The comments 1 were changed and the filter statement for Logfile_Snmpd 2 was added.

We then stopped and restarted the logfile adapter with the following commands:

```
/etc/Tivoli/tecad/bin/init.tecad_logfile stop
/etc/Tivoli/tecad/bin/init.tecad_logfile start
```

For more information on using the ACP, please refer to the *TME 10 Enterprise Console User's Guide*.

7.2 Manually Adding a New Logfile for Monitoring

This section shows how to use the logfile adapter to monitor a log file created from an application. In this example we use the oracle listener log. The process can be used for any text file that is updated and requires monitoring.

The steps are as follows:

1. Work out the format of the logfile
2. Create a new baroc file
3. Import the new baroc file into the current rule base
4. Append to the tecad_logfile_fmt
5. Generate the cds file
6. Add log sources to the conf file
7. Stop and start the logfile adapter

The file we used contains Oracle errors as listed below:

```
TNSLSNR for IBM/AIX RISC System/6000: Version 2.3.3.0.0 - Production

Copyright (c) Oracle Corporation 1994.  All rights reserved.

Listening on: (ADDRESS=(PROTOCOL=ipc)(DEV=6)(KEY=tec.world))
Listening on: (ADDRESS=(PROTOCOL=ipc)(DEV=10)(KEY=tec))
Attempted to listen on: (DESCRIPTION=(CONNECT_TIMEOUT=10)(ADDRESS=(COMMUN
 .world)(PROTOCOL=TCP)(Host=rs60028)(Port=1521)))

03/10/98 10:16 TNS-12545: Connect failed because target host or object do
exist
03/10/98 10:17 TNS-12560: TNS:protocol adapter error
03/10/98 10:18 IBM/AIX RISC System/6000 Error: 78: Connection timed out
```

The requirement is to generate an event when these are logged. We added the following lines to the logfile adapter baroc file /etc/Tivoli/tecad/tecad_logfile.baroc.

```
#
#   tecad_logfile_oracle.baroc

TEC_CLASS :
        Oracle_Base ISA Logfile_Base
        DEFINES {
                sub_source: default= "ORALOG";
                ora_date: STRING, default="";
                ora_time: STRING, default="";
        };
END

TEC_CLASS :
        ORA_TNS-12545 ISA Oracle_Base
        DEFINES {
                severity: default = CRITICAL;
        };

TEC_CLASS :
        ORA_TNS-12560 ISA Oracle_Base
        DEFINES {
                severity: default = HARMLESS;
        };
```

The Oracle_Base class definition comprises of the source, date and time fields. We defined a separate class for each message we are looking for within our logfile. The class definition ORA_TNS_12545 will find the message contained in the logfile as shown below:

```
03/10/98 10:17 TNS-12560: TNS:protocol adapter error
```

Next we had to append the following lines to the tecad_logfile.fmt file:

```
FORMAT Oracle_Base
%s %s TNS%s: %s*
hostname DEFAULT
origin DEFAULT
ora_date: $1
ora_time: $2
date PRINTF("%s %s", ora_date, ora_time)
ora_error $3
msg $4
END

FORMAT ORA_TNS-12545 FOLLOWS Oracle_Base
%s %s TNS-12545: %s*
ora_error $2
msg $3
END

FORMAT ORA_TNS-12560 FOLLOWS Oracle_Base
%s %s TNS-12560: %s*
ora_error $2
msg $3
```

To compile our new definition we change the directory to /etc/Tivoli/tecad on rs600028 and compile the new cds using the command:

```
./bin/logfile_gencds ../etc/tecad_logfile.fmt > ../etc/tecad_logfile.cds
```

We also added the name of the oracle logfile to the tecad_logfile.conf by appending the line shown below:

```
LogSources=/usr/oracle/listener.log
```

Now we can stop and restart the logfile adapter with the following commands:

```
../bin/init.tecad_logfile stop
../bin/init.tecad_logfile start
```

When the error was reported in the oracle error an event was sent to the TEC. This is shown in Figure 61.

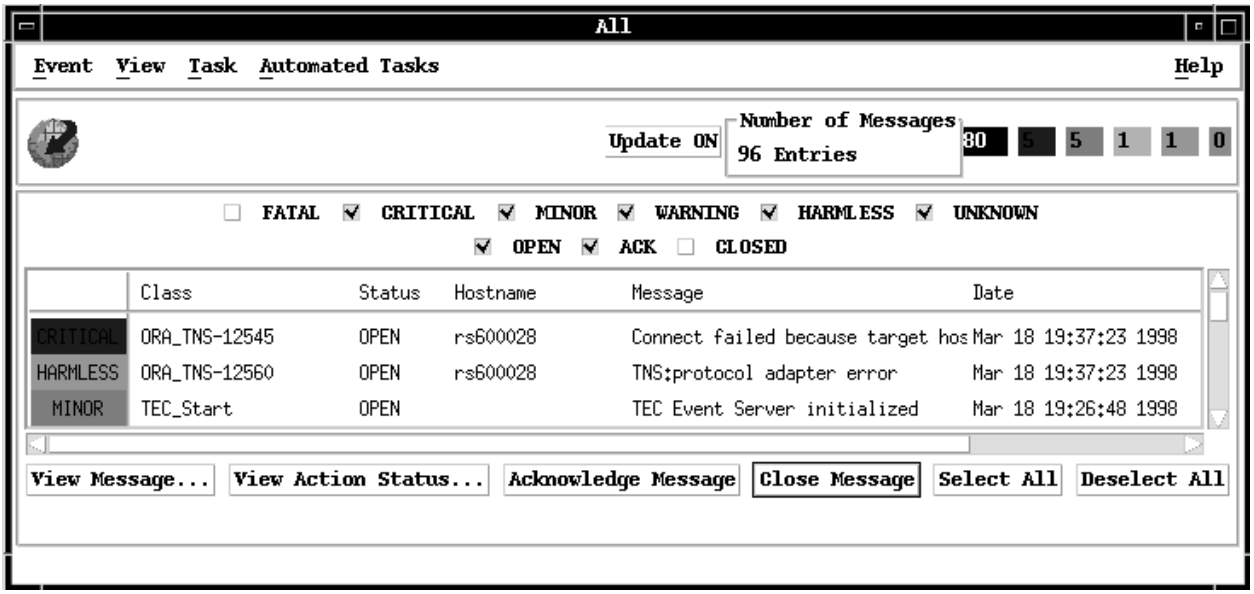


Figure 61. The Oracle Event

The amending of the logfile configuration files can also be performed using the GUI.

7.3 Using the Logfile Format Editor

The T/EC Logfile Format Editor is used to add new format definitions (.fmt) of event messages and their mapping to the T/EC events for the logfile event adapter.

The Format Logfile Editor must be installed on the machine where the T/EC server is installed. In our environment, it was installed on the T/EC server rs600028. The installation follows the standard TME 10 product installation.

You can use the Logfile Format Editor (only through the GUI) in two ways:

- It is one of the options in the ACP Edit Adapter Configuration profile as shown in Figure 53 on page 137.

- From the Administrator desktop, right click on **EventServer** icon to get the pull-down menu and select **Configure Logfile...** (see Figure 62 on page 147).

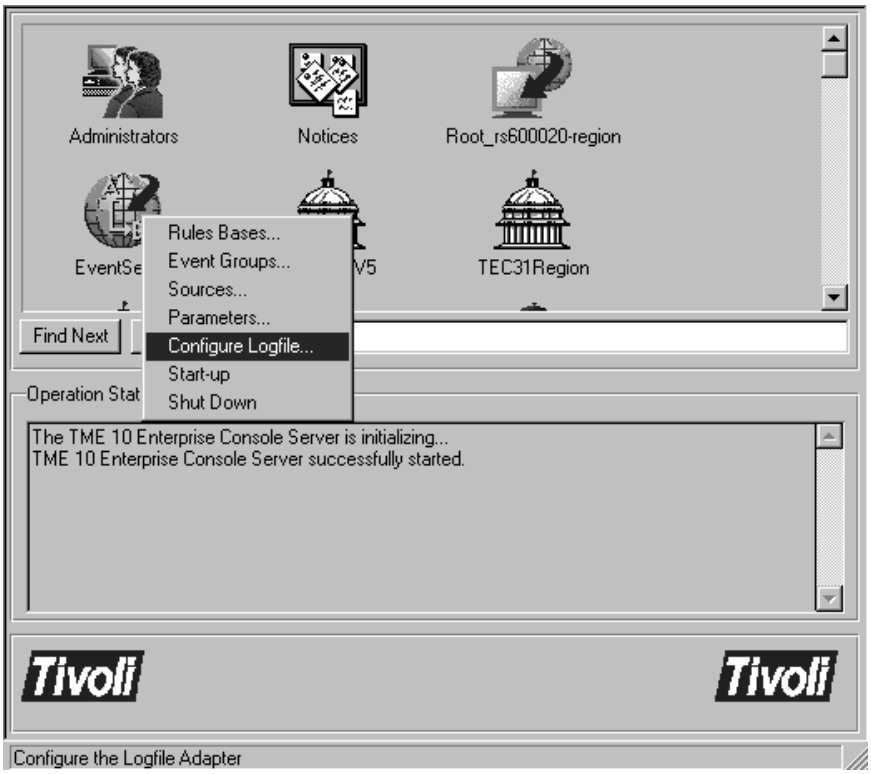


Figure 62. TME Desktop for Administrator

The *TME 10 Enterprise Console User's Guide* has a detailed description and examples on how to add new format definitions (.fmt) of event messages and their mapping using the GUI Logfile Configuration facility.

The major steps involved in using the LCF are:

1. Setting up which format file, selecting the class from the rule base and opening the logfile (that is, /var/adm/sulog) to be used.
2. Editing the format file by setting up the format string, selecting or creating an event class (baroc) and assigning the slot mapping.
3. Saving the changes.
4. Regenerating the .cds from the new fmt created.
5. Stopping and restarting the logfile adapter.

7.4 Installing the Logfile Format Editor

The following examples show to include a new logfile for monitoring by editing the files using the Logfile Configuration Facility.

The Logfile Format Editor edits the logfile configuration files using the GUI. The files that are modified are the baroc and fmt configuration files.

We installed the Logfile Format Editor on our TEC server rs600028 as shown in Figure 63 on page 148.

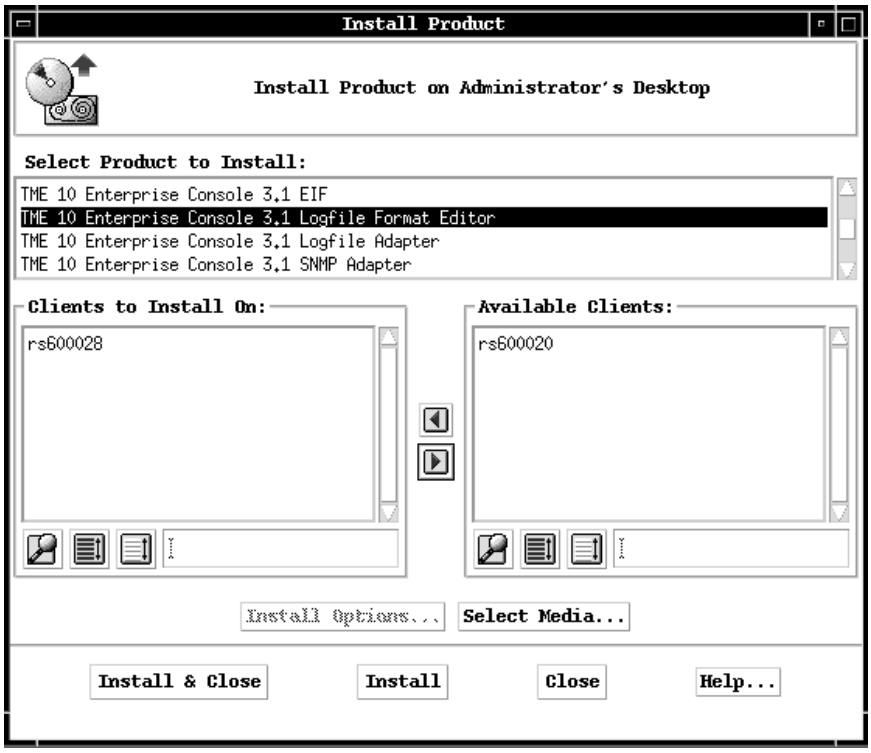


Figure 63. Installing the Logfile Format Application

Once the Logfile Format Editor is installed we start the Logfile Configuration Facility by selecting **Configure Logfile** as shown in Figure 64 on page 149.

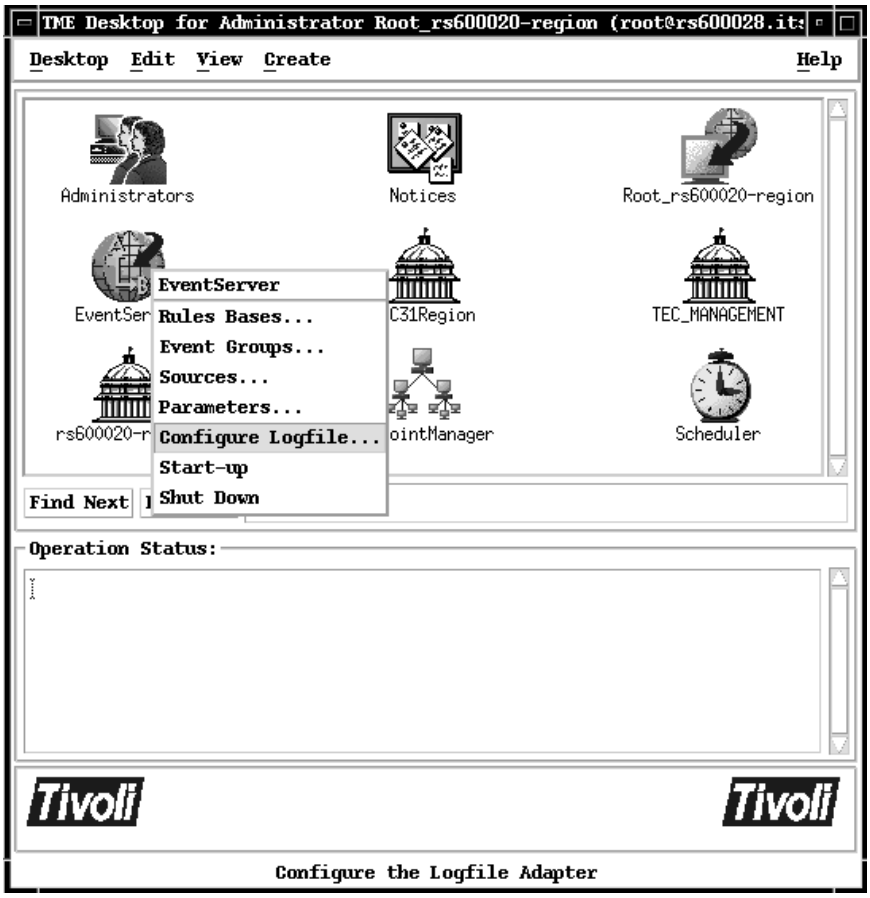


Figure 64. Starting the Configuration Application

Next we need to select the format file we will update. Select **File** followed by **Open Format** (see Figure 65 on page 150).

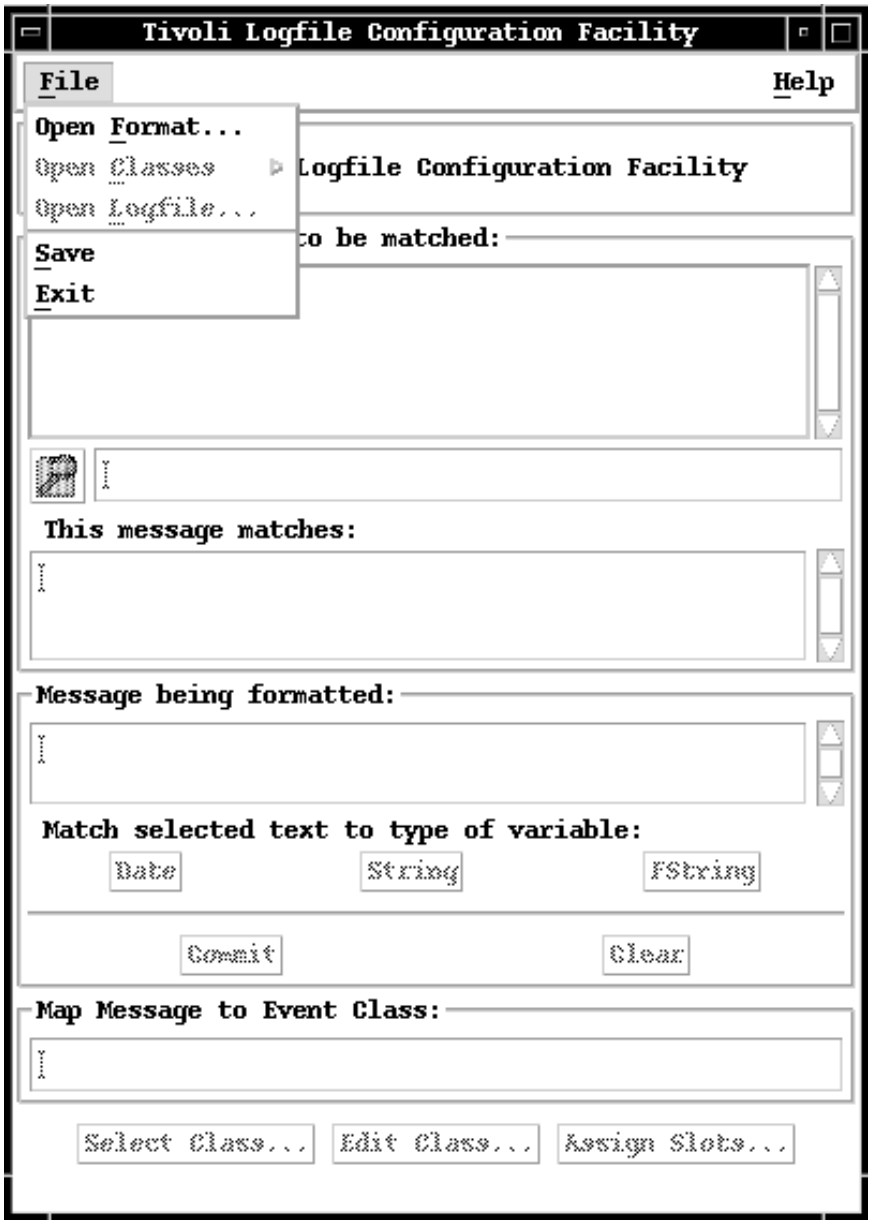


Figure 65. Opening the Format File

The format file we are amending is the logfile format file called tecad_logfile.fmt located in the directory /etc/Tivoli/tecad/etc. We entered this information as shown in Figure 66 on page 151.

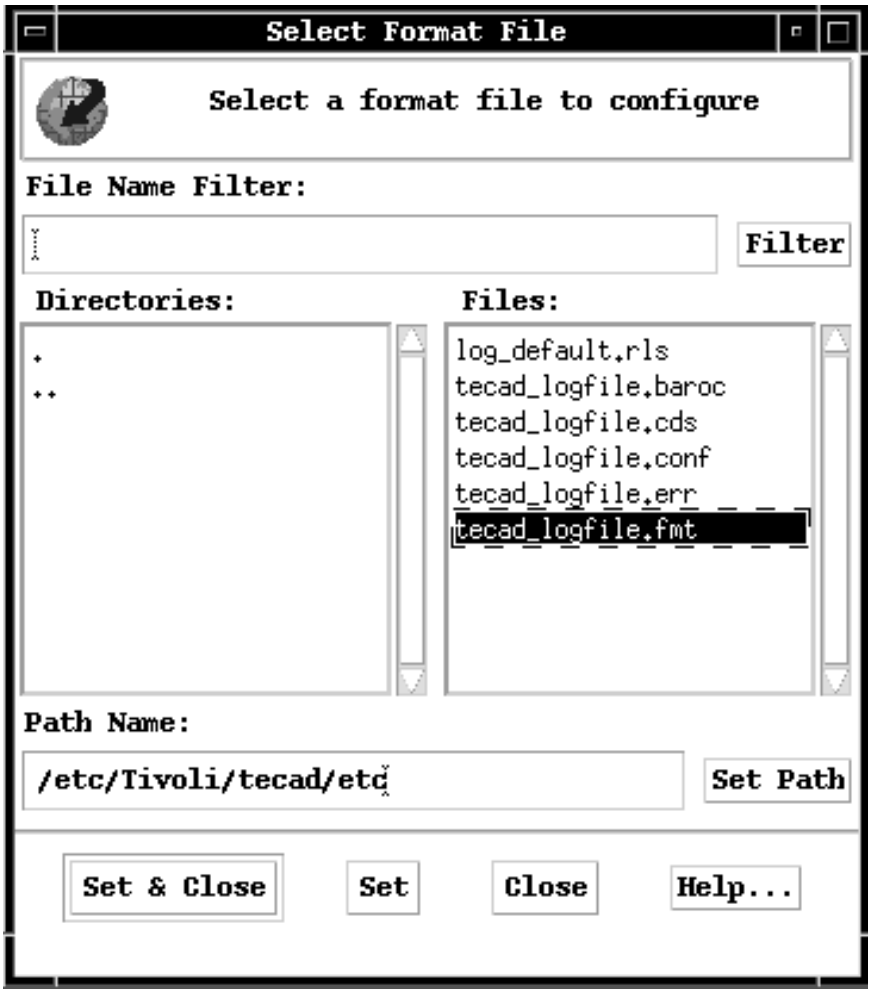


Figure 66. Format File Options

Once entered select **Set & Close**.

From the screen Figure 65 on page 150 we selected the class by choosing **Open Classes**. All defined rule bases will appear (see Figure 67 on page 152).

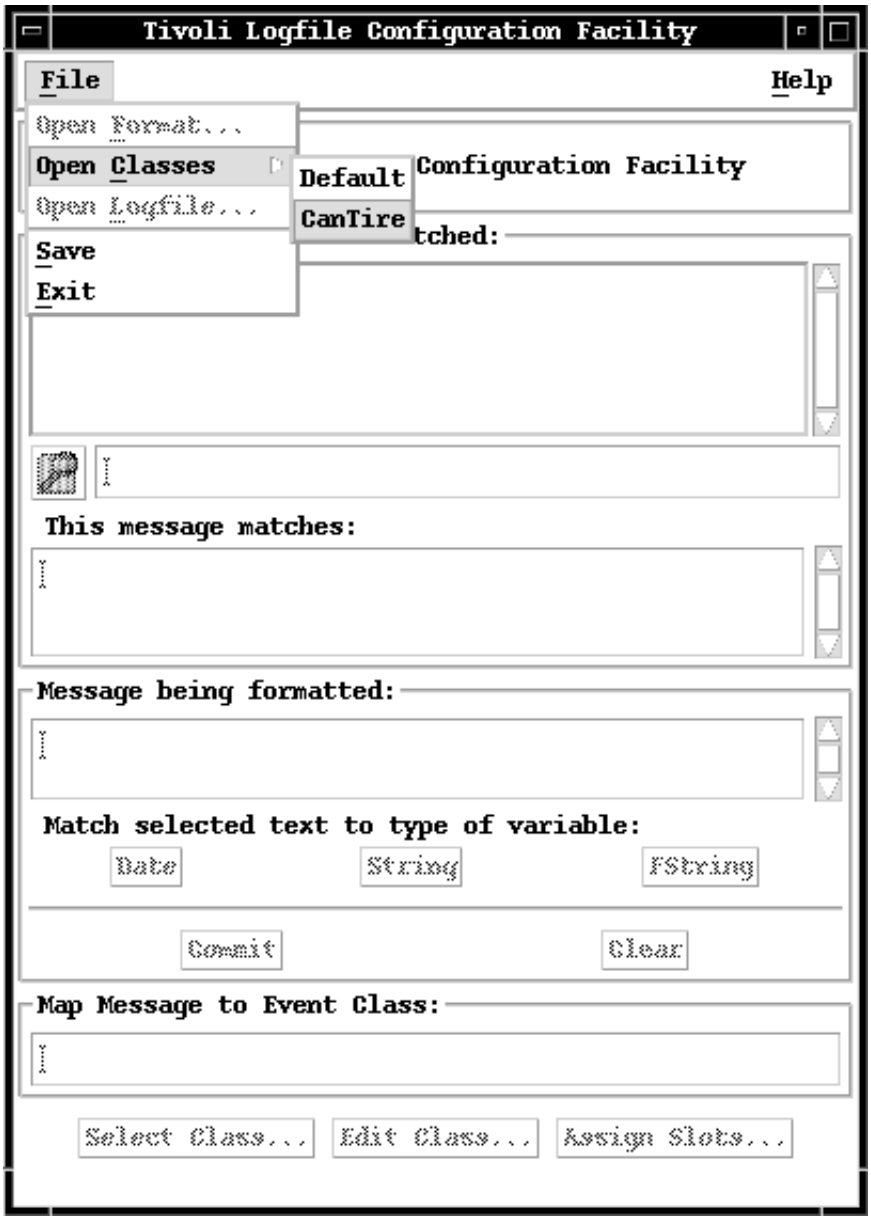


Figure 67. Selecting the Rule Base

We selected our rule base CanTire. The next step is to open the logfile we want to monitor.

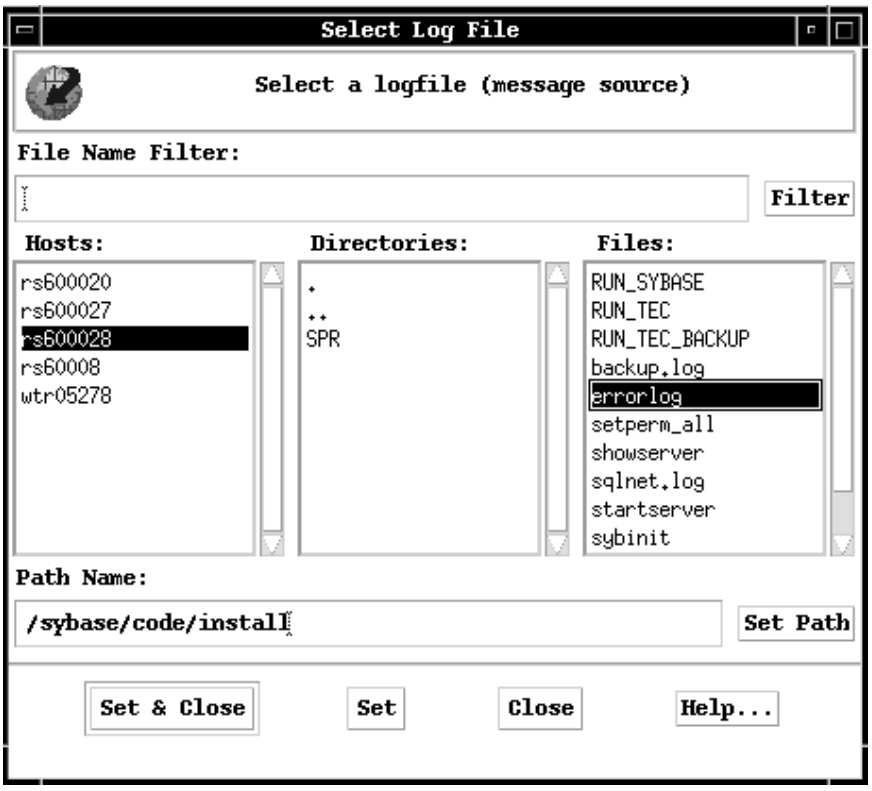


Figure 68. Selecting the Logfile

Open the new logfile by selecting **File->Open Format**. We selected the Sybase error log in the directory /sybase/code/install (see Figure 65 on page 150).

The logfile that we want to monitor is located in the directory /sybase/code/install and is called errorlog. To select this file select **Set & Close**.

The sybase error logfile called /sybase/code/install contains Sybase information. The message we use in this example, which is contained in the error log file is shown below:

```
00:98/02/23 12:12:23.63 kernel WARNING: SIGDANGER Signal received
System low on swap space
```

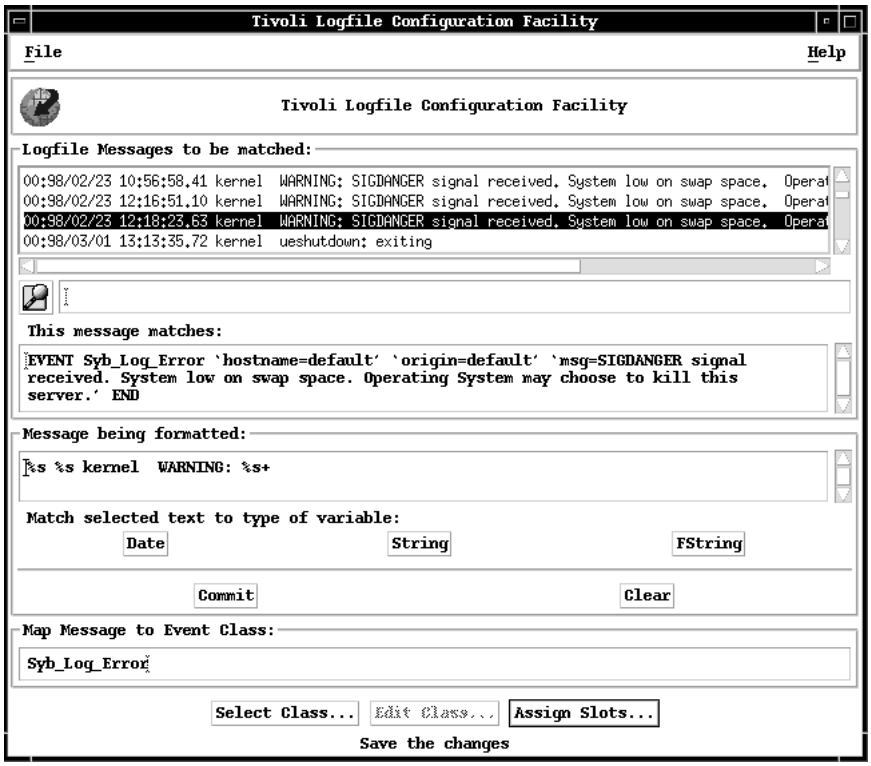


Figure 69. The Main Logfile

We scroll down the messages window and select the error message we are looking for. The contents are displayed in the area This message matches.

Next we need to create our new class definition for this message. We used the logfile_base class as the superclass definition. Do this by clicking on the **Select Class** button (see Figure 70 on page 155).

Figure 70. Selecting the baroc File

The file selected will be the baroc file that is updated with the new class we will define for our error message.

To create a new class first select **Logfile_Base** from the available list. Next select **Class** followed by **New Class** (see Figure 71 on page 156).

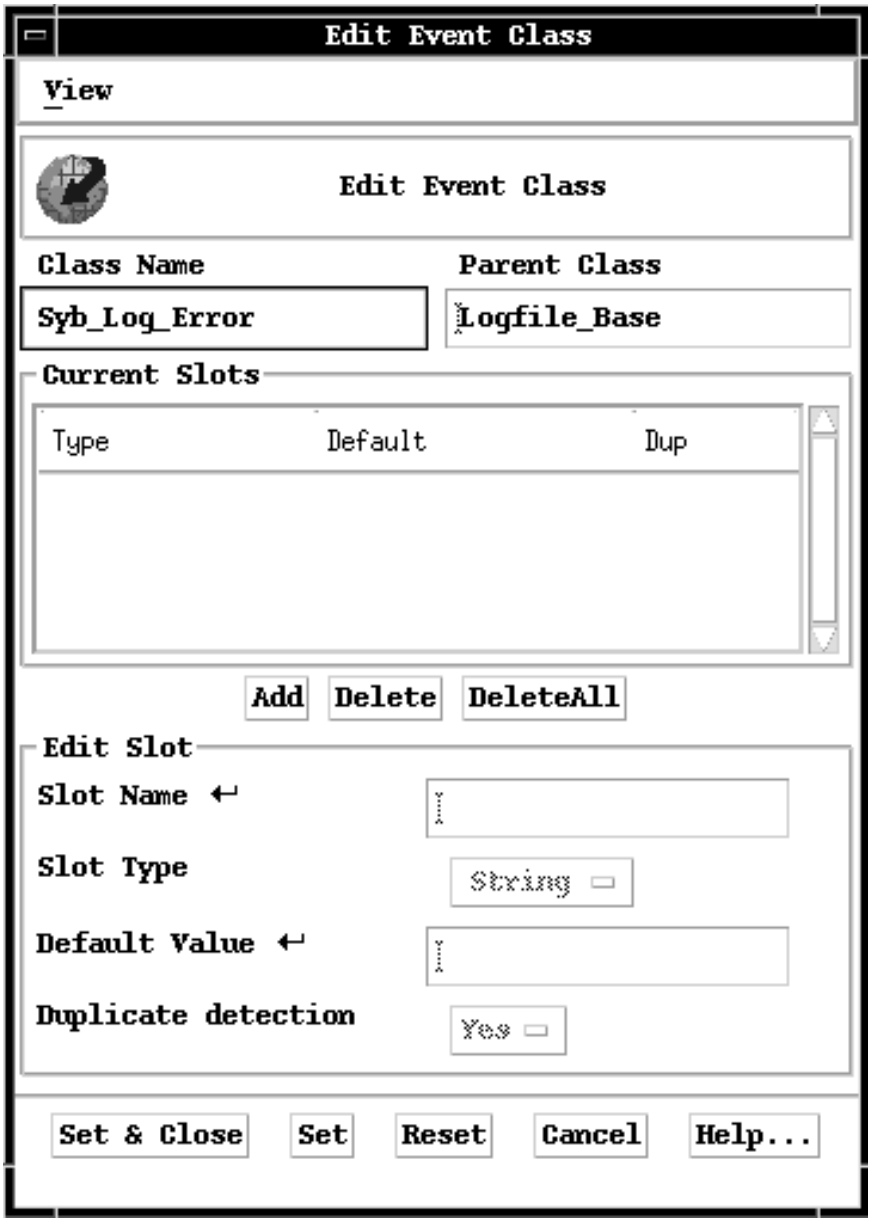


Figure 71. Creating a New Class Definition

The new class name we used was Syb_Log_Error. We entered this as shown in Figure 72 on page 157.

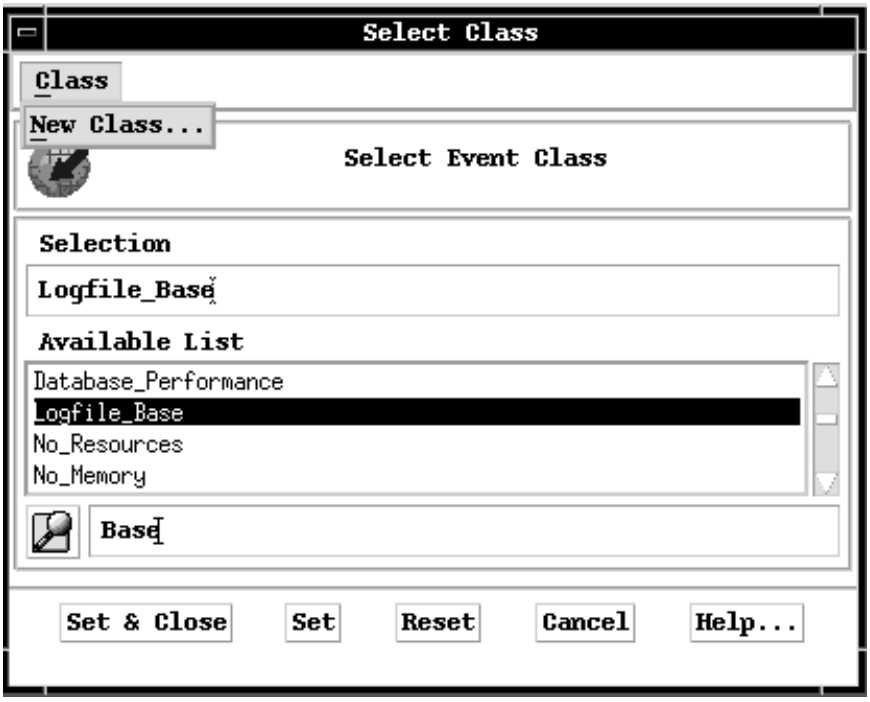


Figure 72. Creating a New Class

Select **Set and Close** to define the class.

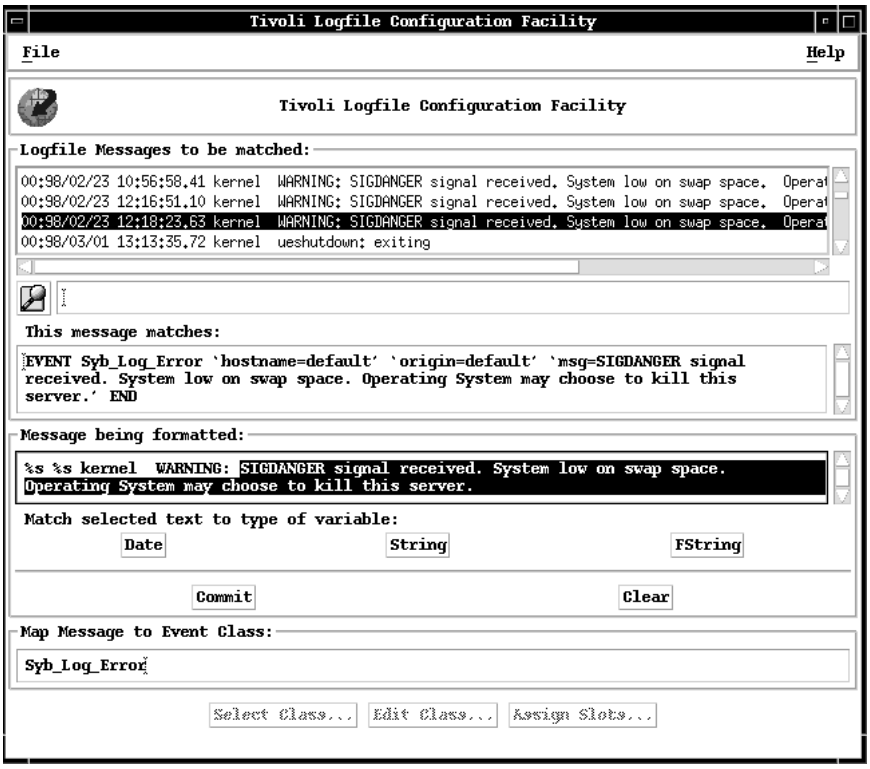


Figure 73. Configuring The Message Format

Now we must create the format for the message. Here we selected the message portions using the right-hand mouse button and then clicked on **String**. This replaces the format field with the percent (%) symbol. This procedure will take a

few attempts to get the format message correct. You must click on **Commit** each time the message is modified to test if the correct strings are set.

Finally from the main screen we select **Edit Slots**. We assigned the slot values as shown in Figure 74 to parse the $3 to the msg slot.

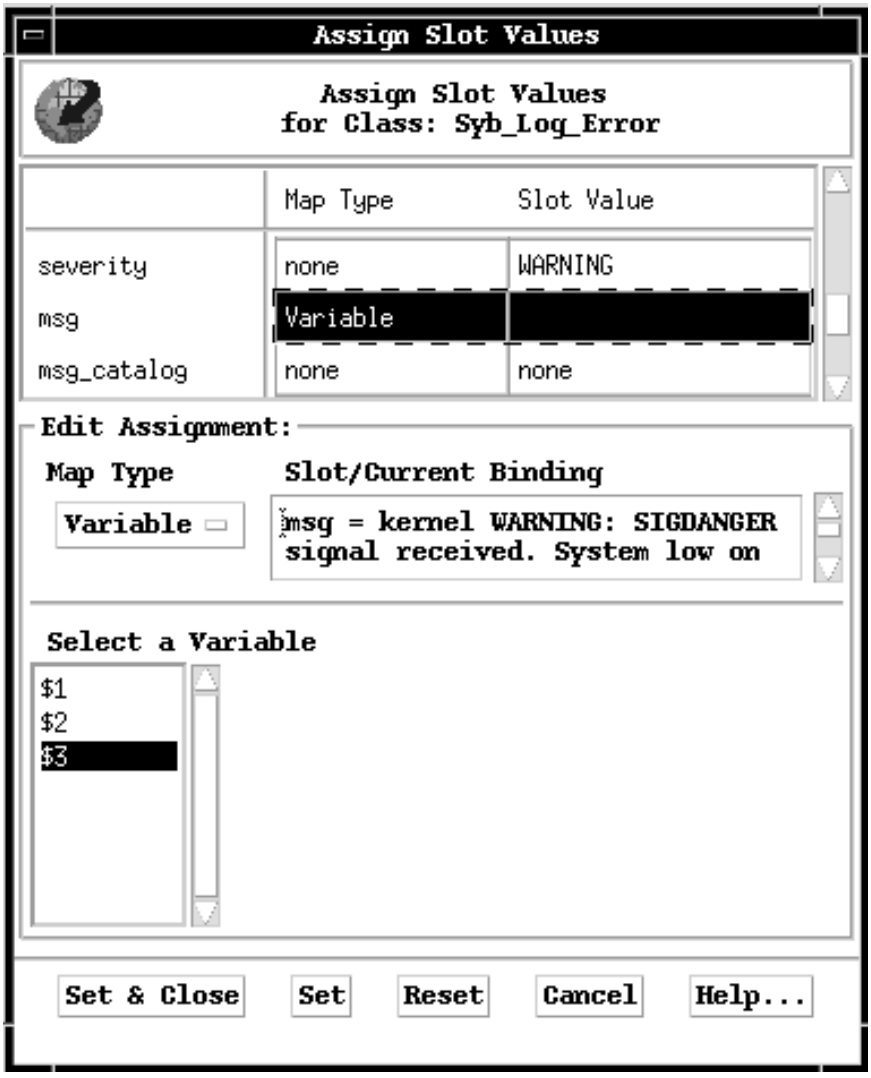


Figure 74. Modifying the Slot Values

We also had to clear the date field. Here we set this to blank (see Figure 75 on page 159).

Figure 75. Removing the Default Date Slot Value

This proved to be a useful tool for creating the correct format for a specific string contained in a new logfile. For additional information refer to the *Enterprise Console User's Guide*.

7.4.1 Logfile Configuration Summary

What this configuration has done is update the logfile adapter files with our new definitions. The tecad_logfile.baroc file is updated with the following lines:

```
TEC_CLASS :
        Syb_Log_Error ISA Logfile_Base
;
END
```

It also appended the following lines to the tecad_logfile.fmt file:

```
FORMAT Syb_Log_Error FOLLOWS Logfile_Base
%s %s kernel WARNING: %s+
END
```

Before this will work we still need to manually add the following new source file to the tecad_logfile.conf configuration file:
`LogSources=/sybase/code/install/errorlog`

The format file has been amended so we need to compile this to create the .cds file. This is done by issuing the command:

```
logfile_gencds tecad_logfile.fmt > tecad_logfile.cds
```

The final two steps are to compile the new tecad_logfile.baroc file and stop and restart the logfile adapter.

Due to the fact that we have modified the tecad_logfile.baroc file we need to remove the old configuration from the rule base and replace it with the new one. The commands are listed below:

```
wdelrbclass tecad_logfile.baroc CanTire
wimprbclass tecad_logfile.baroc CanTire
wcomprules -t CanTire
wloadrb CanTire
wstopesvr
sleep 10
wstartesvr
```

These configuration files can now be files can be distributed by the adapter configuration utility.

Chapter 8. The TME 10 NetView for AIX Event Adapter

This chapter covers Version 5 of TME 10 NetView. This adapter allows the mapping of SNMP traps to specific TEC event classes and provides a mechanism to assign SNMP trap arguments to the TEC event slots.

There are a number of ways to assign these traps:

- Modify the /usr/OV/conf/C/trapd.conf: file directly
- Use the NetView addtrap command
- Use the xnmtrap command

Any of the above techniques may be used to define the behavior of the event adapter. The entries in the trapd.conf file are used by the trap reception daemon, trapd, to determine how each trap is to be filtered or forwarded.

We use two of the methods. First we show how to change the trap definition using the GUI, then we show how to modify the trapd.conf file directly.

Due to the amount of potential traps that are required for forwarding to the TEC console we devised a process to modify the required configuration files. These steps are outlined below.

The examples show how to set up the NetView and TEC server to show events that are generated from the TME 10 NetView application such as the Node Down event. Then we expand this example to show how events from the SNMP sources listed in Chapter 2, “Planning for the TEC Installation” on page 11 can send events. This process is not automatic; there are a number of configuration changes required.

First we discuss the class definitions for the TME 10 NetView adapter.

8.1 Event Class Structure for NetView Events

The class definitions are contained in the file nvserverd.baroc. This file is located on the NetView server in the directory /usr/OV/conf.

The classes defined in this file are for the most part a collection of event classes previously defined in separate files.

The NetView adapter supports classes initially defined for the SNMP event adapter, the OpenView event adapter, and the original NetView/6000 event adapter. Therefore, the user must take care in the selection of baroc files when building rule bases.

When using the nvserverd.baroc file there is no real need to also import the tecad_OV.baroc, tecad_snmp.baroc, or tecad_nv6k.baroc files. Compilation of the rule base may fail if any of these three files are included in the rule base with nvserverd.baroc as duplicate definitions for event classes may occur. This can be easily modified by removing the duplicate slot definition from one of the class definition files.

We build all our required baroc files using the Nvserverd_Event class definition. This served as a template for all the enterprise specific traps. It uses generic names for the SNMP trap arguments, such as nv_var1, as the first SNMP trap argument as defined in the MIB file.

The default nvserverd.baroc file is shown below:

```
TEC_CLASS :
        Nvserverd_Event ISA EVENT
        DEFINES {
                source: default = nvserverd;
                nv_enterprise: STRING;
                nv_generic: INT32;
                nv_specific: INT32;
                nv_var1: STRING;
                nv_var2: STRING;
                nv_var3: STRING;
                nv_var4: STRING;
                nv_var5: STRING;
                nv_var6: STRING;
                nv_var7: STRING;
                nv_var8: STRING;
                nv_var9: STRING;
                nv_var10: STRING;
                nv_var11: STRING;
                nv_var12: STRING;
                nv_var13: STRING;
                nv_var14: STRING;
                nv_var15: STRING;
                };
END
```

We made some modifications to the default baroc file in order to allow specific NetView events to appear in the TEC. We added new class definitions to the nvserverd baroc file as we only wanted to receive four events from the NetView application itself.

After the rule set has selected the traps for forwarding the event customization considerations described below then handle the assignment of event classes to the traps and the mapping of trap variables to event slots.

The two key fields are the the T/EC Event Class field and the T/EC Slot Map. The T/EC Event Class field assigns the selected trap to a T/EC event class. The button is used to designate the slots and slot variables which will comprise the event message.

The NetView event adapter does not use a .fmt or a .cds file for storing the slot mapping information.

The source of the event is nvserverd, the NetView event adapter daemon.

Once all of the class definitions had been corrected we needed to refresh our rule base. As we had modified the tecad_hpov.baroc file in order to correct the enumeration problem, we started by importing the file into the rule base.

8.2 Preparatory Steps

These are the steps required to select and forward SNMP traps as events to the TEC event server, and to create a complete test environment for handling those events at the event server. This description leads up to but does not include the actual TEC rules we wrote for handling these events.

We found that many of these steps were common to the preparation required for handling events from various sources, although this chapter refers only to events coming from the NetView integrated event adapter.

1. We obtained the trapd.conf or Management Information Base (MIB) TRAP-TYPE definitions for each of the SNMP sources of traps we might expect in the customer's environment.
2. The full list of traps was filtered, and only those which required event forwarding by NetView to TEC were used. For example, all log-only traps and informational traps were discarded.

   Where we received trapd.conf definitions for an application, we appended them to the master trapd.conf file. Where we received the MIB definitions containing trap descriptions, these had to be installed to NetView.
3. We created a baroc file for the Canada Tire-specific events from all sources. We conformed to the general philosophy of defining slots common to all subclasses at the superclass level, and overriding these definitions where necessary at the subclass level, for example, to specify certain slot values. When we noticed that some of the trap names and slot names were by nature very large, we devised naming conventions by removing all vowels and truncating the names. This made the files more readable.
4. Based on the event classes and the slot definitions within the baroc files we created, we edited the trapd.conf on the NetView server to reflect the desired mapping of trap fields to event slots.
5. We created and activated a NetView rule set that would filter in only the traps that the customer chose to forward to the TEC event server. We created the rule set via the NetView GUI, and activated the rule set via the SMIT interface to NetView.
   - All variables and slot definitions will be lowercase.
   - All class definition and rule names will be in CamelCase with underscore instead of space (for example, rule Beacon_State).
   - All rules will have descriptions.
   - All actions will have names which indicate the structure in use. These will provide more information when debugging with the trace option.
6. Test scripts were built which would allow us to simulate trap generation to NetView and subsequent forwarding of these events to the TEC for rule-based correlation.
7. With the infrastructure for event generation and forwarding to TEC in place we proceeded to use the flow charts and psuedo-code to start developing rules.

The following examples show what we did for a selection of NetView events and one of the SNMP sources. We used Cabletron.

8.3 The TME 10 NetView Events

This section shows what we configured for the traps that the TME 10 NetView application will generate.

Using the command event -l we can view what NetView traps are generated from the NetView software. Here we are only interested in the default Node Down and Node Up traps.

The only NetView specific events that we want to send to the TEC are:

- Node Down
- Node Up
- Interface Down
- Interface Up

A section of the output is shown below:

```
# event name   number     catg  description
NUP_EV       0058916864    3    Node Up
NDWN_EV      0058916865    3    Node Down
IUP_EV       0058916866    3    Interface Up
IDWN_EV      0058916867    3    Interface Down
```

These are the NetView events that we want to send to the TEC. Next we have to modify the nvserverd.baroc file that is shipped with NetView.

If you are required to send all the events from NetView by default, then you can use the baroc file called tecad_ov.baroc. We chose not to do this as we only wanted four events. We appended the current nvserverd.baroc file by creating new class definitions for the events.

We defined the four events with the following class definitions:

- Node Up event as OV_Node_Up
- Node Down as OV_Node_Down
- Interface Up as OV_IF_Up
- Interface Down as OV_IF_Down

We could have given these classes any value. The additions to our nvserverd.baroc file are shown below:

```
TEC_CLASS:
        OV_Node_Down ISA Nvserverd_Event;
END

TEC_CLASS:
        OV_Node_Up ISA Nvserverd_Event;
END

TEC_CLASS :
        OV_IF_Down ISA Nvserverd_Event;
END

TEC_CLASS :
        OV_IF_Up ISA Nvserverd_Event;
END
```

To add the new baroc file we imported the new definition. Using this baroc file we can now see the NetView events being sent to the TEC console.

By default the NetView traps are assigned class values in the trapd.conf file.

The definition as seen in the trapd.conf file for the node down event is shown below:

```
IBM_NVNDWN_EV {1.3.6.1.4.1.2.6.3} 6 58916865 N 2 0 "Status Events"
NetView Event : $3
EVENT_CLASS OV_Node_Down
SDESC
This event is generated by TME 10 NetView when
it detects a node is down

The data passed with the event are:
    1) ID of application sending the event
    2) Name or IP address
    3) Formatted description of the event
    4) Timestamp of the event and objid of the node object
    5) Database name
EDESC
```

8.4 Configuring the NetView Rule Set

To start the NetView rule editor issue the command nvrsEdit. This command is located on the NetView server in the directory /usr/OV/bin. This will show two screens. The first one shown in Figure 77 on page 166 is the default templates that we can use for the rule development.

Note: We do not cover in any detail the NetView rule development as all the rules are developed using the TEC. This is covered in *TME 10 NetView*.

The rule set screen will also appear as shown in Figure 76 on page 166.

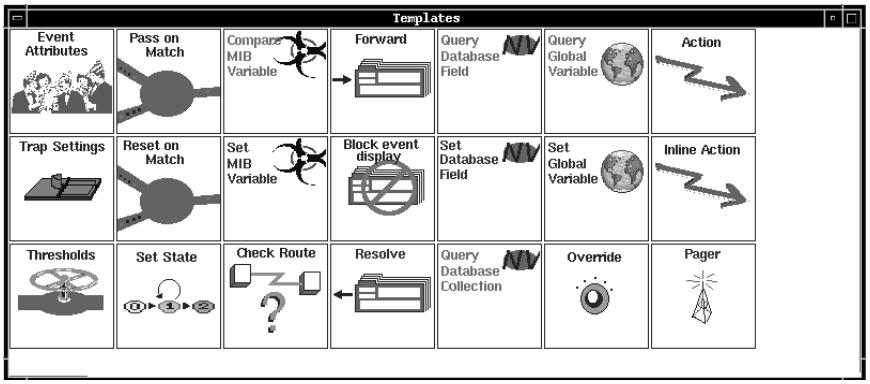


Figure 76. NetView Templates

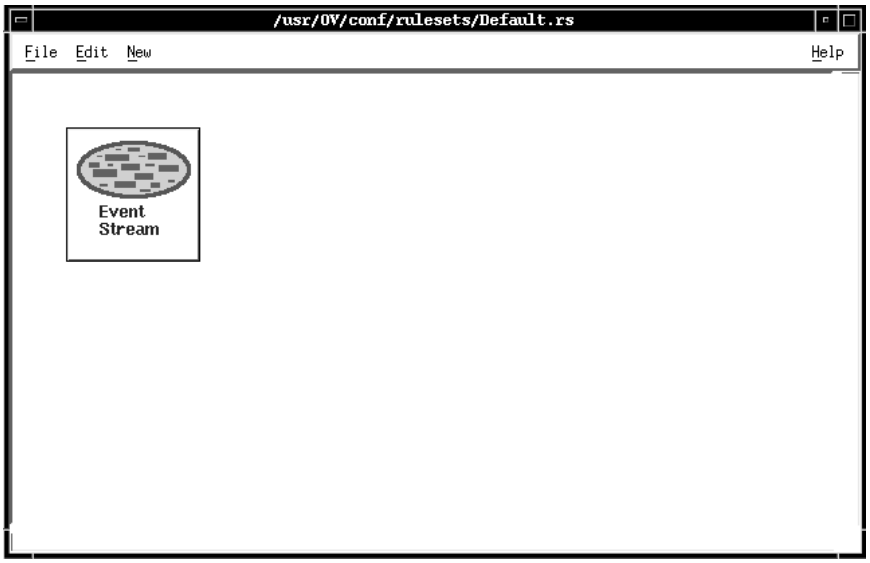


Figure 77. The NetView Rule Set Screen

From here we can create our filter rule. We dragged the template called Trap settings from Figure 77 to screen Figure 76.

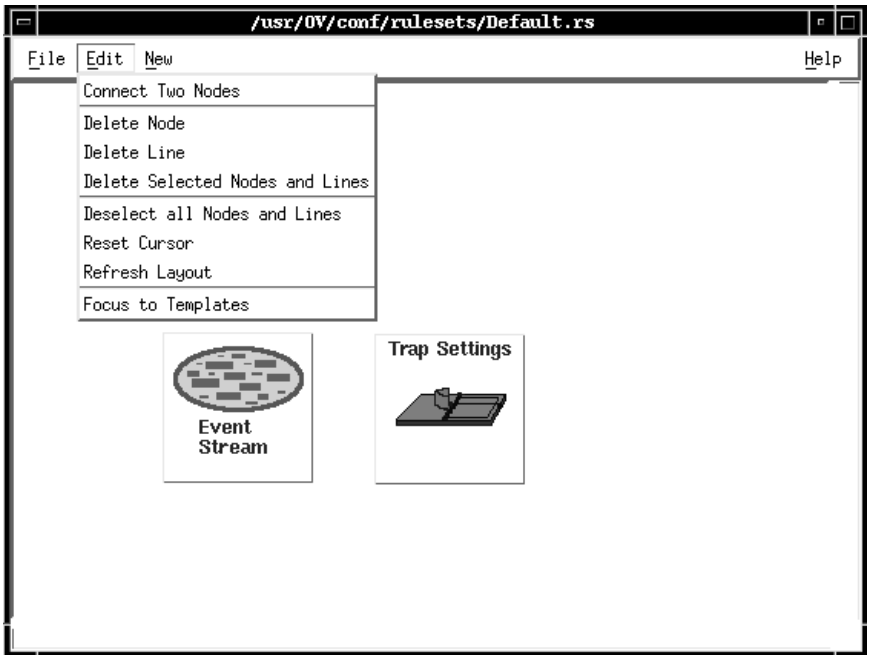


Figure 78. Connecting the Nodes

From the pull-down menu select **Edit->Connect Two Nodes**, then click on the **Event Stream** icon then single click on the **Trap Settings** icon (see Figure 78).

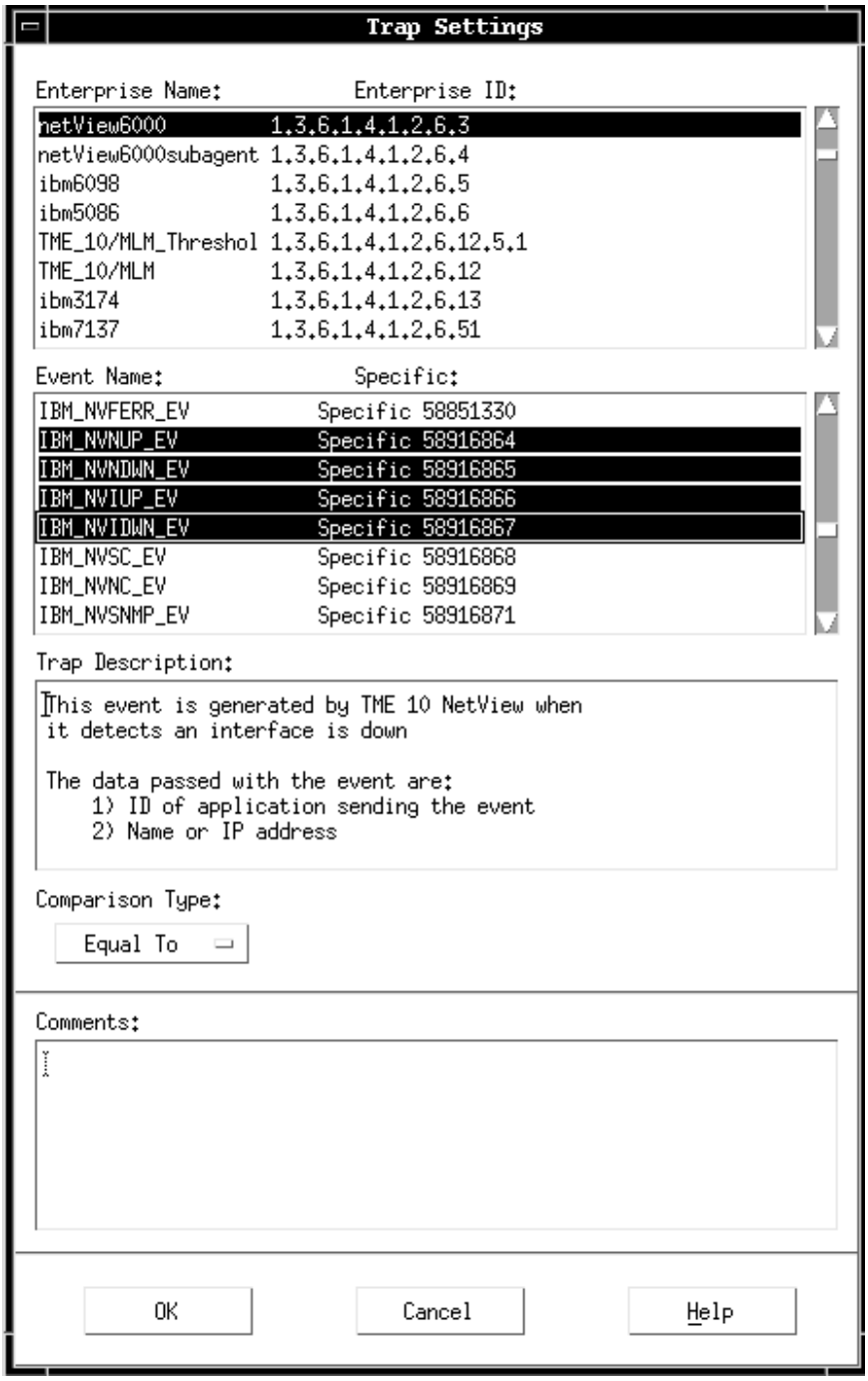


Figure 79. Selecting the Filter Events

Select the enterprise **netView6000** and select the four events we want to send to the TEC. Also set the Comparison Type to be Equal to.

Drag the template icon labeled Forward to the rule set window, then connect the Trap settings to the Forward icons as shown in Figure 80 on page 169.

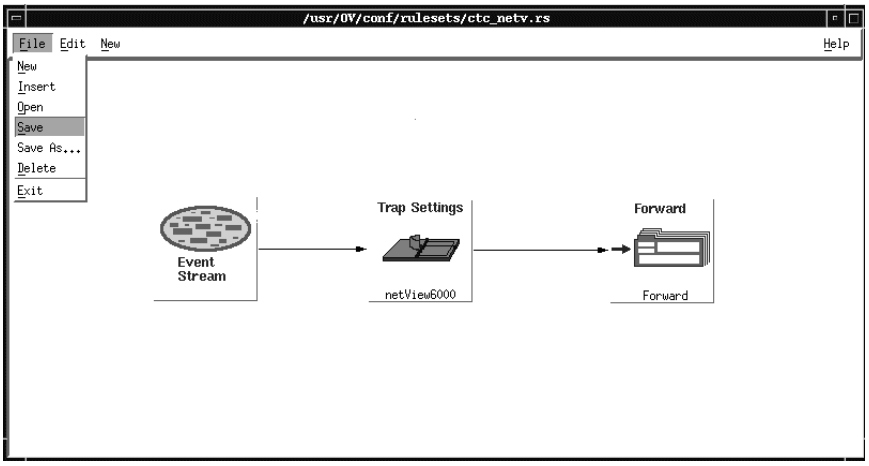


Figure 80. The Complete Rule

Finally we can save the new configuration by selecting **File->Save** from the pull-down menu as shown in Figure 80.

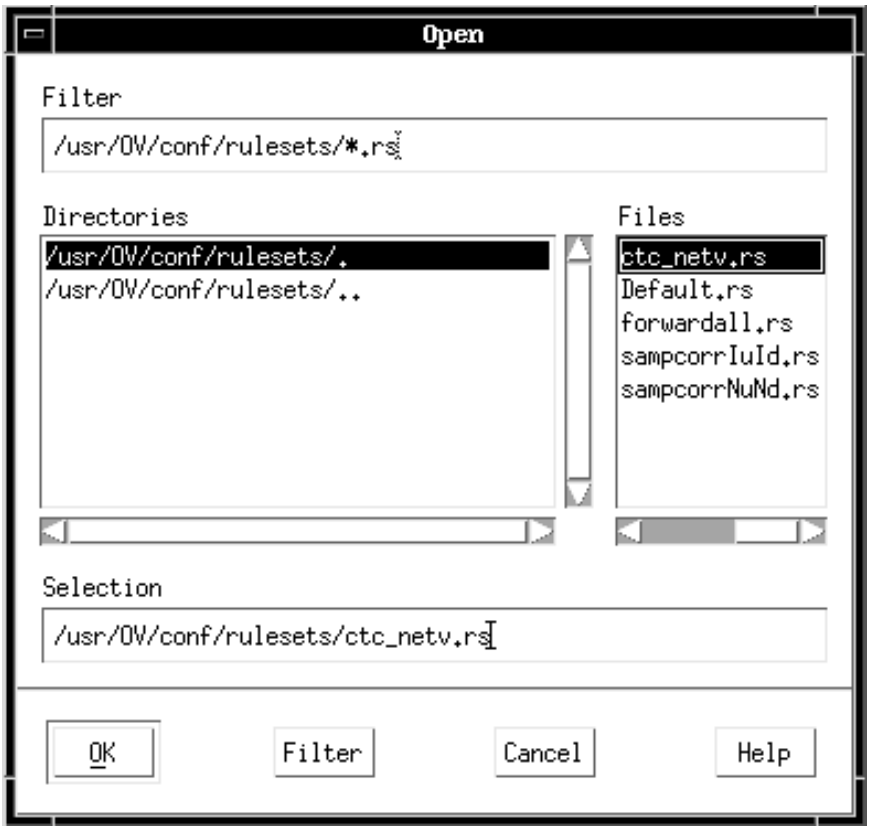


Figure 81. Saving the New Rule

We saved the new rule set as ctc_netv.rs.

8.4.1 Modifying the Trap Description

We also wanted to modify the event description so that when the event is displayed on the TEC console we can see that it was sent from the NetView application. To modify the NetView node down event we selected the node down event as shown in Figure 82.

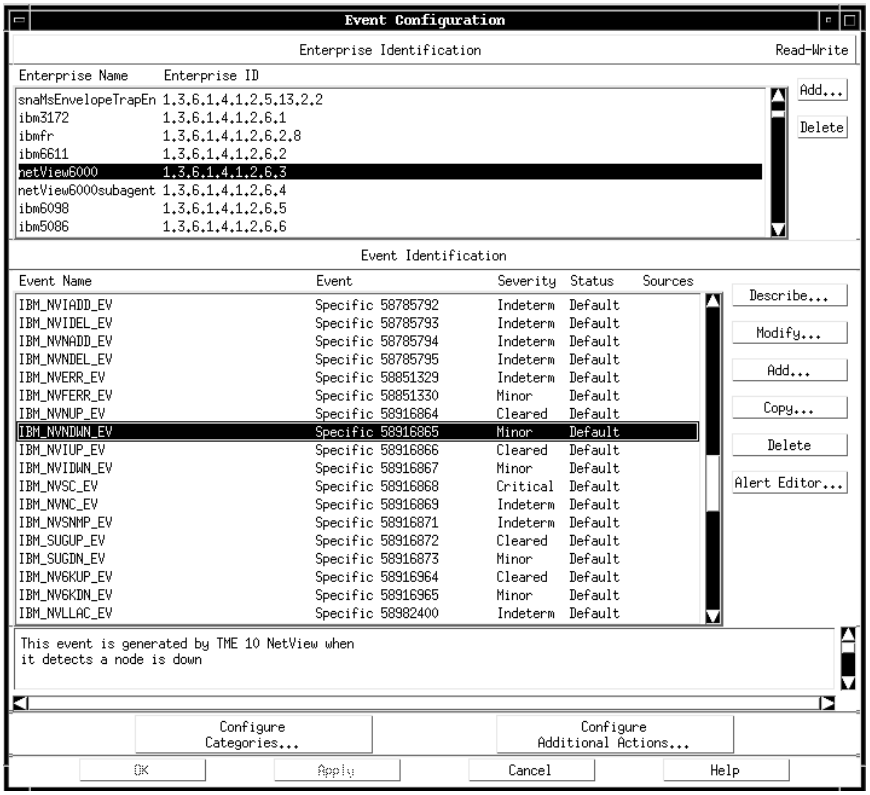


Figure 82. Event Configuration

Scroll down the enterprise list until you can see the netView enterprise and then select it. All the NetView events will appear in the Event Identification window.

First verify that the class definition is set to OV_Node_Down. We also changed the event log message field to include a reference to NetView. This field is parsed to the TEC msg field (see Figure 83 on page 171).

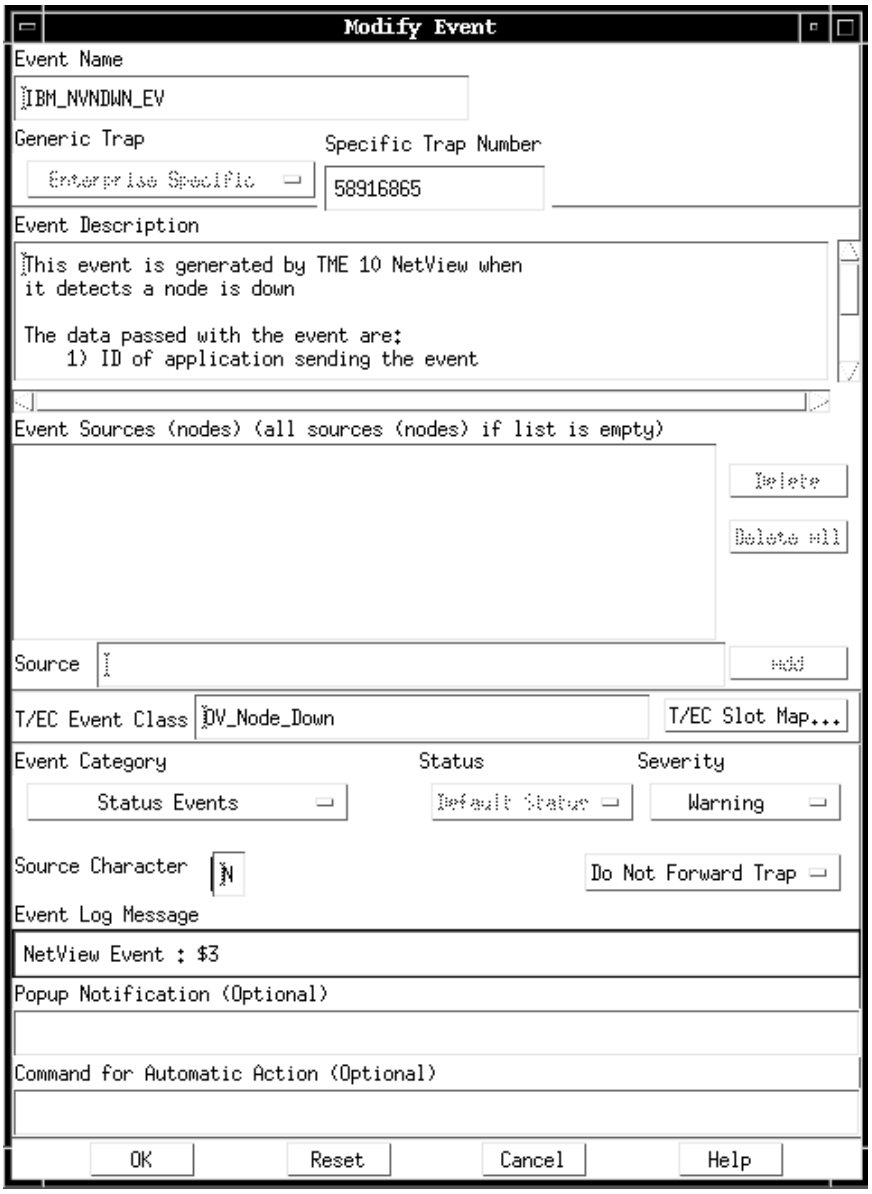


Figure 83. Modifying the Event

Our trap definition has now been modified. Next we have to activate this new rule to send events to the TEC.

8.4.2 Activating the NetView Rule Set

The NetView rule set is activated by modifying the file /usr/OV/conf/tecint.conf shown below.

```
ServerLocation=rs600028
TecRuleName=ctc_netv.rs
```

On a few occasions when modifying the NetView rule set we had to also stop and restart the NetView daemon nvserverd using the commands on rs600027:

```
ovstop nvserverd
ovstart nvserverd
```

Alternatively you can issue the command smitty nv6000. Select **Configure->Configure Event Forwarding to TEC**.

```
            Configure event forwarding for T/EC

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    Entry Fields
  Forward events to Tivoli event server?             yes
    Tivoli event server host name:                   rs600028
    NetView rule name:                               ctc_netv.rs
```

The new filter will now be active.

The entry ServerLocation in the NetView adapter configuration file is used the same way as in other .conf files. However, the TecRuleName entry is unique to NetView. This entry points to a rule set defined within NetView that processes each incoming trap and determines which traps are to be forwarded to the TEC as events.

8.4.3 Testing the Event Forwarding

To test the event we issued the command event -E 58916865 from the NetView server rs600027. Figure 84 shows how the event appears on the NetView events console.

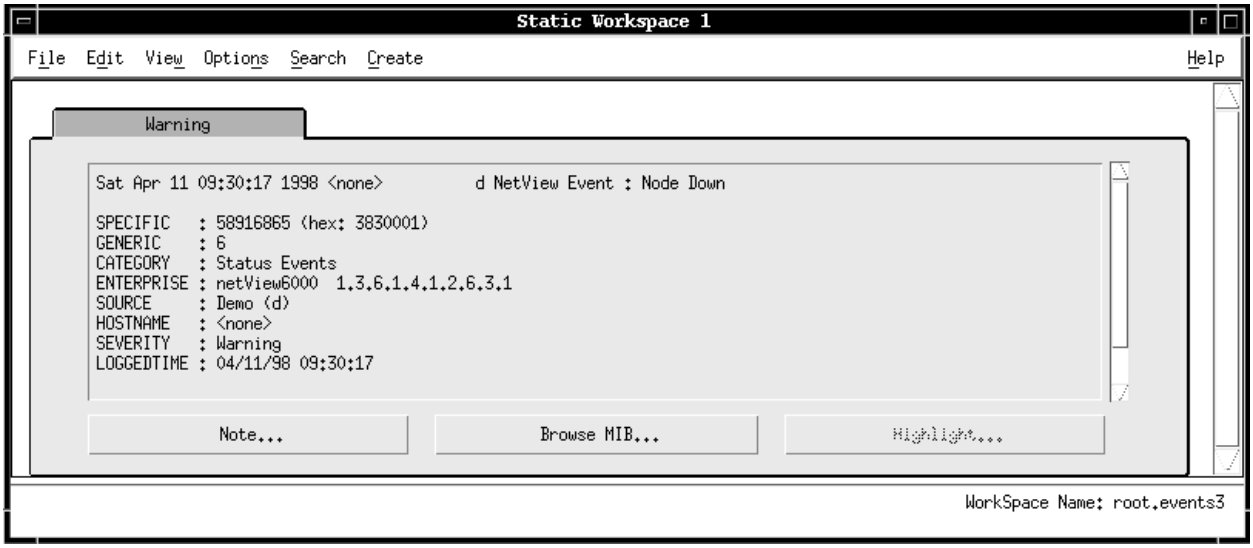


Figure 84. The NetView Event

Figure 85 on page 173 shows what the event looks like from the TEC console.

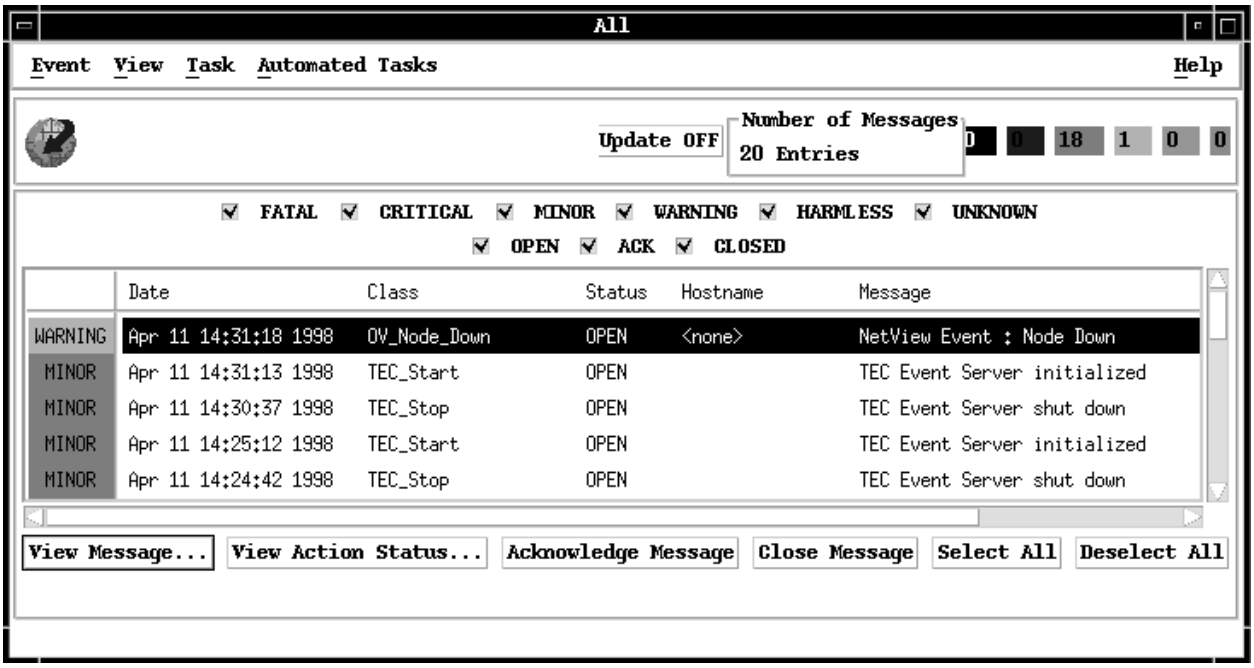


Figure 85. The TEC Event

Now that we have the event forwarding mechanism working we can define our SNMP sources. The example below shows what we did for the Cabletron hubs.

8.5 Example Using Cabletron Hubs

After obtaining the trapd.conf format file for the Cabletron hubs, we added it to the NetView configuration.

We were provided with the trapd.conf file for all the SNMP traps that we required directly from the customer. Should you need to find the trap definition formats for specific devices, they can usually be found on the relevant Web page for the device manufacturer or by loading the management software, such as ManageWise for NetWare.

We started creating a new baroc file called ctccable.baroc for all the Cabletron events. This is shown in the following figure.

```
TEC_CLASS :
        Ctccable_Event ISA EVENT
        DEFINES {
                sub_source: default = "Cabletron";
                severity: default = WARNING;
                last_occur: STRING;
                tt: STRING;
                category: INT32;
                ctcvar1: INT32;
                ctcvar2: INT32;
                ctcvar3: INT32;
                };
END

TEC_CLASS :
        Trouble_Ticket ISA EVENT
        DEFINES {
                nv_enterprise: INT32;
                };
END

# 550
TEC_CLASS :
        Beacon_State ISA Ctccable_Event
        DEFINES {
                ifindex: INT32, dup_detect = yes;
                prtgrpid: INT32, dup_detect = yes;
                mgtstnme: STRING, dup_detect = yes;
                mgtstnadd: INT32, dup_detect = yes;
                mgtstnuna: INT32, dup_detect = yes;
                mgtstnbrd: INT32, dup_detect = yes;
                mgtstnprt: INT32, dup_detect = yes;
                stsrlstbcont: INT32, dup_detect = yes;
                };
END

# 551
TEC_CLASS :
        Beacon_State_Cleared ISA Ctccable_Event
        DEFINES {
                stsrlstbcon: INT32, dup_detect = yes;
                ifindex: INT32, dup_detect = yes;
                };
END
```

We imported our new baroc file into our TEC rule base called CanTire using the following commands:

```
wimprbclass ctccable.baroc CanTire
wcomprules -t CanTire
wloadrb CanTire
wstopesvr
sleep 10
wstartesvr
```

Next we modified the trap definitions using the NetView GUI. The example we show uses the specific trap 550, which is the Beacon State trap.

To customize the trap issue the command xnmptrap from the NetView server, in our case machine rs600027.

Scroll down the window and select the Cabletron enterprise. This will only appear if the Cabletron events have been created for TME 10 NetView for AIX.

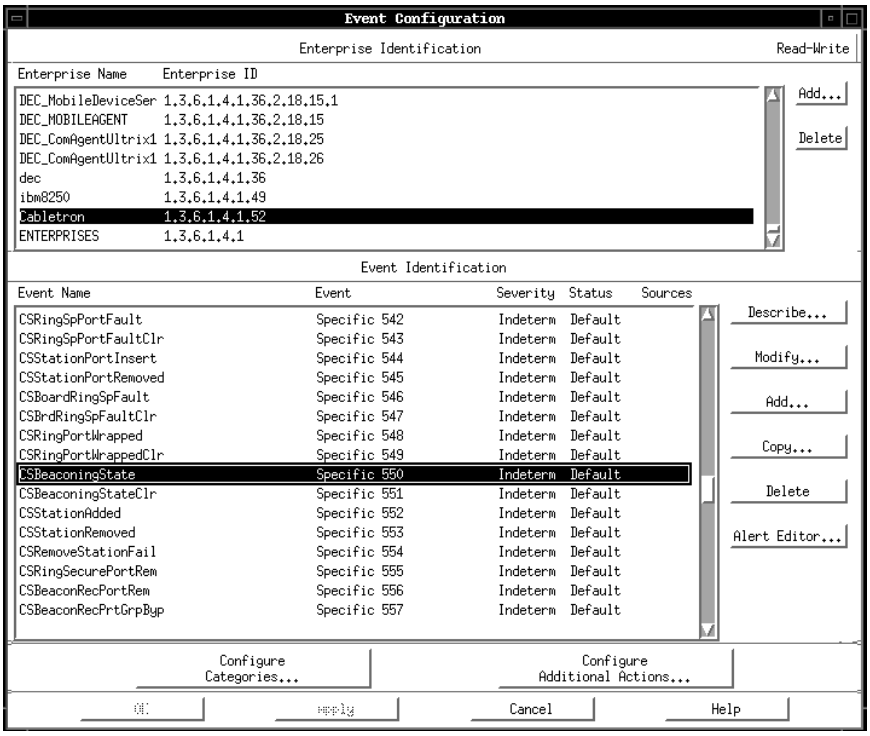


Figure 86. Modifying the Trap Definition

Click on **Modify** to change the definition.

Figure 87 on page 176 shows what we modified in the trap customization. We added the TEC Event Class Beacon_State, which is the TEC class definition for the event server, and also amended the event log message. This message is what is parsed to the TEC msg field within the event message.

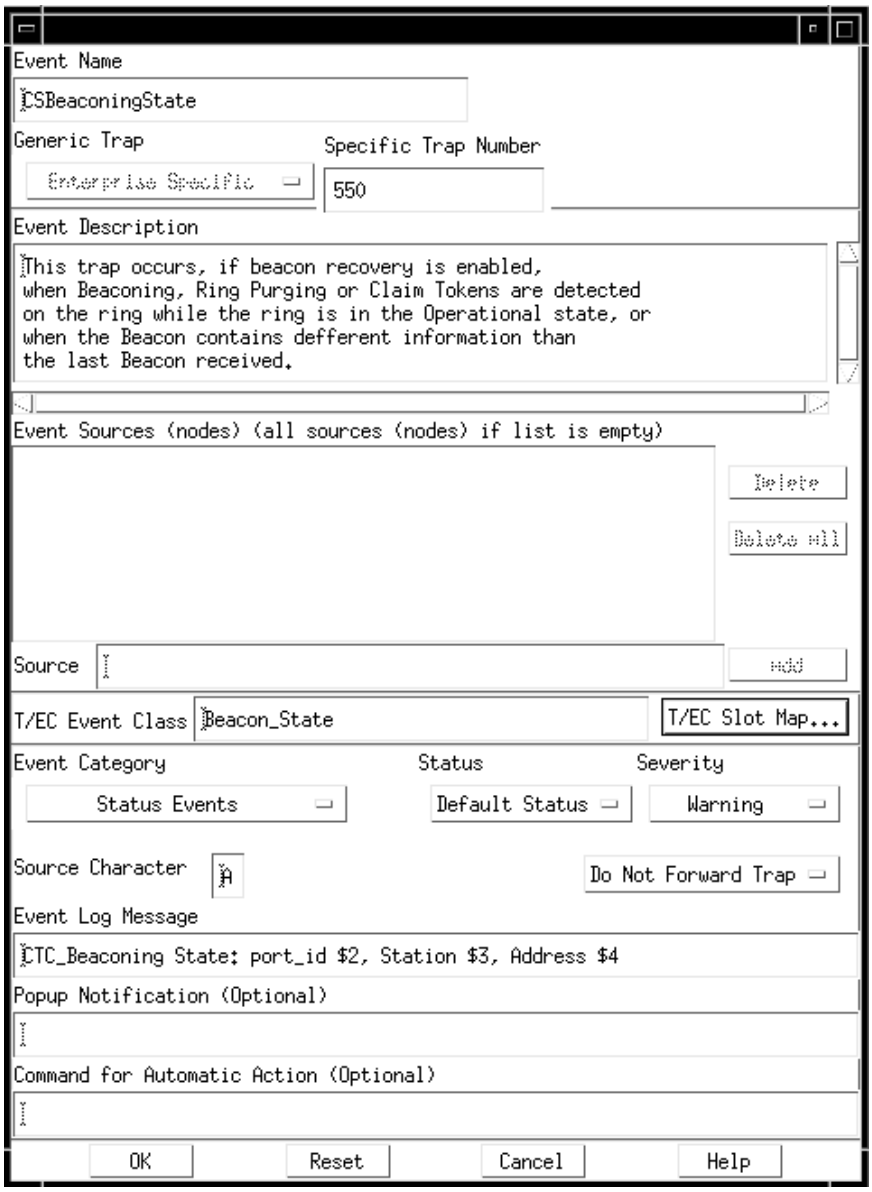


Figure 87. Changing the Event Message

By selecting the **T/EC Slot Map...** button you will see the Edit Trap Slot Map definition screen (see Figure 88 on page 177).

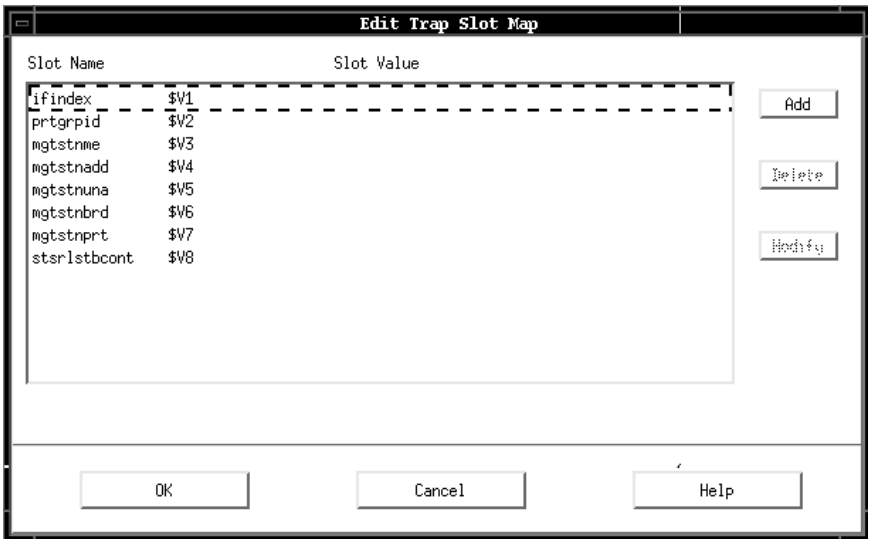


Figure 88. Assigning the Slot Values

Adding the values is illustrated in Figure 89. These values set the slot definitions for the event server. In this case we are parsing the 4th argument of the SNMP trap to the slot defined as ifindex.

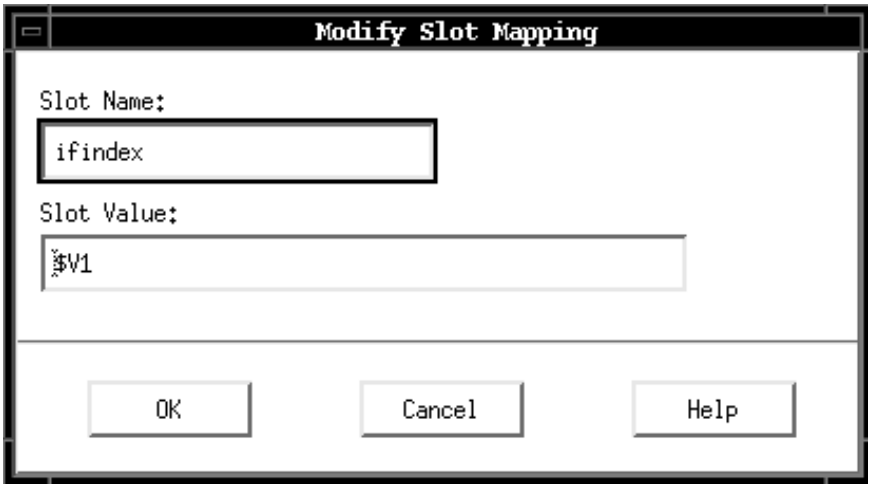


Figure 89. Modify the Event Slot Mapping

Thus the event slots have been defined with contents from the trap.

We now have to create the NetView rule that will allow NetView to forward this event to the TEC server.

8.5.1 Adding This Definition to Our NetView Rule Base

We added the Cabletron events to our filtering as shown in Figure 90 on page 178.

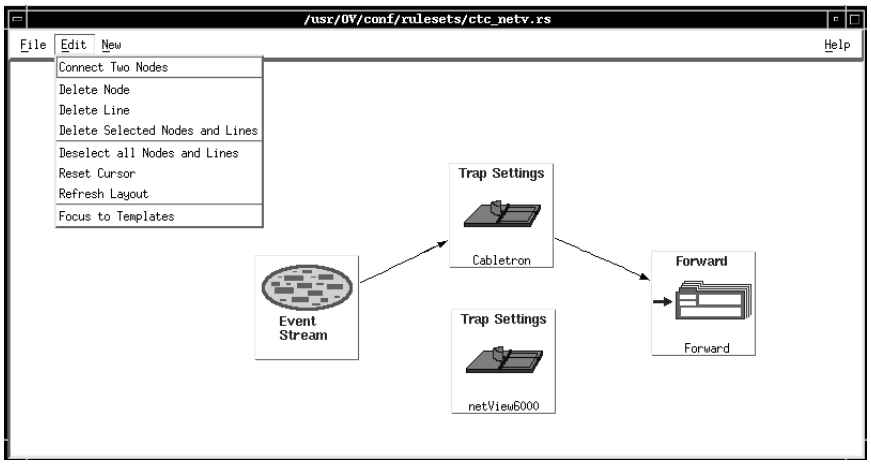


Figure 90. Adding Additional Event Filters

By double-clicking on the Cabletron icon we selected the events shown in Figure 91 on page 179.

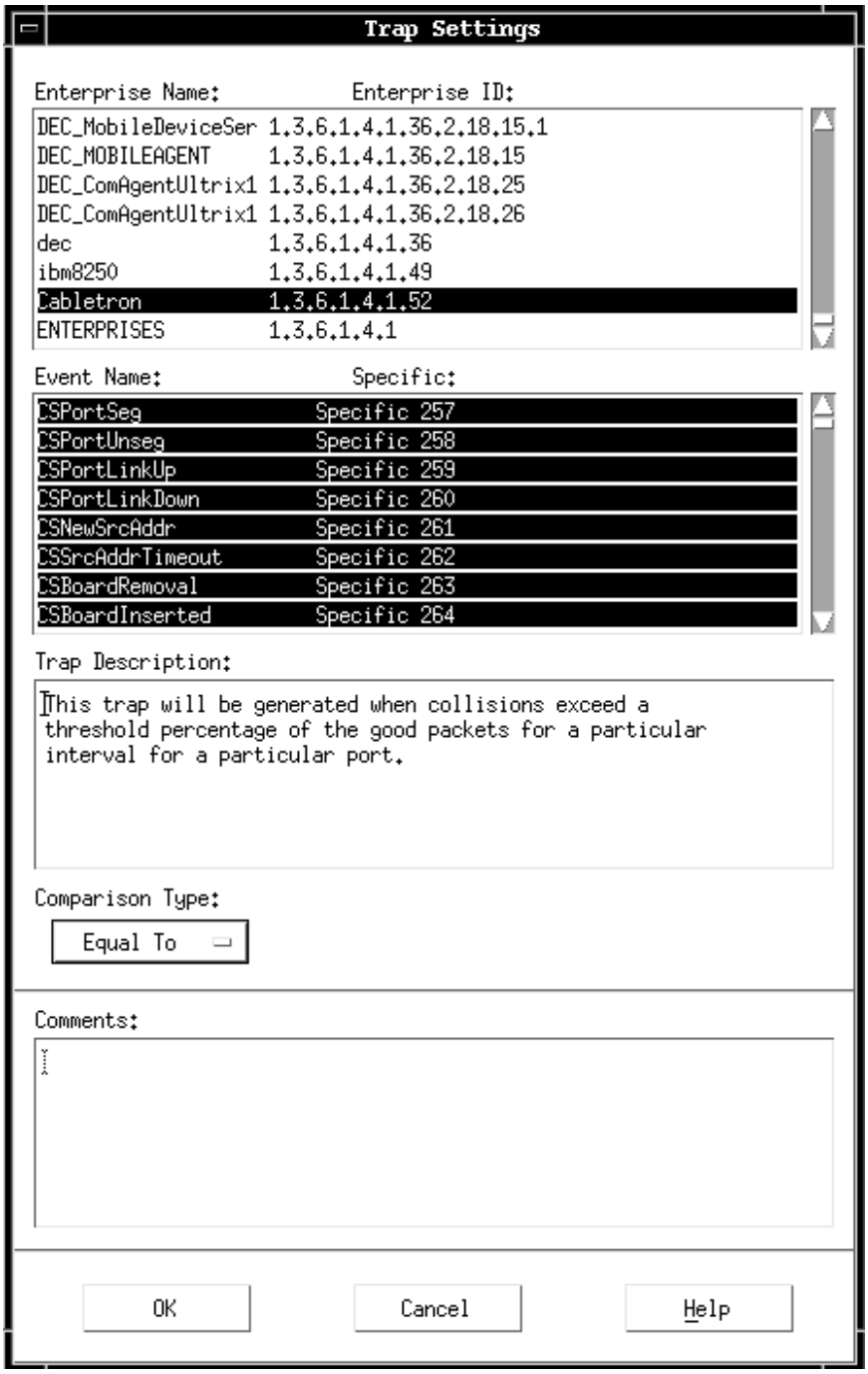


Figure 91. Selecting The Cabletron Events

We located the Cabletron enterprise number in the enterprise list. This reveals the Cabletron traps. Using the right-hand mouse button select the traps that you want to forward to the TEC. Selecting multiple traps is done using the Ctrl key.

Select **OK** to complete the event settings.

Once we had connected the nodes we saved the ruleset by selecting **File->Save**.

8.5.2 Testing the Filtering for the Cabletron Hubs

To test the traps we developed some test scripts that sent SNMP traps. The results are shown in the NetView events window (see Figure 92).

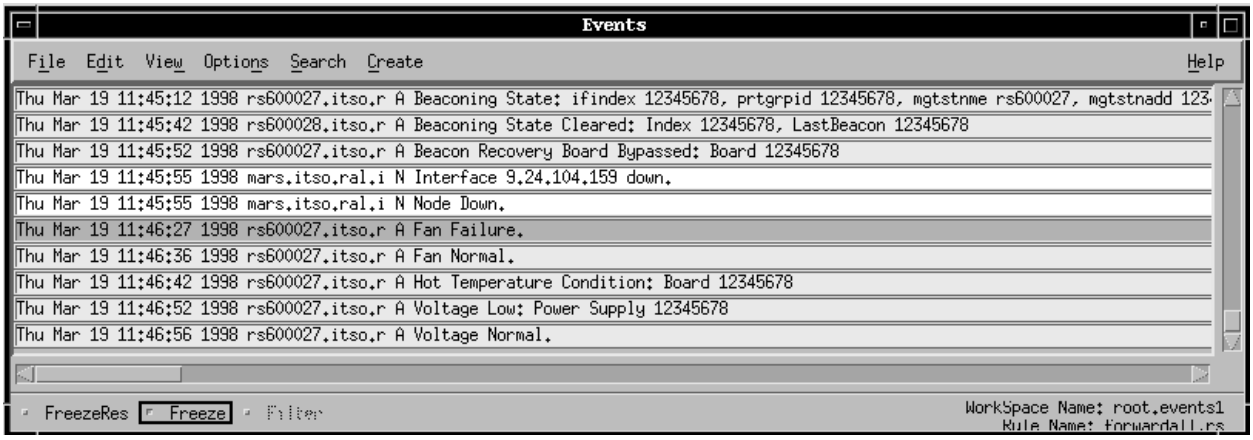


Figure 92. The Events Window

Now that these traps are being sent to the TEC, we are now able to develop the TEC rules that will use these traps. This is covered in Chapter 11, “Rule Development” on page 235.

8.5.3 Modifying the trapd.conf File

We found that directly editing the trapd.conf file was a lot quicker, although it is recommended that this file is not directly edited. If you do manually edit the file, be sure to take a backup of the file first.

```
CSBeaconingState {1.3.6.1.4.1.52} 6 550 A 2 0 "Status Events"
CTC_Beaconing State: port_id $2, Station $3, Address $4
EVENT_CLASS Beacon_State
BEGIN_SLOT_MAPPING
 ifindex $V1
 prtgrpid $V2
 mgtstnme $V3
 mgtstnadd $V4
 mgtstnuna $V5
 mgtstnbrd $V6
 mgtstnprt $V7
 stsrlstbcont $V8
END_SLOT_MAPPING
SDESC
This trap occurs, if beacon recovery is enabled,
when Beaconing, Ring Purging or Claim Tokens are detected
on the ring while the ring is in the Operational state, or
when the Beacon contains different information than
the last Beacon received.
EDESC
```

The second example shows the beacon state clear event definition:

```
CSBeaconingStateClr {1.3.6.1.4.1.52} 6 551 A 1 0 "Status Events"
Beaconing State Cleared: Index $1, LastBeacon $2
EVENT_CLASS Beacon_State_Cleared
BEGIN_SLOT_MAPPING
 stsrlstbcon $V1
 ifindex $V2
END_SLOT_MAPPING
SDESC
This trap occurs when a ring returns to the
Operational state.
EDESC
```

The trap for Beacon_State_Cleared shows that there is only two arguments for this trap.

8.6 Assigning the Event Groups

The events are sent to the TEC using the class definitions. Our requirement is for the root operator and the network operation to receive all these traps. The definition for setting this up is defined below.

To create the new event groups using the GUI, using the right-hand mouse button select **Event Groups**, shown in Figure 93.

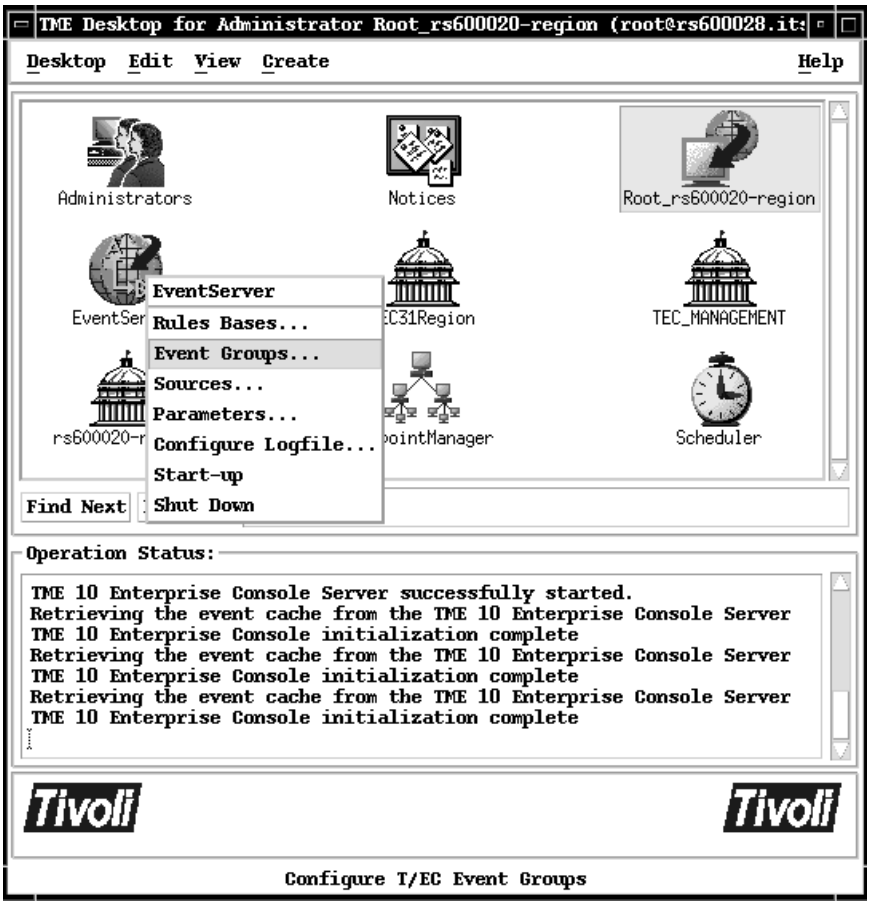


Figure 93. Modifying the Event Groups

We already have two event groups defined from the installation performed in Chapter 4, “Software Installation” on page 49.

Select the **Network** event group shown in Figure 94.

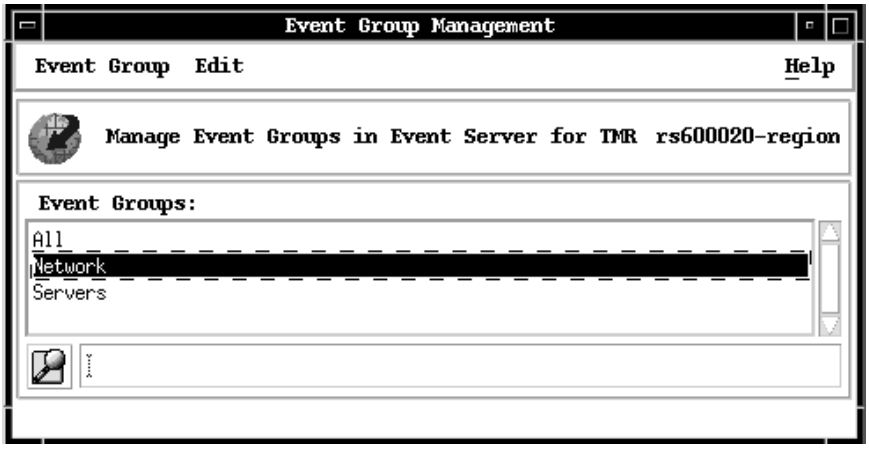


Figure 94. Selecting the Event Group

The event group assignments should be as shown in Figure 95.

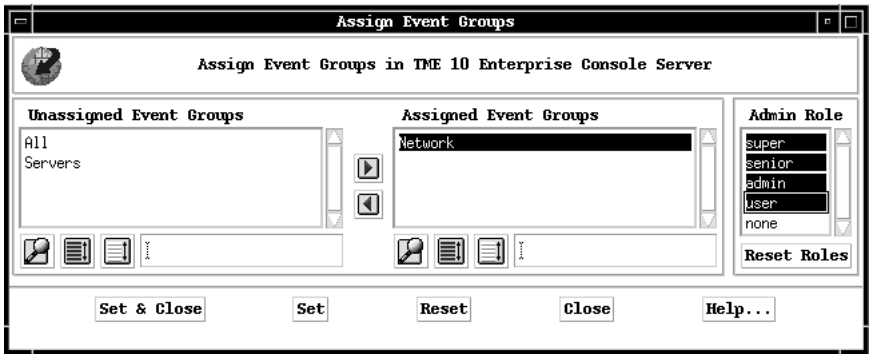


Figure 95. Assigning the Event Group

The filter definitions are shown in Figure 96 on page 183.

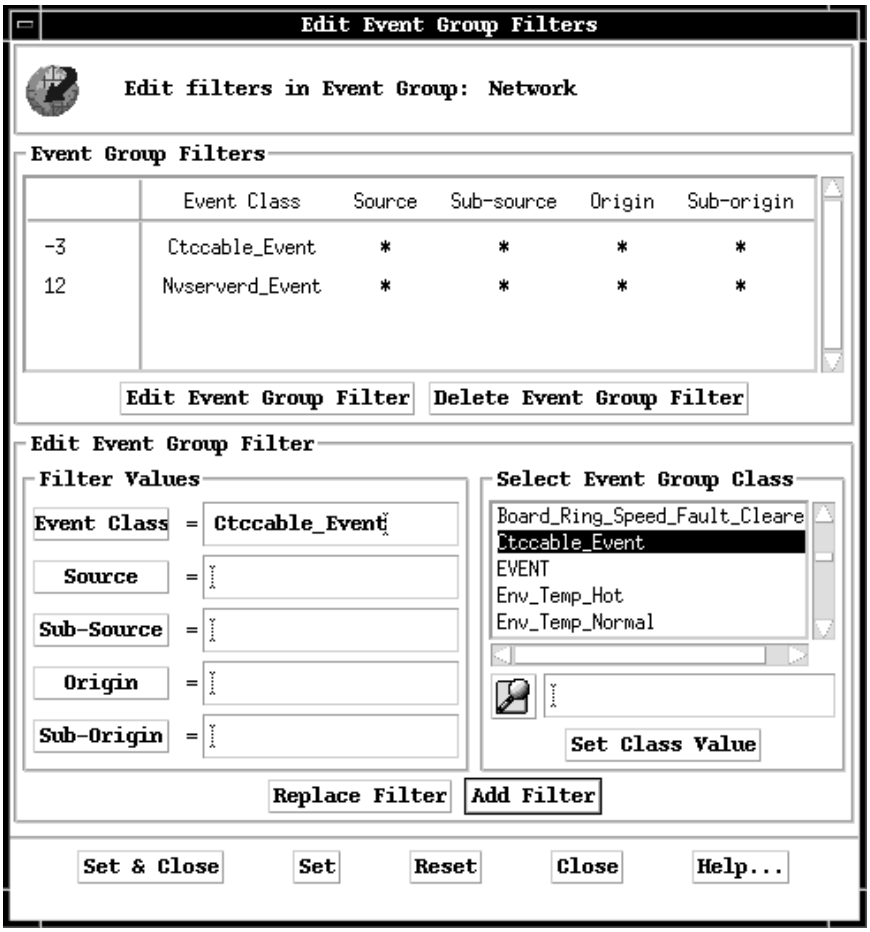


Figure 96. Editing the Event Group Filter

Once the filters are set, the operator can view the specific events by double-clicking on the **Network** event group shown in Figure 97 on page 184.

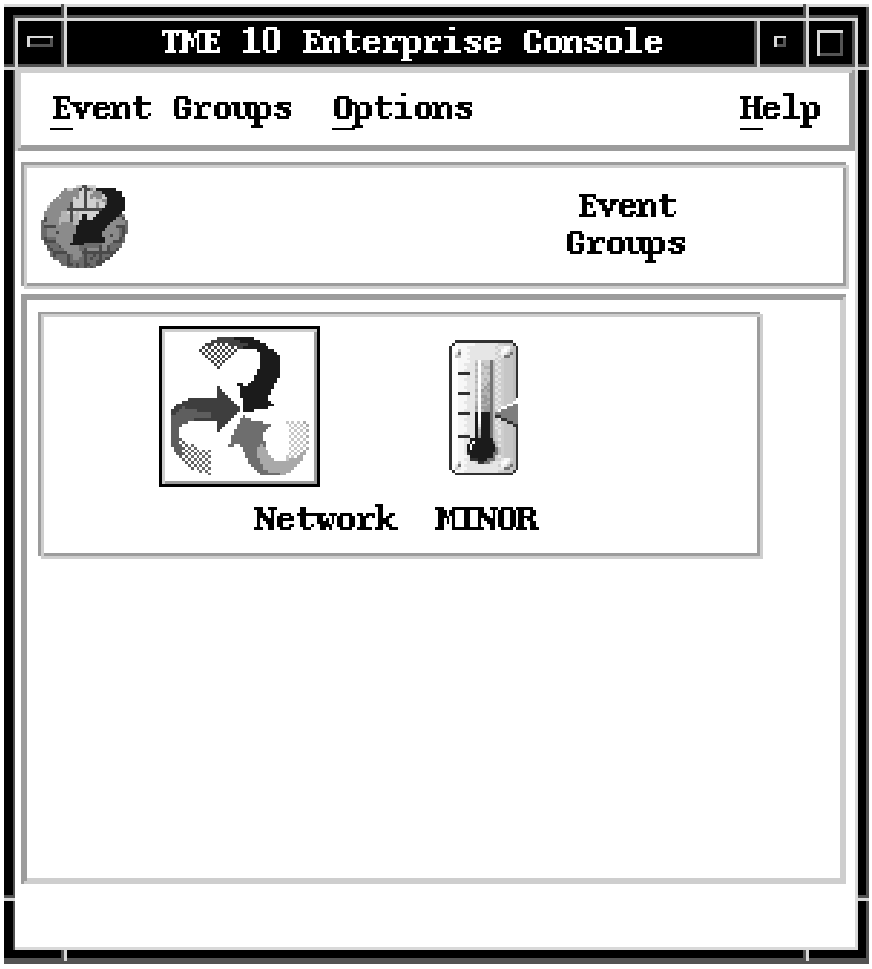


Figure 97. Starting the Console Using the New Event Group

The event window is shown in Figure 98.

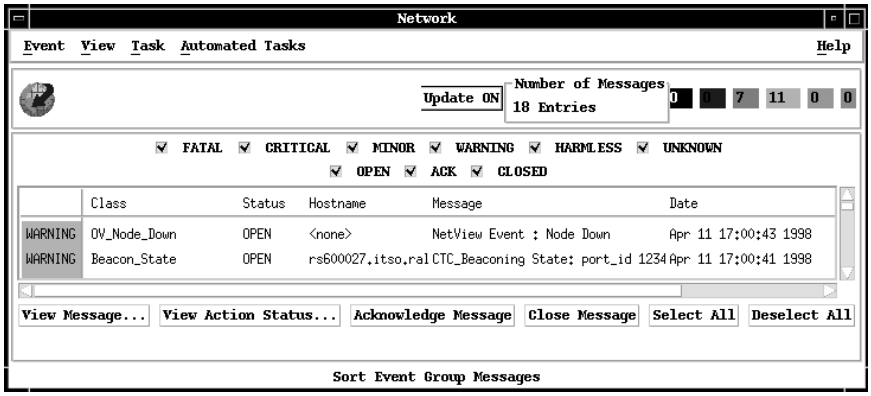


Figure 98. The TEC Events Window

This new event group will only display events with the event class of Nvserverd_Event and Ctcable_Event. As we create the mode of the SNMP sources we can modify the group to include the new class definitions.

Chapter 9. Using Tasks with the TEC

This chapter discusses using the TEC tasks that allow the TEC console operators to perform specific functions either as part of a defined process or in direct response to receiving a specific event. The tasks will be used for automation, driven by a particular event.

The examples we show in this chapter are mainly for providing tasks to manage the RISC System/6000 servers that are used for the management solution. Our requirements are documented below.

9.1 Requirements

We noticed a number of potential problems that may occur while the TEC environment is running. Due to the number of y servers installed in our environment the tasks will assist in managing the TEC infrastructure.

The following outlines the steps we take to implement the tasks:

- Discuss the default TEC task library and what functions are delivered
- Create a new task library that will eventually contain all the new tasks
- Create our own tasks to show how to build new tasks into the existing environment
- Create an automated task to provide automated functions that will be executed when we receive a specific trap
- Create a job that will periodically archive the TEC database

We first concentrated on the pre-installed tasks delivered with the TEC.

9.2 TEC Tasks

By default the TEC is installed with a number of pre-defined tasks. These tasks can be viewed from the Tivoli desktop.

If you double-click on the **TEC31** icon shown in Figure 99 on page 186, to locate the TEC task library.

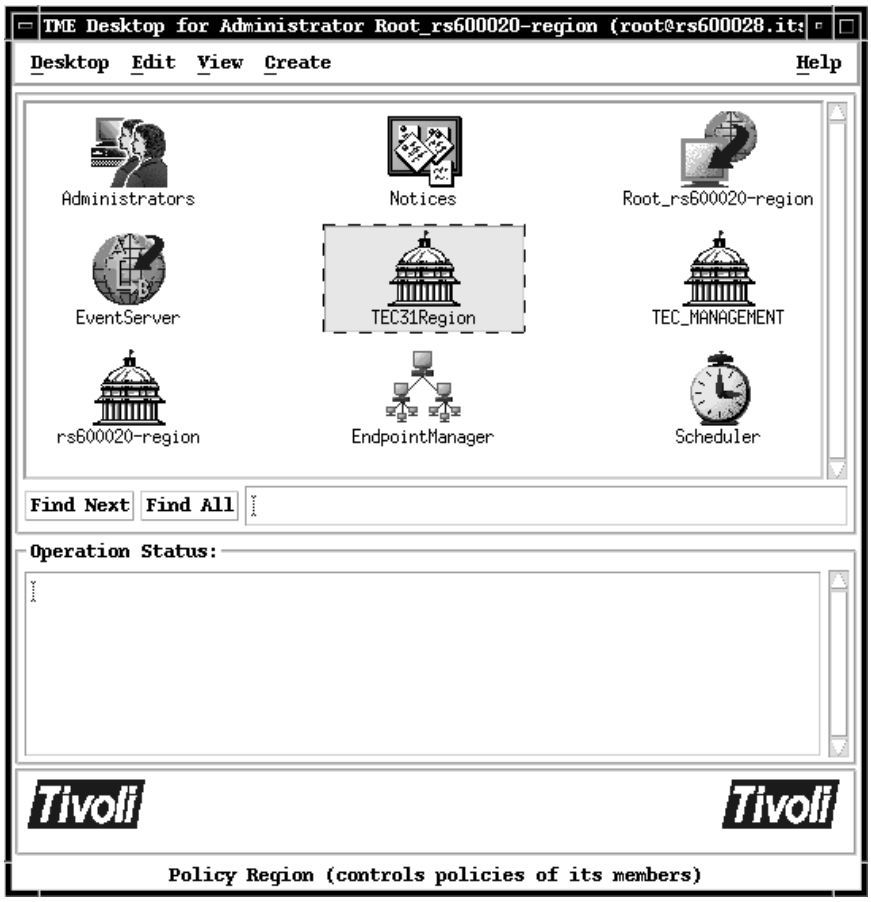


Figure 99. The Tivoli Main Screen

Next click on the **T/EC Tasks** icon shown in Figure 100.

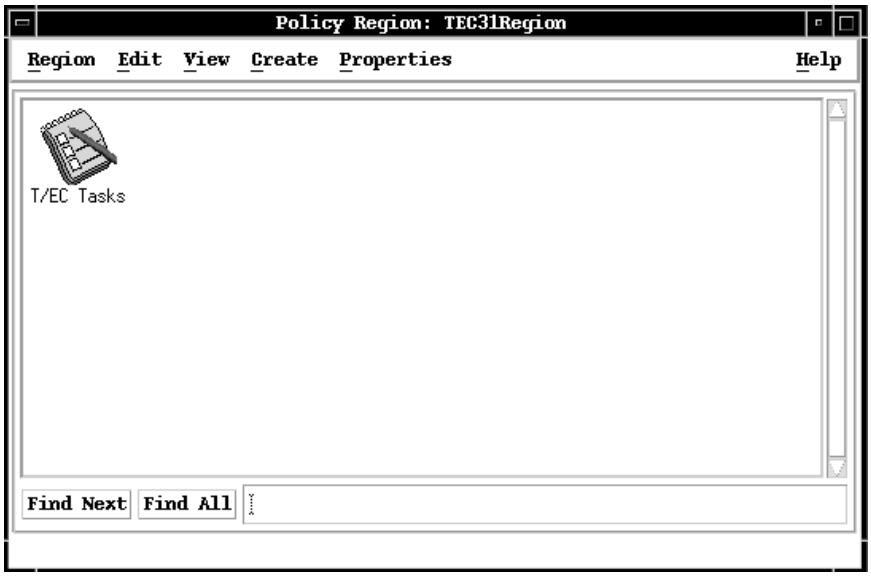


Figure 100. The T/EC Task Region

This lists the tasks shown in Figure 101 on page 187.

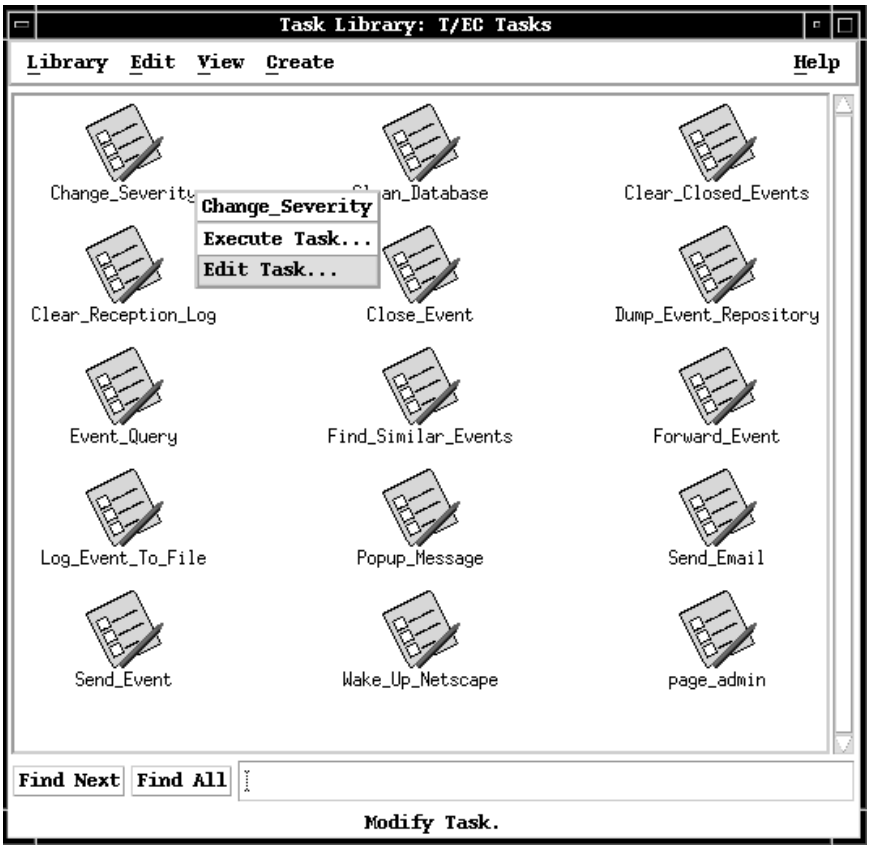


Figure 101. The T/EC Tasks

You will see a task library with the name T/EC Tasks. If you have a single TEC installation in your environment, you will see one occurrence of T/EC Tasks and it will point to the task library on your TMR server. If you connect TMRs, each with its own TEC, you will see the task libraries of all the available TMRs.

To view the tasks from the command line and to list all of the tasks in your system issue the command:

```
wlstlib "T/EC Tasks"
```

Sample output from the task list command follows:

```
(task) T/EC Tasks:
(task) Send_Email
(task) Log_Event_To_File
(task) Popup_Message
(task) Wake_Up_Netscape
(task) Forward_Event
(task) Send_Event
(task) Clear_Reception_Log
(task) Clear_Closed_Events
(task) Change_Severity
(task) Dump_Event_Repository
(task) Event_Query
(task) page_admin
(task) Find_Similar_Events
(task) Clean_Database
(task) Close_Event
```

You can also use the -a argument to show all tasks that have been installed on your system.

The TEC offers many useful predefined tasks that you can execute from your console, and automate, so that they are executed when certain conditions are present.

On UNIX systems, the functions are shell scripts that execute in a controlled environment under the control of the TMR. To view these tasks first we have to find what the TMR region value is. To do this on the TMR server issue the command:

```
odadmin | grep Region
```

For our environment the region value is:

```
Region = 1194067385
```

To view the scripts change the directory as follows: `cd /usr/local/Tivoli/bin/generic_unix/TAS/TASK_LIBRARY/bin/1194067385`

The scripts that are associated with TEC begin with the string T_EC_Tasks.

These tasks perform the following:

- **e-mail event info** - This script uses the UNIX sendmail facility to send an e-mail notification to an administrator.
- **Log event to file** - This script logs an event you have marked in a logfile of your choice.
- **Popup message on desktop** - This script uses the wsendresp command to send your predefined text in a window to a given administrator.
- **Jump Netscape to URL** - This script sets your DISPLAY and PATH variables to the values required to display Netscape where your event console is, and then opens Netscape to the URL that you selected. We found that Netscape also requires the environment variable LANG=C, which was not the default setting in our environment. If you have this problem, you might want to edit this script and add the line export LANG=C.
- **Forward event to TEC server** - This script uses the wpostemsg command to forward the event you mark to the server of your choice.
- **Send Event to TEC server** - This script uses the wpostemsg command to send an event of your design to an event server.
- **Clear reception log** - This script uses wtdbclear to delete events in the event reception log that are older than 30 seconds.
- **Clear closed events** - This script simply issues the command wtdbclear -e -s CLOSED -t seconds, which says delete events from the event repository that have the status CLOSED and are older than the number of seconds given. This is a good task to automate and run at regular intervals, so that your database doesn't run out of space.
- **Change severity of event** - This script uses the wsetemsg command to change the severity of an event.

- **Dump event repository** - This script uses the wtdumper command to dump the number of events that you specify from the event repository.
- **Query event repository** - This script uses your input to form the parameters of the wtdumper command. You can query some of the fields such as event class, status, and severity with this task.
- **Send alphanumeric page** - This script pages someone at a telephone/beeper number that you customize. It uses the fbeep command.
- **Find matching event** - This script uses wtdumper to find events in the event repository that match criteria you define, for example, all events from a certain host and a given event class.
- **Clean out TEC database** - This script removes all events in the reception log that are older than the aging parameter that you configured for your event server. It also removes all events in the event repository that have the status CLOSED and are older than the aging parameter you specified. This task functions only in a single TEC environment.
- **Close event** - This script closes an event. You must first click on an event in the Event Group Message List window, then choose **Task**, and then **Execute on Selected Event**. The script issues the command wsetemsg -t CLOSED for the current event console and the event that you marked.

The respective names for the scripts can be found by issuing the command wgettask. The example below shows how to resolve the name for the Change_Severity tasks:

```
wgettask Change_Severity "T/EC Tasks"
```

The output is shown below:

```
Task Name               Change_Severity
User Name
Group Name
Task ACL             super:senior:admin:user
Supported Platforms
default
<install-dir>/generic_unix/TAS/TASK_LIBRARY/bin/1194067385/T_EC_Tasks_C_zonctxba

Task Comments
        Task Name            : T/EC Tasks/Change_Severity
        Task Created         : Mon Mar 30 16:50:02 1998
        Task Created By      : root@rs600020.itso.ral.ibm.com
        Task Files
            default          rs600020         /tmp/taskbNuxvVj
        Distribution Mode    : ALI
        Task Comments        :

        ------------------------------------------------------------
```

We can now view the script that provides the task function to change the event severity.

```
#!/bin/sh
echo Changing severity of event $ev_key to $1 ...
wsetemsg -r $1 @"$CONSOLE_NAME" $ev_key
rc=$?
echo done
exit $rc
```

The script is passed arguments from the TEC in order to execute the wsetemsg function. Later in this chapter we show how to create a new task that shows what additional values can be accessed while in the shell script.

The next example shows how to execute the default tasks.

9.2.1 Running the Default Tasks

From the Event Group Message List window, as shown in Figure 102, you can execute individual tasks, effective for individual events or for an entire system, as well as view and define automated tasks that are triggered by events. In order to set that up you will need to use the pull-down menus shown in the following figure. To execute the tasks select from the events window the pull-down menu **Tasks** (see Figure 102).

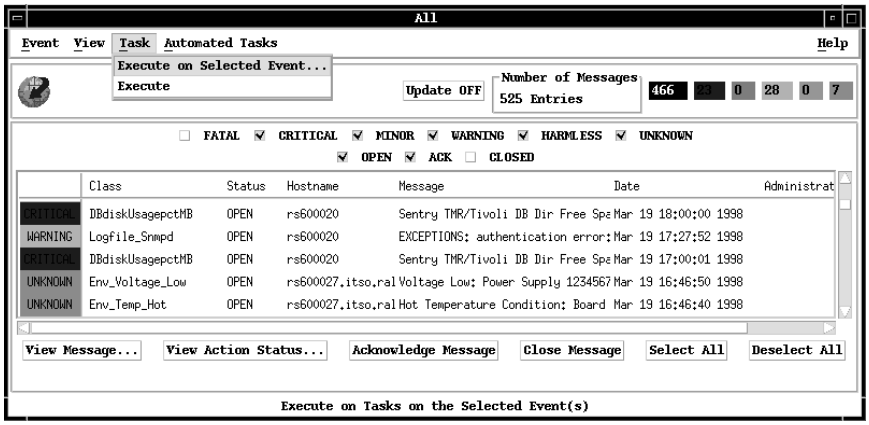


Figure 102. Event Group Message List

Here you can execute a task for a specific event or run a generic task not associated with any specific event.

The tasks are displayed by double-clicking on the task library **T/EC Tasks**. The tasks will appear in the Task window (see Figure 103 on page 191).

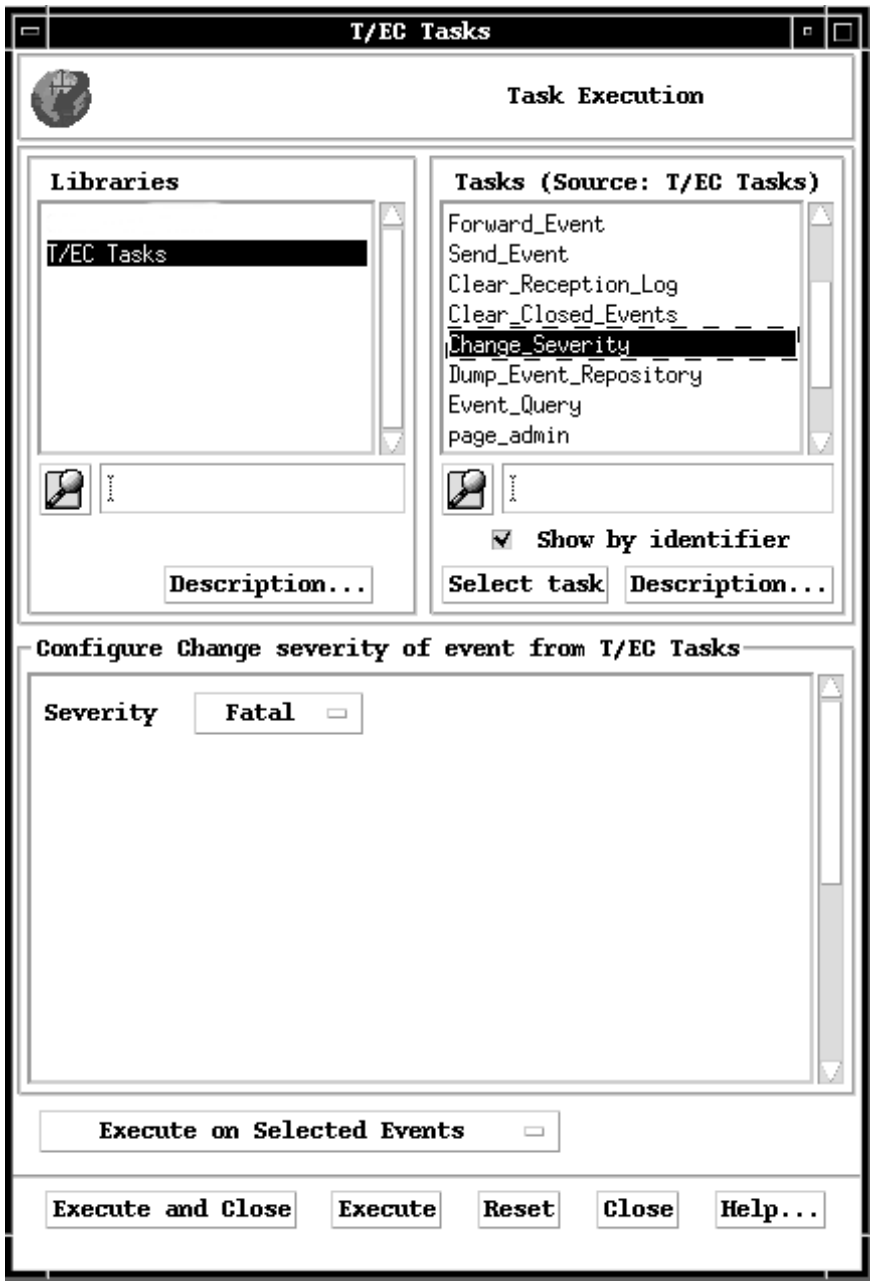


Figure 103. Executing a TEC Task

The default TEC tasks will ask for arguments, for instance the severity level for the event (see Figure 103).

9.2.2 Changing Options for the Default Tasks

The example below shows how to change the permissions for the change severity task to only allow operators with the senior priority to be able to execute. From the TEC task library select **Edit Task**.

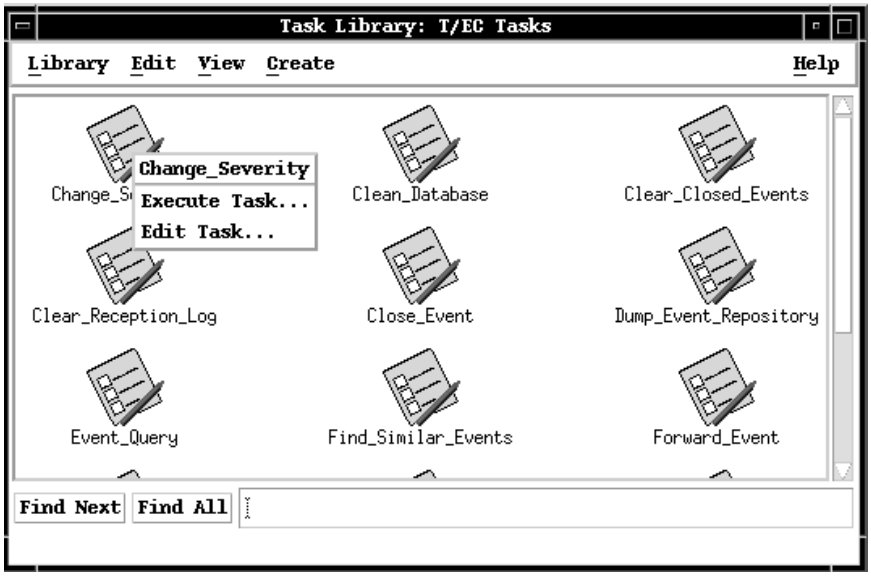


Figure 104. Editing the Task

The edit task will be displayed as shown in Figure 105 on page 193.

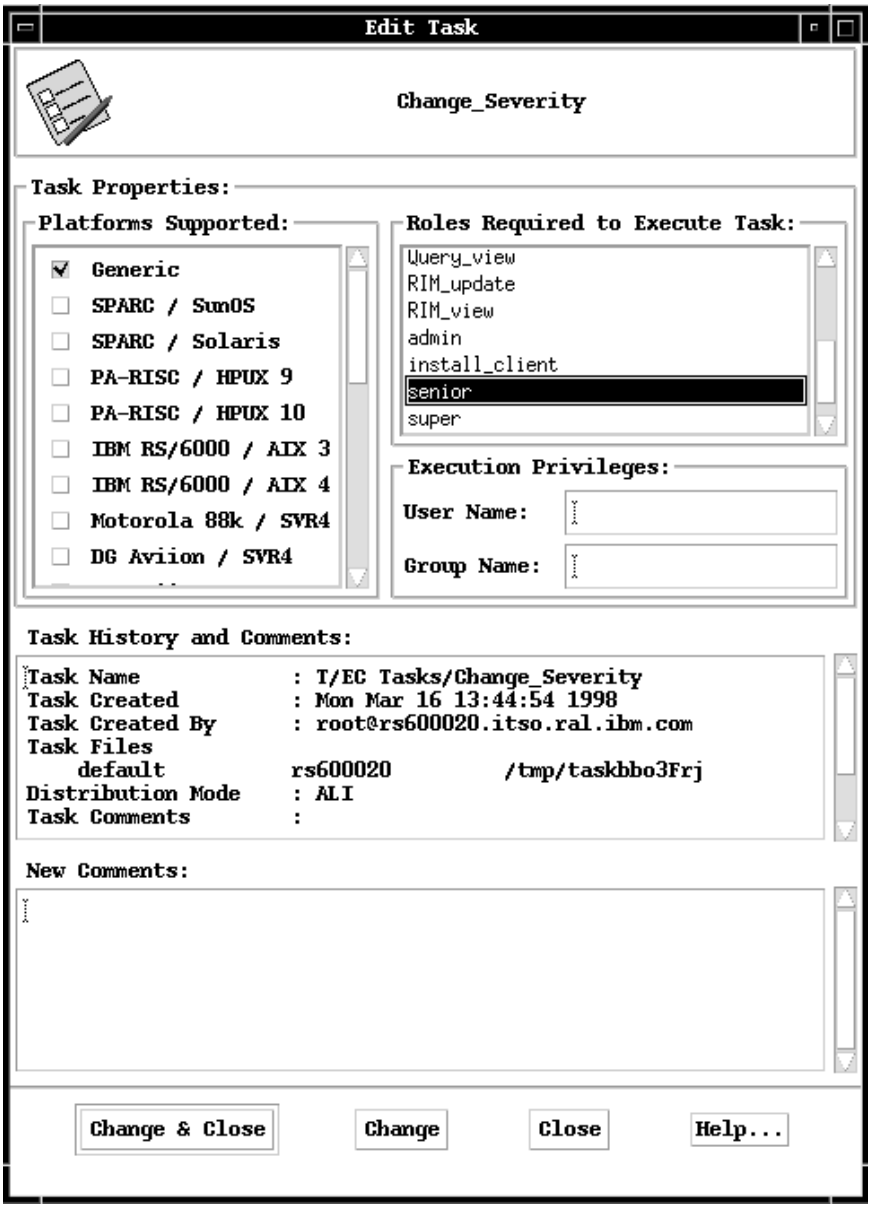


Figure 105. Editing a Task

There may be a requirement to stop certain operators from executing a particular task. The example shows that we changed the role to senior and clicked on **Change & Close**.

Once the task had been modified we ran the task as shown in Figure 106 on page 194.

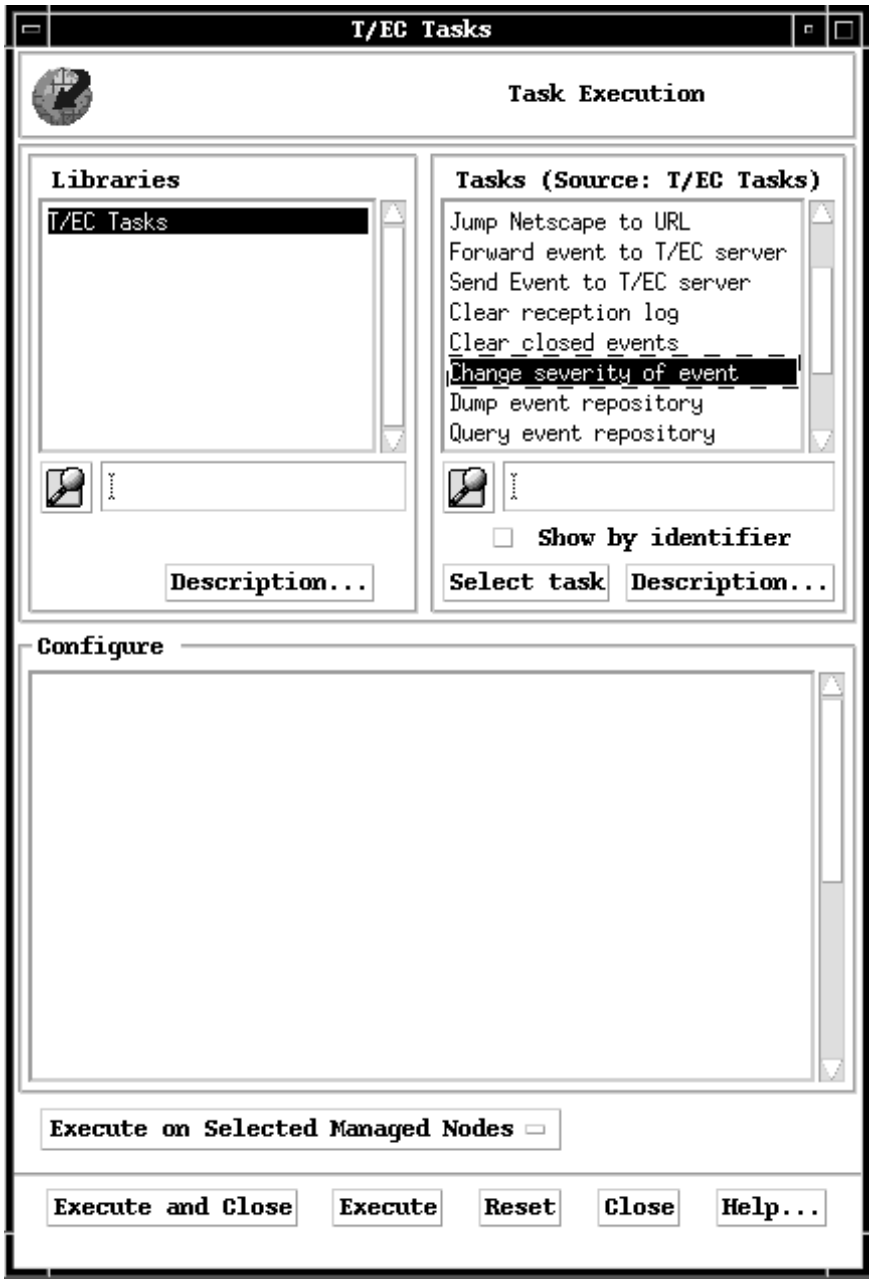


Figure 106. The TEC Tasks

When we ran the task as the operator Server_Operator, the task ran with the output shown in Figure 107 on page 195.

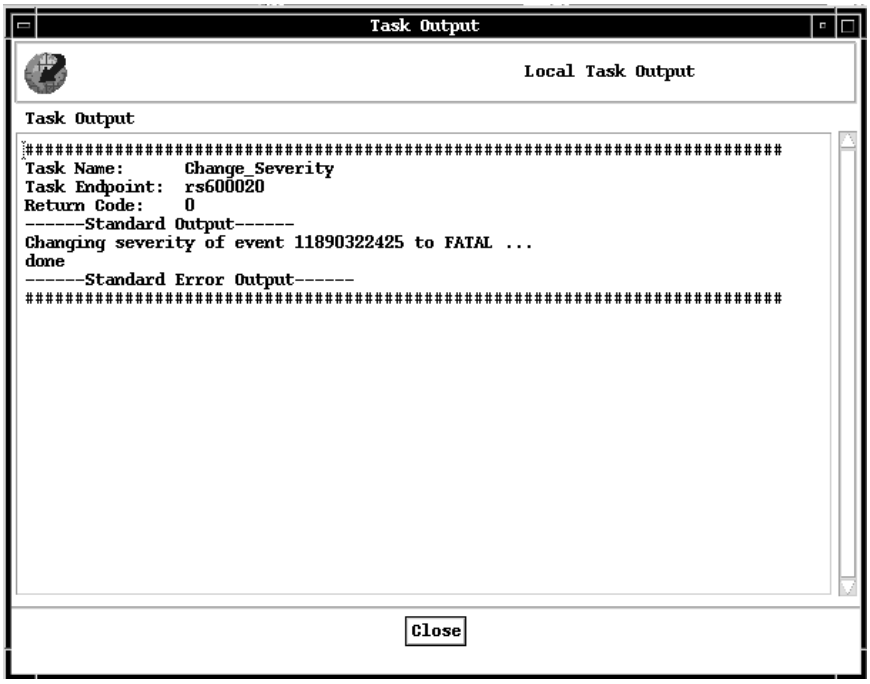


Figure 107. Running the Set Severity Task

When we ran the task as the Network_Operator without the required permission Figure 108 was displayed.

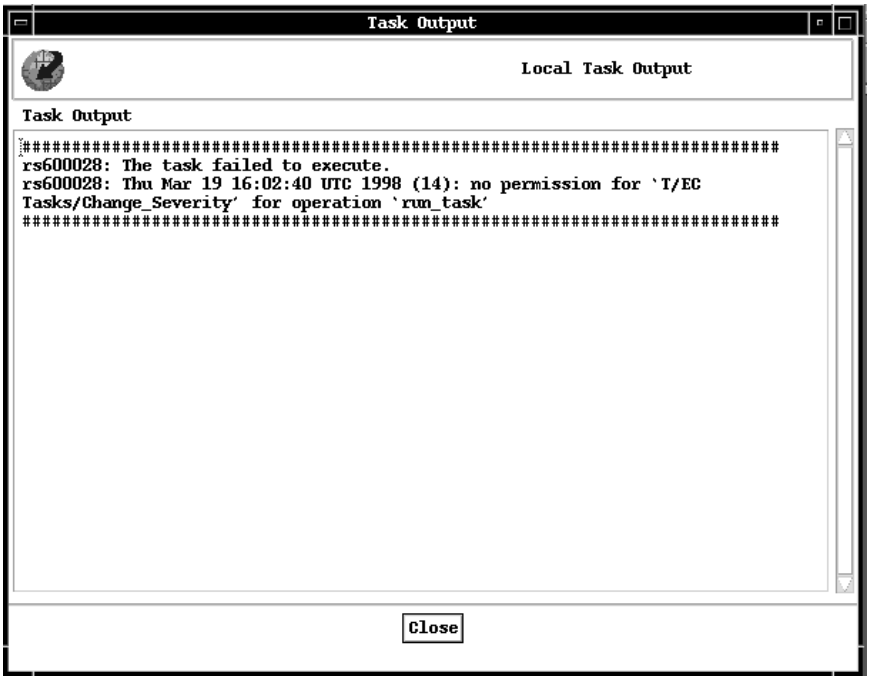


Figure 108. No Permission to Execute Task

9.3 Creating New Tasks

To show how to add new TEC tasks we developed some simple shell scripts to perform useful tasks for our environment. Our new tasks are created in the directory /usr/development/tasks.

Table 2. New Tasks

Task Name	Description	Permissions
ctc_ping_host	Pings the hostname that sent the event.	Admin
ctc_db_space	Uses wtdbspace to view database information.	Admin
ctc_set_event_status	Sets the current event to ACK and changes severity to Minor.	Senior
ctc_db_events	Checks how many occurrences of the current event are still in the cache and have a status of OPEN.	Senior
ctc_stop_sentry	Stops the sentry engine on the current host.	Senior
ctc_start_sentry	Re-starts the sentry engine on the current host.	Senior
ctc_open_events	Lists all open events older than 24 hours.	Admin

In the following sections we list the actual scripts and show how to add these scripts to a library. The task output is also shown.

9.3.1 The ctc_ping_host Task

This task will be passed the hostname from the current event and will ping the hostname twice. The script is shown below:

```
############################################################
#!/bin/ksh
# Task Name:ctc_ping.sh
# Task to ping the host that sent the event to the Console
############################################################

/usr/sbin/ping -c 2 $hostname
exit 0
```

This task was created using the command:

```
wcrttask -t ctc_ping -l OPERATOR_TASKS -r admin\
         -i aix4-r1 rs600028 /usr/development/tasks/ctc_ping.sh
```

When this task is executed the following screen will be displayed as shown in Figure 109 on page 197.

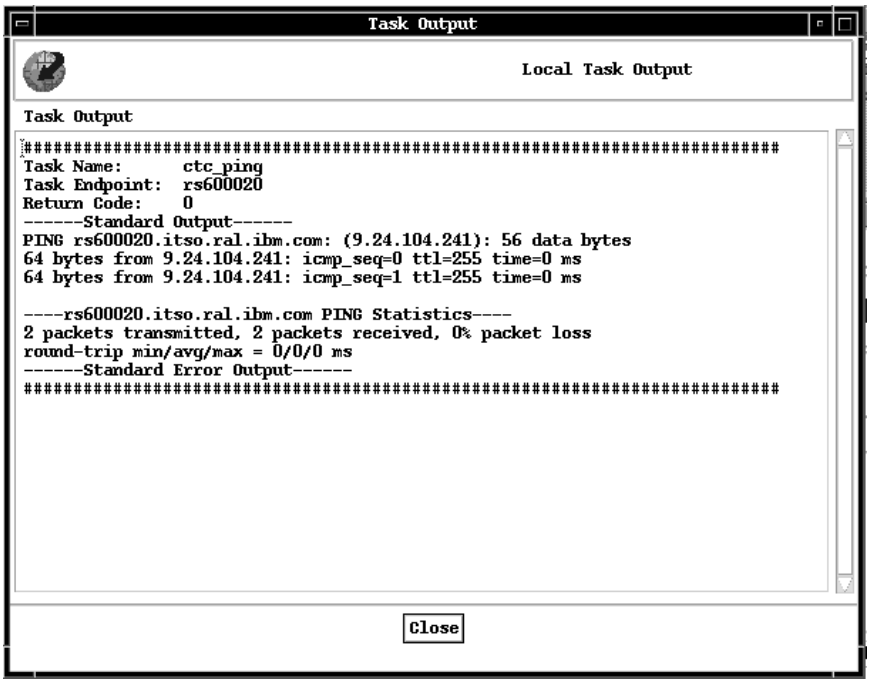


Figure 109. Output from Task ctc_ping

We can see that task performed a ping on the host rs600020.

9.3.2 The ctc_db_space Task

The ctc_db_space command runs the Tivoli command wtdbspace. The script is shown below:

```
############################################################
#!/bin/ksh
# # task Name: "ctc_db_space.sh"
# Task to show the dbspace for the TME Database
#
############################################################

 . /etc/Tivoli/setup_env.sh

wtdbspace

exit 0
```

To add this script to the library execute the command:

```
wcrttask -t ctc_db_space -l OPERATOR_TASKS -r admin \
         -i aix4-r1 rs600028 /usr/development/tasks/ctc_db_space.sh
```

When this task is executed from the TEC the following output will be displayed as shown in Figure 110 on page 198.

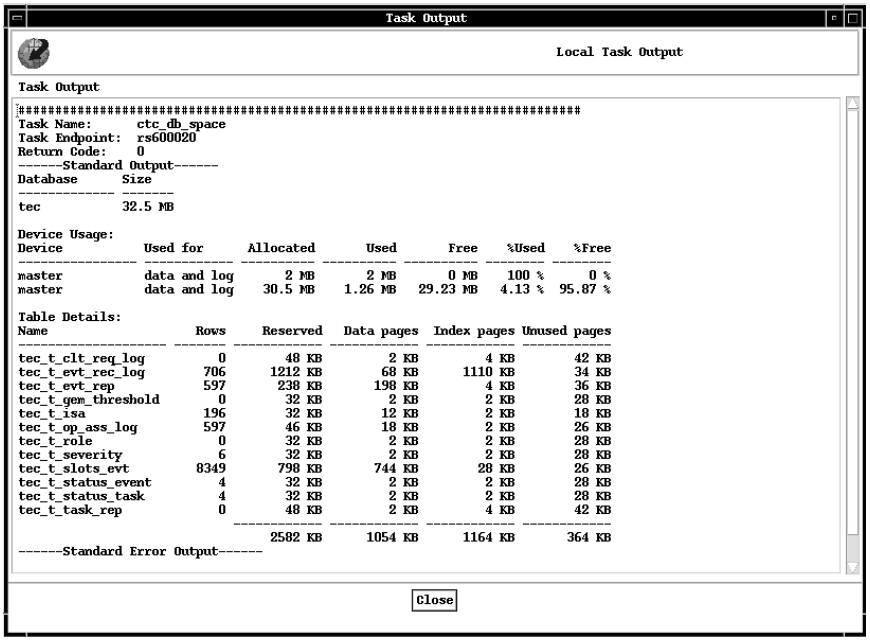


Figure 110. Output from Task ctc_db_space

9.3.3 The ctc_set_event Task

The ctc_set_event script shows how variables can be passed from the TEC event. These variables can be used from within a script. This example creates a logfile that contains certain information. The task changes the event to be Acknowledged, and records which operator performed this action and when. The script is shown below.

```
#############################################################################
#!/bin/sh
# This task is called from the TEC Interface
# Task Name: ctc_set_event.sh
# Set the event status to ACKNOWLEGDE and record the information in
# the LOGFILE
#############################################################################

LOGFILE=/usr/development/tasks/task.log

echo "****************  date  *************" >> $LOGFILE
echo "Event Server " $EventServer >>$LOGFILE
echo "wsetemsg -t ACK @"$CONSOLE_NAME" $ev_key" >> $LOGFILE
echo "Event Console Class Hostname Severity origin message "
 >> $LOGFILE
echo $ev_key $CONSOLE_NAME $class_name $hostname $severity $origin $msg
 >> $LOGFILE
echo "wsetemsg -t ACK @"$CONSOLE_NAME" $ev_key"
echo "**************** END *************" >> $LOGFILE

wsetemsg -t ACK @"$CONSOLE_NAME" $ev_key
rc=$?
echo done
exit $rc
```

This script could be modified to add additional functions, such as changing slot values. In fact any Tivoli command can be used and an audit log can be recorded containing the command output.

This task is created using the command:

```
wcrttask -t ctc_set_event -l OPERATOR_TASKS -r admin \
         -i aix4-r1 rs600028 /usr/development/tasks/ctc_set_event.sh
```

An example of the logfile generated by this script is shown below:

The logfile is called /usr/development/tasks/task.log.

```
*************** Thu Mar 19 18:30:36 UTC 1998 *************
Event Server
wsetemsg -t ACK @Root_rs600020-region 11890331478
Event Console Class  Hostname  Severity  origin  message
11890331478 Root_rs600020-region Logfile_Snmpd rs600020 WARNING 9.24.104.
EXCEPTIONS: authentication error: invalid community name: ITSO'
*************** END *************
```

9.3.4 The Tasks to Stop and Start Sentry

The tasks to stop and start Distributed Monitoring could be called when you want to stop any further events from being sent from a managed node. An alternative to this would be the the logfile adapter.

The scripts are shown below:

```
#!/bin/ksh
# Task Name ctc_start_sentry
# Task to Re-start the Sentry Engine on a specific Managed Node
#
#
 . /etc/Tivoli/setup_env.sh

SENTRY_LOG=/usr/development/tasks/sent_engine.log

echo "Starting Sentry_Engine on $hostname at  date "
wlseng $hostname

exit 0
```

```
#!/bin/ksh
# ctc_stop_sentry
# Task to stop the Sentry Engine from Sending Any more events
#
#
 . /etc/Tivoli/setup_env.sh

SENTRY_LOG=/usr/development/tasks/sent_engine.log

echo "Stopping Sentry_Engine on $hostname at  date " >> $SENTRY_LOG
wstopeng $hostname

exit 0
```

To add the tasks to the library execute the command:

```
wcrttask -t ctc_start_sentry -l OPERATOR_TASKS -r admin \
         -i aix4-r1 rs600028 /usr/development/tasks/ctc_start_sentry.sh
wcrttask -t ctc_stop_sentry -l OPERATOR_TASKS -r admin
         -i aix4-r1 rs600028 /usr/development/tasks/ctc_stop_sentry.sh
```

The task output example for the ctc_stop_sentry task is shown in Figure 111.

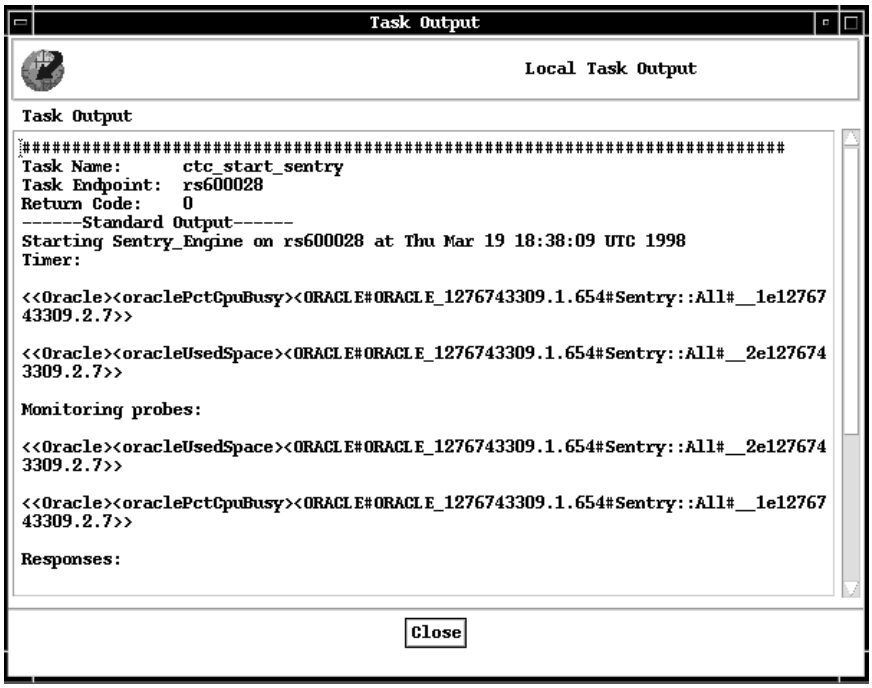


Figure 111. Output from Task ctc_start_sentry

9.3.5 The ctc_db_events Task

This script initially resolves what type of database is being used by the TEC. It will then access the database using the relevant sql command. This script could be modified to include any sql statements.

The script is listed below:

```
################################################3
#!/bin/ksh
# Script to check what RIM is being used then
# show how many events are still OPEN for that
# event processed

function sybase_lookup
{
export DSLISTEN=TEC
export DSQUERY=TEC
export SYBASE=/sybase/code
export PATH=$PATH:$SYBASE/bin

isql -Utec -Ptectec <<!
select count(*) "Total Number of Events" from tec_t_evt_rep
where hostname='$HOSTNAME' and class='$CLASS' and status=0
go
!
}

function oracle_lookup
{
DBDIR='/usr/local/oracle/7.3.3/bin'
export ORACLE_BASE=/usr/local/oracle
export ORACLE_HOME=/usr/local/oracle/7.3.3
export
ORACLE_PATH=.:/usr/local/oracle/7.3.3/bin: \
    /usr/local/oracle/7.3.3/obackup/bin:/opt/bin: \
    /bin:/usr/bin:/GNU/bin/make:/usr/ccs/bin
export ORACLE_SID=tec
export ORACLE_TERM=vt100

$DBDIR/sqlplus -s tec/tectec@tec  <<!
SET ECHO OFF

select count(*) "Total Number of Events " from tec_t_evt_rep
 where class = '$CLASS'
 and hostname = '$HOSTNAME'
 and status = 0;
 exit;
!
}

#Resolve Database
CLASS=$class_name
HOSTNAME=$hostname

echo "Class Type is $CLASS from Source is $HOSTNAME"
RIM= wlookup -r RIM tec 2>/dev/null
RIM_TYPE= idlcall $RIM get_type_name

case $RIM_TYPE in
        *Sybase*)
                DBTYPE="Sybase"
          sybase_lookup
                ;;
        *Oracle*)
                DBTYPE="Oracle"
          oracle_lookup
                ;;
esac
exit 0
```

To add this script to the library we ran the command:

```
wcrttask -t ctc_db_events -l OPERATOR_TASKS -r admin \
```

```
-i aix4-r1 rs600028 /usr/development/tasks/ctc_db_events.sh
```

When this task is executed it will first resolve what type of database is running, then it will access the database and show the output shown in Figure 112.

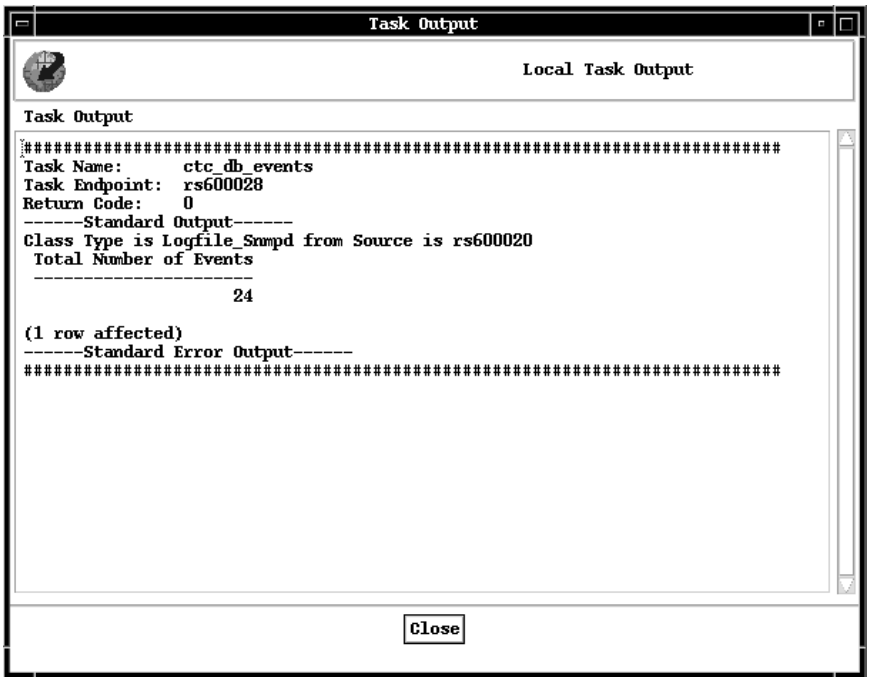


Figure 112. Output from Task ctc_db_events

This script is executed on a specific event. The example shows that we have 24 open events for the logfile_snmp class.

9.3.6 Task Summary

To define our tasks to a new library, our new library has already been defined in Chapter 4, “Software Installation” on page 49 using the command wcrttlib OPERATOR_TASKS TEC_MANAGEMENT.

We developed a set of scripts and added these scripts using the wcrttask command.

To verify that the task has been created issue the following command:

```
wgettask ctc_db_space OPERATOR_TASKS
```

Our example for this task was:

```
Task Name               ctc_db_space
User Name               *
Group Name
Task ACL                admin
Supported Platforms
        aix4-r1   <install-dir>/aix4-r1/TAS/TASK_LIBRARY/bin/1276743309/O
_TAS_upatpwfa
Task Comments
        Task Name            : OPERATOR_TASKS/ctc_db_space
        Task Created         : Thu Mar 19 13:09:21 1998
        Task Created By      : root@rs600028.itso.ral.ibm.com
        Task Files
         aix4-r1          rs600028 /usr/development/tasks/ctc_db_space.sh
        Distribution Mode    : ALI
        Task Comments        :
        ------------------------------------------------------------
```

When all the tasks are created they will appear on the desktop as shown in Figure 114 on page 204.

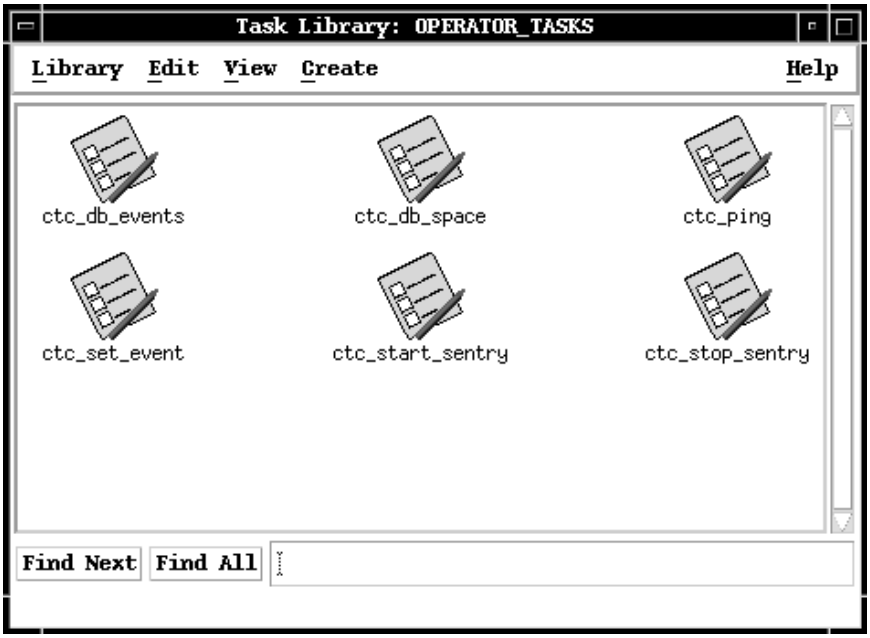


Figure 113. New Tasks Created in Library OPERATOR_TASKS

Each task can be executed from the TEC console from the pull-down menu as shown in Figure 114 on page 204.

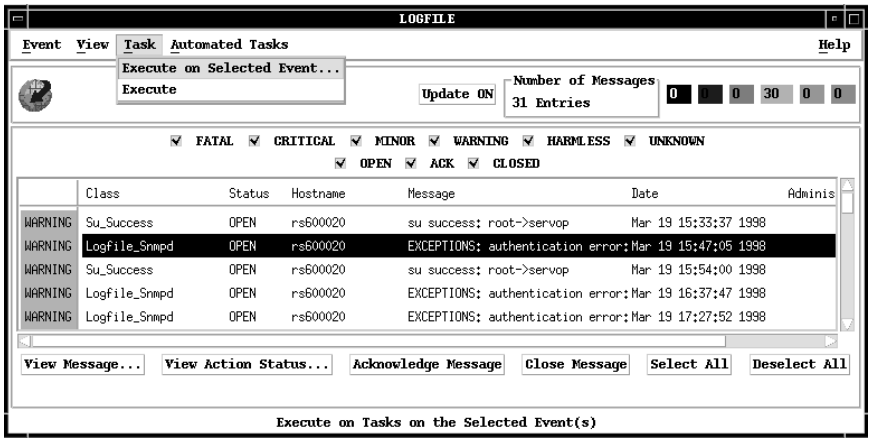


Figure 114. Running a Task

The list of tasks will appear (see Figure 115 on page 205).

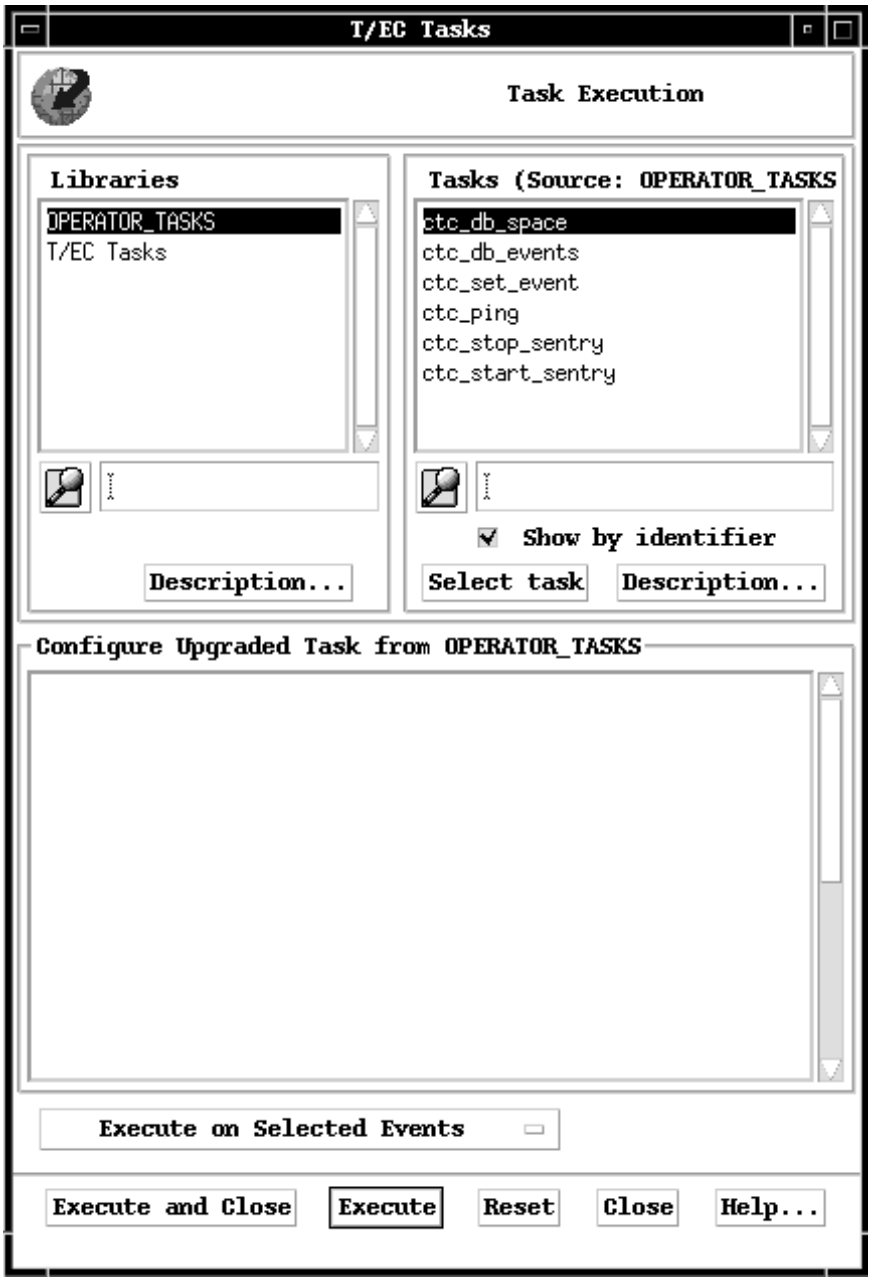


Figure 115. New Tasks

Certain tasks have to be executed on specific nodes. For example, the ctc_db_events must be executed on select rs600028. This is because the managed node rs600028 is the RIM host machine. If the task is to be executed on a specific node, then you must change the execute option for the task. This option is located on Figure 115.

Figure 116 on page 206 shows how to select a specific node.

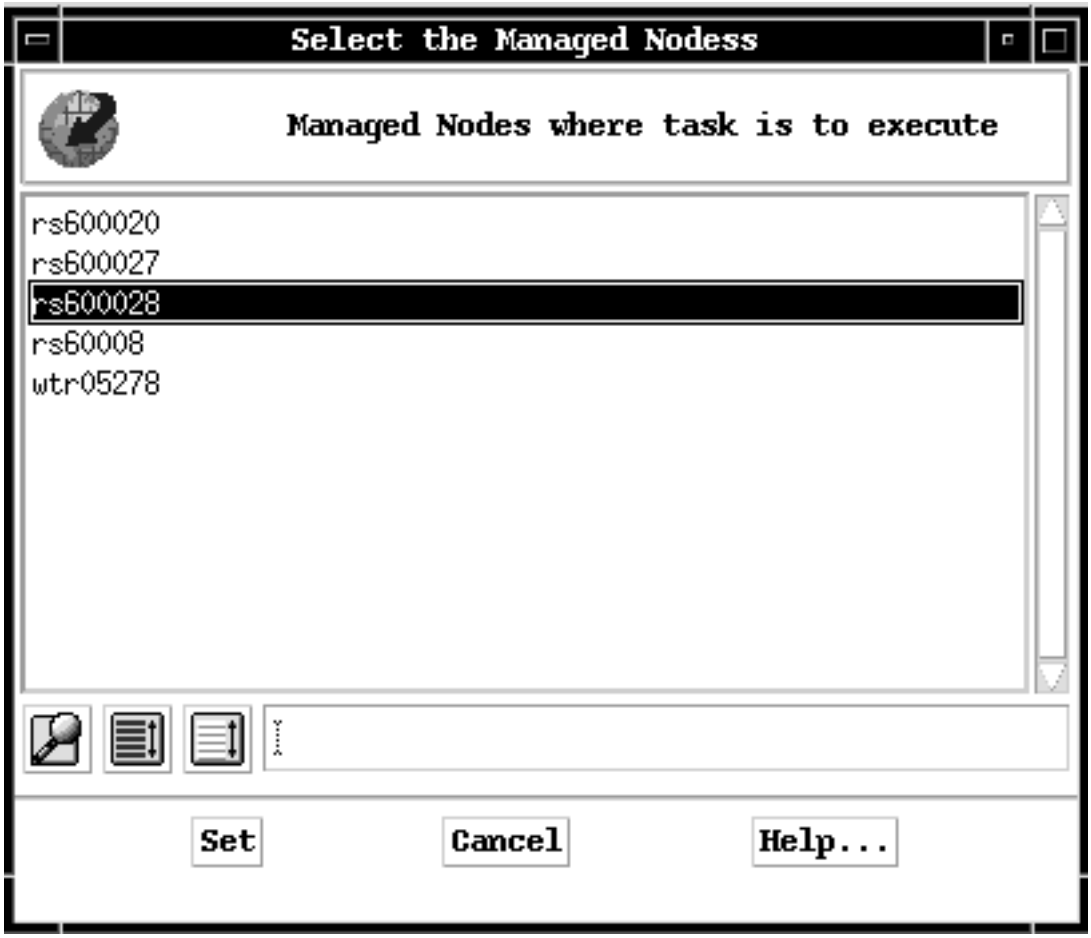


Figure 116. Execute on a Specific Managed Node

The tasks also record information in the TME notices as shown in Figure 117.

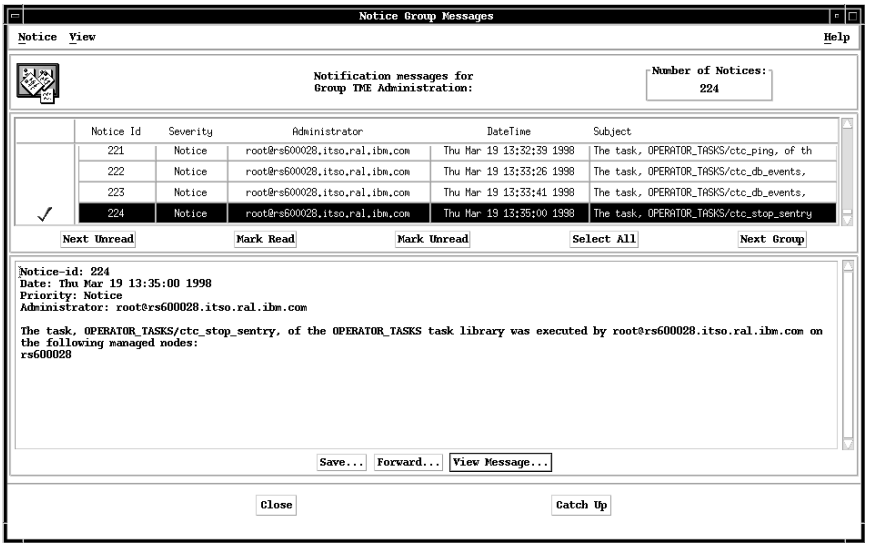


Figure 117. Viewing the Notice Board

9.4 Automating a TEC Task

The first example of automating an event is to create an automated task that will check for an event that will be checked for duplicates within the rule engine. If this specific event has been received more that five times, we will amend the severity. Our script uses information contained in the specific event to display on the operators desktop. The script ctc_au_repeat.sh is shown below:

```
#!/bin/ksh

#  Example of the Automated task
#  Task to popup pre-defined message to the Console
#

 . /etc/Tivoli/setup_env.sh

TEMP_FILE=/tmp/ctc_message

#Set up temp file
echo "*****WARNING EVENT OCCURED MORE THAN 5 TIMES ****" > $TEMP_FILE
echo "Event Details:\n\n" >> $TEMP_FILE
echo "Hostname  origin  " >> $TEMP_FILE
echo "$hostname $origin " >> $TEMP_FILE
echo "message: $msg" >> $TEMP_FILE

wsendresp "$CONSOLE_NAME" rs600028:/tmp/ctc_message

rm $TEMP_FILE

rc=$?
echo done
```

9.4.1 Creating the Task

The command to create the task is as follows:

```
wcrttask -t ctc_au_repeat.sh -l OPERATOR_TASKS -r admin \
-i aix4-r1 rs600028 /usr/development/tasks/ctc_au_repeat.sh
```

9.4.2 Defining the Automated Task

To create an automated task using the GUI, select from the TEC events window **Automated Tasks->New** as shown in Figure 119 on page 208.

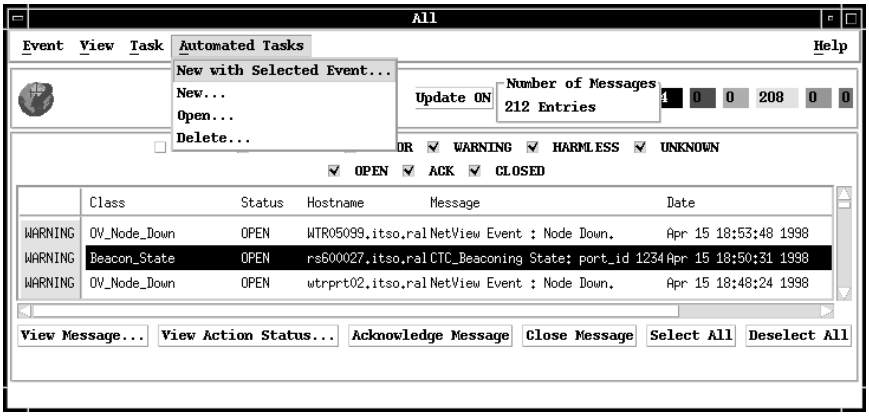


Figure 118. Creating the Automated Task

Next we selected the event class of **Beacon_State**. This class is the class for our beacon state event. Specify single classes here. Our selections are shown in Figure 119 on page 208.

Figure 119. Using the Classes to Run Automated Tasks

Figure 120 appears. From here click on **Edit Criteria**.

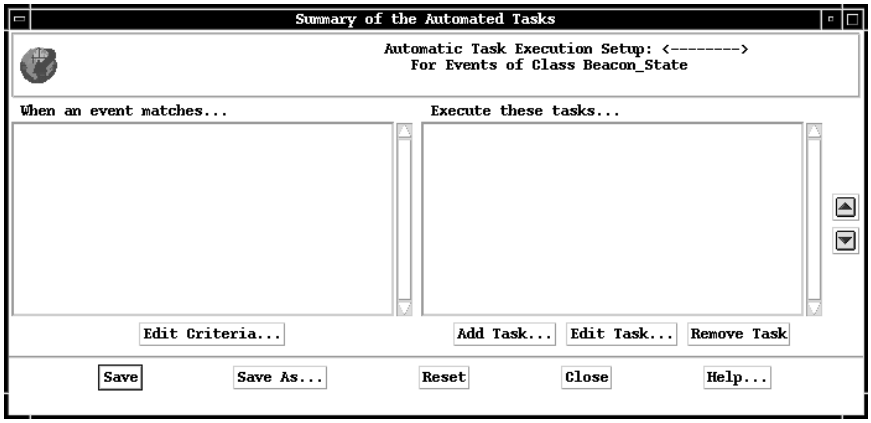


Figure 120. Automated Tasks

The rules that are developed for the events within the ctcable_Event class add 1 to the repeat_count field when a duplicate event is detected. We wanted to alert the TEC operator when this event has been sent five times. To to this we selected the **repeat_count** field from the Available slot window as shown in Figure 121 on page 209.

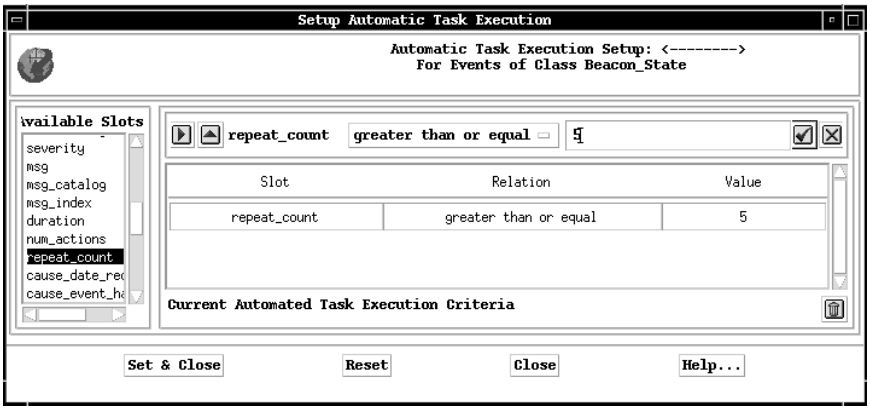


Figure 121. Setting Up the Automated Task

Next select **Add Task** from Figure 120 on page 208. Select the task **ctc_au_repeat.sh**.

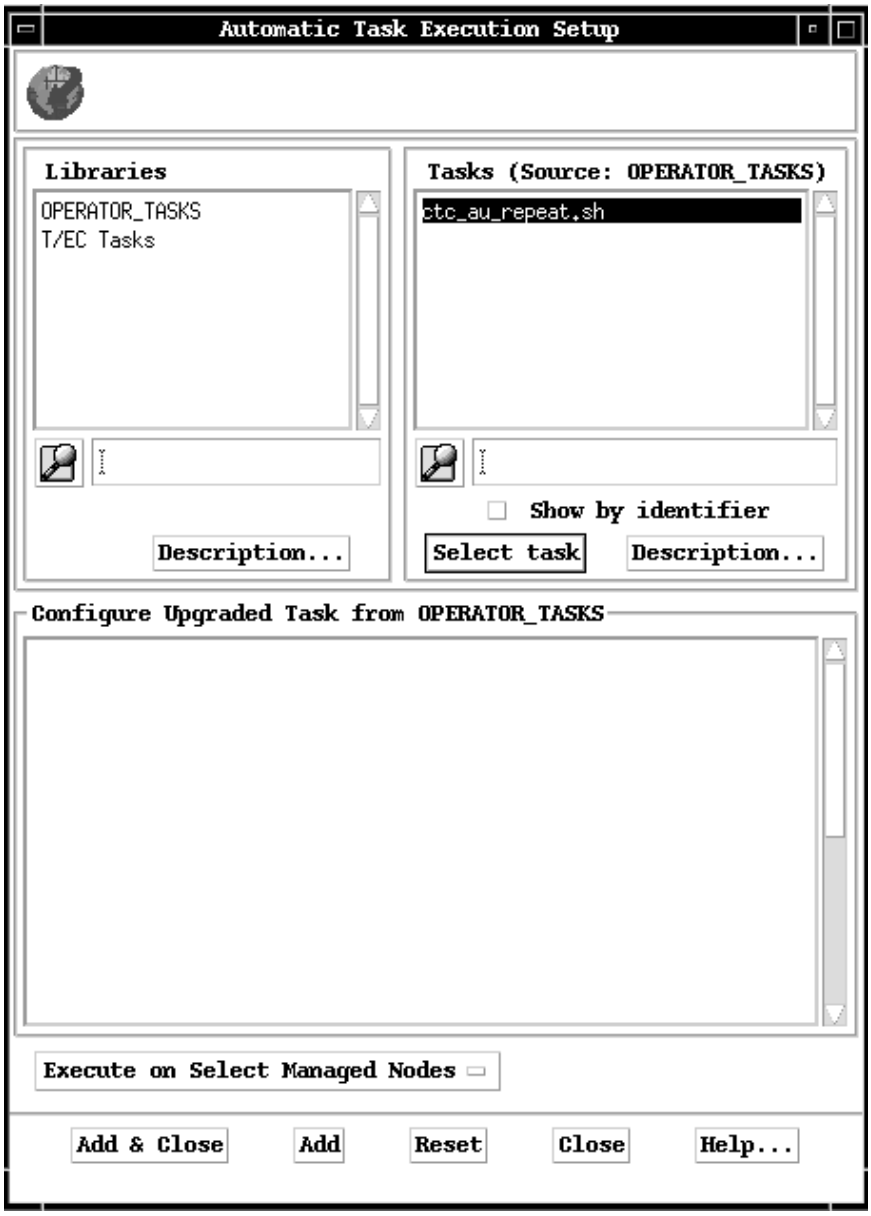


Figure 122. Automatic Task Execution Setup

Select the node to execute this task. We chose rs600028 (see Figure 123 on page 211).

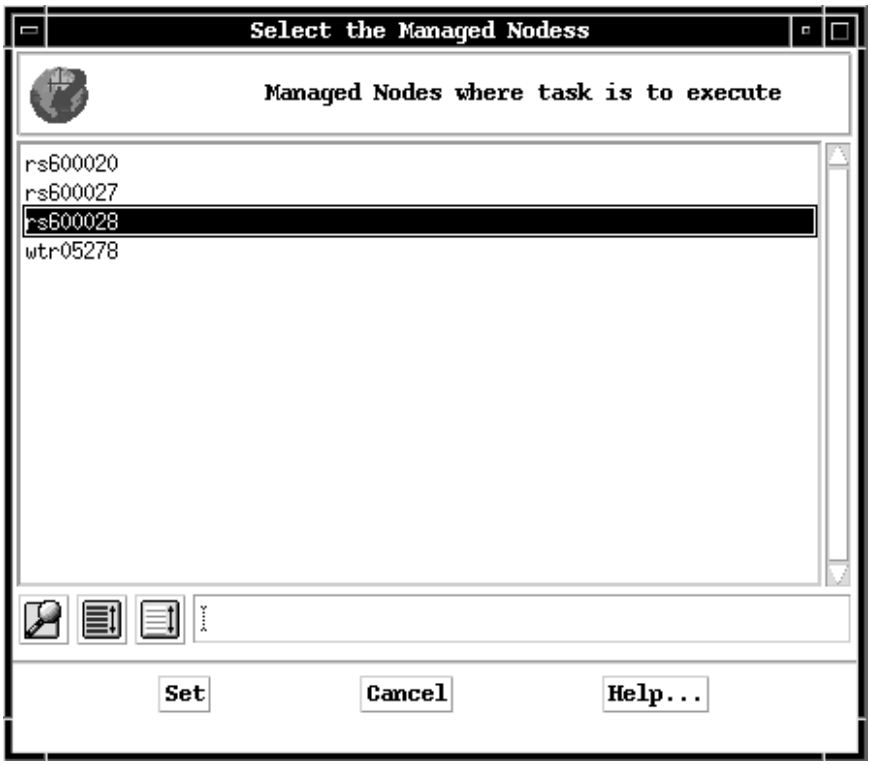


Figure 123. Selecting the Managed Node

Once the managed node has been selected then from the summary screen shown in Figure 124 click on **Save**.

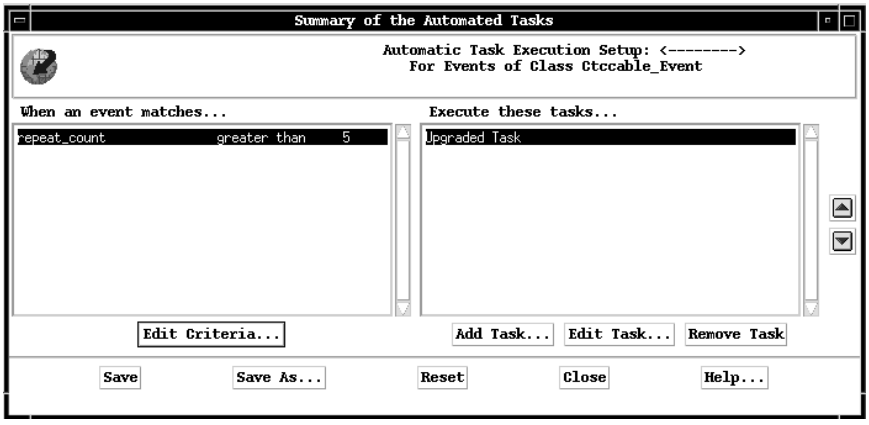


Figure 124. Summary of Defined Tasks

Finally save the automated task as shown in Figure 125 on page 212 by clicking on **Save**.

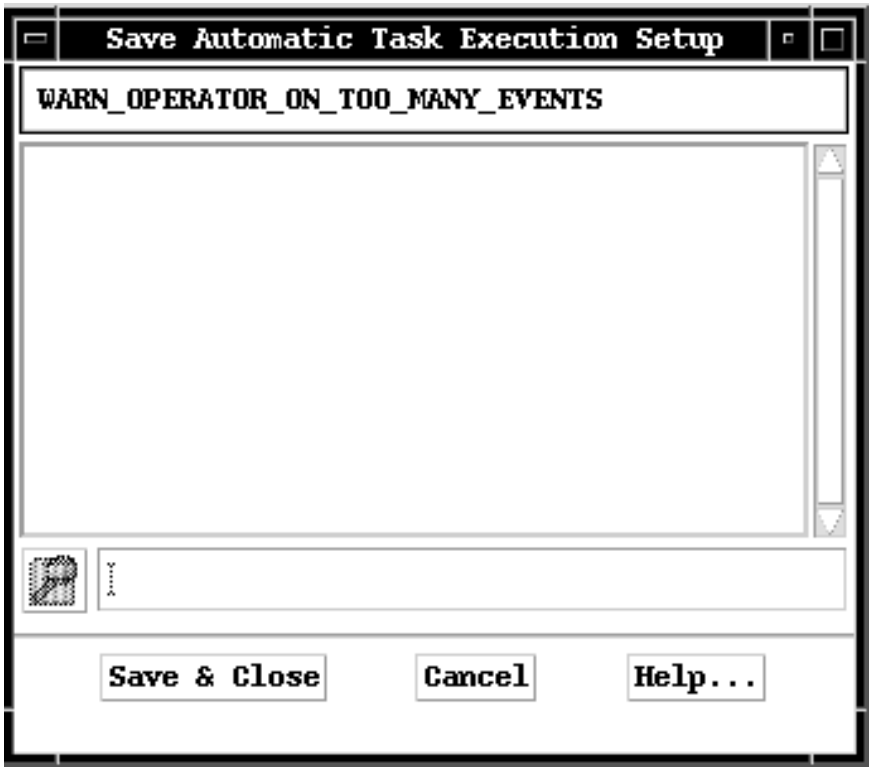


Figure 125. Saving the Task

When we tested this task, we got the output shown in Figure 126.

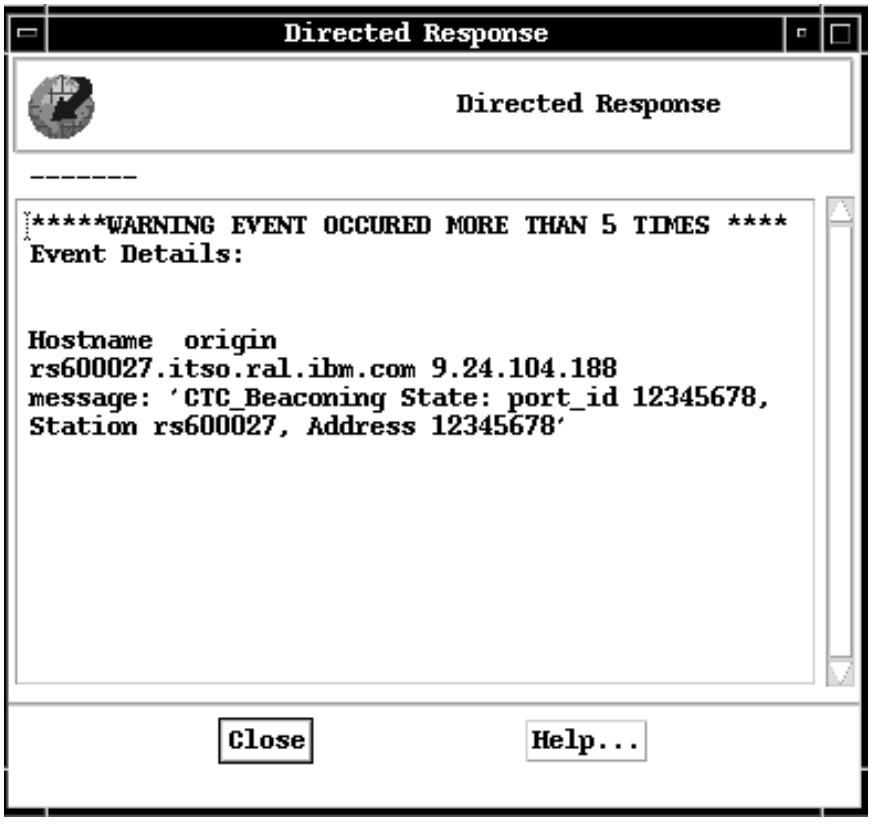


Figure 126. Task Output

9.5 Create a Job to Back Up the TEC Database

We show you an example of defining your own task to back up the TEC database, taking you through the steps to automate the procedure.

Script to backup the sybase directory. We can set this script to run once every day at 6 p.m.

```
#!/bin/sh
# Name: ctc_sybase_backup.sh
#
#Script to backup the Sybase

export DSLISTEN=TEC
export DSQUERY=TEC
export SYBASE=/sybase/code
DATE= date | awk '{print $1$2$3}'

SYB_USER=sa
$SYBASE/bin/isql -Usa -P"" <<!

dump database master to "/sybase/backups/master.$DATE"
go
dump database model to "/sybase/backups/model.$DATE"
go
dump database sybsystemprocs to "/sybase/backups/sybsys.$DATE"
go
dump database tec to "/sybase/backups/tec.$DATE"
go
!

echo "backup complete"
exit 0
```

The following process will create the task and schedule the job:

- Create the task
- Create a job
- Schedule the job to run once per day

To create the task use the following command:

```
 wcrttask -t ctc_sybase_backup -l OPERATOR_TASKS -r senior -i
-i aix4-r1 rs600028 /usr/development/tasks/ctc_sybase_base.sh
```

To create the job:

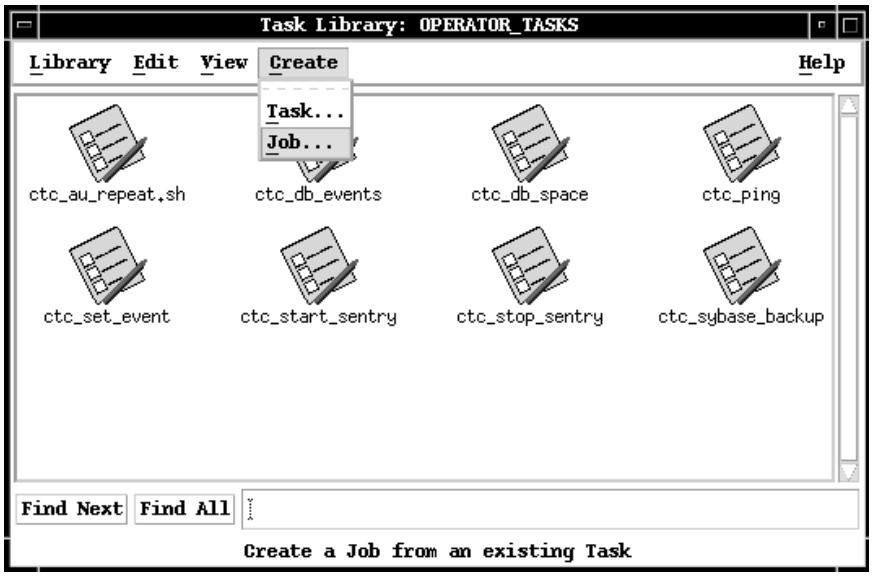


Figure 127. Creating a Job

Select **Create->Job** from the pull-down menu as shown in Figure 128.

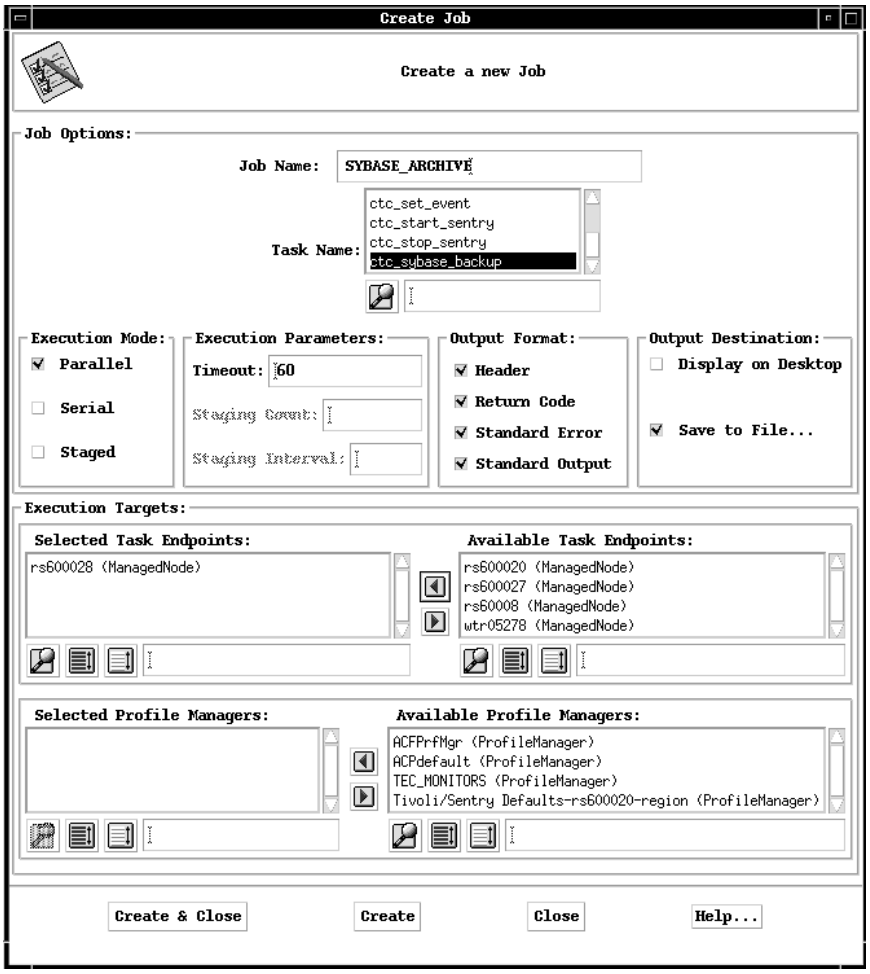


Figure 128. Creating a Job

We entered the details as shown in Figure 129 on page 215.

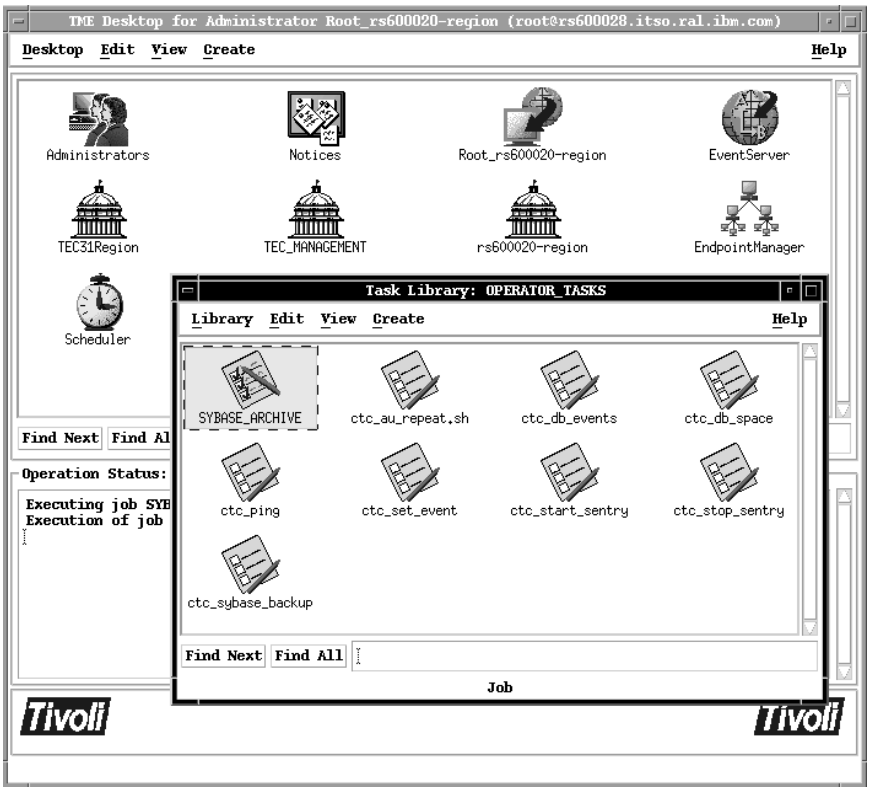


Figure 129. Entering the Job Parameters

To schedule the job move the SYBASE_ARCHIVE icon to the Scheduler icon on the desktop (see Figure 130 on page 216).

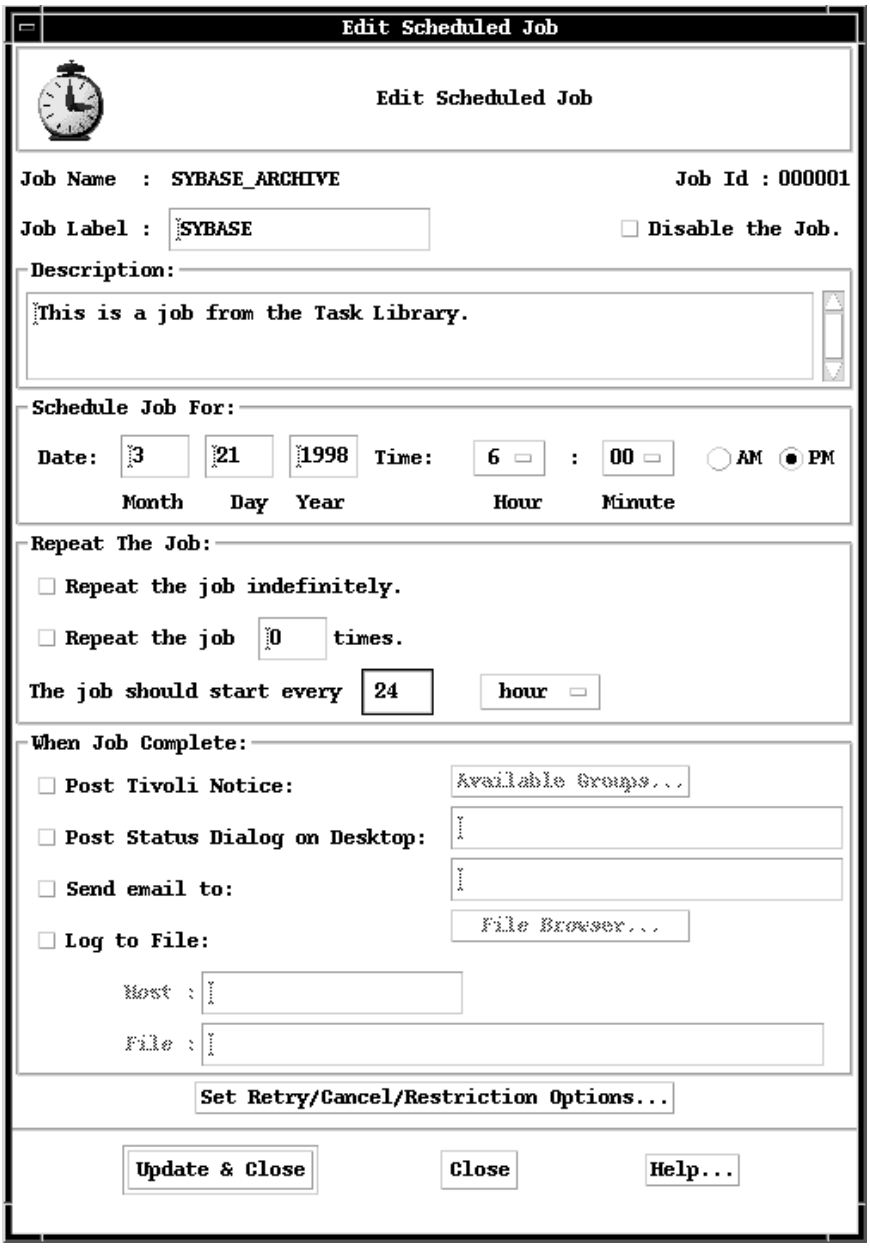


Figure 130. Setting Up the Scheduler

The scheduled jobs are shown in Figure 131 on page 217.

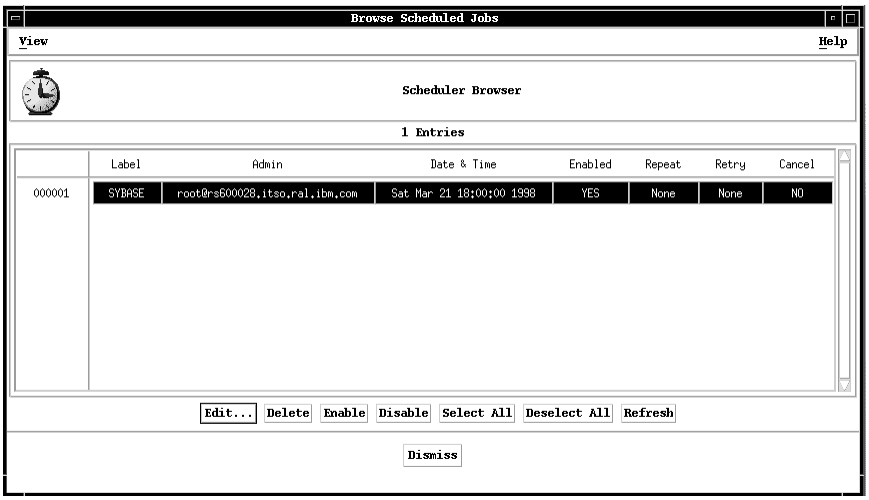


Figure 131. Browsing the Scheduled Jobs

The command to create a scheduled job is wschedjob.

This job will perform a Sybase database backup every 24 hours.

Chapter 10. The TEC Rules

This chapter contains an introduction to the rules engine, an overview of the rules logic available, and a list of the information you should gather prior to developing new rules.

We show some examples of existing rule code and analyze it line by line.

We use a standard approach when developing new rules. For each rule required we collect or develop the following information:

- A table describing all related events
- A flowchart showing each step in the logic of the rule
- Sample rule code we can copy and change

10.1 TEC Processes and the Event Flow

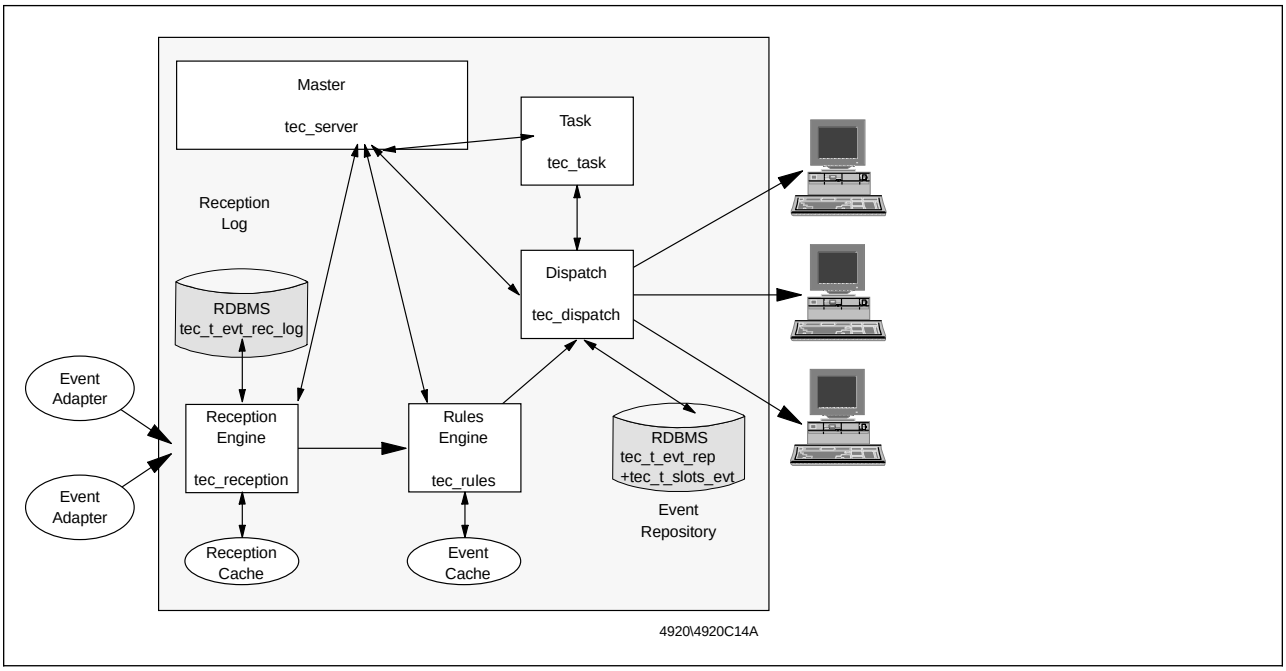


Figure 132. Event Flow

In the diagram above, you can follow the path of events through the TEC. Events arrive from event adapters over an agreed-upon port.

The reception engine logs all incoming events in the reception log, assigning each one a unique ID and time stamp, and then passes them along to the rules engine. If the rules engine is busy, the events are buffered in the reception cache until the rules engine can process them.

The rules engine then uses the active rule base to analyze whether the events are valid, and, if so, what should be done with them. This can include initiating automatic actions, reevaluating the status of previous events, notifying a person, etc. The rules engine also stores current events, up to a pre-configured limit, in a

memory-resident event cache, so that they are quickly available for correlation with other incoming events.

The rules engine informs the dispatch engine of any changes to events, and the dispatch engine updates the event consoles. The dispatch engine also handles communication in the reverse direction, by receiving requests from event consoles for event status changes, and makes sure that the event repository is updated with all changes.

When a rule specifies that a task is to be carried out, the dispatch engine passes the request to the task engine.

A normal startup of the event server causes the start of the process tec_server, which in turn starts the four processes tec_reception, tec_task, tec_rule, and tec_dispatch, all of which are started with the parameter -config /usr/local/Tivoli/bin/aix4-r1/TME/TEC/.tec_config.

The file .tec_config defines the port over which the process tec_reception on UNIX receives events from event adapters, in the event the portmapper deamon is not used for assigning and mapping ports. The file contains the line shown below:

```
tec_recv_agent_port=5529
```

Event reception on NT runs over a fixed port.

10.2 The Rules Engine

The T/EC event server consists of five daemon processes:

1. **tec_server** is the server process that coordinates the other four processes.

   Should one of these processes fail, the tec_server attempts to restart it. If it cannot restart the process, the tec_server will bring the others down gracefully.

2. **tec_reception** is the reception process that receives incoming event instances from all the various sources.

   When an event comes into tec_reception, the event is cached in memory and written to the reception log in the database (RDBMS). As each event instance is received, it is validated against the event class definitions in the active rule base. If there is no corresponding definition, the event instance is not advanced past the reception log. If the appropriate event class definition is found, the event is sent to the tec_rule process, where it is processed.

3. **tec_rule** is the rules process (also called the rule engine), which defines the action or actions to be taken and passes the action or actions to the tec_dispatch process.

   Event instances are advanced from the reception cache to the rules cache one at a time in the order in which they were received (FIFO). The tec_rule process compares each event instance against each rule in the rule base sequentially. If the state of the event instance changes, tec_rule communicates this change to the tec_dispatch process, which insures that the update is seen everywhere.

4. **tec_dispatch** is the dispatch process (also called the dispatch engine), which writes the event instances to the event repository in the database (RDBMS) and updates the event console(s) with current information.

The tec_dispatch process must communicate with the tec_rule process and the tec_task process in order to know when information needs updating. If there are tasks or programs to be run for an event, the tec_task process is contacted.

5. **tec_task** is the task process (also called the task engine), which is responsible for executing any programs, tasks, scripts or commands initiated by rules or consoles.

   The tec_task process monitors the execution of these programs, tasks, scripts or commands, and can return their exit status to tec_dispatch, which will write that information into the appropriate task table in the database (RDBMS).

These processes may be viewed using the UNIX ps command or through the Task Manager on NT.

If any of these five processes fails, error conditions are written to a file in the /tmp directory. The output files for errors are:

- /tmp/tec_master (for tec_server)
- /tmp/tec_reception
- /tmp/tec_rule
- /tmp/tec_dispatch
- /tmp/tec_task

The names of these files are specified in the $BINDIR/TME/TEC/.tec.diag.config and can be changed.

The event server machine must be a managed node client within a TMR. There can be only one T/EC event server in a TMR, and it is recommended that the event server run on a machine that is solely dedicated to event processing.

10.3 Basic Rule Writing

There are three basic types of rules, categorized by what triggers them:

- *Plain rules* are triggered by receipt of a new event.
- *Change rules* are triggered by a request to change a previously received event.
- *Timer rules* are triggered by the expiration of a timer set on a previously received event.

A rule consists of a *header*, for the purpose of matching an event or events, and an *action body*, which carries out some operation or operations in the event of a match.

There are three types of primitives used in action bodies of rules:

1. There are four *event specifiers*, which are used in action bodies to search the event cache for events meeting specified criteria. The available event specifiers are shown in the following table.

Table 3. The Four Event Specifiers	
Event Specifier	**Function**
all_instances	Searches the event cache for all events that fulfill the specified criteria
first_instance	Searches the event cache for the first (the most recent) event that fulfils the specified criteria
all_duplicates	Searches the event cache for all events that match the event class and the dup_detect=yes slots of the reference event
first_duplicate	Searches the event cache for the most recent event that matches the class and the dup_detect=yes slots of the reference event

2. There are 21 (or so) templates, which are predefined frequently used actions. These are shown in the following table.

Table 4. The Action Templates	
Template	**Function**
add_to_repeat_count	Adds the specified number to the slot repeat_count for an event
change_event_administrator	Changes the administrator for an event
change_event_severity	Changes the severity of an event
change_event_status	Changes the status of an event
decrement_slot	Subtracts a number from the integer slot specified
drop_change_request	Prevents a change request from being applied after change rules are executed
drop_received_event	Discards a received event after all applicable rules are executed
exec_program	Executes a program
exec_task	Executes a task from a TME task library
forward_event	Forwards an event to the event server defined in tec_forward.conf in the currently active rule base
generate_event	Generates an internal event
increment_slot	Adds a number to the integer slot specified
link_effect_to_cause	Links an effect event to a cause event by placing the contents of two slots from the cause event into other slots in the effect event
place_change_request	Updates the contents of the specified slot
redo_analysis	Requests that an event be reprocessed by the rules engine
set_event_administrator	Sets the administrator for an event
set_event_severity	Sets the severity of an event
set_event_status	Sets the status of an event
set_timer	Sets a timer for an event
tec_agent_demo	Reads a list of file names and uses their contents to generate events for testing purposes
unlink_from_cause	Removes the slot values from an effect event that were copied in from a cause event by link_effect_to_cause

3. There are three *control primitives* which can be used to alter the default execution flow of the rules engine. By default, all rules are examined for any given event, in the order in which the rules were loaded. It is sometimes

desirable, however, to end the execution flow after a certain action, rule, or rule set. To do this, use the control primitives shown in the following table.

Table 5. The Three Control Primitives

Control Primitive	Function
commit_action	Completes the current action, then jumps to the next rule
commit_rule	Completes the current action, then jumps to the next rule set
commit_set	Completes the current action, then leaves rule processing for this event

10.4 Simple Rule Examples

Let's begin examining the functions provided by the rules engine by analyzing some of the rules delivered with the UNIX logfile adapter. By default, when you install the logfile adapter, the predefined rules are copied into /etc/Tivoli/tecad/etc/log_default.rls, so whether or not you've imported them into a rule base, modified them, or used them at all, you should be able to find the original rules, as delivered, in this file.

The rules we examine are:

- Eliminating duplicate events
- Correlating two events
- Setting a timer
- Responding to the expiration of a timer
- Re-analyzing an event
- Re-acting to a changed event

10.4.1 Eliminating Duplicate Events

The first rule in log_default.rls is a classic rule for eliminating duplicate events.

Imagine that a printer repeatedly sends out the message that toner is low. The operator is aware of the situation, wishes for the original event to remain visible on the event console so that the problem cannot be forgotten, but does not wish to see the event repeated frequently.

The typical way to handle this situation is to drop the repeat (or duplicate) events, but to update the repeat_count slot on the original event, so that there is still a record of how often the event has arrived.

But before we look at the rule itself, we need to understand what the event server means by the term *duplicate event*. A duplicate event is an event with the same event class as the reference (or original) event, and in which the contents of all slots marked dup_detect = yes are the same as in the reference event.

To understand what a duplicate event of class Printer_Toner_Low means, we need to examine the file /etc/Tivoli/tecad/etc/tecad_logfile.baroc, which defines the class structure for the logfile adapter. Since event classes inherit the characteristics of higher level classes in the hierarchy, we've reproduced the relevant upper level classes for Printer_Toner_Low. (We've left out the uppermost class EVENT, as no slots in EVENT are defined as dup_detect slots.)

```
TEC_CLASS:
        Logfile_Base ISA EVENT
        DEFINES {
                source: default= "LOGFILE";
                sub_source: default= "LOGFILE";
                sub_origin: default= "N/A";
                adapter_host: default= "N/A";
                msg_catalog: default= "none";
                msg_index: default= 0;
                repeat_count: default= 0;
                pid: STRING, default="N/A";
                severity: default = WARNING;
        };
END

TEC_CLASS:
        Logfile_Printer ISA Logfile_Base;
END

TEC_CLASS:
        Printer_Toner_Low ISA Logfile_Printer
        DEFINES {
                hostname: dup_detect = yes;
        };
END
```

From these definitions we can see that a duplicate event of class Printer_Toner_Low is one that has the same value in the slot hostname as does the original reference event.

Now we are ready to look at the rule that eliminates duplicate events. The rule actually handles multiple event classes at once. (We've numbered the lines to make it easier to discuss the individual parameters.)

```
Line 1  rule:
Line 2  dup_toner_low:
Line 3    (
Line 4    event: _event of_class
Line 5                  within ['Printer_Paper_Out',
Line 6                          'Printer_Toner_Low',
Line 7                          'Printer_Offline',
Line 8                          'Printer_Output_Full',
Line 9                          'Printer_Paper_Jam',
Line 10                         'Printer_Door_Open'],
Line 11   reception_action:
Line 12     (
Line 13     first_duplicate( _event,
Line 14             event: _dup_toner_ev
Line 15                 where [
Line 16                     status: outside ['CLOSED']
Line 17                     ],
Line 18           _event - 600 - 600
Line 19          ),
Line 20     commit_rule,
Line 21     add_to_repeat_count(_dup_toner_ev, 1),
Line 22     drop_received_event
Line 23   )
Line 24 ).
```

Let's look at the rule line by line.

Line 1 `rule:` tells us that this rule is eligible to be matched every time the rules engine processes its active rules, as opposed to timer rules and change rules, which are eligible only under certain conditions.

Line 2 `dup_toner_low:` is simply a name given to the rule for purposes of documentation and to make trace output more meaningful.

Line 4 `event: _event of_class` begins the identification of the event and the conditions for which this rule can be triggered. If the conditions are met, a unique identifier of this event instance will be stored in the variable `_event`.

Lines 5 through 10

```
within ['Printer_Paper_Out',
        'Printer_Toner_Low',
        'Printer_Offline',
        'Printer_Output_Full',
        'Printer_Paper_Jam',
        'Printer_Door_Open'],
```

set the restrictions on what event classes are capable of triggering the actions in this rule. In this example, only leaf event classes (the bottom most classes in the hierarchy, or, in other words, classes that do not branch off into other classes) are listed. Upper level classes may, however, be defined in rules, meaning that all their leaf classes would cause the the rule to be triggered. Remember that only leaf class events cause rules to be triggered.

Line 11 `reception_action:` indicates that the actions described in this section of the rule are to be carried out only in the event that this rule has been triggered by a new, freshly received event (and not, for example, by a redo request). Within the body of this reception_action there are four individual actions to be carried out.

The first action to be carried out in the event of a match on the listed event classes is described in Lines 13 through 19:

```
first_duplicate( _event,
        event: _dup_toner_ev
            where [
                status: outside ['CLOSED']
                ],
       _event - 600 - 600
      ),
```

The action first_duplicate causes a search through the event cache for the most recent duplicate of the newly received event. (Let's call it the reference event.) When we looked at the BAROC definition of the event class Printer_Toner_Low, we saw one slot defined with dup_detect = yes, so we know that a duplicate event will have the same hostname value as the reference event. The additional condition placed on the search in this case in Line 16 is only that the status of the duplicate event may not be CLOSED.

If such a duplicate event is found in the event cache, a unique identifier for that event is stored in the variable _dup_toner_ev as described in Line 14 `event: _dup_toner_ev`.

Line 18 `_event - 600 - 600` further limits the search through the event cache for duplicate events, in that it sets a time limit with reference to the reference event, whose identifier we stored earlier in the variable _event. The first - 600 parameter limits the search to 600 seconds *before* the reference event, and the second - 600 parameter limits the search to 600 seconds *after* the reference event. Given that the reference event in this case is a newly received event (otherwise it wouldn't have triggered a reception_action), you would be justified in asking how another event could be found in cache that might have arrived after the reference event. So Line 18 would more sensibly read:

```
_event - 600 - 0
```

Line 20 `commit_rule,` causes the rules engine to finish the actions in this rule, but not examine any other rules in the current rule set, and continue processing with the next active rule set. This is an efficient thing to do if your rules are grouped according to topic, and you know that no other rules in this rule set can apply to a given event.

Line 21 `add_to_repeat_count(_dup_toner_ev, 1),` is an action which increments by 1 the value of the slot repeat_count in the event whose identifier is given as a parameter. In this case, the variable _dup_toner_ev contains the identifier of the first duplicate found, so that it is the duplicate event whose repeat_count is increased.

Line 22 `drop_received_event` is an action that causes the newly received event to be discarded. This means that the event is not stored in the event cache and is not recorded in the event repository in the TEC database.

In summary, this rule reacts to newly received events of the specified classes. For each such new event, the event cache is searched for a possible duplicate within the last 600 seconds whose status is not CLOSED. If a duplicate is found, its repeat_count is incremented and the new event is discarded. If no appropriate duplicate event is found, none of these actions is carried out.

10.4.2 Correlating Two Events

Remaining with our printer problem from the last example, let's imagine that the toner is refilled, and the printer finally sends an indication that the error has been cleared. Our operator does not wish to see the Printer_Error_Cleared event, but wants all open printer errors of the last hour from the printer in question to be closed automatically. The following rule does that job (again, we have numbered the lines for convenience):

```
Line 1  rule:
Line 2  print_reset:
Line 3    (
Line 4    event: _event of_class 'Printer_Error_Cleared'
Line 5           where [
Line 6                   status: equals 'OPEN' ,
Line 7                   hostname: _hostname
Line 8                ] ,
Line 9    reception_action:
Line 10     (
Line 11     all_instances(event: _prt_ev of_class
Line 12                   within ['Printer_Paper_out',
Line 13                           'Printer_Toner_Low',
Line 14                           'Printer_Offline',
Line 15                           'Printer_Output_Full',
Line 16                           'Printer_Paper_Jam',
Line 17                           'Printer_Door_Open']
Line 18                   where [
Line 19                           hostname: equals _hostname,
Line 20                           status: outside ['CLOSED']
Line 21                         ],
Line 22                   _event - 3600 - 3600 ),
Line 23     change_event_status( _prt_ev, 'CLOSED' ),
Line 24     drop_received_event
Line 25     )
Line 26 ).
```

Line 4 `event: _event of_class 'Printer_Error_Cleared'` identifies the class of event that can cause this rule to be triggered.

Line 6 `status: equals 'OPEN' ,` further limits the events that can trigger this rule to those whose status is OPEN.

Although Line 7 `hostname: _hostname` is syntactically part of the event identification, all that happens in this statement is that the current value of the hostname slot in the event instance that has satisfied the above criteria is stored in the variable _hostname for future reference. Variables defined in the header of a rule may be used in any subsequent actions in the rule.

Line 9 `reception_action:` shows us that what follows will be executed only when triggered by reception of a new event, not as a result of a request for redo analysis.

Line 11 `all_instances(event: _prt_ev of_class` searches the event cache for all instances of events belonging to the classes listed in Lines 12 through 17. In addition, when an event instance is found that fulfills all criteria, this statement causes the unique event identifier to be stored in the variable _prt_ev for future reference within this action. For each such event found, the remaining actions defined in this reception_action body will be carried out. This is like coding loop logic without having to code the loop.

Line 19 `hostname: equals _hostname,` contained within a where clause helps to restrict the events extracted from the event cache to those whose hostname slot contains the same value as was stored in the variable _hostname in the rule header. In this way, we insure that the events from cache that we are about to close are for the same printer that sent the Printer_Error_Cleared event that triggered this rule execution.

With Line 20 `status: outside ['CLOSED']` we limit our effort to events that are still open.

Line 22 `_event - 3600 - 3600 ),` instructs the rules engine to search the event cache for events as old as one hour earlier than the reference event, in this case the Printer_Error_Cleared event. As in the previous example, the parameter limiting the search to one hour later than the reference event doesn't make much sense in a reception_action.

Line 23 `change_event_status( _prt_ev, 'CLOSED' ),` acts upon all events found in the search of the event cache, since it acts upon each event whose unique identifier is stored in the variable _prt_ev. Each such event will have its status changed to CLOSED.

And finally, in Line 24 `drop_received_event`, the Printer_Error_Cleared event that triggered the rule execution is dropped.

In summary, when a printer sends a cleared event, the rule looks for any open problem events from that same printer that are not more than one hour old, closes those problem events, and discards the cleared event.

10.4.3 Setting a Timer

It's all well and good to close old events that have been cleared up, but what do we do with problems that haven't been resolved within a reasonable time period? One answer might be to notify a responsible person by e-mail. (There are many other possibilities, of course, including escalating the problem to a higher severity, sending a pop-up window to a particular administrator, etc.)

Two rules are necessary to accomplish this goal: a rule to set a timer when a problem event arrives, and a rule to send the e-mail when the timer pops. Here is a rule, also taken from log_default.rls, for setting a 90-second timer on printer problems:

```
Line 1  rule:
Line 2  print_assist:
Line 3    (
Line 4    event: _event of_class
Line 5           within ['Printer_Paper_Out',
Line 6                   'Printer_Toner_Low',
Line 7                   'Printer_Offline',
Line 8                   'Printer_Output_Full',
Line 9                   'Printer_Paper_Jam',
Line 10                  'Printer_Door_Open']
Line 11          where [
Line 12                  status: _status equals 'OPEN',
Line 13                  hostname: _hostname,
Line 14                  msg: _msg
Line 15                  ] ,
Line 16   reception_action:
Line 17     (
Line 18      set_timer(_event, 90, '')
Line 19     )
Line 20 ).
```

Lines 4 through 10 identify the event classes that can cause this rule to fire, and in the event of a match, the event instance identifier is stored in the variable _event.

Lines 11 through 15 define further criteria for an event match. Specifically, Line 12 limits this rule to OPEN events, and stores the status of the found event in the variable _status.

In Lines 13 and 14, contents of slots are stored in variables, but these variables are not referenced again in the rule.

Line 18 `set_timer(_event, 90, '')` is the only action taken by this rule, and only on newly received events. A timer is set on the event whose identifier is in the variable _event, meaning the event that satisfied the header of the rule. The timer is set for 90 seconds from the arrival of the event. The third parameter has been left empty, but could have been used to store information to be referenced later in a rule triggered by the popping of the timer. This is the only possibility to communicate between the rule that sets the timer and the rule that responds to the timer pop.

In summary, this rule does nothing more than to set a 90-second timer on all printer error events.

10.4.4 Responding to Expiration of a Timer

If a given printer error is solved within 90 seconds, then we don't need to do anything when the timer pops. However, if the problem is still open after 90 seconds, we want to send an e-mail to the user tec_print.

In preparation for this rule, let's look at the script called TEC_Send_Mail.sh, delivered with the TEC product and installed into the path $BINDIR/TME/TEC/scripts.

```
#!/bin/sh
# This script assumes that sendmail is available and is in
# /usr/lib.
#
PATH=/bin:/usr/bin:/usr/ucb:/usr/lib
export PATH
sendmail -F'T/EC' -t << __EOF__
To: $2
Subject: $1
$3
__EOF_
```

This simple script uses the UNIX sendmail command to send a formatted message to a user. The significant portions of the script for our purposes are the variable fields the script expects from the caller, and our caller is the timer rule that follows.

$1 will be a string describing the subject matter of the mail.

$2 will be the user ID of the recipient of the mail.

$3 will be the body of the mail.

The rule will have to give this information out in the correct order. Here's the rule delivered with the logfile adapter:

```
Line 1  timer_rule:
Line 2  timer_print_assist:
Line 3    (
Line 4    event: _event of_class
Line 5               within ['Printer_Paper_Out',
Line 6                       'Printer_Toner_Low',
Line 7                       'Printer_Offline',
Line 8                       'Printer_Output_Full',
Line 9                       'Printer_Paper_Jam',
Line 10                      'Printer_Door_Open']
Line 11             where [
Line 12                      status: _status equals 'OPEN',
Line 13                      hostname: _hostname,
Line 14                      msg: _msg
Line 15                    ] ,
Line 16   action:
Line 17     (
Line 18     exec_program(_event,
Line 19          'scripts/TEC_Send_Mail.sh',
Line 20          '"T/EC - %s: %s" tec_print "The Printer %s has the \
        following conditions:\n\t%s"',
Line 21          [_status, _msg, _hostname, _msg],
Line 22          'YES')
Line 23     )
Line 24 ).
```

Line 1 `timer_rule:` tells us that this rule can be fired only by the expiration of a timer. (Line 2 is, as usual, the name of the rule.)

In lines 4 through 10, the class of the event whose timer just expired is examined, and if it belongs to one of the six named classes, processing continues.

Line 12 `status: _status equals 'OPEN',` further restricts matching events to those with the status OPEN, and stores this status in the variable _status.

Line 13 `hostname: _hostname,` and Line 14 `msg: _msg` store the two slot values in, respectively, the variables _hostname and _msg for future reference. Remember, these are the slot values from the event whose timer was set 90 seconds ago in another rule, and whose timer just popped.

Line 18 `exec_program(_event,` begins the description of a call to a program or script from the rule. The rules engine expects the program or script to be located in $BINDIR/TME/TEC, which explains the relative path used in Line 19 `'scripts/TEC_Send_Mail.sh',`.

Line 20 describes the parameters to be passed to the script, and Line 21 lists the values to be stuffed into the parameters in Line 20. Let's examine this line by line.

Line 20 `'"T/EC - %s: %s" tec_print "The Printer %s has the \ following conditions:\n\t%s"',` defines the three parameters expected by the script:

`"T/EC - %s: %s"` ($1 is the subject of the mail.)

`tec_print` ($2 is the recipient of the mail.)

`"The Printer %s has the following conditions:\n\t%s"` ($3 is the body of the mail.)

Each percent (%) will be replaced, in order, by the contents of the variables in Line 21.

Line 22 `'YES'`) indicates that the completion status of the script should be reported in the event console window.

In summary, this rule is fired when a timer pops. If the event connected with the timer is a printer error and it is still open, an e-mail is sent to the user tec_print detailing the name of the printer and the error condition.

10.4.5 Reanalyzing an Event

We often make the assumption that events arrive at the event server in the order of their creation. This is not a safe assumption. The nature of an IP network and the fact that events come from various pieces of software at various locations mean that we have to prepare our rules for the possibility that events will arrive out of logical sequence.

For example, let's imagine that a certain printer goes offline when it runs out of paper. Logically, we consider the event Printer_Paper_Out to be the initial event, or the *cause* event. The event Printer_Offline is a consequence of the first event, or an *effect* event. We wish to link these two events and document their cause-effect relationship, so that we can handle them together in the future. We might want, for example, only the cause event to be closed by an operator or a rule, and that any related effect events be automatically closed.

Let's write a rule that covers the first possibility, namely that the cause event Printer_Paper_Out arrives before the effect event Printer_Offline, so when Printer_Offline arrives, we search the event cache for the most recent Printer_Paper_Out event and link the two events.

```
Line 1  rule:
Line 2  prt_offline:
Line 3    (
Line 4    event: _event of_class 'Printer_Offline'
Line 5           where [
Line 6                  origin: _origin
Line 7                ],
Line 8    action:
Line 9    offline_link_nopaper:
Line 10         (
Line 11          first_instance(event:
Line 12                 _prt_ev of_class 'Printer_Paper_Out'
Line 13                 where [
Line 14                        origin: equals _origin,
Line 15                        status: outside ['CLOSED']
Line 16                       ],
Line 17                 _event - 60 - 60),
Line 18          link_effect_to_cause(_event, _prt_ev)
Line 19         )
Line 20   ).
```

The header of the rule, Lines 4 through 7, identifies the class of event, Printer_Offline, that can trigger processing of the rule. There are no further restrictions, but in Line 6 the value of the slot origin is stored in the variable _origin for future reference in the action part of the rule.

Line 8 is coded as `action:` and not `reception_action`. This means that the action is not limited to being triggered by newly received events, but can potentially be triggered by a redo request in some other rule.

Lines 11 and 12 begin a search through the event cache for the most recent instance of an event of class Printer_Paper_Out.

In Line 14 we limit the search to those events whose origins match the origin of the Printer_Offline event that triggered this rule execution.

In Line 15 we limit the search to non-CLOSED events.

In Line 17 we limit the search to Printer_Paper_Out events that arrived within 60 seconds on either side of the Printer_Offline event under analysis.

If we find such a Printer_Paper_Out event that meets all our criteria, we link in Line 18 the Printer_Offline event (whose identifier is in the variable _event) as an effect to the Printer_Paper_Out event (whose identifier is in the variable _prt_ev) as the cause. In practical terms, this means that values in the two slots date_reception and event_handle of the cause event are copied respectively into the two slots cause_date_reception and cause_event_handle of the effect event.

Now let's imagine that Printer_Offline (the effect event) arrives before Printer_Paper_Out (the cause event). At the time Printer_Offline is received, there is no Printer_Paper_Out event in the event cache, so the above rule cannot link any events. When the Printer_Paper_Out event finally does come, we'd like the chance to link the two events as cause and effect, but we don't want to have to rewrite the link (and any other) logic in another rule, just because the order of arrival is backwards. So we write a rule that is triggered by reception of the cause event (Printer_Paper_Out) that looks for any instances of an effect event (Printer_Offline) from the same origin that might be hanging around in the event cache, and if any such effect events are found, they are processed with the action redo_analysis. This causes the rules engine to start from the beginning of the rules for the event being processed to look for any applicable rules, and our previous rule prt_offline will be reexecuted.

Here is what the redo rule might look like:

```
Line 1  rule:
Line 2  printer_paper_out:
Line 3    (
Line 4     event: _event of_class 'Printer_Paper_Out'
Line 5            where [
Line 6                   origin: _origin
Line 7                  ],
Line 8     reception_action:
Line 9     (
Line 10     first_instance(event:
Line 11           _prt_ev of_class 'Printer_Offline'
Line 12           where [
Line 13                  origin: equals _origin
Line 14                 ],
Line 15           _event - 60 - 0 ),
Line 16     redo_analysis(_prt_ev)
Line 17    )
Line 18   ).
```

Receipt of a new event of class Printer_Paper_Out triggers this rule. In Line 6 the contents of the origin slot are stored in the variable _origin.

In Line 8 we see a reception_action, meaning that what follows is executed only upon receipt of a fresh new event.

In Lines 10 through 15 we search the event cache for an instance of Printer_Offline with the same origin as the received Printer_Paper_Out event and not older than 60 seconds.

If such a Printer_Offline event is found in cache, then the action redo_analysis is carried out for that event. Thus the rules engine reevaluates from the beginning whether any rules are applicable to the Printer_Offline event.

10.4.6 Reacting to a Changed Event

Now that we have two events linked as cause and effect, we can put that relationship to work to save our operators work. If an operator acknowledges a cause event, we would like the status of the effect event to change to ACK also. If a cause event is automatically closed by a rule that uses the change template change_event_status when the solution has been found (when a *clear* event arrives), then we want the effect event to be closed automatically also. (The other templates that can trigger change rules are change_event_administrator and change_event_severity.) We can accomplish this automation with a change_rule:

```
Line 1  change_rule:
Line 2  chg_stat_Printer_Offline:
Line 3    (
Line 4     event: _event of_class 'Printer_Paper_Out'
Line 5            where [
Line 6                   event_handle: equals _ev_handle,
Line 7                   date_reception: equals _date_rec
Line 8                  ],
Line 9     slot: status set_to _status,
Line 10    action:
Line 11    chg_prt_cause_effect:
Line 12      (
Line 13       all_instances(event:
Line 14          _prt_offline of_class 'Printer_Offline'
Line 15          where [
Line 16                 cause_event_handle: equals _ev_handle,
Line 17                 cause_date_reception: equals _date_rec
Line 18                ],
Line 19          _event - 60 - 60 ),
Line 20       set_event_status(_prt_offline, _status),
Line 21   )
Line 22 ).
```

Line 1 `change_rule:` indicates that this rule is triggered only when the contents of a previously received event are changed.

We then specify in Line 4 which type of changed event causes this rule to continue execution. In this scenario it is the cause event Printer_Paper_Out.

In Lines 6 and 7 we are not further restricting the event criteria, but rather storing the two slot values in variables that are necessary for a unique identification of the event, whose corollaries we expect to see in any linked event.

In Line 9 we use a slot change filter to determine that is was the status slot that was changed, and to store the changed status value in the variable _status for future reference.

If all criteria in the change_rule header have been met, then we instigate in Line 13 a search through event cache for all instances within a range of 60 seconds from the changed event of Printer_Offline (effect) events that are linked through the two cause slots to the changed event.

When such linked effect events are found, their status is automatically set in Line 20 to the value of the changed status of the cause event Printer_Paper_Out.

10.5 Advanced Techniques

There are three utility templates in the rules language that enable you to manipulate strings, known as *atoms*. These are shown in the following table.

Table 6. Templates for Atom Manipulation

Template Name	Function
atomconcat	Concatenates the values in a list of arguments to one value
atomlength	Returns the length of a string
atompart	Extracts a substring from a string

We wrote our own template to extract the first three positions of a hostname, knowing for a given customer that the location code was to be found in those positions. We used the utility atompart to extract a substring. (We've numbered the lines to be able to discuss the logic more easily.)

```
Line 1  rule:
Line 2  boot_get_pref:
Line 3  (event: _start of_class within ['TEC_Start']
Line 4          where [],
Line 5    reception_action:
Line 6    (
Line 7     assert((get_ctc_pref(_ctc_name, _ctc_pref) :-
Line 8        atompart(_ctc_name, _ctc_pref, 1, 3)
Line 9        ))
Line 10 )).
```

This rule will be called when the TEC_Start event is received, as defined in Line 3, and will define the template get_ctc_pref in Line 7, which will then be available for use by other rules.

The atompart statement in Line 8 takes the contents of the variable _ctc_name, starting in position 1, for a length of 3, and stores the result in the variable _ctc_pref. Since this algorithm is defined in our template, a rule that calls the template get_ctc_pref need supply only two variables, as described in Line 7. The first three characters of the first variable will be stored in the second variable, as seen for example in this excerpt from a rule.

```
action:
      get_ctc_pref(_hostname, _host_pref)
```

Now with a basic understanding of rules we can begin to develop rules based on the output of the methodology covered in Chapter 2, “Planning for the TEC Installation” on page 11.

Chapter 11. Rule Development

This chapter illustrates the steps involved in actually coding rules. We used output from a thorough event analysis, in our case the IBM Event Management Methodology, as input to rule development.

The input we required for each set of events to be correlated and managed consisted of:

- An event relationship diagram
- The baroc class definition for each event, in order to know which slots are available for correlation and for duplicate detection
- Policy decisions made by the enterprise's subject matter experts
- A flowchart depicting the rule logic
- A test plan

The examples covered in this chapter deal with events from various sources. These sources were chosen to illustrate various coding techniques, such as:

- Closing a dependent event automatically, when its causal event is closed
- Searching the event cache for related events within a given time range
- Setting a timer to escalate the severity of an event if it is not closed within a reasonable time period
- Eliminating duplicate events, but keeping a record of the time of the latest duplicate event and the number of duplicate events
- Allowing for the possibility in a network that a *clear* event may actually arrive at the event server before its related *problem* event

The logic used in these examples can be applied to events from any source.

Our examples cover:

- Power loss events generated by an APC UPS and a Cisco router
- Beaconing events generated by a Cabletron hub on a token-ring
- Fan failing events generated by a Cabletron hub

11.1 Development Standards

To define a common look and feel for our baroc and rules files, we created names according to a naming convention. This was very useful when we began debugging the rules via the rules trace mechanism, as it enabled us to track the specific rules and rule sets being processed.

11.1.1 The Baroc Files

We created a new event class definition for each type of trap. Although this was time consuming initially, there are coding and performance benefits of being able to use specialized classes in rule development. When the rules engine searches the event cache, it is more efficient to weed out inapplicable rules by virtue of no matching event class, rather than performing comparisons on lists of slots.

All event class names follow the convention of beginning with a capital letter and being separated by underscores, for example: `Beacon_Recovery_Port_Removed`.

The slot values contained in the event class definition files use names that reference the specific slot contents. In defining event classes for SNMP traps we used the MIB argument definition from the MIB file. For non-SNMP sources we used a standard naming policy.

The baroc files are named according to the source of the events. For instance, the Cabletron hub event definitions are called ctccable.baroc.

11.1.2 The Rules Files

We wrote one rule set for each event requiring rules, or in some cases, for two related events that are handled similarly. So, for example, the rules for the Beacon_State event generated by the Cabletron hub are found in ctcbeacst.rls, and the rules for the Beacon_State_Clear event are in ctcbeacclr.rls.

The rules files have extensive comments describing the rule logic and indicating the event being handled.

All actions within a rule have action names that describe the logic covered in that action, for example:

```
'Beacon_Related_Propagate_Status_To_Linked_Beacon_State'
```

Note: If you use a capital letter to begin a rule name or an action name, enclose it in single quotes (for example, `'Env_Volt_Low'`) to avoid the name being interpreted as a variable.

11.1.3 Directory Structures

All our definitions and rules were developed in a development directory, separate from all TEC directories. We used:

- /usr/development/rules
- /usr/development/classes

The rule base we defined in Chapter 4, “Software Installation” on page 49 is called CanTire. We created a shell script to import and compile the new class definitions, and one to import and compile new rule sets.

Note: It is a good idea to separate your import and compile commands from your wloadrb command in any scripts you write to automate your testing, since you should attempt to load only a successfully compiled rule base.

11.1.4 Rule Development Policies

When defining rules, we tried to:

- Use the commit control primitives (commit_action, commit_rule, commit_set) to guide the execution flow of the rule engine. This necessitates putting your rules in a logical sequence, so that you can skip unnecessary actions, rules, or rule sets.
- Use a first_ template instead of an all_ template, if it meets your needs just as well. The first_instance and first_duplicate templates will search the event cache for only the most recent occurrence of this event, while the all_instances

and all_duplicates templates will search the event cache for all occurrences of the named event (within the given time range, if used).

- Put the most restrictive condition first in a where clause. This filtering compares the slot contents with a value, using operators such as equals, outside, or within. The sooner the rule engine can eliminate unsuitable event candidates, the sooner it is free to handle the next event.

11.1.5 General Testing Guidelines

In order to be debug rules effectively, we compiled our rules with the trace option:

```
wcomprule -t <rulebasename>
```

We can also turn tracing on for the event server:

```
wsetesvrcfg -t /tmp/rules.trace
```

By default, the trace is written to /tmp/rules.trace, but you may choose another file. Between tests, we cleared the trace file with:

```
cat /dev/null > /tmp/rules.trace
```

(Deleting the file between tests proved problematic, as the event server reallocated the file only the next time it was started, and it didn't write to a file we allocated with the touch command.)

For debugging, we found it useful to open a UNIX terminal and issue the command:

```
tail -f /tmp/rules.trace
```

Prior to beginning testing by sending events, we closed the TEC console, issued wtdbclear -el -t0 to clear all events from both the reception log and the event repository, and then re-opened the event console. We then began sending test events.

If the events did not appear as expected on the event console, we used wtdumprl -o DESC to find out whether the event had been received at all by the event server.

The rule sets were loaded individually and tested, and then when each rule produced the correct outcome, we added the rule set to those already tested, and re-ran the tests.

For testing purposes we created a number of scripts to send both SNMP traps and TEC events. An SNMP script sample is shown below:

```
/tmp/traps
```

For example, when we send the SNMP trap from a device we use the snmptrap command as shown below:

```
/usr/OV/bin/snmptrap rs600027 .1.3.6.1.4.1.52 rs600027 6 282 0 \
1.1 octetstring "arg1"
```

These scripts used the snmptrap command from TME 10 NetView for AIX. This simulates an SNMP trap, which NetView then forwards to the TEC event server for correlation.

We allocated a directory called tests in the same development path /usr/development in which we stored all our rules and classes. Here we evolved testing scripts for sending a series of TEC events in various sequences and timing relationships to give our rules a workout. Sample scripts are reproduced in the descriptions of the three individual correlation cases that follow.

11.1.6 Miscellaneous Rules Errors

We discovered that rules often compiled successfully, although there were spelling errors. For example, an occurrence of all_instance within one of our rules (which should have been all_instances) compiled correctly, but the rule failed on execution. We saw the mistake only in the trace file.

In testing, we learned that using wtdbclear -el -t0 to clear the reception log and the event repository did not cause the event cache to be cleared. So, for example, our link_effect_to_cause logic sometimes linked an effect event to an older cause event that we were no longer able to see on our console and believed to be deleted. The solution was to stop and restart the event server, so that the event cache was rebuilt from the events truly remaining in the TEC database.

11.1.7 Trouble Ticket Integration

One of the main reasons for developing the rules was to limit the number of trouble tickets being created. To simulate the trouble ticket creation we decided to send an event to the TEC reflecting that a trouble ticket action has been performed. The shell scripts we created are shown below:

```
#!/bin/ksh
#ctc_ttproc.sh

/usr/local/Tivoli/bin/aix4-r1/bin/wpostemsg -m
"Trouble Ticket Created" -r FATAL  origin=$1 msg="$2" Trouble_Ticket nvserverd
```

```
#!/bin/ksh
#ctc_ttproc_upd.sh

/usr/local/Tivoli/bin/aix4-r1/bin/wpostemsg -m "Trouble Ticket Updated"
 -r HARMLESS origin="$1" msg="$2" Trouble_Ticket nvserverd
```

If there is a requirement to link with a trouble ticket application then these scripts will be replaced with the relevant trouble ticket creation and update scripts for the application.

11.2 Power Supply Example

An uninterrupted power supply (UPS) is a battery-generated electrical supply that switches on automatically in the event of a failure of the usual source of power, used to insure the continuous availability of critical network resources.

In this example we show the sequence of events when the UPS from the company APC switches on, when its battery starts losing energy, and when its battery is

entirely depleted, leading to the shutdown of a Cisco router that was supplied by the APC UPS.

The APC PowerNet SNMP agent running on the UPS reports to to the network management product (in our case TME 10 NetView) by sending SNMP traps. We customized these traps in NetView such that the NetView integrated event adapter sends a unique event to the TEC event server for each possible APC trap.

11.2.1 The Events

The events used in this example are described in the Table 7.

Table 7. UPS Events

Original SNMP Trap Number	TEC Event Class Name	Event Description
1	Ups_Power_Fail_Backup_On	The power supply has failed; UPS is supplying backup power.
7	Ups_Battery_Low	The UPS battery is low and may not support load.
4	Ups_Battery_Discharged	The UPS battery is entirely depleted.
9	Ups_Power_Restored	The normal power supply has been restored; UPS no longer supplying power.
11	Ups_Battery_No_Longer_Low	The UPS battery is no longer depleted.

This exercise correlates the worsening sequence of UPS events to, ultimately, the shutting down of a Cisco router powered by the UPS. The router is capable of sending a shutdown notification to NetView before dying, in the event that its environmental monitor detects a loss of power. Once again, we defined to NetView that this trap should be converted into a TEC event and forwarded to the event server.

The Cisco-generated event is described in Table 8.

Table 8. Cisco Events

Original SNMP Trap Number	TEC Event Class Name	Event Description
1	Cisco_Shutdown_Notification	The environmental monitor has detected a critical state and is about to initiate a shutdown.

11.2.2 The Event Relationships

The event relationships for the UPS events have been divided up into three categories:

- One cause event
- Three effect events
- Two clearing events

The most critical event is the cause event. To handle this we must check for existing events that where a direct effect of the UPS power fail event.

The clear events will basically clear up both the cause and the effect events.

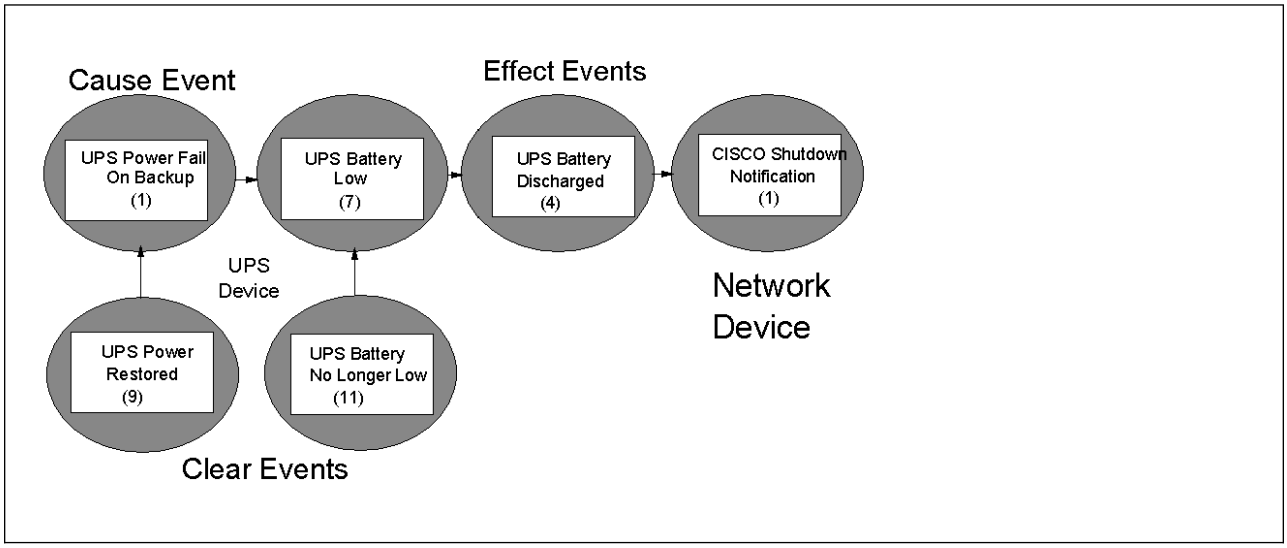


Figure 133. Event Relationships

The relationships among the events in this example are depicted in Figure 133.

Based on experience, we are making the assumption that the event Ups_Power_Fail_Backup_On will always precede Ups_Battery_Low, and that Ups_Battery_Low will precede Ups_Battery_Discharged, before we receive notice that the router is shutting down. Therefore we are considering the event telling us that the UPS is supplying power to be the primary and causal event, and the others, although carrying critical information, to be effects of the original causal event.

However, the UPS battery running low is logically an additional problem to the original loss of power problem, and consequently has its own clear event. Thus Ups_Battery_No_Longer_Low clears the Ups_Battery_Low follow-on problem, and Ups_Power_Restored clears the original power failure problem.

If we were to handle the Cisco_Shutdown_Notification event fully, we would have to discuss many more possible event interrelationships, as there are many possible causes of a router shutting down. For purposes of this example, however, we are considering the Cisco_Shutdown_Notification event only as an effect event of a loss of power to the router.

11.2.3 The Event Class Definitions

The class definition file we developed is shown in Figure 134 on page 241.

```
#########################################################
# Class Definitions for the APC UPS                     #
#########################################################
TEC_CLASS:
        Ups_Event ISA Nvserverd_Event
        DEFINES {
                sub_source: default = "UPS";
                origin: dup_detect = yes;
                severity: default = WARNING;
                last_occur: STRING;
                location: STRING;
                tt: STRING;
                };
END
# Trap 1
TEC_CLASS :
        Ups_Power_Fail_Backup_On ISA Ups_Event
        DEFINES {
                severity: default = CRITICAL;
                };
END

# Trap 9
TEC_CLASS :
        Ups_Power_Restored ISA Ups_Event
        DEFINES {
                severity: default = HARMLESS;
                };
END
# Trap 7
TEC_CLASS :
        Ups_Battery_Low ISA Ups_Event
        DEFINES {
                severity: default = MINOR;
                };

END
# Trap 11
TEC_CLASS :
        Ups_Battery_No_Longer_Low ISA Ups_Event
        DEFINES {
                severity: default = HARMLESS;
                };
END
# Trap 4
TEC_CLASS :
        Ups_Battery_Discharged ISA Ups_Event
        DEFINES {
                severity: default = FATAL;
                };
END
```

Figure 134. The UPS Class Definitions File (ctcups.baroc)

The class definition file was derived from the NetView trap definition file that was shipped with the UPS software.

The baroc class definitions we coded for the Cisco shutdown notification from the router and are shown below:

```
#####################################################################
# Class Definitions for Cisco Routers                               #
#####################################################################
TEC_CLASS:
        Cisco_Event ISA Nvserverd_Event
        DEFINES {
                sub_source: default = "Cisco";
                origin: dup_detect = yes;
                severity: default = WARNING;
                last_occur: STRING;
                location: STRING;
                tt: STRING;
                };
END

# Enterprise Specific Trap 1
TEC_CLASS:
        Cisco_Shutdown_Notification ISA Cisco_Event;
END
```

Figure 135. The Cisco Class Definitions File (ctccisco.baroc)

Here are a few points to notice about these event definitions.

- Our UPS superclass Ups_Event is defined as a child of Nvserverd_Event, the superclass for all events coming from the NetView integrated event adapter.
- All UPS events inherit the slot last_occur, which we use in conjunction with checking for duplicate events. If a duplicate event arrives, and we choose not to display that event, we do, however, insert the date and time of the duplicate event into the slot last_occur of the currently displayed event, so that there is a record of the arrival of the duplicate event.
- All UPS events inherit the slot tt, which we use to indicate whether the event has opened a trouble ticket (tt=1) or whether the event has updated an existing trouble ticket (tt=2).
- The class Trouble_Ticket serves testing purposes only. Since this project did not include interfacing to a specific trouble ticketing system, we tested our rule logic by calling scripts to open and update a trouble ticket, which do nothing more than to send an event of class Trouble_Ticket.
- What came to our attention in examining these event class definitions was that we had no slots particular to the UPS classes that would help us correlate the events to a specific network resource, in this case to a specific Cisco router connected to the UPS.

  We decided to add a slot called location and fill it with the information defined at the SNMP agents in the MIB variable SysLocation. We were able to rely on the contents of this slot being the same for devices at a particular physical location, because this customer has a location naming convention that is strictly defined and enforced.

  When planning for the correlation of events from multiple sources, as in this example, analyze your environment, your addressing scheme, and your naming conventions, to make sure you have reliable identifying information available across event sources.

11.2.4 The Event Policy

Understanding the relationships among related events is only the beginning. Many additional decisions are necessary before you can code rules that meet the needs of your enterprise. Here are the customer policy decisions that affected our logic for the UPS rules:

- The Ups_Power_Fail_Backup_On will be considered a duplicate event if it arrives within eight hours of an open event of the same class from the same IP address. In this way, if the power should fluctuate during the course of a whole day, a given operator on a given shift would not be bothered with multiple open events for the same problem, but the operator on the next shift would be reminded of the problem at some point.
- Ups_Power_Fail_Backup_On (when not a duplicate event) will cause a trouble ticket to be opened.
- Ups_Battery_Low and Ups_Battery_Discharged will update the trouble ticket opened by the causal event Ups_Power_Fail_Backup_On, as will the Cisco_Shutdown_Notification event.
- Ups_Battery_Low and Ups_Battery_Discharged will be linked to each other in the classical cause-effect relationship, so that Ups_Battery_Discharged will be automatically closed when Ups_Battery_Low is closed by its clear event.
- Ups_Power_Fail_Backup_On will be closed by receipt of the event Ups_Power_Restored, but will not cause any of the related events to be cleared, since Ups_Battery_Low is actually a problem in its own right.
- Cisco_Shutdown_Notification will not be automatically closed by closure of any of the other related events, because a dead router may require additional operator intervention. From the point of view of the event server, this event will be closed only by an operator or by closure of the related trouble ticket.
- The rules should allow for the reasonable possibility, due to network delivery timing inconsistencies, that a problem event might be preceded by a related clear event by up to 1 minute.
- Because Ups_Battery_Low is a problem we want someone to solve *before* it comes to a case of Ups_Battery_Discharged, we give this event special emphasis by defining an escalation policy. Ups_Battery_Low is normally issued with a severity of MINOR. If this event is not closed within one hour, it should automatically be escalated to a status of FATAL.

Note:

Using a timer set for one hour (or some other time period) to check whether an event is still open, and if so, instigate some escalation activity, is a perfectly acceptable way to deal with follow-up on important events. You should not, however, simply start a timer on every event arriving at the event server, in order to check the status at a later time. As you are limited to a maximum of 1000 timers at any given time, a few of which are used by the TEC itself, such a policy could quickly cease to function. Instead, use this mechanism for only your most time-critical events.

11.2.5 Rule Information

We developed the flowcharts from the event relationship diagrams. In order to understand the rule and event relationships we recommend that when reading the rules that the relationship rules are also view concurrently as there are a number of cross references in the flowcharts.

Directly after each flowchart we have described the rule flow, then we have included the actual rule code. The related flowcharts are listed in the grey boxes at the top of the flowchart.

The following information is contained in this example:

- For the cause event (UPS_Power_Fail_On_Backup)
  - ctcups.baroc file
  - ctccisco.baroc file
  - ctcupc.rls flowchart
  - ctcups.rls rule code
- For the effect events
  - ctcupsbat.rls rule
- For the clear events
  - ctcupsbatclr.rls flowchart
  - ctcupsbatclr.rls rule
  - ctcupsclr.rls rule

We did not include the flowchart for the effect events, however the flowchart shown in Figure 148 on page 268 contains similar flow.

11.2.6 The Cause Event

Figure 136 on page 245 shows the flowchart that depicts the rule logic for the UPS_Power_Fail_On_Backup event. This event as shown in Figure 133 on page 240.

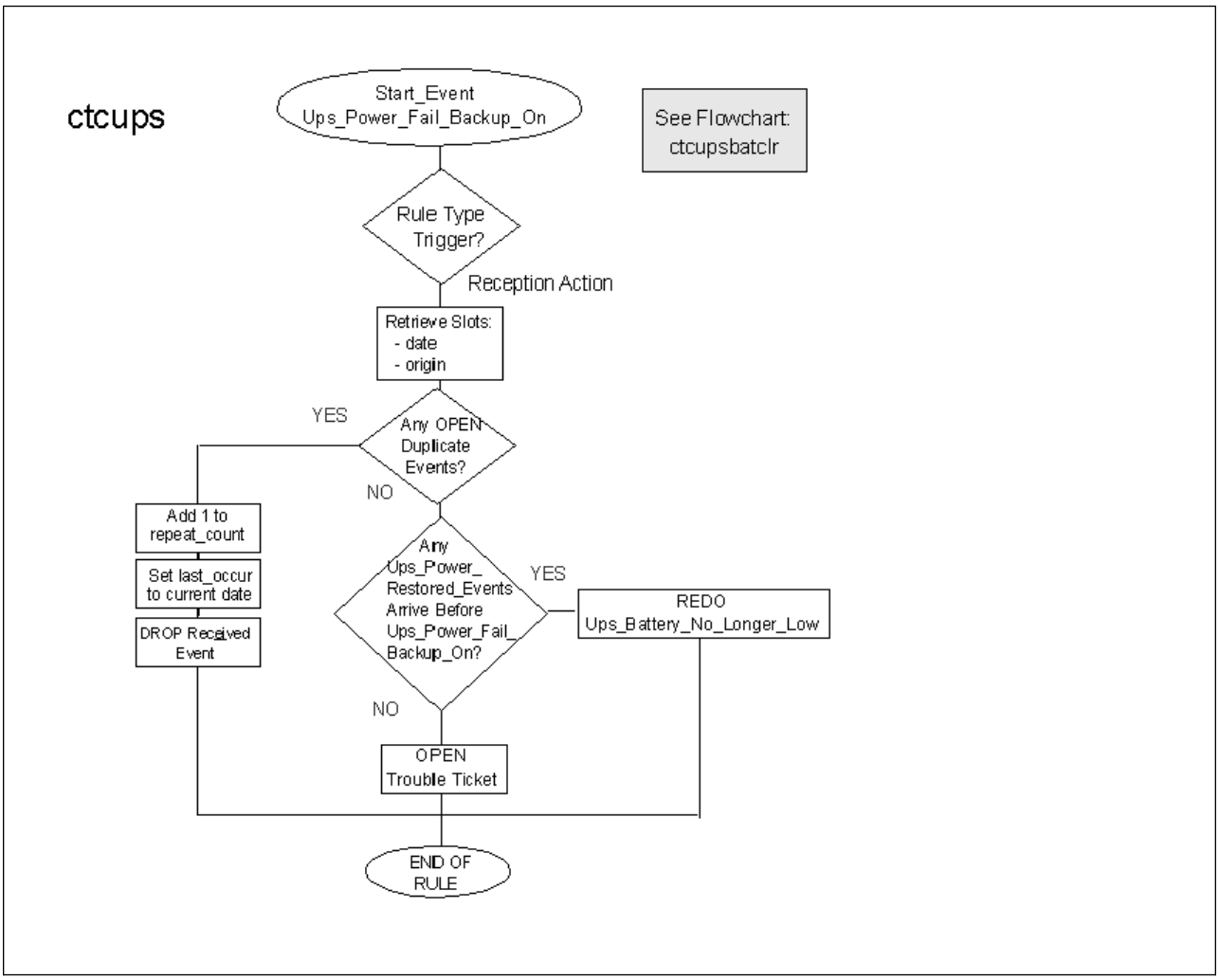


Figure 136. The Rule ctcups.rls

We can see from Figure 136 that the rule is activated on reception of the event UPS_Power_Fail_Backup_On. The flowchart logic is outlined below:

- On reception of the event assign slot origin and date
- Check for any duplicate events
- If a duplicate event is found, then:
  - Add 1 to the repeat count variable for that event.
  - Assign the current date to the last_occured variable for the event.
  - Drop the received event.
  - End the rule.
- If no duplicate events are found, then:
  - Check to see if there are any Power_Restored events arrive before the power fail event.
  - If the event is found then:
    - Re-do the analysis for the Ups_battery_no_longer_low rule.
    - End rule processing.
- If no Power_restored event is found, then:
  - Open a new trouble ticket.

– End rule processing.

The reason we checked for the clear event was in case the clear had arrived before the Ups_powerfail_backup_on event event due to, for instance, a network delay.

The rule that was developed from the flowchart shown in Figure 137.

```
/*   */
/* ctcups.rls         */

rule:
'Ups_Power_Fail_Backup_On':
(
   event: _event of_class 'Ups_Power_Fail_Backup_On'
  where [
   origin: _origin,
   date: _date
                     ],


reception_action:
'Ups_Power_Fail_Elim_Dup_Events':
  (
 first_duplicate(_event,
              event: _dup_ev
   where [
    status: outside ['CLOSED']
                 ],
   _event - 28800 - 0),

 add_to_repeat_count(_dup_ev,1),
        place_change_request(_dup_ev,last_occur,_date),
 drop_received_event,
 commit_rule
    ),

reception_action:
'Ups_Power_Fail_Check_For_Recent_Clear':
  (
 first_instance(event: _clr_ev of_class 'Ups_Power_Restored'
  where [
   origin: equals _origin,
   status: outside ['CLOSED']
        ],
  _event - 60 - 0),
 redo_analysis(_clr_ev)
  ),

reception_action:
'Ups_Power_Fail_Raise_TT':
 (
 exec_program(_event,'/usr/development/rules/ctc_ttproc.sh','-s "%s"',
       [_origin],'NO'), place_change_request(_event,tt,'1')
   )
).
```

Figure 137. The ctcups.rls Rule

We used the re-do analysis for this rule. We found this process reduced the amount of actual rule code required.

11.2.7 The Effect Events

The effect events where grouped together. The example rule below shows performs additional functions based on the rule policy. The additional functions include:

- Adding a one hour escalation policy
- Modifying the status of the event

The main flow of this rule is detailed below:

- Rule is activated when either a Ups_Battery_Low or a Ups_Battery_Discharged event is received.
- Check for any duplicates.
- If any duplicates are found, then:
  - Add 1 to the repeat count for the event.
  - Place a change request for the event.
  - Drop the event from the cache.
  - Stop rule processing.
- If no duplicates are found, then:
  - Check for the event Ups_Battery_No_Longer_Low.
  - If found, then:
    - Re-do the Clear analysis. This will clear up the events.
    - End rule processing.
  - If not found, then:
    - Check if a trouble ticket was created with the event Ups_Power_Fail_Backup_on.
    - If found, then:
      - Update the existing trouble ticket.
      - Place a change request.

  This rule file also contains a timer rule:

  - On Reception of a Ups_Battery_Low event.
  - Set a timer and wait for a clear event.
  - Once the timer has expired and the event still exists then change the severity of the event to FATAL.

  The change rule when activated will:

  - Set a new event status.
  - Escalate the problem by assigning a new administrator.

The rule is shown in Figure 138 on page 248.

```
/* handling of UPS battery events          */
/* ctcupsbat.rls                           */

rule:
'Ups_Battery_Low_Or_Discharged':
(
   event: _event of_class within
  [
   'Ups_Battery_Low',
   'Ups_Battery_Discharged'
  “
  where [
   origin: _origin,
   date: _date
                      ],

reception_action:
'Ups_Battery_Elim_Dup_Events':
  (
 first_duplicate(_event,
              event: _dup_ev
   where [
    status: outside ['CLOSED']
                 ],
   _event - 28800 - 0),

 add_to_repeat_count(_dup_ev,1),
        place_change_request(_dup_ev,last_occur,_date),
 drop_received_event,
 commit_action
    ),


reception_action:
'Ups_Battery_Check_For_Recent_Clear':
  (
 first_instance(event: _clr_ev of_class 'Ups_Battery_No_Longer_Low'
  where [
   origin: equals _origin,
   status: outside ['CLOSED']
        ],
  _event - 60 - 0),
 redo_analysis(_clr_ev),
 commit_rule
  ),
```

Figure 138 (Part 1 of 4). The Rule File ctcupsbat.rls

```
reception_action:
'Ups_Battery_Check_For_Power_Fail':
   (
 first_instance(event: _pow_ev of_class 'Ups_Power_Fail_Backup_On'
  where [
   status: outside ['CLOSED'],
   tt: equals '1',
   origin: equals _origin
                ],
  _event -86400 - 0),
  exec_program(_pow_ev,'/usr/development/rules/ctc_ttproc_upd.sh',
'-s "%s"',[_origin],'NO'),
  place_change_request(_event,tt,'2')
 )
  ).

rule:
'Ups_Battery_Low':
(
   event: _event of_class 'Ups_Battery_Low',
reception_action:
  (
 set_timer(_event, 3600, '')
  )
).

rule:
'Ups_Battery_Discharged':
(
   event: _event of_class 'Ups_Battery_Discharged'
  where [
   origin: _origin
                      ],
reception_action:
'Ups_Battery_Discharged_Link_To_Low':
   (
 first_instance(event: _low_ev of_class 'Ups_Battery_Low'
  where [
   origin: equals _origin
        ],
  _event -86400 - 0),
 link_effect_to_cause(_event,_low_ev)
   )
).
```

Figure 138 (Part 2 of 4). The Rule File ctcupsbat.rls

```
change_rule:
'Ups_Battery_Low_Propagate_Status_To_Linked_Discharged':
   (
    event: _event of_class 'Ups_Battery_Low'
 where [
  date_reception: _date_rec,
  event_handle: _ev_handle,
  administrator: _admin
       ],
       slot: status set_to _status
         within ['ACK', 'CLOSED'],
action:
   (
    first_instance(event: _effect_ev of_class 'Ups_Battery_Discharged'
 where [
  status: outside ['CLOSED'],
  cause_date_reception: equals _date_rec,
  cause_event_handle: equals _ev_handle
       ],
       _event - 0 - 86400),
 set_event_status(_effect_ev, _status),
 set_event_administrator(_effect_ev, _admin)
   )
).
```

Figure 138 (Part 3 of 4). The Rule File ctcupsbat.rls

```
timer_rule:
'Ups_Battery_Low_One_Hour_Escalation':
(
  event: _event of_class 'Ups_Battery_Low'
 where ]
  status: outside ]'CLOSED'[
       ],
action:
  (
 set_event_severity(_event, 'FATAL')
  )
).
```

Figure 138 (Part 4 of 4). The Rule File ctcupsbat.rls

11.2.8 The Clear Events

The flowchart for the clear events are very similar. We have included the flowchart for ctcupsbatclr.rls (see Figure 139 on page 251).

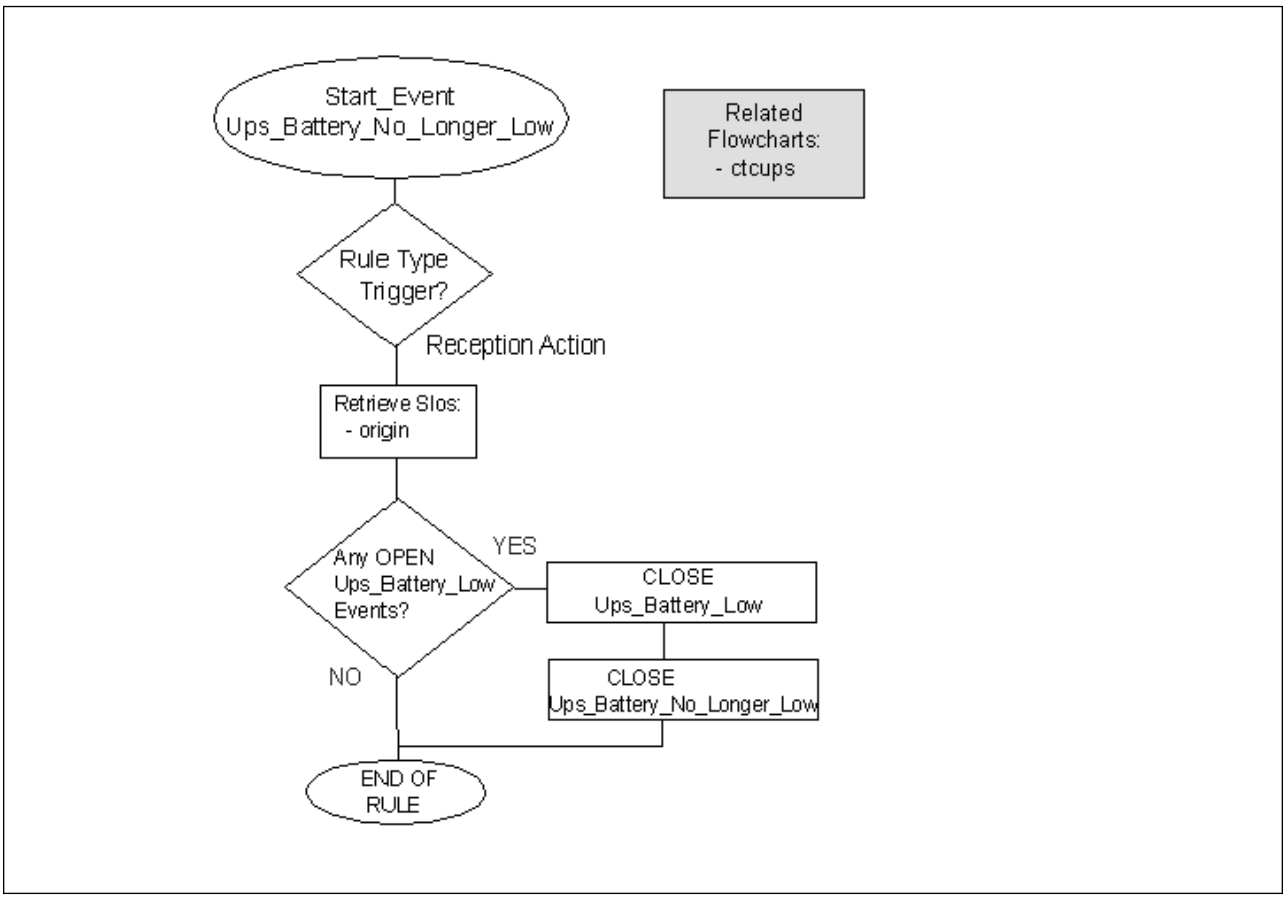


Figure 139. The Flowchart for ctcupsbatclr.rls

On reception of the clear event Ups_Battery_no_Longer_Low we perform the following functions:

- Retrieve the slot value for origin for the received event.
- Check for any event with the class type of Ups_Battery_Low from the same origin.
- If found, then:
  - Close the Ups_Battery_Low event.
  - Close the Ups_Battery_No_Longer_Low.
  - End rule processing.
- If not found, then:
- End rule processing.

The rule for the clear event is shown below:

```
/* Ups_Battery_Low cleared by Ups_Battery_No_Longer_Low */
rule:
'Ups_Battery_No_Longer_Low':
 (
 event: _event of_class 'Ups_Battery_No_Longer_Low'
 where [
  origin: _origin
       ],

action:
'Ups_Battery_No_Longer_Low_Close_Open_Bat_Low':
  (
 first_instance(event: _ups_ev of_class 'Ups_Battery_Low'
  where [
   origin: equals _origin,
   status: outside ['CLOSED']
        ],

  _event - 86400 - 60),

 change_event_status(_ups_ev,'CLOSED'),
        set_event_status(_event,'CLOSED')
 )
).
```

Figure 140. The Rule for ctcupsbatclr.rls

The ctcupsclr.rls is shown in Figure 141.

```
/* Ups_Power_Fail_Backup_On cleared by Ups_Power_Restored */
rule:
'Ups_Power_Restored':
 (
 event: _event of_class 'Ups_Power_Restored'
 where [
  origin: _origin
       [,

action:
'Ups_Power_Restored_Close_Open_Power_Fail':
  (
 first_instance(event: _ups_ev of_class 'Ups_Power_Fail_Backup_On'
  where [
   origin: equals _origin,
   status: outside ['CLOSED']
        ],

  _event - 86400 - 60),

 set_event_status(_ups_ev,'CLOSED'),
        set_event_status(_event,'CLOSED')
 )
).
```

Figure 141. The Rule for ctcupsclr.rls

11.2.9 Testing the Rules

Once we had developed the rules we imported them into the current rule base, compiled them and re-started the event server.

We tested that NetView was sending the events properly by issuing each of the SNMP traps and checking the events that arrived at the TEC event server. If you need to simulate the trap sending, then you can use the TME 10 NetView command snmptrap.

Once we knew that the NetView definitions for restructuring the events were working, we confined our further testing to sending events directly to the event server with the wpostemsg command, since this phase concerns testing the rules logic.

We tested each possible sequence of events and each rule with individual commands, looking in /tmp/rules.trace along the way to correct any logic errors. We used our policy decisions as a checklist for the testing scenarios, reasoning that if the results conform to our established policy, then the rules are doing what they are supposed to do.

We then combined the various tests into one script that issues the events in usual and unusual sequences.

For testing this example we used a script that generated a set of events in a specific sequence.

The test script we used is shown in Figure 142 on page 254.

```
#!/bin/ksh
#############################################################################
echo "Power_Fail followed by Power_Restored for same origin"
wpostemsg -m "test 1" sub_source=UPS origin=1.1.1.1
     Ups_Power_Fail_Backup_On nvserverd
sleep 5
wpostemsg -m "test 1" sub_source=UPS origin=1.1.1.1
     Ups_Power_Restored nvserverd
echo "Enter for next test..."
read

#############################################################################
echo "Power_Fail followed by Power_Restored for different origin"
wpostemsg -m "test 2" sub_source=UPS origin=2.2.2.2
     Ups_Power_Fail_Backup_On nvserverd
sleep 5
wpostemsg -m "test 2" sub_source=UPS origin=2.1.2.1
     Ups_Power_Restored nvserverd
echo "Enter for next test..."
read

#############################################################################
echo "Power_Restored arriving before Power_Fail"
wpostemsg -m "test 3" sub_source=UPS origin=3.3.3.3
     Ups_Power_Restored nvserverd
sleep 5
wpostemsg -m "test 3" sub_source=UPS origin=3.3.3.3
     Ups_Power_Fail_Backup_On nvserverd
echo "Enter for next test..."
read

#############################################################################
echo "Battery followed by No_Longer_Low for same origin"
wpostemsg -m "test 4" sub_source=UPS origin=4.4.4.4
     Ups_Battery_Low nvserverd
sleep 5
wpostemsg -m "test 4" sub_source=UPS origin=4.4.4.4
     Ups_Battery_No_Longer_Low nvserverd
echo "Enter for next test..."
read
```

Figure 142 (Part 1 of 3). Script to Test the Rules

```
#############################################################################
echo "Battery followed by No_Longer_Low for different origin"
wpostemsg -m "test 5" sub_source=UPS origin=5.5.5.5
     Ups_Battery_Low nvserverd
sleep 5
wpostemsg -m "test 5" sub_source=UPS origin=5.4.5.4
     Ups_Battery_No_Longer_Low nvserverd
echo "Enter for next test..."
read

#############################################################################
echo "No_Longer_Low arriving before Battery"
wpostemsg -m "test 6" sub_source=UPS origin=6.6.6.6
     Ups_Battery_No_Longer_Low nvserverd
sleep 5
wpostemsg -m "test 6" sub_source=UPS origin=6.6.6.6
     Ups_Battery_Low nvserverd
echo "Enter for next test..."
read

#############################################################################
# Normal Sequence: Power_Fail, Battery_Low, Battery_Discharged,
#   Cisco_Shutdown_Notification.
#   Sometime later, Battery_No_Longer_Low, then Power_Restored
echo "Normal sequence of all 5 events"
wpostemsg -m "test 7" sub_source=UPS origin=7.7.7.7
     Ups_Power_Fail_Backup_On nvserverd
sleep 2
wpostemsg -m "test 7" sub_source=UPS origin=7.7.7.7
     Ups_Battery_Low nvserverd
sleep 2
wpostemsg -m "test 7" sub_source=UPS origin=7.7.7.7
     location=ABC Ups_Battery_Discharged nvserverd
sleep 2
wpostemsg -m "test 7" sub_source=UPS origin=7.7.7.7
     location=ABC Cisco_Shutdown_Notification nvserverd
sleep 2
wpostemsg -m "test 7" sub_source=UPS origin=7.7.7.7
     Ups_Power_Restored nvserverd
sleep 2
wpostemsg -m "test 7" sub_source=UPS origin=7.7.7.7
     Ups_Battery_No_Longer_Low nvserverd
echo "Enter for next test..."
read
```

Figure 142 (Part 2 of 3). Script to Test the Rules

```
#####################################################

echo "Normal Sequence, where Cisco location is different"
wpostemsg -m "test 8" sub_source=UPS origin=8.8.8.8
     Ups_Power_Fail_Backup_On nvserverd
sleep 2
wpostemsg -m "test 8" sub_source=UPS origin=8.8.8.8
     Ups_Battery_Low nvserverd
sleep 2
wpostemsg -m "test 8" sub_source=UPS origin=8.8.8.8
     location=BCD Ups_Battery_Discharged nvserverd
sleep 2
wpostemsg -m "test 8" sub_source=UPS origin=8.8.8.8
     location=CDE Cisco_Shutdown_Notification nvserverd
sleep 2
wpostemsg -m "test 8" sub_source=UPS origin=8.8.8.8
     Ups_Power_Restored nvserverd
sleep 2
wpostemsg -m "test 8" sub_source=UPS origin=8.8.8.8
     Ups_Battery_No_Longer_Low nvserverd
echo "Enter for next test..."
read
#####################################################
echo "Power_Fail duplicates"
wpostemsg -m "test 9" sub_source=UPS origin=9.9.9.9
     Ups_Power_Fail_Backup_On nvserverd
wpostemsg -m "test 9" sub_source=UPS origin=9.9.9.9
     Ups_Power_Fail_Backup_On nvserverd
wpostemsg -m "test 9" sub_source=UPS origin=9.9.9.9
     Ups_Power_Fail_Backup_On nvserverd
wpostemsg -m "test 9" sub_source=UPS origin=9.9.9.9
     Ups_Power_Fail_Backup_On nvserverd
echo "Enter for next test..."
read
#########################################################################
echo "Battery duplicates"
wpostemsg -m "test 10" sub_source=UPS origin=1.2.3.4
     Ups_Battery_Low nvserverd
wpostemsg -m "test 10" sub_source=UPS origin=1.2.3.4
     Ups_Battery_Low nvserverd
wpostemsg -m "test 10" sub_source=UPS origin=1.2.3.4
     Ups_Battery_Low nvserverd
wpostemsg -m "test 10" sub_source=UPS origin=1.2.3.4
     Ups_Battery_Low nvserverd
sleep 2
wpostemsg -m "test 10" sub_source=UPS origin=2.3.4.5
     location=BCD Ups_Battery_Discharged nvserverd
wpostemsg -m "test 10" sub_source=UPS origin=2.3.4.5
     location=BCD Ups_Battery_Discharged nvserverd
wpostemsg -m "test 10" sub_source=UPS origin=2.3.4.5
     location=BCD Ups_Battery_Discharged nvserverd
wpostemsg -m "test 10" sub_source=UPS origin=2.3.4.5
     location=BCD Ups_Battery_Discharged nvserverd
echo "Testing finished for now."
```

Figure 142 (Part 3 of 3). Script to Test the Rules

An excerpt of the results of this test as seen on the event console can be seen in Figure 143 on page 257.

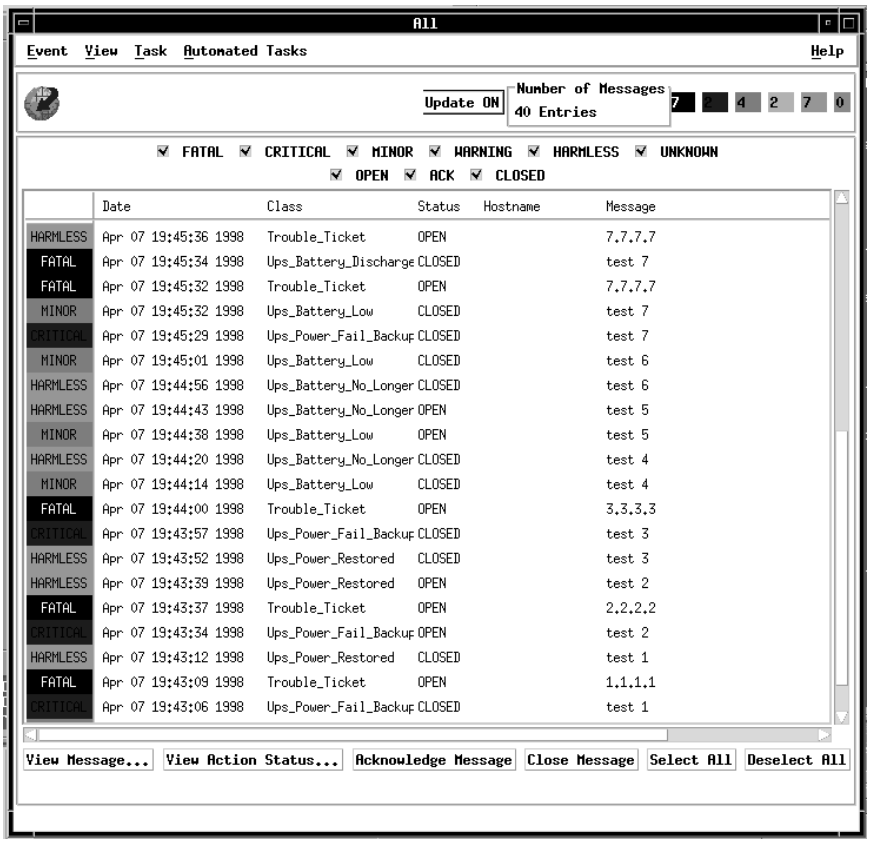


Figure 143. Test Output for UPS Correlation Example

We observed that the correct flow was being performed, and the correct events were being closed and the required number of trouble tickets were being generated. We also checked that the repeat_count and date slots were being set correctly.

We also used the trace during the testing of the rules. This shows that the correct rules are being processed when a specific event arrives.

11.3 Beacon State Example

A Cabletron hub will send a beacon state trap to its network manager whenever it detects beaconing on the token-ring. Token-ring beacons are an aspect of the token-ring architecture used to signal ring problems. As described at length in Chapter 8, “The TME 10 NetView for AIX Event Adapter” on page 161, we customized NetView to forward certain traps in the form of events to the TEC event server. As no baroc class structure existed for these Cabletron events, we defined our own baroc file, in which each trap potentially forwarded by NetView is reflected as an individual event class. The variables contained in the SNMP trap are converted to slots in the TEC event.

For this example, let us examine four Cabletron events that stand in a somewhat complex relationship to one another.

11.3.1 The Events

Table 9. Beacon State Events

Original SNMP Trap Number	TEC Event Class Name	Event Description
550	Beacon_State	Indicates a problem of undetermined type on the ring, but contains the information on what interface the problem is found.
551	Beacon_State_Cleared	Problem that caused Beacon_State is no longer detected.
556	Beacon_Recovery_Port_Removed	Hub has achieved a workaround of the problem by removing a port.
558	Beacon_Recovery_Board_Bypassed	Hub has achieved a workaround of the problem by bypassing a board.

Typically, the hub sends a Beacon_State immediately upon detecting a problem. In many cases, the problem resolves itself, and the hub sends a Beacon_State_Cleared within a matter of seconds.

If the problem is a bad port or a bad board, the hub is often capable of bypassing the bad hardware, and sends an event Beacon_Recovery_Port_Removed or Beacon_Recovery_Board_Bypassed, followed by a Beacon_State_Cleared, indicating that the original problem can no longer be detected by the hub. However, this is only a temporary workaround.

It is possible that the hub will directly send the information Beacon_Recovery_Port_Removed or Beacon_Recovery_Board_Bypassed, without it being preceded by a Beacon_State event.

11.3.2 The Event Relationships

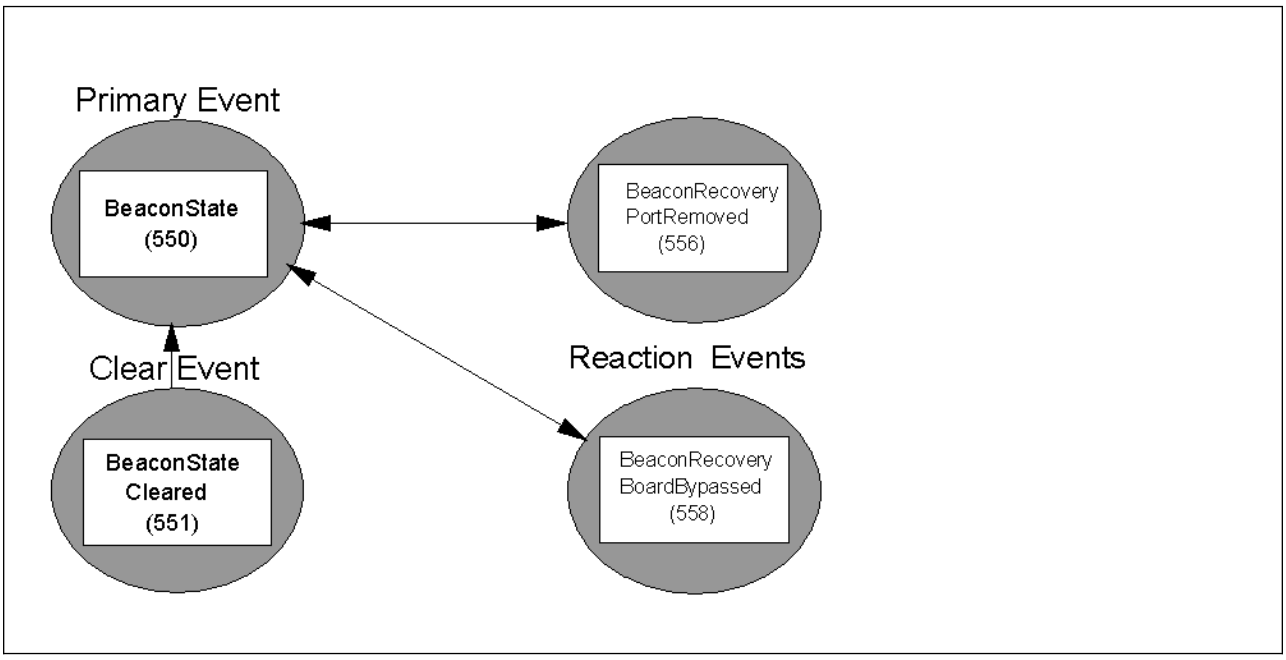


Figure 144. Event Relationship Diagram for Beacon State Events

Figure 144 illustrates the relationships just defined.

We describe Beacon_State as the primary event, because it is typically the first event to arrive, serving sometimes as a significant occurrence in its own right, and serving sometimes as a warning, followed by the more specific information contained in the reaction events.

We describe Beacon_Recovery_Port_Removed and Beacon_Recovery_Board_Bypassed as reaction events, because they are the hub's reaction to detecting a Beacon_State. These two events will be handled in the same way. However, these reaction events are not to be seen as effects caused by the Beacon_State, because the two events can be sent without a Beacon_State, and because the information in them is significant, with or without a preceding Beacon_State.

The event Beacon_State_Cleared is a straightforward clear event, which negates the significance of the Beacon_State. However, it does not mean that the problem temporarily bypassed by one of the reaction events is gone.

11.3.3 The Event Class Definitions

```
# Cabletron Specific Class Definitions

TEC_CLASS :
        Ctccable_Event ISA Nvserverd_Event
        DEFINES {
                sub_source: default = "Cabletron";
                severity: default = WARNING;
                last_occur: STRING;
                tt: STRING;
                category: INT32;
                ctcvar1: INT32;
                ctcvar2: INT32;
                ctcvar3: INT32;
          };
END

TEC_CLASS :
     Trouble_Ticket ISA EVENT
     DEFINES {
          nv_enterprise: INT32;
          };
END

# 550
TEC_CLASS :
     Beacon_State ISA Ctccable_Event
     DEFINES {
          ifindex: INT32, dup_detect = yes;
          prtgrpid: INT32, dup_detect = yes;
          mgtstnme: STRING, dup_detect = yes;
          mgtstnadd: INT32, dup_detect = yes;
          mgtstnuna: INT32, dup_detect = yes;
          mgtstnbrd: INT32, dup_detect = yes;
          mgtstnprt: INT32, dup_detect = yes;
          stsrlstbcont: INT32, dup_detect = yes;
          };
END

# 551
TEC_CLASS :
     Beacon_State_Cleared ISA Ctccable_Event
     DEFINES {
          stsrlstbcont: INT32, dup_detect = yes;
          ifindex: INT32, dup_detect = yes;
          };
END
```

Figure 145 (Part 1 of 2). The Cabletron Class Definition File (ctccable.baroc)

```
# 556
TEC_CLASS :
      Beacon_Recovery_Port_Removed ISA Ctccable_Event
      DEFINES {
          chslotid: INT32, dup_detect = yes;
          prtgrpid: INT32;
          prtmgtprtid: INT32;
          severity: default = CRITICAL;
          };
END

# 558
TEC_CLASS :
     Beacon_Recovery_Board_Bypassed ISA Ctccable_Event
     DEFINES {
          chslotid: INT32, dup_detect = yes;
};
END
```

Figure 145 (Part 2 of 2). The Cabletron Class Definition File (ctccable.baroc)

Notice a few things about this excerpt of the baroc file:

- Our Cabletron superclass Ctccable_Event is defined as a child of Nvserverd_Event, the superclass for all events coming from the NetView integrated event adapter.
- All Cabletron events inherit the slot last_occur, which we use in conjunction with checking for duplicate events. If a duplicate event arrives, and we choose not to display that event, we do, however, insert the date and time of the duplicate event into the slot last_occur of the currently displayed event, so that there is a record of the arrival of the duplicate event.
- All Cabletron events inherit the slot tt, which we use to indicate whether the event has opened a trouble ticket (tt=1) or whether the event has updated an existing trouble ticket (tt=2).
- The class Trouble_Ticket serves for testing purposes only. Since this project did not include interfacing to a specific trouble ticketing system, we tested our rule logic by calling scripts to open and update a trouble ticket, which do nothing more than to send an event of class Trouble_Ticket.

11.3.4 The Event Policy

Many policy decisions were discussed and agreed upon before we started coding the actual rules for the Beacon_State example:

- If a Beacon_State arrives and is cleared by a Beacon_State_Cleared within 20 seconds, and no reaction event arrives in the interim, then no trouble ticket should be opened.
- If a Beacon_State is not cleared within 20 seconds, and no reaction event arrives in the interim, then a trouble ticket should be opened.
- If a Beacon_State opens a trouble ticket, and a reaction event arrives within 24 hours, its information should be added to the open trouble ticket.

- If one of the two reaction events arrives on its own and there is no open Beacon_State within the last 24 hours, then the reaction event should open a trouble ticket.
- If a reaction event opens a trouble ticket, and a Beacon_State arrives within 24 hours, the Beacon_State event should add its information to the open trouble ticket.
- A Beacon_State event is considered a duplicate of an open event if it is from the same IP address (origin), for the same interface (ifindex), and arrives within eight hours of the open event.
- A Beacon_Recovery_Port_Removed or Beacon_Recovery_Board_Bypassed event is considered a duplicate of an open event if it is from the same IP address, has the same contents in the slot chslotid, and arrives within eight hours of the open event.
- A Beacon_State_Cleared event should close only a Beacon_State event, and not any of the reaction events.
- The rules should allow for the reasonable possibility, due to network delivery timing inconsistencies, that a Beacon_State event might be preceded by a Beacon_State_Cleared event by up to 1 minute.
- When a reaction event is acknowledged by an operator, the status and administrator information should be propagated to any related Beacon_State event.
- When a reaction event is closed, whether by the trouble ticketing system or by an operator, any related Beacon_State event should be closed, also.

11.3.5 The Rule Flowcharts and Rules

For this example we include the following flowcharts and rules:

- For the cause event:
  - ctccable.baroc file
  - ctcbeacst.rls flowchart
  - ctcbeacst.rls rule code
- For the reaction events:
  - ctcbeacsec.rls flowchart
  - ctcbeacsec.rls rule
- For the clear events:
  - ctcbeacclr.rls flowchart
  - ctcbeacclr.rls rule

The flowchart for the beacon state event is shown in Figure 146 on page 263.

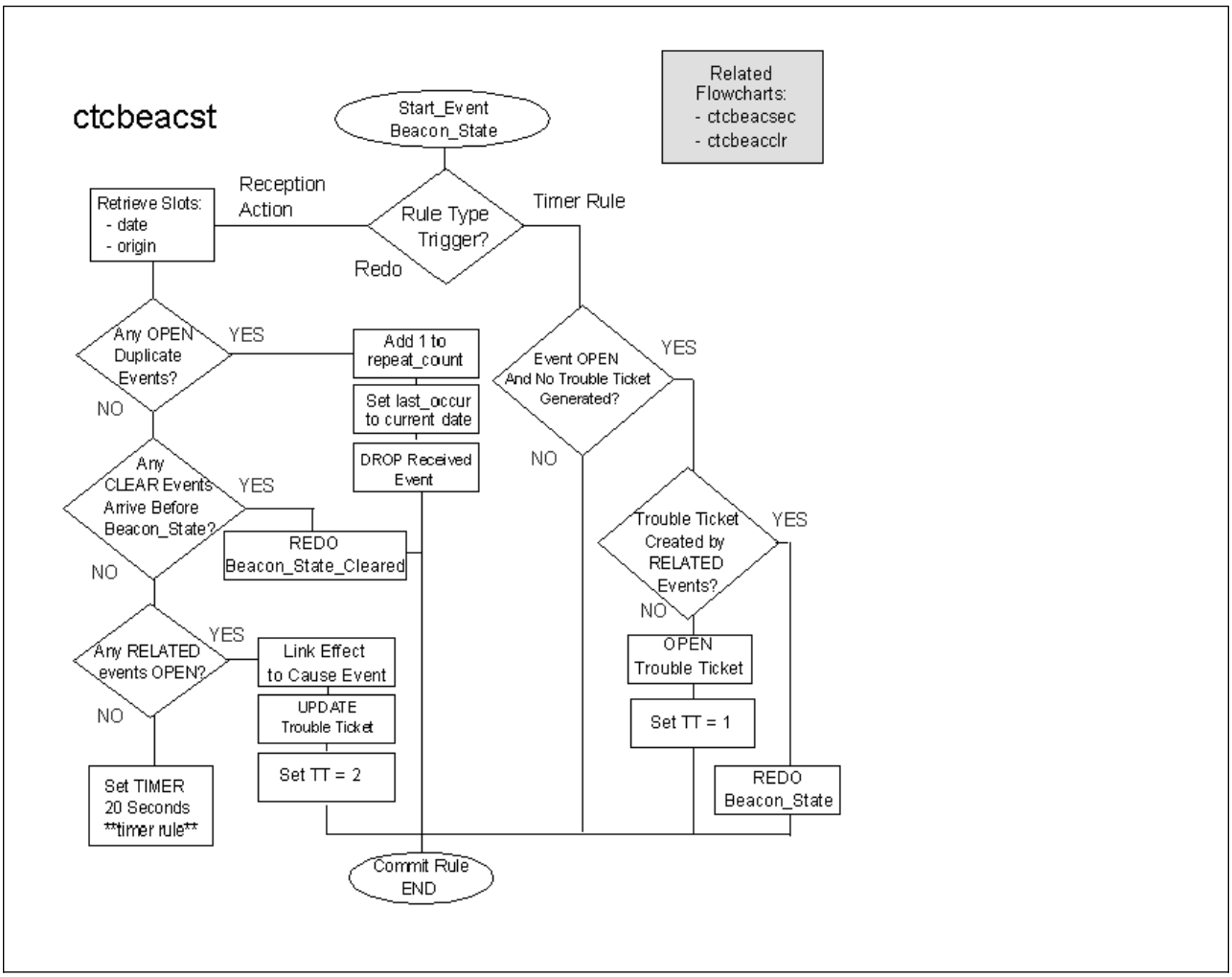


Figure 146. The Flowchart for ctcbeacst.rls

The rule is outlined below:

- The rule is activated on reception of Beacon_State event.
- Retrieve date and origin values for the event.
- If any duplicates are found, then:
  - Add 1 to the repeat count.
  - Set the last occur date to date on the current event.
  - Drop the event.
  - End rule processing.
- If no duplicates are found, then:
  - If any clear beacon events have arrived, then
    - Re-do the analysis for clear event.
    - End rule processing.
  - If any reaction events exist, then:
    - Assume a trouble ticket has been created by the reaction rule processing.

- Link the current event to the reaction event.
- Update the trouble ticket information.
- Set the tt variable to 2.

– If no reaction events have been received, then set a timer for 20 seconds. This is set in case a beacon_state clear event is sent.

The processing for the timer rule is:

- If no new closed events have been sent, then create a new trouble ticket for beacon state.
- If a trouble ticket has been created from the reaction events, then redo the analysis for the beacon state event.

The rule is shown in Figure 147 on page 265.

```
/* handling of Beacon_State event             */

rule:
'Beacon_State':
(
   event: _event of_class 'Beacon_State'
  where [
   origin: _origin,
   date: _date
                     ],

reception_action:
'Beacon_State_Elim_Dup_Events':
  (
 first_duplicate(_event,
              event: _dup_ev
   where [
    status: outside ['CLOSED']
                 ],
   _event - 28800 - 0),

 add_to_repeat_count(_dup_ev,1),
        place_change_request(_dup_ev,last_occur,_date),
 drop_received_event,
 commit_rule
    ),

reception_action:
'Beacon_State_Check_For_Recent_Clear':
  (
 first_instance(event: _clr_ev of_class 'Beacon_State_Cleared'
  where [
   origin: equals _origin,
   status: outside ['CLOSED']
        ],
  _event - 60 - 0),
 redo_analysis(_clr_ev),
 commit_rule
  ),
```

Figure 147 (Part 1 of 3). Rule for Beacon State (ctcbeacst.rls)

```
action:
'Beacon_State_Check_Related_Events':
   (
 first_instance(event: _beacon_ev of_class _class
  within [
  'Beacon_Recovery_Port_Removed',
  'Beacon_Recovery_Board_Bypassed'
  ]
  where [
   origin: equals _origin,
   status: outside ['CLOSED'],
   tt: equals '1'
                      ],
  _event - 86400 - 86400),
 link_effect_to_cause(_event,_beacon_ev),
 exec_program(_beacon_ev,'/usr/development/rules/ctc_ttproc_upd.sh',
  '-s "%s"',[_origin],'NO'),
 place_change_request(_event,tt,'2'),
 commit_action
      ),
reception_action:
'Beacon_State_New_Set_Timer':
  (
 set_timer(_event, 20, '')
  )
).
```

Figure 147 (Part 2 of 3). Rule for Beacon State (ctcbeacst.rls)

```
/* First real processing of new Beacon_State after 20 second delay */
timer_rule:
'Beacon_State_Check_After_Timer':
   (
   event: _event of_class 'Beacon_State'
  where [
   tt: outside ['1','2'],
   status: outside &lbrk'CLOSED'],
   origin: _origin
                     ],

action:
'Beacon_State_Check_Related_Events':
   (
 first_instance(event: _beacon_ev of_class _class
  within [
  'Beacon_Recovery_Port_Removed',
  'Beacon_Recovery_Board_Bypassed'
  ]
  where [
   origin: equals _origin,
   status: outside ['CLOSED'],
   tt: equals '1'
                     ],
  _event - 86400 - 20),
 redo_analysis(_event),
 commit_action
      ),
action:
'Beacon_State_Open_Raise_TT':
 (
 exec_program(_event,'/usr/development/rules/ctc_ttproc.sh',
  '-s "%s"',[_origin],'NO'),
 place_change_request(_event,tt,'1')
   )

).
```

Figure 147 (Part 3 of 3). Rule for Beacon State (ctcbeacst.rls)

11.3.6 The Reaction Events

The reaction events are generated from the device and provide a solution to the existing problem. In this example the device will remove the port that is in the beacon state. The flowchart that shows the process is shown in Figure 148 on page 268.

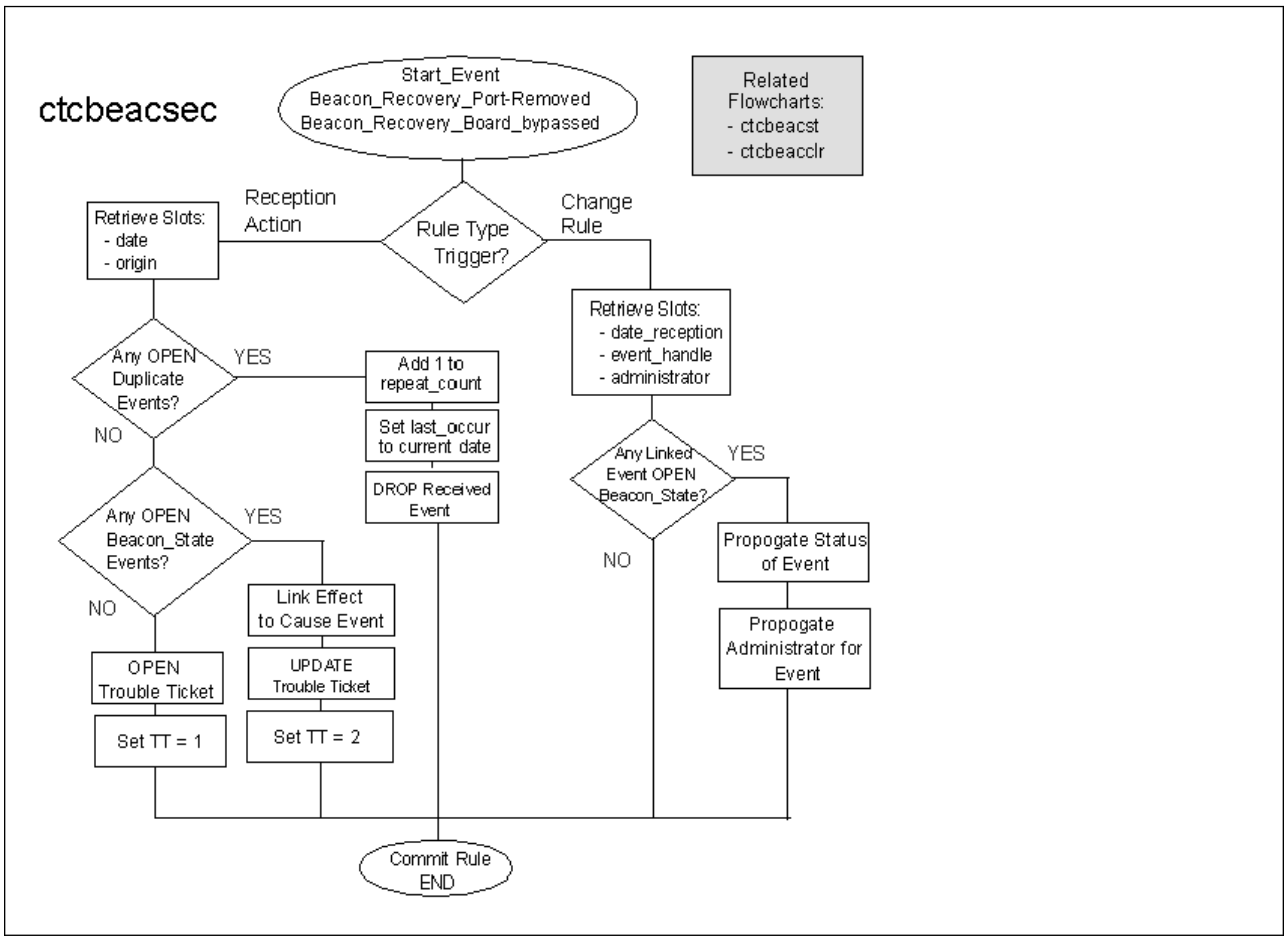


Figure 148. The Flowchart for ctcbeacsec.rls

The rule process is outlined below:

- The rule is activated when a port removed or board bypassed event arrives.
- Retrieve date and origin slot values.
- If any Beacon state events exist, then:
  - Link the current event to the beacon state event.
  - Update the trouble ticket.
  - Set the TT variable to 2 to signify the trouble ticket has been updated.
  - End rule processing.
- If there are no beacon state events, then:
  - Open a new trouble ticket.
  - Set the TT variable to 1 to signify new trouble ticket.
  - End rule processing.

The change rule is activated when the status of the events has been changed.

The rule is shown in Figure 149 on page 269.

```
/* Beacon_Recovery_Port_Removed and Beacon_Recovery_Board_Bypassed */
/* are handled the same way.  If a Beacon_State has opened a TT,   */
/* these events update it, otherwise they open their own TT.       */
/* An open Beacon_State will be linked as effect for closure       */
/* purposes.                                                       */

rule:
'Beacon_State_Related_Event_Arrives':
(
   event: _event of_class within
 [
  'Beacon_Recovery_Port_Removed',
  'Beacon_Recovery_Board_Bypassed'
 [
  where [
   date: _date,
   origin: _origin
                     ],

reception_action:
'Beacon_Related_Elim_Dup_Events':
  (
 first_duplicate(_event,
              event: _dup_ev
   where [
    status: outside ['CLOSED']
                 ],
   _event - 28800 - 0),
 add_to_repeat_count(_dup_ev,1),
       place_change_request(_dup_ev,last_occur,_date),
 drop_received_event,
 commit_rule
    ),

reception_action:
'Beacon_Related_Check_For_Beacon_State':
   (
 first_instance(event: _beacon_ev of_class 'Beacon_State'
  where [
   tt: equals '1',
   origin: equals _origin,
   status: outside ['CLOSED']
                    ],
  _event - 86400 - 0),
 link_effect_to_cause(_beacon_ev,_event),
 exec_program(_beacon_ev,'/usr/development/rules/ctc_ttproc_upd.sh',
  '-s "%s"',[_origin],'NO'),
 place_change_request(_event,tt,'2'),
 commit_action
     ),
```

Figure 149 (Part 1 of 2). The Rule for Reaction Events (ctcbeacsec.rls)

```
reception_action:
'Beacon_Related_No_Beacon_State_Raise_TT':
   (
 exec_program(_event,'/usr/development/rules/ctc_ttproc.sh',
  '-s "%s"',[_origin],'NO'),
 place_change_request(_event,tt,'1')
   )

).

/*********************************************************************/
/* Change Rule                                                       */
/* When the status of the Beacon- related events is changed          */
/* (either by an operator or by a retro-action from a TT),           */
/* the new status will be propagated to the linked Beacon_State events*/

change_rule:
'Beacon_Related_Propagate_Status_To_Linked_Beacon_State':
   (
    event: _event of_class within
 [
  'Beacon_Recovery_Port_Removed',
  'Beacon_Recovery_Board_Bypassed'
 ]
 where [
  date_reception: _date_rec,
  event_handle: _ev_handle,
  administrator: _admin
       ],
       slot: status set_to _status
  within ['ACK', 'CLOSED'],
    action:
    (
     first_instance(event: _effect_event of_class 'Beacon_State'
 where [
  status: outside ['CLOSED'],
  cause_date_reception: equals _date_rec,
  cause_event_handle: equals _ev_handle
  ],
  _event - 86400 - 86400),
 set_event_status(_effect_event, _status),
 set_event_administrator(_effect_event, _admin)
    )
   ).
```

Figure 149 (Part 2 of 2). The Rule for Reaction Events (ctcbeacsec.rls)

11.3.7 The Clear Rule

The clear event will only clear the beacon state event. The reaction events will either be cleared manually or by the trouble ticket application.

The flowchart is shown in Figure 150 on page 271.

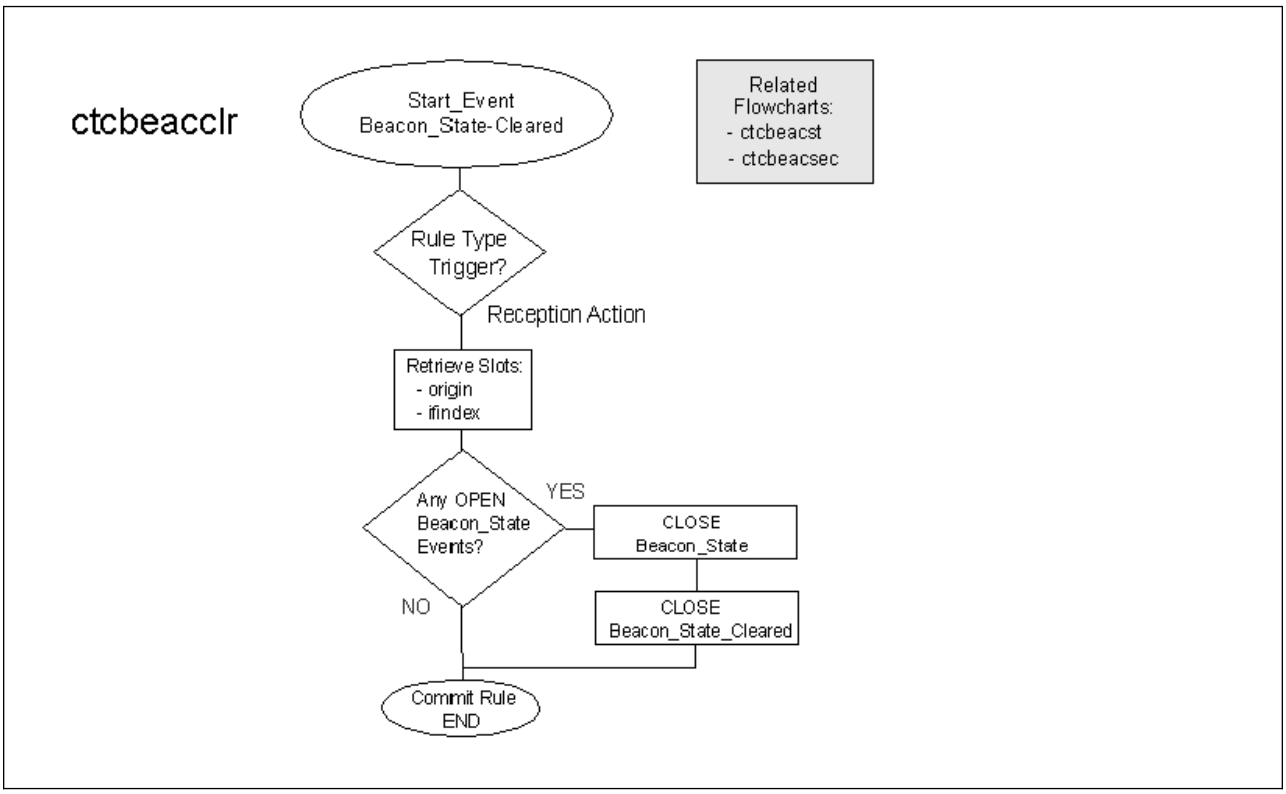


Figure 150. The Flowchart for ctcbeacclr.rls

The rule flow is as follows:

- The event Beacon_State_Cleared will activate the rule.
- Retrieve the values for origin and ifindex.
- If any beacon state events are OPEN, then:
  - Close the beacon state event.
  - Close the beacon clear event.

The clear rule is shown in Figure 151 on page 272.

```
/* Beacon State Cleared Rules                                    */
/* Beacon_State_Cleared closes itself and Beacon_State.          */
/* Action can be called at reception or due to redo_analysis. */

rule:
'Beacon_State_Cleared':
 (
 event: _event of_class 'Beacon_State_Cleared'
 where [
  origin: _origin,
  ifindex:  _ifindex
       ],

action:
'Beacon_State_Cleared_Close_Open_Beacon_State':
  (
 first_instance(event: _bea_ev of_class 'Beacon_State'
  where [
   origin: equals _origin,
   ifindex: equals _ifindex,
   status: outside ['CLOSED']
        ],
  _event - 86400 - 60),
 set_event_status(_bea_ev,'CLOSED'),
        set_event_status(_event,'CLOSED')
 )
).
```

Figure 151. The Rule for ctcbeacclr.rls

11.3.8 Testing the Rules

We tested the rules by issuing the expected events with the wpostemsg command for each of the four event classes, in every plausible order and number. We then examined the /tmp/rules.trace file to make sure the sequence of commands was as we expected.

We then combined the individual tests into a script that issues the events in various sequences.

To test the full end to end process for the event we created a set of shell scripts to send SNMP traps from the NetView server to the TEC. An example of one of these scripts is shown below.

```
/usr/OV/bin/snmptrap rs600027 .1.3.6.1.4.1.52 rs600027 6 550 0
        1.1 octetstring "12345678" \
        2.2 octetstring "12345678" \
        3.3 octetstring "rs600027" \
        1.4 octetstring "12345678" \
        2.5 octetstring "12345678" \
        3.6 octetstring "12345678" \
        1.7 octetstring "12345678" \
        2.8 octetstring "12345678" \
        3.6 octetstring "12345678"
```

The command snmptrap is passed dummy arguments using the format shown. These will be passed to NetView as the eight required arguments for this specific trap definition.

An excerpt of the results of this test as seen on the event console can be seen in Figure 152.

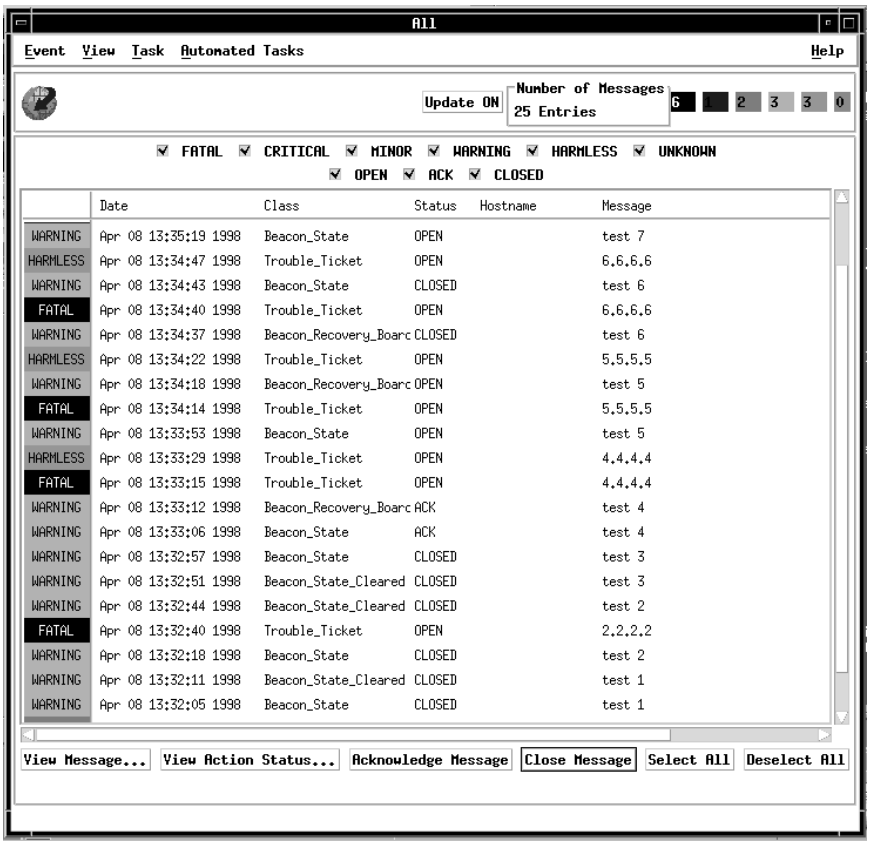


Figure 152. Test Output for Beacon State Correlation Example

11.4 Temperature Example

This example covers the temperature events generated by the Cabletron HUBs.

11.4.1 The Events

The events used in this example are shown in Table 10

Table 10 (Page 1 of 2). Power Supply Events

Original SNMP Trap Number	TEC Event Class Name	Event Description
281	Env_Temp_Warm	The hub has detected that the temperature within a MIM has reached a warm condition.
282	Env_Temp_Hot	The hub has detected that the temperature within a MIM has reached a hot condition.

Table 10 (Page 2 of 2). Power Supply Events		
Original SNMP Trap Number	**TEC Event Class Name**	**Event Description**
284	Env_Temp_Normal	The hub has detected that the temperature within a MIM has regained a normal condition.
286	Fan_Fail	The device has detected that the system fans have failed.
287	Fan_Normal	The device has detected that the system fans have returned to normal.

11.4.2 The Event Relationships

The events relationship diagram is shown in Figure 153.

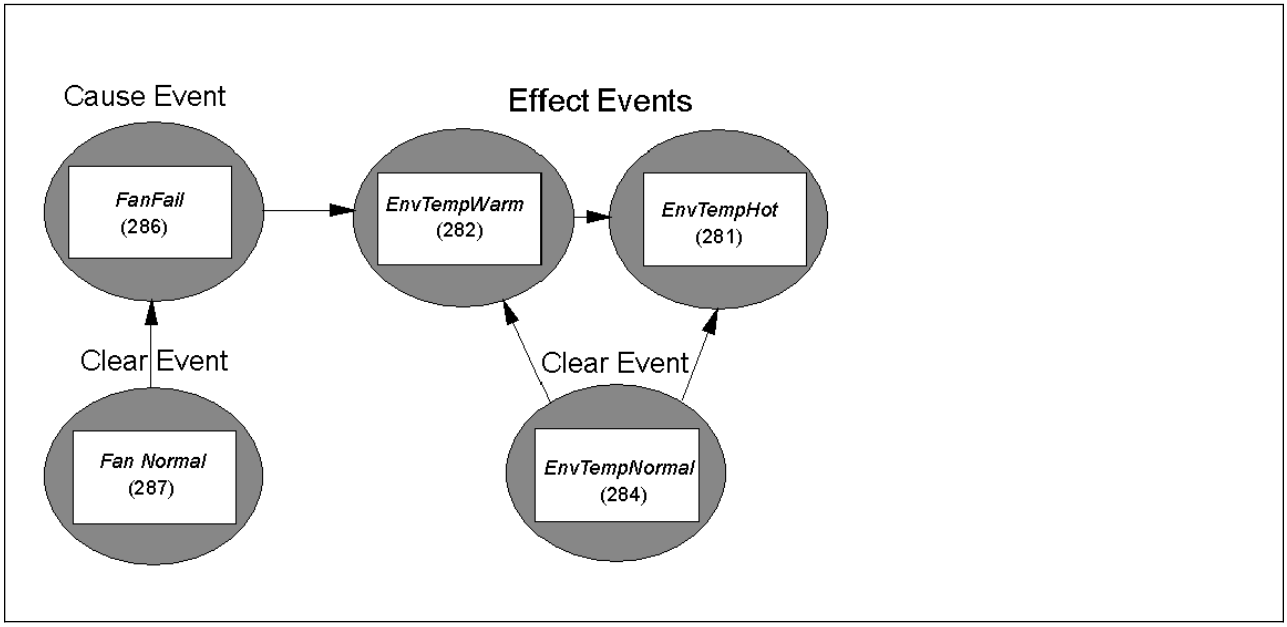


Figure 153. Flowchart For ctcfanfail.rls

It seems logical that a fan failing could and should lead to a warm or hot environment, and it often does. So we can look at the Fan_Fail event as causing the events Env_Temp_Warm and Env_Temp_Hot.

However, it is also possible to have overheating of the hub without the fan failing. So we need to handle Env_Temp_Warm and Env_Temp_Hot as individual problems if they arrive without being preceded by a Fan_Fail event.

For this reason, these two overheating events are cleared by a different event from the one that clears Fan_Fail. Env_Temp_Normal clears Env_Temp_Warm and Env_Temp_Hot; and Fan_Normal clears Fan_Fail.

In what way, then, are these events practically linked or related? For our purposes, they are linked only in that they all update one and the same trouble ticket when they arrive as a complete package, as opposed to being seen as separate problems.

11.4.3 The Event Class Definitions

The event class definitions are shown below:

```
# 281
TEC_CLASS :
     Env_Temp_Warm ISA Ctccable_Event
     DEFINES {
          chslotid: INT32;
          };
END

# 282
TEC_CLASS :
     Env_Temp_Hot ISA Ctccable_Event
     DEFINES {
          chslotid: INT32;
     };
END

# 284
TEC_CLASS :
     Env_Temp_Normal ISA Ctccable_Event
     DEFINES {
          chslotid: INT32;
     };
END

# 286
TEC_CLASS :
     Fan_Fail ISA Ctccable_Event
     DEFINES {
           severity: default = CRITICAL;
     };
END

# 287
TEC_CLASS :
      Fan_Normal ISA Ctccable_Event
      DEFINES {
          severity: default = HARMLESS;
      };
END
```

11.4.4 The Event Policy

The following policy decisions were made about how to handle the events involved in this example:

- The three problem events are considered to be duplicates of an open event of the same type if they arrive within eight hours of the open event.
- The rules should allow for the reasonable possibility, due to network delivery timing inconsistencies, that a problem event might be preceded by a related clear event by up to 1 minute.
- A Fan_Fail event should always open a trouble ticket, even if it is closed quickly by a Fan_Normal.
- An Env_Temp_Warm or Env_Temp_Hot event preceded by a Fan_Fail by up to eight hours should update the trouble ticket opened by the Fan_Fail.
- An Env_Temp_Warm or Env_Temp_Hot event not preceded by a Fan_Fail should open a trouble ticket.

- An Env_Temp_Hot event not preceded by an Env_Temp_Warm event should open a trouble ticket.
- A Fan_Normal clears a Fan_Fail for the same IP address if received within 24 hours.
- An Env_Temp_Normal event clears all Env_Temp_Warm and Env_Temp_Hot events for the same chslotid (?) received within the last 24 hours.

11.4.5 The Rules

We did not include any of the flowcharts for this example as they are very similar to the flowcharts discussed in the previous examples.

We have included the following rules:

- Cause rule
  - ctcfanfail.rls
- Effect events
  - ctcfantemp.rls
- Clear events
  - ctcfanfailclr.rls
  - ctcfantempclr.rls

```
/* handling of Cabletron Fan_Fail events          */

rule:
'Fan_Fail':
(
   event: _event of_class 'Fan_Fail'
  where [
   origin: _origin,
   date: _date
                      ],

reception_action:
'Fan_Fail_Elim_Dup_Events':
  (
 first_duplicate(_event,
              event: _dup_ev
   where [
    status: outside ['CLOSED']
                 ],
   _event - 28800 - 0),

 add_to_repeat_count(_dup_ev,1),
        place_change_request(_dup_ev,last_occur,_date),
 drop_received_event,
 commit_rule
    ),

reception_action:
'Fan_Fail_Check_For_Recent_Clear':
  (
 first_instance(event: _clr_ev of_class 'Fan_Normal'
  where [
   origin: equals _origin,
   status: outside ['CLOSED']
        ],
  _event - 60 - 0),
 redo_analysis(_clr_ev)
  ),
```

Figure 154 (Part 1 of 3). Rule for ctcfanfail.rls

```
reception_action:
'Fan_Fail_Check_For_Related_Events':
  (
 first_instance(event: _temp_ev of_class _temp_class
  within [
   'Env_Temp_Warm',
   'Env_Temp_Hot'
  ]
  where [
   origin: equals _origin,
   status: outside ['CLOSED'],
   tt: equals '1'
        ],
  _event - 28800 - 0),
  exec_program(_temp_ev,'/usr/development/rules/ctc_ttproc_upd.sh',
  '-s "%s"',[_origin],'NO'),
  place_change_request(_event,tt,'2'),
  commit_rule
  ),
```

Figure 154 (Part 2 of 3). Rule for ctcfanfail.rls

```
reception_action:
'Fan_Fail_Open_TT':
  (
 exec_program(_event,'/usr/development/rules/ctc_ttproc.sh','-s "%s"
  ',[_origin],'NO'),
 place_change_request(_event,tt,'1')
  )
).
```

Figure 154 (Part 3 of 3). Rule for ctcfanfail.rls

11.4.6 The Effect Events

The rules for the effect events are shown in Figure 155 on page 279.

```
/* handling of Cabletron Environment Warm and Hot events          */
/* as related to Fan_Fail event                                   */

rule:
'Env_Warm_or_Hot_Event':
(
   event: _event of_class _class
 within [
  'Env_Temp_Warm',
  'Env_Temp_Hot'
        ]
  where [
   chslotid: _chslotid,
   origin: _origin,
   date: _date
                     ],

reception_action:
'Env_Temp_Elim_Dup_Events':
  (
 first_duplicate(_event,
              event: _dup_ev
   where [
    status: outside ['CLOSED']
                 ],
   _event - 28800 - 0),

 add_to_repeat_count(_dup_ev,1),
        place_change_request(_dup_ev,last_occur,_date),
 drop_received_event,
 commit_rule
    ),

reception_action:
'Env_Temp_Check_For_Recent_Clear':
  (
 all_instances(event: _clr_ev of_class 'Env_Temp_Normal'
  where [
   origin: equals _origin,
   status: outside ['CLOSED']
        ],
  _event - 60 - 0),
 redo_analysis(_clr_ev)
  ),
```

Figure 155 (Part 1 of 2). The ctcfantemp.rls File

```
reception_action:
'Env_Temp_Check_For_Related_Fan_Event':
  (
 first_instance(event: _fan_ev of_class 'Fan_Fail'
  where [
   origin: equals _origin,
   status: outside ['CLOSED'],
   tt: equals '1'
        ],
  _event - 28800 - 0),
  exec_program(_fan_ev,'/usr/development/rules/ctc_ttproc_upd.sh',
  '-s "%s"',[_origin],'NO'),
  place_change_request(_event,tt,'2'),
  commit_rule
  ),

reception_action:
'Env_Temp_Check_For_Related_Env_Temp_Event':
  (
 first_instance(event: _temp_ev of_class _temp_class
  within [
   'Env_Temp_Warm',
   'Env_Temp_Hot'
        ]
  where [
   chslotid: equals _chslotid,
   status: outside ['CLOSED'],
   tt: equals '1'
        ],
  _event - 28800 - 0),
  exec_program(_temp_ev,'/usr/development/rules/ctc_ttproc_upd.sh',
  '-s "%s"',[_origin],'NO'),
  place_change_request(_event,tt,'2'),
  commit_rule
   ),

reception_action:
'Env_Temp_Open_TT':
  (
 exec_program(_event,'/usr/development/rules/ctc_ttproc.sh','-s "%s"'
  ,[_origin],'NO'),
 place_change_request(_event,tt,'1')
  )
).
```

Figure 155 (Part 2 of 2). The ctcfantemp.rls File

11.4.7 The Clearing Events

The clear rule is shown in Figure 156 on page 281.

```
/* Env_Temp_Warm and Env_Temp_Hot cleared by Env_Temp_Normal */

rule:
'Env_Temp_Normal':
 (
 event: _event of_class 'Env_Temp_Normal'
 where [
  chslotid: _chslotid
       ],

action:
'Env_Temp_Normal_Close_All_Open_Env_Temp':
  (
 all_instances(event: _temp_ev of_class _temp_class
  within [
   'Env_Temp_Warm',
   'Env_Temp_Hot'
        ]
  where [
   chslotid: equals _chslotid,
   status: outside ['CLOSED']
        ],

  _event - 86400 - 60),

 set_event_status(_temp_ev,'CLOSED'),
        set_event_status(_event,'CLOSED')
 )
).
```

Figure 156. The ctcfanfailclr.rls Rule

The clear event ctcfantempclr rule is shown in Figure 157 on page 282.

```
/* Fan_Fail cleared by Fan_Normal */
rule:
'Fan_Normal':
 (
 event: _event of_class 'Fan_Normal'
 where [
  origin: _origin
       ],

action:
'Fan_Normal_Close_Open_Fan_Fail':
  (
 first_instance(event: _fan_ev of_class 'Fan_Fail'
  where [
   origin: equals _origin,
   status: outside ['CLOSED']
        ],

  _event - 86400 - 60),

 set_event_status(_fan_ev,'CLOSED'),
        set_event_status(_event,'CLOSED')
 )
).
```

Figure 157. The ctcfantempclr.rls File

11.4.8 Testing the Rules

After testing the individual events using the wpostemsg command, we combined our events once again into a script that issues the events in various sequences.

An excerpt of the results of this test as seen on the event console can be seen in Figure 158 on page 283.

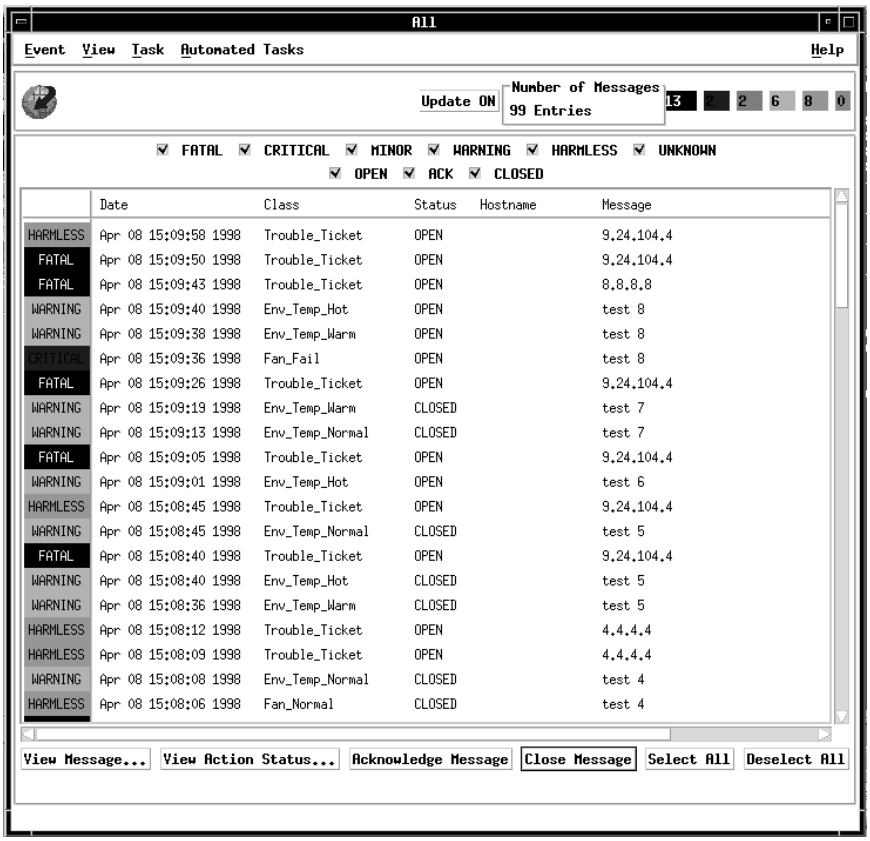


Figure 158. Test Output for Fan Failing Correlation Example

Appendix A. TEC Installation on Windows NT

This chapter shows how to install a TME 10 Enterprise Console on Windows NT. The first step is to install the relational database to be used by the TEC which in this example is Sybase. The second step is to install the TME 10 Enterprise Console Server. The third step is to verify that the RIM object was installed correctly and install the TEC database components. The fourth step is to install the TME 10 Enterprise Console.

Also note that TMR Framework was already installed and Tivoli Enterprise Console will be installed on the same machine, which is called WTR05212.

A.1 RDBMS Database Sybase Installation

Since the installation of the Sybase database is a new process for TEC users we show the installation of Sybase on a single machine using Sybase 11.0.1. Also note that TMR Framework was already installed.

Figure 159. Windows NT Run Setup Program

Run the setup program for Sybase and click **OK**.

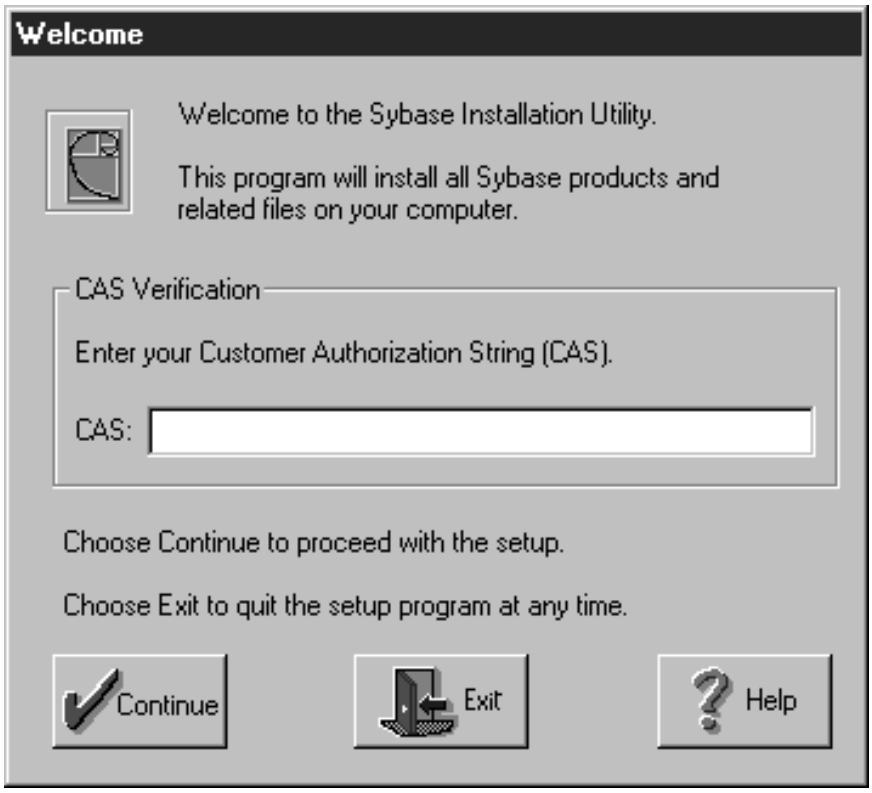


Figure 160. Sybase CAS Verification

Enter the license key (Customer Authorization String (CAS)) and click **Continue**.

Figure 161. Sybase Release Directory

Enter the path for the installation directory and click **Continue**.

Figure 162. Sybase Product Set Selection

Select the 32-bit option and click **Continue**.

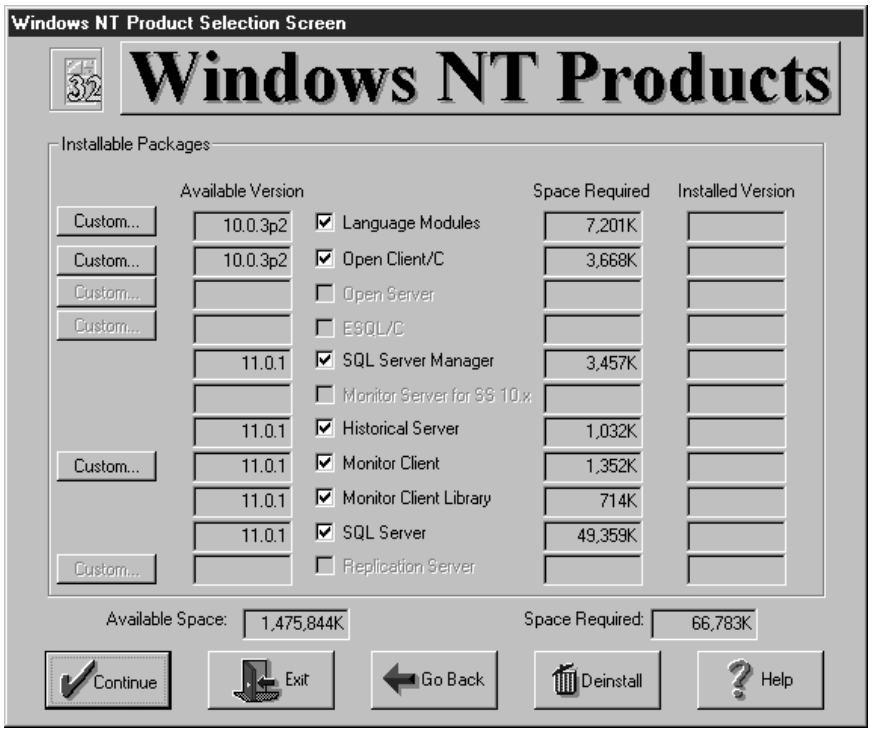


Figure 163. Sybase Windows NT Product Selection Screen

Accept defaults on the installable packages window and click **Continue**.

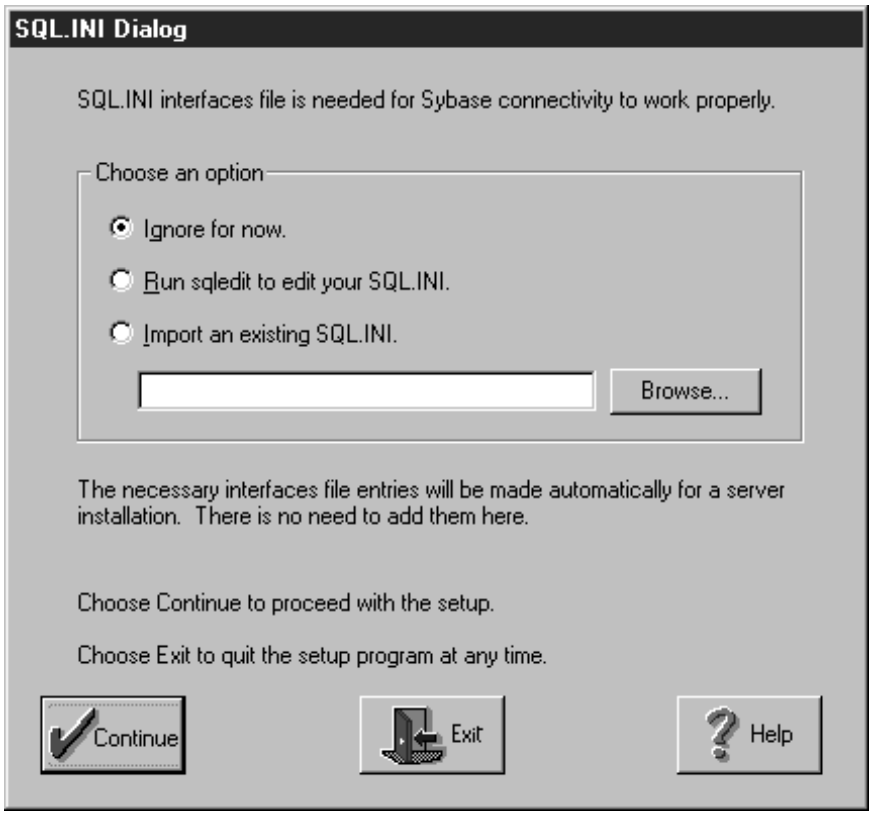


Figure 164. Sybase SQL.INI Dialog

Select **Ignore for now** for the SQL.INI interface file options and click **Continue**.

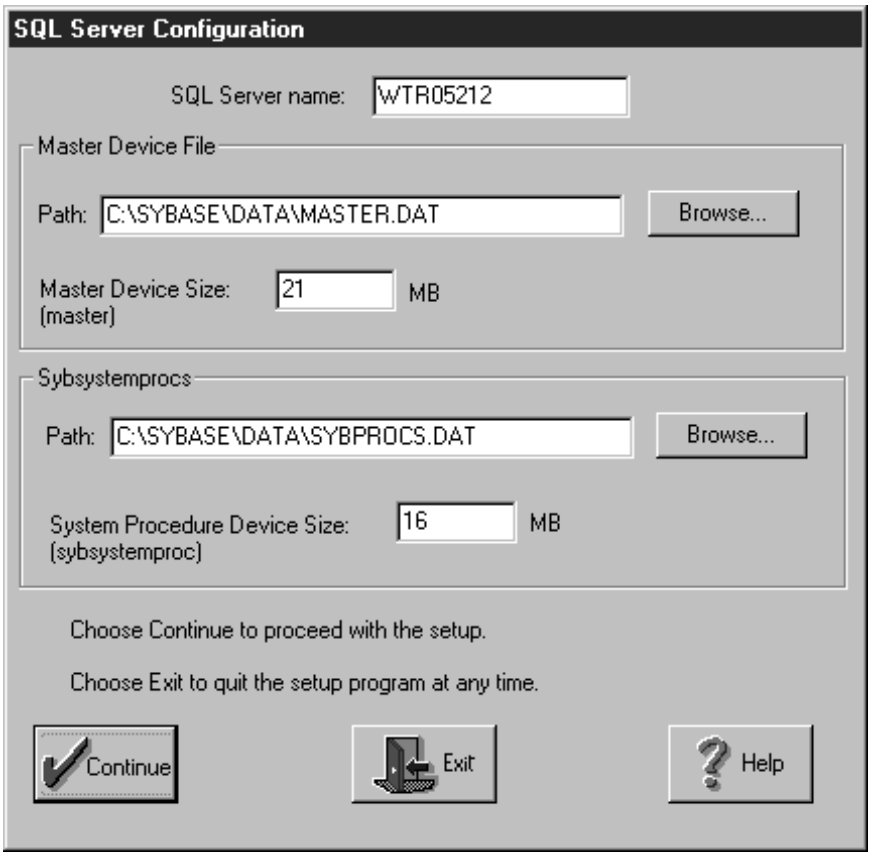


Figure 165. Sybase SQL Server Configuration

Accept the defaults for the server configuration screen click **Continue**.

Figure 166. Sybase Installation Complete

Select **Reboot machine** and click **Finish**.

After your machine reboots, do the following:

1. Select **Start menu->Programs->Sybase for Windows NT->Services Manager**.

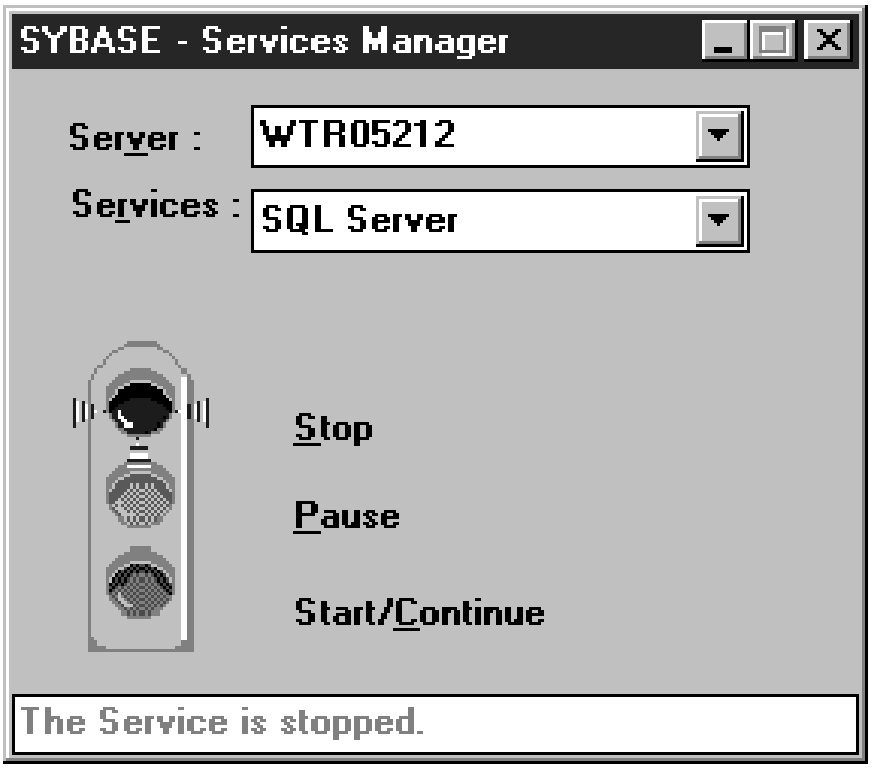


Figure 167. Sybase - Services Manager

2. From there, click on the **Start/Continue** button (the green light).

After you start the service, you have to do the following:

1. Select **Start menu->Programs->Sybase for Windows NT->SQL Server Manager**.

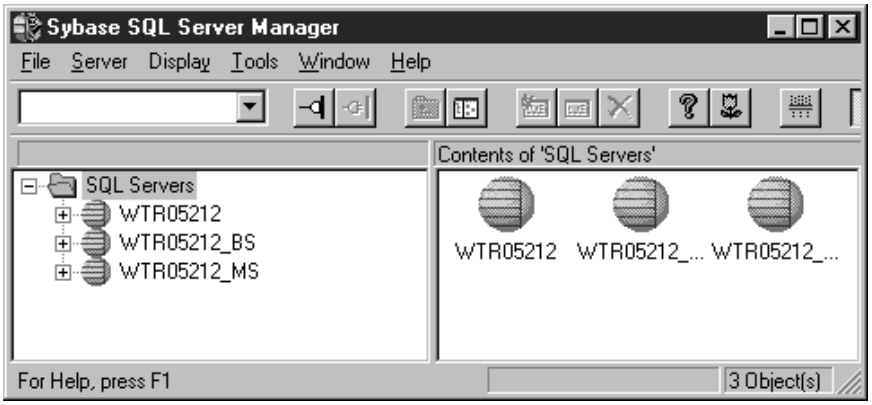


Figure 168. Sybase - SQL Server Manager

2. Select the first database of the three listed on the left.

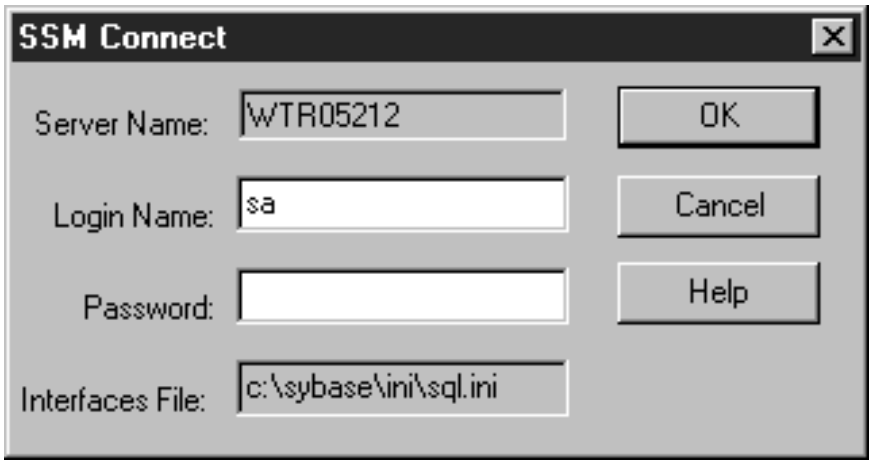


Figure 169. Sybase - SSM Connect

3. Log in as sa with no password. Click **OK**.

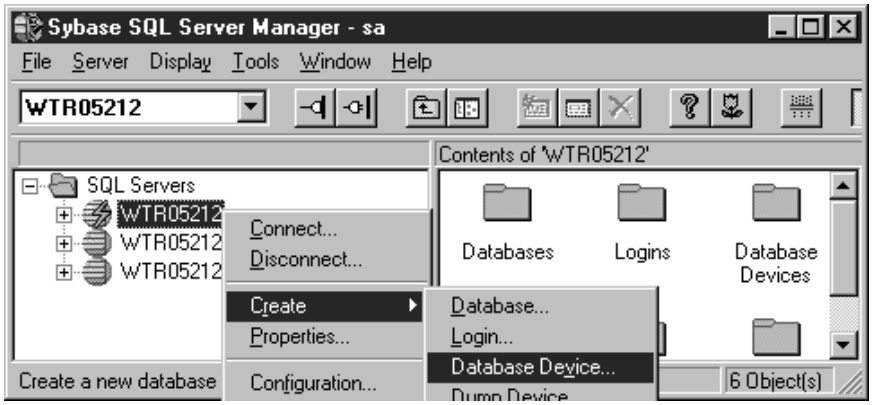


Figure 170. Sybase SQL Server Manager - sa

4. Right click on the database you logged into and select **Create->Database Device**.

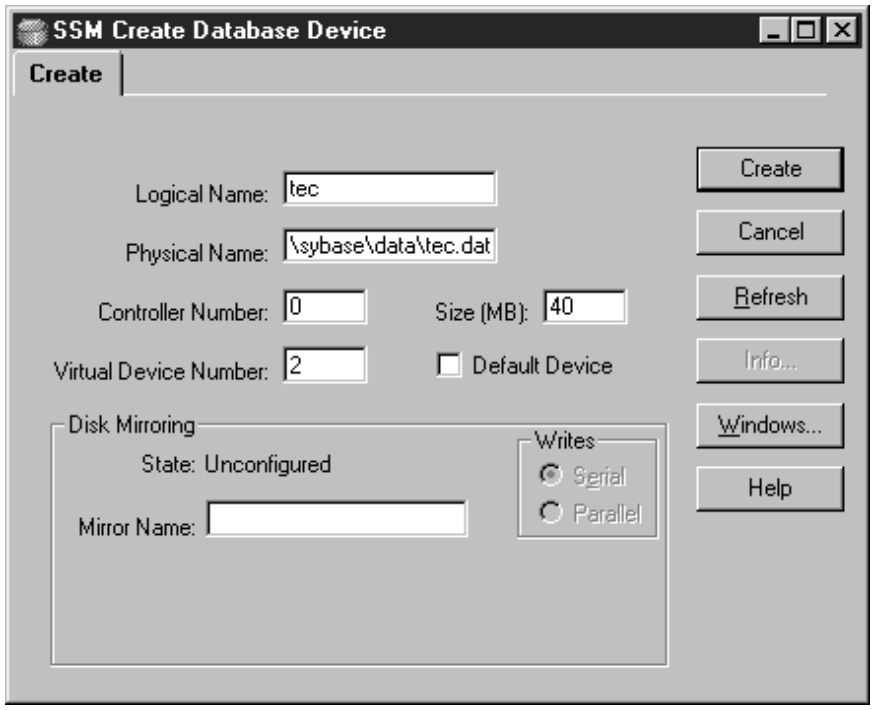


Figure 171. Sybase SSM Create Database Device

5. Name the Logical area (that is, tec).
6. Name the physical device (that is, c:\sybase\data\tec.dat).
7. Make the database 40M.
8. Select **Create**.

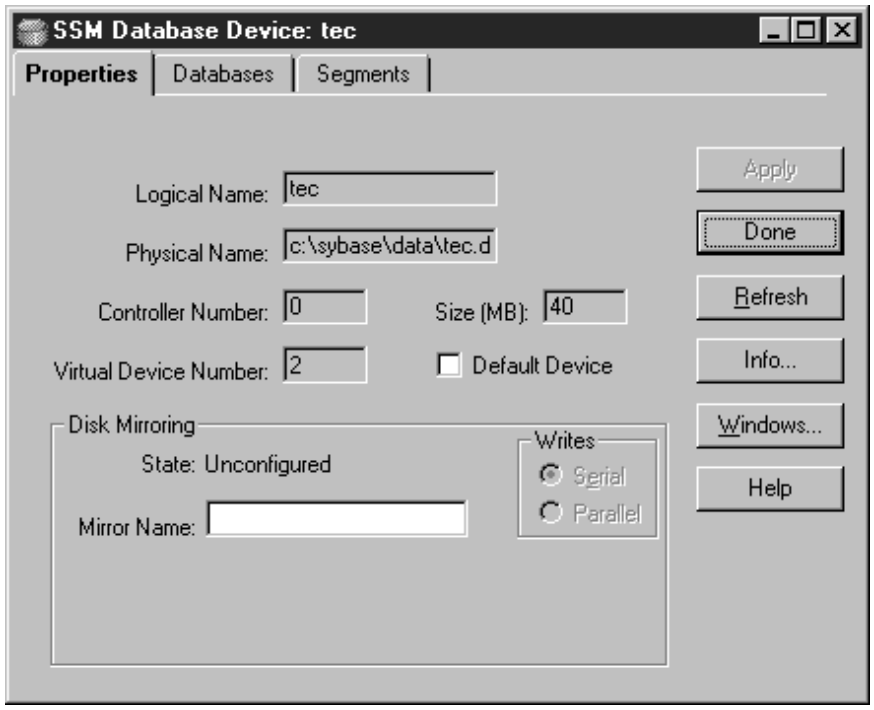


Figure 172. Sybase SSM Database Device: tec

9. Select **Done**.

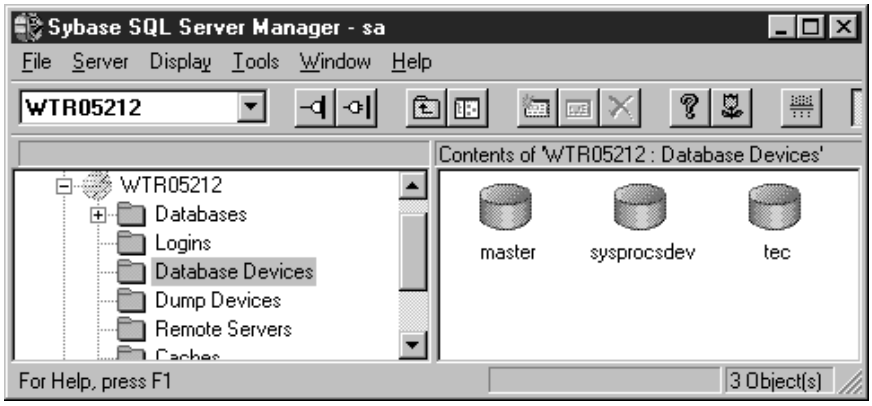


Figure 173. Sybase SQL Server Manager - sa

10. Confirm you have the device by double-clicking on the active database, (the one with the lightening bolt on it), then on **Database Devices**, and the tec device you created should appear there.

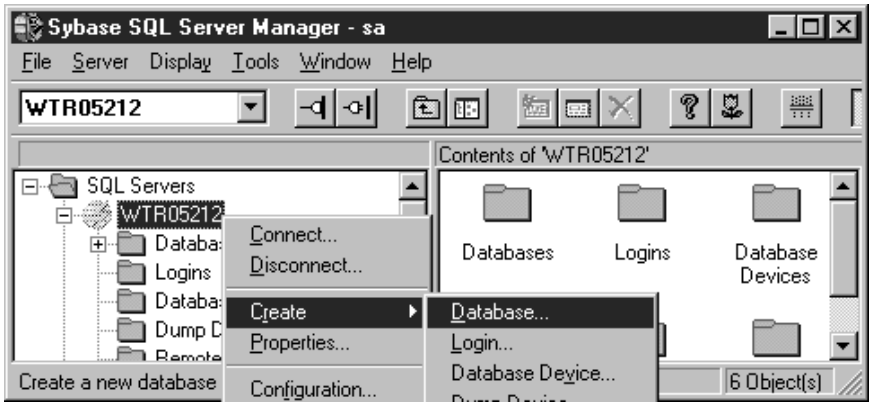


Figure 174. Sybase SQL Server Manager - sa

11. Right click on the database you're still logged on and select **Create->Database**.

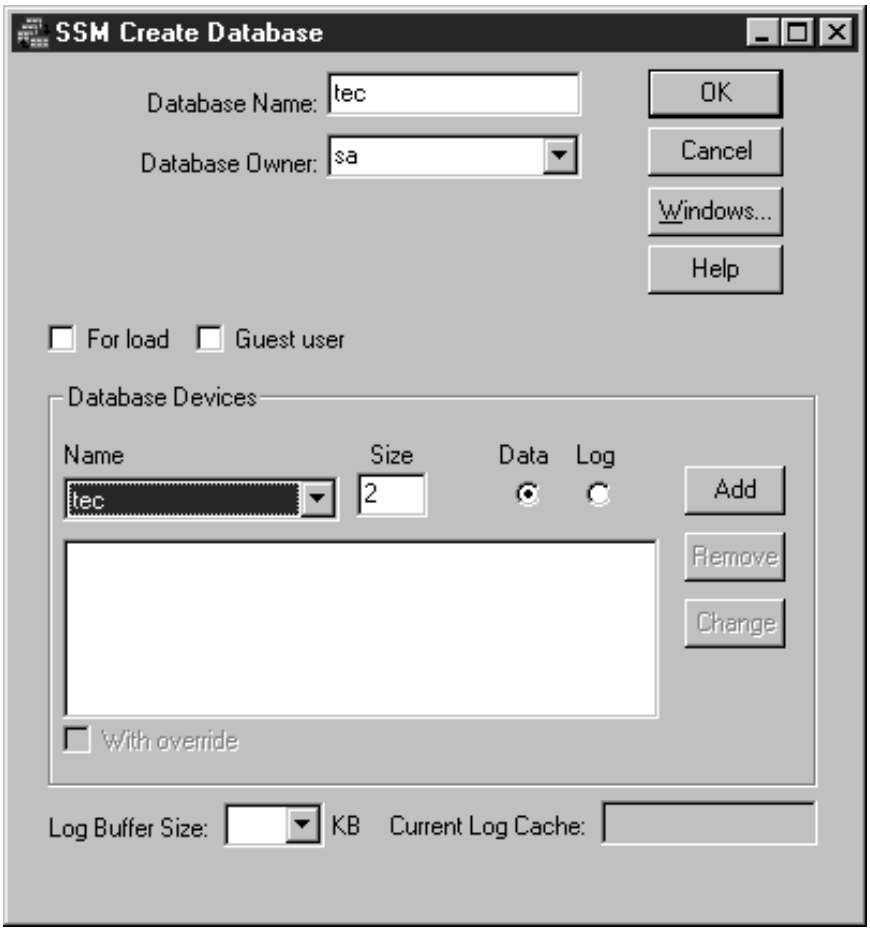


Figure 175. Sybase SSM Create Database

12. Name db the same as your server ID for the RIM object (tec).
13. Set the owner to sa for now. This will be changed later.
14. Set the db device to tec as created in the above.
15. Click **Add**.
16. Click **OK**.

If the installation did not add the environment variables DSQUERY, DSLISTEN and SYBASE, you should add them manually. An example of what they should be set to is shown in Figure 176 on page 296.

Update the PATH Windows/NT variable with the directory path of the Sybase executables and Dynamic Link Libraries (DLLs). All this can be done by clicking on the **System** icon on the Control Panel Settings group.

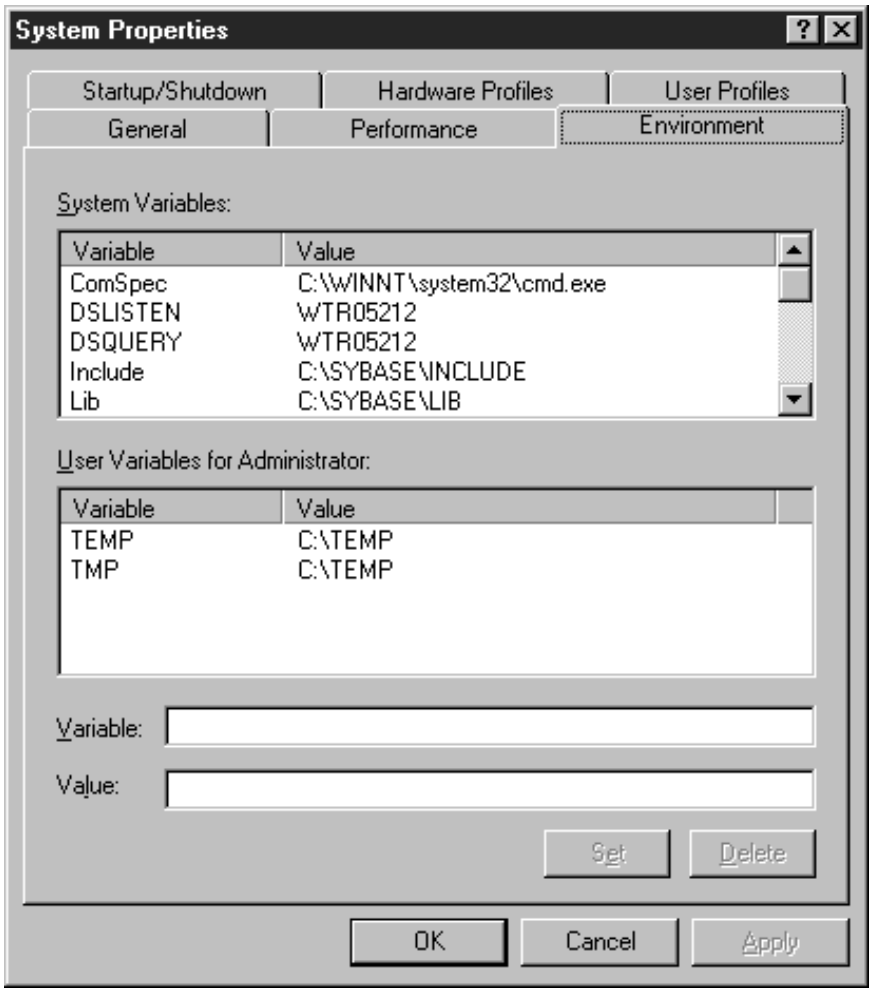


Figure 176. Windows NT Environment Menu

The following variables are set:

- DSQUERY defines the SQL server name that the client program tries to connect to if no name is specified with a command line option.
- DSLISTEN defines the name SQL server uses to listen for client connections.
- SYBASE defines the path of the Sybase directory.

Now we need to start Sybase SQL_Server.

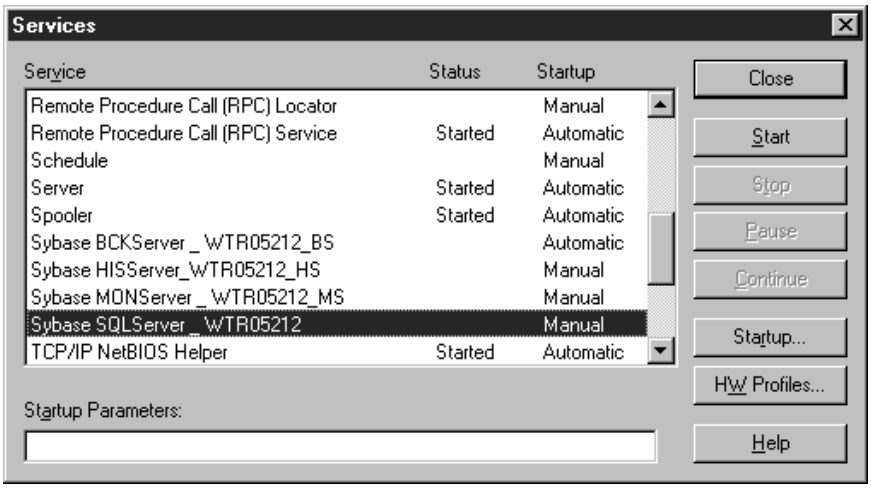


Figure 177. Windows NT Services Menu

Click **Start**.

To do automatic startup at system reboot, from the same screen, change startup option as follows:

- Click **Startup**.
- Change startup type to Automatic.
- Click **OK**.

To test Sybase functionality, we issue the following command (sp_help may produce lots of output):

```
bash$ isql -Usa
Password:
1> sp_help
2> go
1> exit
```

A.2 Installing the TME 10 Enterprise Console Server

We installed TME 10 Framework 3.2. All details about the installation and setup of the framework can be found in the base publications, as well as in *Setting UP a TME NT Environment*, SG24-4819-00.

We installed the TME 10 Enterprise Console server using the desktop dialogs. Make sure that the oserv database is backed up at all stages during this process. This provides for a recoverable checkpoint, which will not be possible if there is some failure during the installation due to lack of space or some other problem.

With TEC 3.1 the console is packaged separately from the server. The console is called TME 10 Enterprise Console 3.1 whereas the server is called the TME 10 Enterprise Console Server. Both are required for a complete installation.

The following shows TME 10 Enterprise Console and RIM installation.

Figure 178. Install Product

Select **TME 10 Enterprise Console 3.1 Server** and the client to install on.

During the installation of the server, we were prompted for certain pieces of information that are used to build the RDBMS Interface Module (RIM) for use by the TEC to connect to the SYBASE RDBMS:

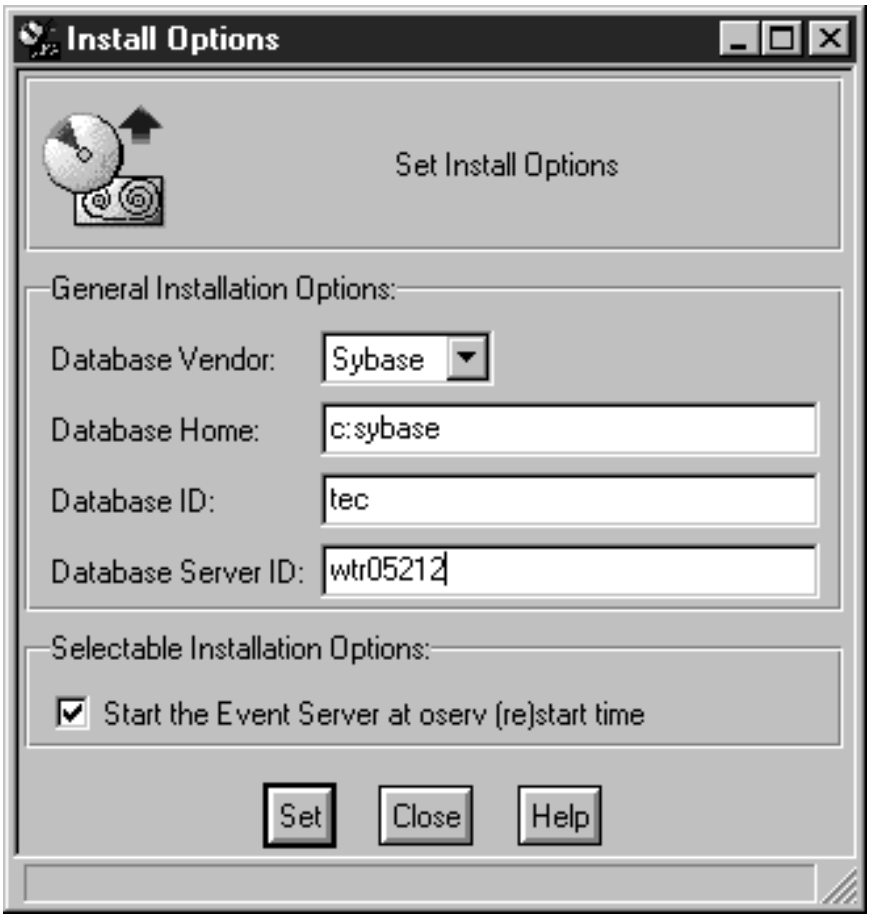


Figure 179. TEC Install Options

Fill in the general installation options as follows:

- Database Vendor

  Here we used SYBASE.

- Database Home

  This was c:sybase and should be the same value as in the SYBASE environment variable. (SYBASE environment must be set to c:\sybase.)

- Database ID

  This entry is the name of the Sybase database. We used tec.

- Server ID

This is the name of the RDBMS server and should be same as the value that you have specified for the DSQUERY NT environment variable. This is *not* the hostname of the RDBMS server machine. In our case this was wtr05212.

Click **Set and Close**.

Figure 180. TEC Install Options

Click **Install & Close**.

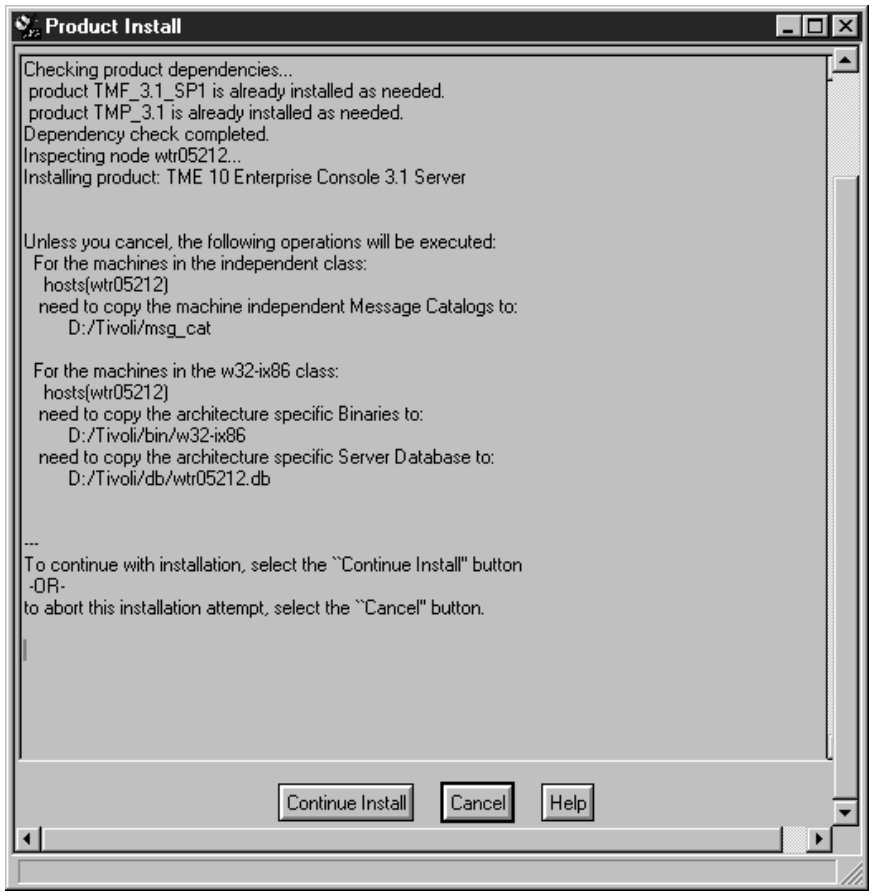


Figure 181. TEC Product Install

Click **Continue Install.**

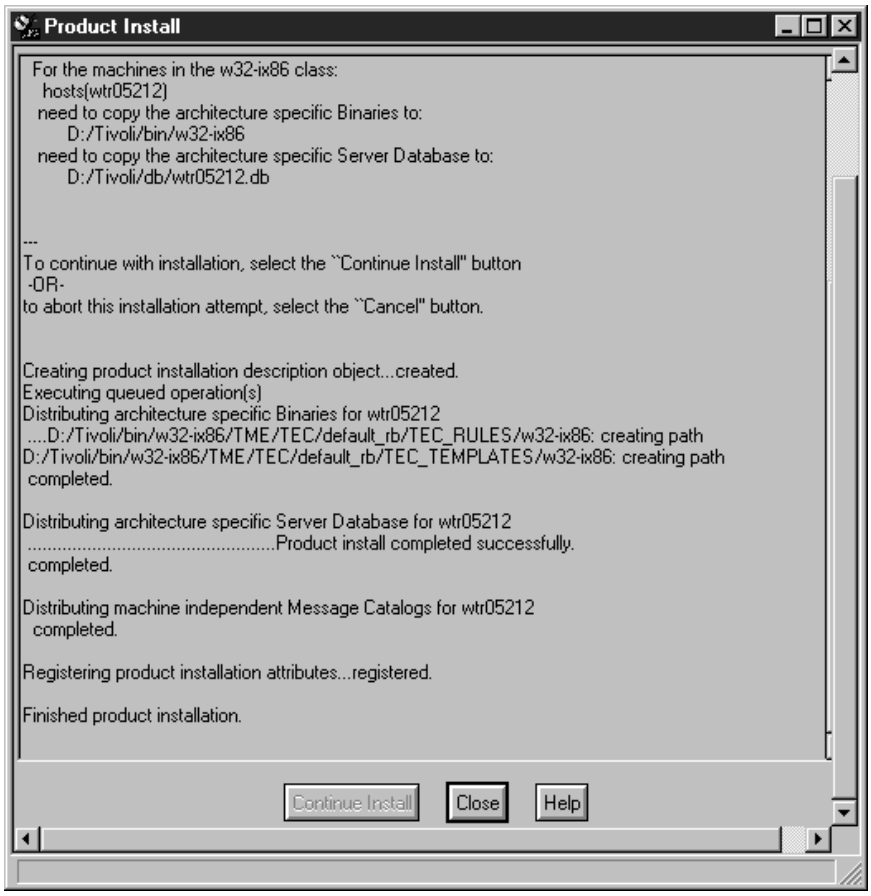


Figure 182. TEC Product Install

Click **Close**.

Next we have to complete the connection between the TEC server and the SYBASE database.

You will find it useful at this point to add some TME environment variables. From the System->Environment menus within the NT Control Panel (see Figure 176 on page 296) define the following variables. (D: is the drive where Tivoli was installed.)

1. Select the PATH variable, and add D:\tivoli\bin\w32-ix86\bin and D:\Tivoli\bin\w32-ix86\tools. You will have to log off and log on again for this to take effect.
2. DBDIR to D:\Tivoli\db.
3. BINDIR to D:\Tivoli\bin\w32-ix86.

A.2.1 Verify the RIM Object Installation

After the TEC Server installation has completed, there are a few things to check before the customization can continue.

First, check that the RIM built by the installation process contains the correct information. This can be listed with the wgetrim tec command.

Sample output from the wgetrim command follows:

```
bash$ wgetrim tec
@objcall/tcp service not found--using default
RIM Host:       wtr05212
RDBMS User:     tec
RDBMS Vendor:   Sybase
Database ID:    tec
Database Home:  c:sybase
Server ID:      wtr05212
Instance Home:
bash$
```

If any of the fields in the object are found to be incorrect, then the wsetrim command can be used to change their values.

Format of wsetrim command follows:

```
wsetrim -n new_name -d database -u user -H rdbms_home
-s server_id -I instance_home rim_name
```

If you need to change the vendor, you must delete the RIM using the command wdel @RIM:rimname and recreate it using the command wcrtrim.

If the RIM was not built, you can use the wcrtrim command to create it.

Format of wcrtrim command follows:

```
wcrtrim -i -v vendor -h host_name – -o host_oid‘ -d database
-u user -H rdbms_home -s server_id -I instance_home rim_name
```

A.2.2 Install the Database Components.

After we have checked that the RIM is valid, we need to define the TEC database. We did this with the BASH %BINDIR%\TME\TEC\SQL\ cr_tec_db.sh command. You are prompted for the password for the SYBASE super user, which if you have not changed it, will be null, so just press Enter in response to this prompt. Also ignore the message that database tec already exists since it was defined at Sybase install.

```
cd $BINDIR/TME/TEC/SQL
bash$ . cr_tec_db.sh

Looking up TME 10 system information...
The RIM object within this TMR is configured to define the
following database:

   Database Vendor:    Sybase
   Database Server ID: wtr05212
   Database Home:      c:sybase
   Database ID:        tec
   Database User ID:   tec

If this is correct, enter 'Y' to continue.  Otherwise, check the
configuration of the local RIM object or run this script from
the RIM host within the TMR you wish to configure.
y

What device do you want to put the TME 10 Enterprise Console Server
database in? (default is 'master')
tec

What size, in megabytes, do you want to make the
TME 10 Enterprise Console Server database? (default is 50)

At the Password: prompt, enter the DB 'sa' user ID's password...

++ isql -Usa -i ./cr_db.syb.sql
Password:

Msg 1801, Level 16, State 1:
Line 2:
Database 'tec' already exists
Extending database by 19456 pages on disk tec
Database option 'trunc log on chkpt' turned ON for database 'tec'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
Password correctly set.
Account unlocked.
New login created.
(return status = 0)
New user added.
(return status = 0)
++ set +x
++ isql -Utec -Ptectec -i ./cr_tbl.syb
++ set +x
bash$
```

When the script has completed, then we can check that all is well. You can do SQL query from the BASH shell or from the desktop WISQL32 by selecting **Programs->Sybase for Windows NT**. The next sample was done from the BASH shell.

```
bash$ isql -Utec -Ptectec
1> use tec
2> go
1> sp_help
2> go
```

The above command showed some of the system and data tables created by cr_tec_db.sh Script.

```
Name                           Owner                          Object_type
------------------------------ ------------------------------ ----------------------
tec_t_clt_req_log              tec                            user table
tec_t_evt_rec_log              tec                            user table
tec_t_evt_rep                  tec                            user table
tec_t_gem_threshold            tec                            user table
tec_t_isa                      tec                            user table
tec_t_op_ass_log               tec                            user table
tec_t_role                     tec                            user table
tec_t_severity                 tec                            user table
tec_t_slots_evt                tec                            user table
tec_t_status_event             tec                            user table
tec_t_status_task              tec                            user table
tec_t_task_rep                 tec                            user table
sysalternates                  dbo                            system table
sysattributes                  dbo                            system table
syscolumns                     dbo                            system table
syscomments                    dbo                            system table
sysconstraints                 dbo                            system table
sysdepends                     dbo                            system table
sysgams                        dbo                            system table
sysindexes                     dbo                            system table
syskeys                        dbo                            system table
syslogs                        dbo                            system table
sysobjects                     dbo                            system table
syspartitions                  dbo                            system table
sysprocedures                  dbo                            system table
sysprotects                    dbo                            system table
sysreferences                  dbo                            system table
sysroles                       dbo                            system table
syssegments                    dbo                            system table
systhresholds                  dbo                            system table
systypes                       dbo                            system table
sysusermessages                dbo                            system table
sysusers                       dbo                            system table
tec_p_clear_events             tec                            stored procedure

User_type       Storage_type    Length Nulls Default_name    Rule_name
--------------- --------------- ------ ----- --------------- ---------------

(return status = 0)
```

A.3 Installing the TME 10 Enterprise Console

We installed the TME 10 Enterprise Console using the desktop dialogs. Make sure that the oserv database is backed up at all stages during this process.

Figure 183. Install Product

Select **TME 10 Enterprise Console 3.1** and the client to install on. Click **Install & Close**.

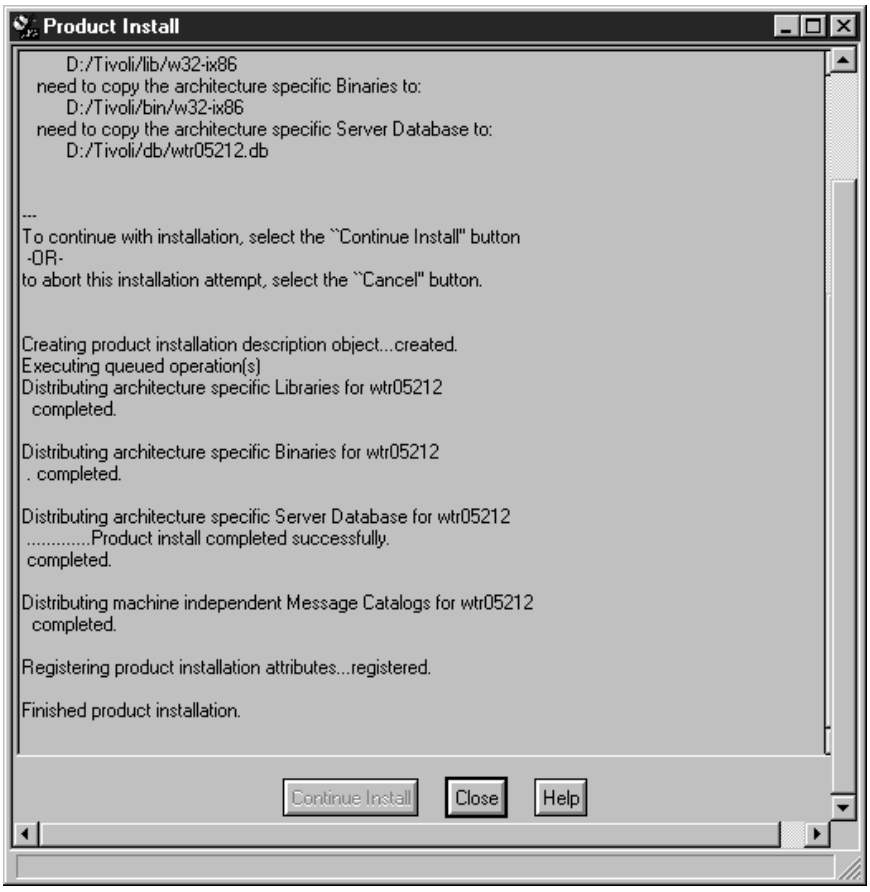


Figure 184. TEC Product Install

Click **Continue Install**. Once completed, click on **Close**.

A.4 Starting the TME 10 Enterprise Console Server

Before starting T/EC you *must* create a directory called TMP if it does not exist yet. Tivoli Enterprise Console will use this directory to allocate temporary space when compiling rules, and to allocate the following log files:

- Tec_dispatch
- Tec_master
- Tec_reception
- Tec_rule
- Tec_task

At this point you can start the TEC server with the wstartesvr command or from the desktop.

A.5 Setup TME 10 Enterprise Console

Before you can use the Enterprise Console you must do the following. You can use the TME desktop or BASH command to accomplish these tasks.

- Create an event console for each administrator. The command to use is wcrtconsole.
- You must assign an event group to a specific administrator before the event group will show up on the administrator's console. The command to use is wassigneg.

A.5.1 NT Sybase Parameters

The following parameters were modified to provide a better performance for the NT server.

After running the database creation scripts we modified some of the Sybase parameters. These parameters are modified using the Sybase server configuration program. To modify the parameters select the configuration option from the Sybase server manager screen shown in Figure 185.

Figure 185. Sybase Server Manager

The options can be changed by selecting each of the parameters as shown in Figure 186 on page 309.

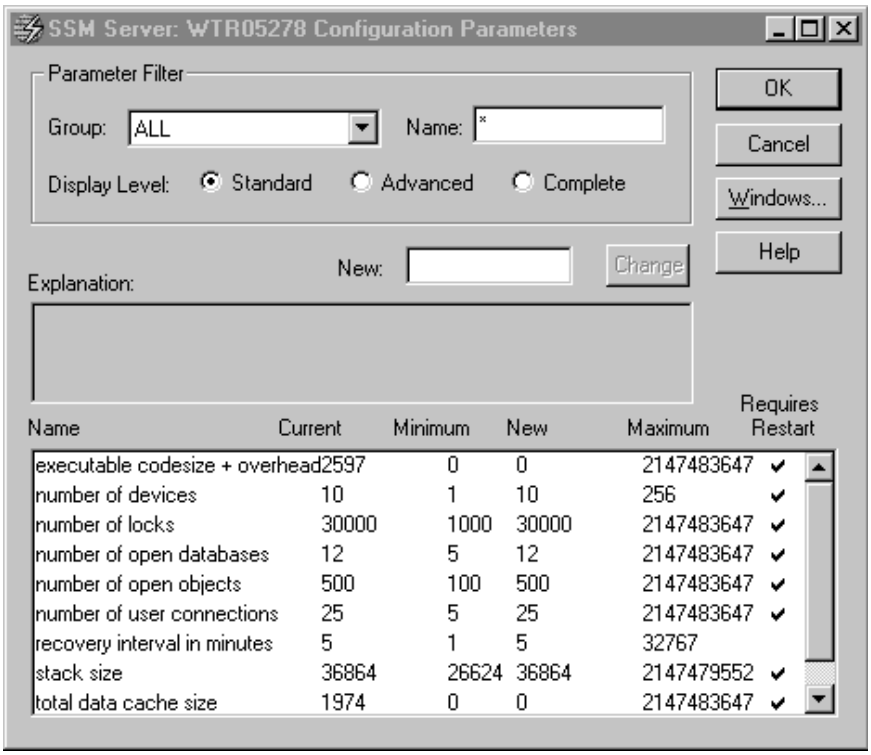


Figure 186. Sybase Configuration Parameters

We changed the following parameters:

```
allow sql server async i/o = 0
disk i/o structures = 64
default network packet size = 10240
max network packet size = 10240
max async i/os per engine = 4096
max async i/os per server = 4096
i/o polling process count = 5
time slice = 50
cpu grace time = 200
number of locks = 30000
```

In addition we changed a parameter in the NT registry as shown below:

```
Go to the following Registry Key and set the decimal value to "0"
(for LargeSystemCache)

\HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SessionManager\
MemoryManagement\LargeSystemCache
```

Appendix B. Special Notices

This publication is intended to help IBM and Tivoli services personnel to inplement the TEC application for customers. See the PUBLICATIONS section of the IBM Programming Announcement for TME and TEC products for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including

these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AS/400
IBM	NetView
RISC System/6000	RS/6000
Trouble Ticket	

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 315.

- *TME 10 Internals and Problem Determination*, SG24-2034
- *Implementing TME 10 in High Availability Environments*, SG24-2032
- *An Introduction to Tivoli's TME 10*,SG24-4948
- *A First Look at TME 10 Distributed Monitoring 3.5*, SG24-2112
- *Examples of Using TME 10 NetView for AIX V5 and NT*, SG24-4898
- *TME 10 Cookbook for AIX*, SG24-4867

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

C.3 Other Publications

These publications are also relevant as further information sources:

- *Enterprise Console User's Guide Version 3.1*, GC31-8506-00
- *Enterprise Console Rule Builder's Guide*, SC31-8508-00
- *Enterprise Console Adapter's Guide*, SC31-8507-00
- *Framework Reference Manual Version 3.2*, SC31-8432-01

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at `http://www.redbooks.ibm.com/`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type `GOPHER.WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks/`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to `softwareshop@vnet.ibm.com`
- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com/
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.html`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

□ Invoice to customer number

□ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

Special Characters

A

B

C

D

E

I

L

M

N

O

P

R

S

T

U

W

ITSO Redbook Evaluation

TEC Implementation Examples
SG24-5216-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
□Customer □Business Partner □Independent Software Vendor □IBM employee
□None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

_____

SG24-5216-00
Printed in the U.S.A.

SG24-5216-00