Tivoli. software

A Netcool OMNIbus Primer for IBM Tivoli Enterprise Console Administrators

White Paper

Todd West and Karen Durston

September 2006

Table of Contents

Introduction

About this Paper

The acquisition of the Netcool product suite presents several challenges for IBM Tivoli Enterprise Console (TEC) consultants, deployment specialists, and administrators. There are several areas where TEC and OMNIbus functionality overlaps and terminology and technology between the two products can be confusing. This white paper is intended to provide people familiar with installation and administration of TEC an overview of the features of OMNIbus and how those features map with similar TEC functionality. There are many products in the Netcool product suite, but this paper is concerned only with the core OMNIbus version 7.1 event server and Webtop version 1.3.1 products and how they relate to TEC 3.9 and the TEC 3.9 Web console.

Audience

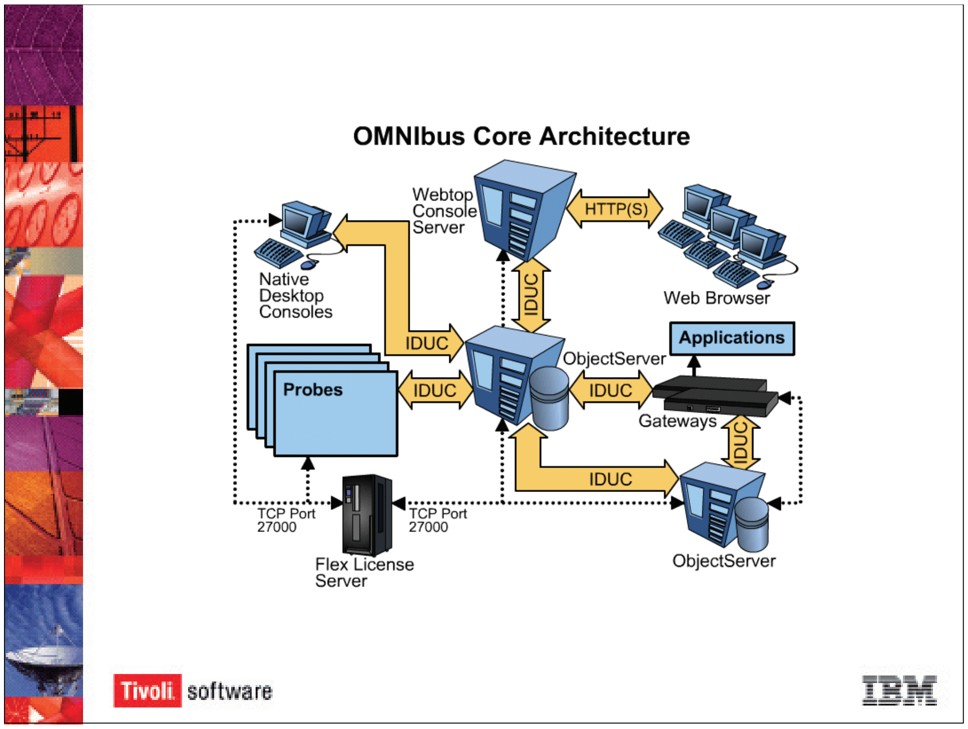
This white paper is intended for consultants, administrators, and deployment specialists with extensive experience working with IBM Tivoli Enterprise Console, but little or no exposure to Netcool OMNIbus.

White Paper

1 Overview

Netcool OMNIbus and Webtop provide a set of core products that perform a number of functions that overlap IBM Tivoli Enterprise Console (TEC) features. This white paper is intended to provide an overview of the core components of OMNIbus and Webtop and how those components relate to TEC. Included are discussions of Netcool terminology and event management as it relates to, and in some cases, differs from TEC. Unless otherwise noted, OMNIbus references are to version 7.1 and Webtop references are to version 1.3.1. For additional information on product convergence and development plans for OMNIbus and TEC, read *Netcool + Tivoli: delivering service management innovation*, at **ftp://ftp.software.ibm.com/software/tivoli/whitepapers/GC28-8458-00.pdf**.

2 OMNIbus Core Architecture



The core components for an OMNIbus installation consist of the following products:

- **Flex license server**: License validation server, queried by Netcool components when they are first run. There must be a minimum of one per Netcool environment. There is no equivalent component in a TEC environment. Keys are provided with each product as part of the installation media. By default, the license server listens for license requests on TCP port 27000.

- **OMNIbus ObjectServer**: This is an SQL database residing completely in memory. It receives events from a variety of monitoring sources and provides several event housekeeping procedures. The ObjectServer performs the functions provided by the TEC server, TEC Relational Database Management System Interface Module (RIM) objects, Framework authentication and authorization, event server database, and some basic TEC rule functionality. Events are delivered to the ObjectServer using an SQL Insert, Delete, Update, and Command (IDUC) communication protocol running over TCP/IP. Authentication and access are maintained and controlled within the ObjectServer database.

- **Probes**: These are passive monitors that listen to a data source, looking for defined events. Events are formatted and sent to the OMNIbus ObjectServer. These provide functionality similar to that provided by TEC event adapters, but the Netcool probes have a significantly more extensive parsing engine and parsing language, allowing greater event evaluation prior to delivery to the ObjectServer.

- **Gateways**: These are software applications designed to provide unidirectional or bidirectional communications between an OMNIbus ObjectServer and an external program. Some examples of gateways would be a trouble-ticket gateway or an OMNIbus-to-TEC gateway. In this instance, the term gateway is used to mean an application-to-application level gateway, rather than one of the network or transport layers of the Open Systems Interconnection (OSI) model. Historically, this type of functionality has been provided with TEC using either Tivoli Plus Modules, the Tivoli Data Warehouse, or custom applications.

- **Webtop**: This is an application that uses the Tomcat Apache Web server to provide OMNIbus Web console communications with an ObjectServer. This component is roughly equivalent to the TEC UI server and the TEC WebSphere-based GUI console. User access and role authorization is controlled through the Webtop server, with account and password information either maintained on the connected ObjectServer, locally on the Webtop server, or some combination of the two.

- **Dedicated consoles and command-line utilities**: Native desktop consoles provide administrative access to the ObjectServer. This functionality is roughly equivalent to the Java TEC console. There is also a set of commands that provide the ability to manage and administer an ObjectServer environment using a command line. Additionally, a complete command-line version of the capabilities provided with the Web console is provided using a Web Administration Application Programming Interface (WAAPI). These are roughly equivalent to Tivoli Framework w-commands, except that WAAPI commands are written and processed in XML formatted files.

2.1 Event Management

Although the ObjectServer can be configured to do high-level correlation similar to that provided by TEC rules, other Netcool products, such as Realtime Active Dashboards (RAD) and Impact, have been designed to retrieve information from the ObjectServer and analyze the events, depending on the focus of the product. Also, for existing TEC customers, a bidirectional event gateway exists to synchronize events between a TEC server and an OMNIbus ObjectServer. This is discussed in detail in "TEC Gateway" on page 23.

When a database is created using the **nco_dbinit** command, it builds and populates all tables and necessary configuration files to begin receiving events. Event information is stored in the **alerts.status** table. Because the ObjectServer is a database, events are delivered and modified using an Insert, Delete, Update, and Command (IDUC) communication protocol running over TCP/IP. This allows the event data and overhead from probes and monitors to be very small.

A difference in approach to event processing from TEC is that an event delivery is a direct update to the ObjectServer database, as opposed to TEC, which requires the event to be processed and evaluated prior to storage in the TEC event database. An OMNIbus event is the data for a network-based SQL command, so all tables, columns, and rows included in the event must match the ObjectServer schema. Because of this, event probes and monitors will validate that they are using the same schema as the ObjectServer when the probe or monitor is started. Part of the probe startup establishes a persistent TCP socket connection with the ObjectServer, facilitating fast event delivery to the ObjectServer.

Another difference in approach to event processing with TEC is that OMNIbus is designed to automatically detect duplicate events and only update those affected fields (such as time of delivery, duplicate count, event metrics). This is configured by default through the use of a database trigger. This duplicate event detection in OMNIbus is called *deduplication*.

Also provided in the default ObjectServer installation is an SQL trigger providing clearing-event correlation. Because OMNIbus started in the telecommunications industry, where thousands of router ports can repeatedly go up and down throughout the day, this is the default functionality, but it can be modified, just as can any database trigger.

2.2 OMNIbus and TEC Component Comparison Summary

TEC Component	OMNIbus Equivalent
Framework (Authentication)	User and group information created and maintained in the ObjectServer database.

TEC Component	OMNIbus Equivalent
Framework (Authorization)	User roles defined and maintained in the ObjectServer database.
Framework (RIM Object)	Probes and monitors communicate with an IDUC communication process running on the ObjectServer, which handles database updates. Uses TCP/IP for underlying communication protocol.
Framework (Event Gateways)	None. If necessary, hierarchical ObjectServers could be deployed.
TEC Event Server	ObjectServer.
UI Server and WebSphere TEC Console Server	Webtop.
Java event console	Conductor, Administrator (nco_config) consoles.
External event database	SQL database running in server memory, part of ObjectServer installation.
Command-line framework commands	**nco_sql** to access ObjectServer tables, columns, and rows; Webtop Administrative Application Programming Interface (WAAPI).

2.3 Flex License Server

When Netcool servers, clients, and many commands are run, they first query a Flex license server to determine if a valid license exists for the software. A minimum of at least one Flex license server per Netcool environment is required, although multiple servers can be configured if network topology or redundancy requires additional servers. Once a Netcool application has obtained a valid license, if the license server goes down for some reason, the application will continue to run for another 24 hours. This provides sufficient time to either fix the problem server or direct the application to a backup server.

The processing overhead for a license server is minimal, allowing the server software to be installed on almost any server-level system available to the Netcool applications with TCP/IP.

Note: License keys are now provided with the installation media. Netcool application versions prior to the IBM acquisition required specially generated license keys from Netcool support.

2.3.1 Flex License Server Installation Overview

The primary installation tasks are:

UNIX or Linux:

1. Set the **$NCLICENSE** environment variable to the common installation directory (default: **/opt/netcool/common/license**).
2. Extract the installation media to the target license server.
3. Run the installation script, **license_install**.
4. Copy all license files to **$NCLICENSE/etc**. License file names must be in the format **<filename>.lic**.

Note: Each license file must be modified to include the host name of the license server.

5. Start the license server using the following command:

```
$NCLICENSE/bin/nc_start_license &
```

6. Test operation using the following command:

```
$NCLICENSE/bin/nc_print_license
```

7. Configure the server to automatically start the license server at system startup.

Note: All variables must be set on the system so as to be available to programs running on the server. For example, include the definitions in the **/etc/profile** file on a Linux system.

Windows:

1. Extract the installation media to the target server.
2. Run the **setup.exe** program.
3. Copy all license files to **%NCLICENSE%\etc**. License file names must be in the format **<filename>.lic**.
4. Restart the **NCO Flex License Manager** service to read all the license files.
5. Verify the license server operation using the tool **Start > Programs > Netcool Suite > Flex License > LMTools**.

2.3.2 Netcool Client Access to License Server

On UNIX, Linux, or Windows systems, any Netcool program requiring license authentication knows the location of the license server by the environment variable NETCOOL_LICENSE_FILE. The default format for the variable is:

```
27000@<license_server_hostname>
```

27000 is the default listening port of the license server.

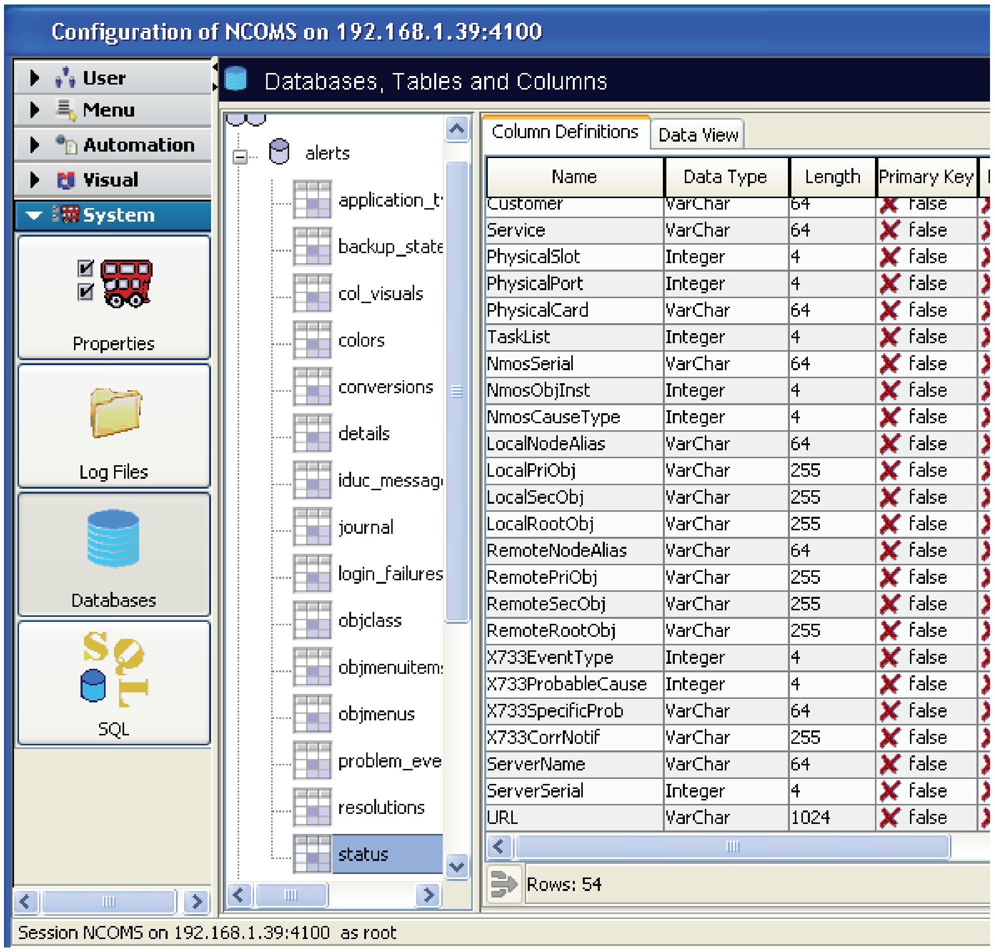
2.4 OMNIbus ObjectServer

The Netcool OMNIbus event server consists of the ObjectServer, an SQL database running entirely in memory, optimized to receive and process incoming events at a very high rate. The general approach to managing events with the ObjectServer is to provide fast event processing so that all enterprise events can be channeled to a single ObjectServer. The ObjectServer provides built-in general event housekeeping, such as automatic elimination of duplicate events, and automatic closing of related effect events when a clearing event is received.

Similar to IBM Tivoli Framework task execution, OMNIbus provides the ability to run automated responses to received events or event state changes. Because OMNIbus uses very limited process control across host systems, automation responses are limited to running programs on the ObjectServer or another system controlled by Netcool Process Automation (PA), or running a Common Gateway Interface (CGI) script on a connected Webtop server.

2.4.1 ObjectServer Instance Creation

Once the ObjectServer software has been installed, you must create the ObjectServer database instance using the **nco_dbinit** command. When this command is run, a default ObjectServer instance is created. The ObjectServer initially consists of nine databases (alerts, catalog, custom, master, persist, security, service, tools, and transfer). Event data is stored in the **alerts.status** table. By default, this table consists of 53 columns, although custom column definitions can be created if necessary. The ObjectServer instance can be viewed and administered using the OMNIbus Administrator console. An example view of the administrator console is shown in the graphic below:



Name	Data Type	Length	Primary Key
Customer	VarChar	64	✗ false
Service	VarChar	64	✗ false
PhysicalSlot	Integer	4	✗ false
PhysicalPort	Integer	4	✗ false
PhysicalCard	VarChar	64	✗ false
TaskList	Integer	4	✗ false
NmosSerial	VarChar	64	✗ false
NmosObjInst	Integer	4	✗ false
NmosCauseType	Integer	4	✗ false
LocalNodeAlias	VarChar	64	✗ false
LocalPriObj	VarChar	255	✗ false
LocalSecObj	VarChar	255	✗ false
LocalRootObj	VarChar	255	✗ false
RemoteNodeAlias	VarChar	64	✗ false
RemotePriObj	VarChar	255	✗ false
RemoteSecObj	VarChar	255	✗ false
RemoteRootObj	VarChar	255	✗ false
X733EventType	Integer	4	✗ false
X733ProbableCause	Integer	4	✗ false
X733SpecificProb	VarChar	64	✗ false
X733CorrNotif	VarChar	255	✗ false
ServerName	VarChar	64	✗ false
ServerSerial	Integer	4	✗ false
URL	VarChar	1024	✗ false

Rows: 54

Session NCOMS on 192.168.1.39:4100 as root

Because the ObjectServer runs in memory, and to increase processing speed, the ObjectServer databases do not support journaling or transaction rollbacks. To provide protection from server power loss or some other type of server failure, the ObjectServer periodically writes memory regions to the hard disk. By default, this is done every sixty seconds. In the event of a system failure, the ObjectServer state is recovered from the latest disk save. Any probes or monitors connected to the ObjectServer will buffer their events at the source until the ObjectServer is again available.

2.4.2 Event Views and Filters

When working with OMNIbus events, either viewing or modifying, the consoles display events using a combination of *filters* and *views*. Filters define which rows of data are displayed. Views define which columns of event data are displayed. A Netcool view is essentially the same as a summary view with a TEC console. A Netcool filter is equivalent to an event group detail within a TEC console.

2.4.3 ObjectServer Automation

Automation is the capability to make an automatic response to an event or pattern of events received at the central event server. The automation capability is similar to TEC's ability to run external programs or tasks during TEC rule processing.

OMNIbus ObjectServer ships with a default set of automations. Like TEC, there is the capability to create custom automations. More complex enhancements to automation can be developed using the additional product, Netcool Impact. Impact allows a business monitor perspective to be layered on a system monitor perspective. Specifically, Impact allows you to implement policy, enrich events and create synthetic events.

An example of a policy would be to check a database of scheduled maintenance when a device reports an outage. Enriching events means to add contextual information to an event and is similar to using TEC BAROC definitions to create new attributes and using TEC rules to fill in additional event values. An example of event enrichment which provides a business perspective is to add the name of the person to contact for a given problem. Synthetic events are created by the ObjectServer in response to reported events. An example of a synthetic event is to report that a service, such as printing, is not available when the printers are not available. TEC has the capability to create synthetic events using the *generate_event* template.

Netcool generally uses a modular approach to product packaging in which the customer gets a basic level of functionality (OMNIbus) and pays for additional function if needed (Impact). TEC provides a more comprehensive set of functions within a standard TEC installation. Advantages to the Netcool approach may not be apparent looking strictly at functionality but can be appreciated when looking at performance and scalability. Netcool Impact uses a separate database so it does not have a large impact on the ObjectServer operation.

Automation in OMNIbus is implemented using *triggers* and *procedures*. Triggers automatically run when the ObjectServer detects an incident associated with a trigger. Procedures either run SQL commands to manipulate data in an ObjectServer database or run external programs on a networked system using Netcool process control. In TEC, when to run is determined by the event header and the type of event (plain, timer, or redo). Procedures are comparable to the actions defined within TEC rules.

Generally, TEC rules are used to modify events under evaluation and events already stored in the event database. With OMNIbus, because the automation language is SQL, events are modified either by triggers, procedures, or direct database modification by an OMNIbus administrator. The database can be modified using SQL commands with the **nco_sql** command. There is also a graphical interface to facilitate the creation of procedures (**nco_config**) in OMNIbus.

2.4.3.1 Triggers

Triggers determine when automation procedures will run.

In OMNIbus, triggers are organized into groups, called trigger groups, for management of multiple triggers. TEC rules are not viewed using a graphical console, so there is not anything comparable to a trigger group.

OMNIbus triggers are significantly different from processing TEC rule actions. Database triggers in OMNIbus are based on SQL statements. There are graphical interfaces to help create triggers and procedures in OMNIbus.

Triggers can be assigned a priority to determine the order of execution. In TEC, the order of execution is determined by the order of rules in the rules file.

In both products it is possible to create circular dependencies and it is up to the administrator creating or modifying the automations to avoid them.

2.4.3.2 Trigger Types

There are three types of triggers:

1. **Database triggers**: These are caused in response to a condition, defined in the trigger, in the ObjectServer database. Examples of database triggers would be:
   - if the day of week is Monday
   - if severity is greater than three
   - if duplicate events are coming in faster than every sixty seconds

   In TEC, this is similar to testing attribute values in an event header.

   Database triggers in OMNIbus can be set to test prior to or after the test condition using the keyword BEFORE and AFTER. Before modification, for example, you might want to test for authorization to modify. An example of an AFTER action would be to signal if the attribute value has crossed a threshold.

2. **Temporal triggers**: These triggers activate after a specified period of time. These are similar to timer rules in TEC. An example of a temporal rule is to purge events which have aged beyond a predefined time without modification.

3. **Signal triggers**: These are actions taken when a signal is raised by a system or sent on a user-defined signal. TEC’s sequential computational model does not

accommodate interrupts. The closest comparison in TEC are change rules, which cause reevaluation of events when field values change, for example, when the severity of an event is increased. However, the PROLOG engine in TEC operates in a sequential manner and does not respond to interrupts. There is no mechanism in TEC for a user to signal the TEC rules engine other than sending in a specifically defined event which is queued up and handled in order. Examples of OMNIbus signals are:

– a system signal can be sent on system shutdown or startup, when a backup succeeds or fails, or when a process connects or disconnects to the ObjectServer among other reasons.
– A signal can be raised if an attribute value has crossed a threshold after a change to the database, for example, the severity is too high.

2.4.3.3 Programming Constructs in Triggers

Regardless of the type of triggers, the following capabilities are common. Each trigger can be individually enabled or disabled (groups of triggers can also be enabled or disabled together). Each trigger can have debugging enabled which will increase logging information. TEC has a similar capability by using the **trace** directive with a TEC rule, then compiling and loading the rule base with tracing on. In OMNIbus, there are implicit variables indicated by **%trigger**. These are variables that are automatically created based on the type of trigger, for example, **%trigger_previous_rowcount**. See the product documentation for details.

The programming constructs are:

– assignment (SET) statement
– conditional statements (IF THEN ELSE and CASE WHEN)
– looping statements (FOR and FOR EACH ROW)

2.4.3.4 Automation Procedures

Procedures specify the action taken when a trigger is activated. They are programmed responses to events invoked by the ObjectServer. They are similar to the action portion of a TEC rule.

Procedures are either SQL procedures or external procedures. SQL procedures use, and potentially modify, the ObjectServer database. External procedures run either on the ObjectServer host or a remote host and can be roughly compared with the TEC **exec_program**.

Authority to modify automations can be controlled through role, group and user configuration.

Tasks within the Tivoli Framework provide the capability to package scripts to run on different platforms within one task. This provides a uniform interface so the rules language does not have to differentiate between, for example, **delete** on a Windows system and **rm** on a UNIX system. There is not a similar capability on OMNIbus. Eternal procedures are platform dependent.

2.4.3.5 Statements

For SQL procedures, the following actions are available:

- ALTER SYSTEM BACKUP
- ALTER SYSTEM SET
- ALTER SYSTEM DROP CONNECTION
- ALTER TRIGGER
- ALTER TRIGGER GROUP
- ALTER USER
- UPDATE
- INSERT
- DELETE
- WRITE INTO
- RAISE SIGNAL
- {CALL | RUN} PROCEDURE
- CANCEL
- BREAK

There are also internal functions which may be used (for example, to get the current time and date) which fall into one of three categories: time functions, data conversion functions, and mathematical functions. External procedures, like scripts in TEC, are not limited to this set of actions.

2.4.3.6 Shipped Automations

Section 4.12 Automation Examples of the *Netcool/OMNIbus Administration Guide V7* provides specifics on predefined automation capabilities. Predefined automation capabilities can be customized. This is a high-level view to provide some examples and is not comprehensive.

Predefined automations are in the file **automations.sql**. Be aware that some of the automation capabilities are disabled by default. Remember to review and select those that you would like to enable.

Although automations provide similar functionality to that provided within TEC rules, there is not a simple direct comparison between the two. The automations provided with a default ObjectServer installation are designed to optimize event reception without sacrificing event processing speed. Additional TEC functionality, such as complex event correlation, is generally handled with another of the Netcool family products, such as Impact or RAD.

2.4.3.6.1 Event Correlation

By default, in OMNIbus, a **Link Up** can cancel a **Link Down,** causing both events to be deleted from the ObjectServer. Any additional correlation capabilities would have to be customized using a trigger or procedure. In general, this is discouraged and would be better performed using a product such as Netcool Impact.

2.4.3.6.2 Event Clearing and Housekeeping

A temporal trigger checks every minute for entries in the details table that do not have a corresponding entry in the alerts.status table. If there is no corresponding status, the details are deleted.

A default trigger records the time when an event state modification is made.

A default trigger deletes an event when severity is equal to zero and the event has not been modified for two minutes.

2.4.3.6.3 Duplication

If a duplicate event arrives, there is a procedure to increment the count of the previously received instance of that kind and not insert a duplicate event.

TEC programmers can use the **dup_detect** facet to specify precisely which events are to be considered duplicates. In OMNIbus, if the following column values are the same in two events, the events are considered duplicates: probe, node, interface, category, problem or resolution, and severity. This can be customized by changing the automation definition.

2.4.3.6.4 Notification

If an event is of status critical and it has not been acknowledged within thirty minutes, send an e-mail to a predefined administrator.

2.4.4 Process Control

OMNIbus provides tools to run programs external to the ObjectServer. These tools are collectively known as *process control*. Process control is used, among other things, to monitor and automatically restart the operational state of OMNIbus components when they have failed. This aspect of process control is similar to the TEC Health Agent introduced in TEC 3.9 Fixpack 5.

Process control is provided by a process control agent installed on a target machine. As agents are defined within an OMNIbus environment, they can be defined so that all

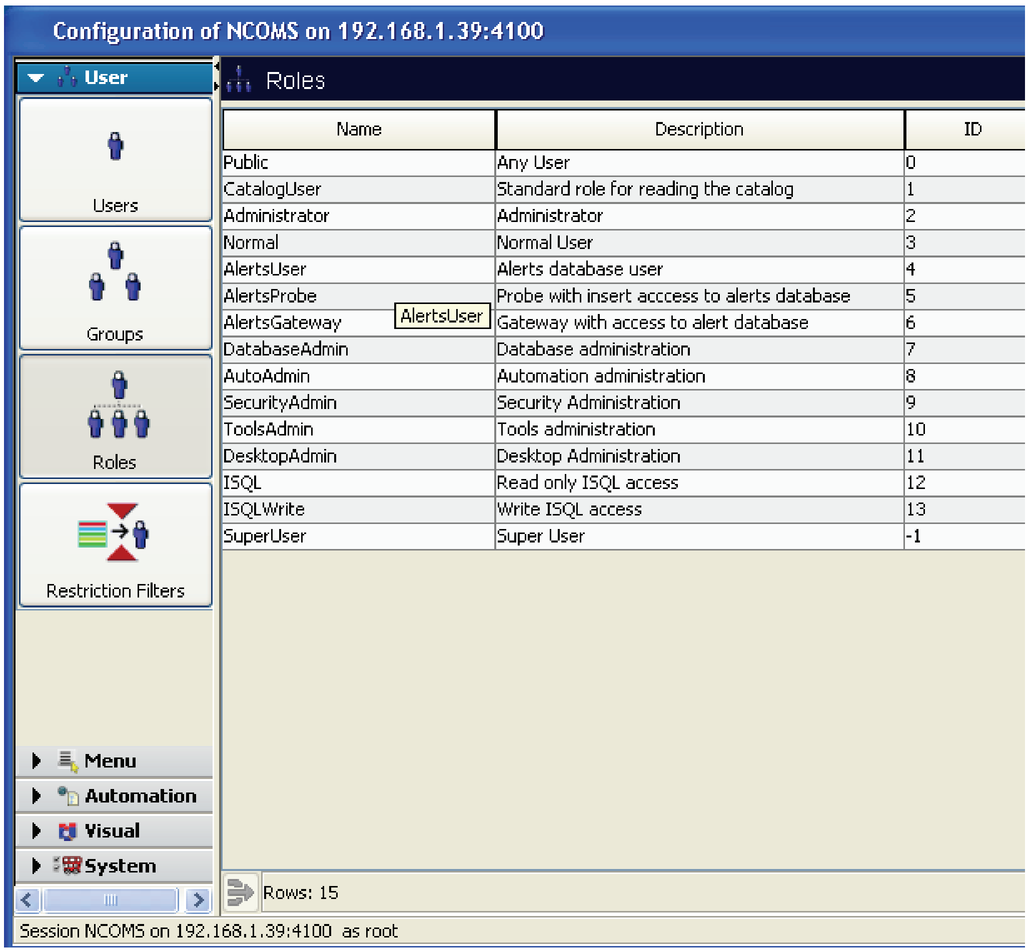
OMNIbus servers are aware of the agent locations. When an automation needs to run a program, the ObjectServer contacts the process control agent on the target server, passing the automation information. The remote target then runs the automation specified.

Note: Process control agents on Windows machines can only connect to process control agents on other Windows machines. Process control agents on UNIX machines can only connect to process control agents on other UNIX machines. External procedures cannot pass between these different environments. This aspect of Netcool OMNIbus does not scale as well as the combination of Tivoli Framework and tasks, but is suitable for most OMNIbus installations.

2.4.5 User Authentication and Authorization

OMNIbus users, groups, and roles are defined with the Administrator console and stored in the ObjectServer instance. As of version 7.1, there is no way to use an external directory, such as LDAP, to manage users, groups, and roles. Webtop also maintains a separate list of user definitions, but this information can be automatically migrated from an OMNIbus ObjectServer.

Shown below is a view of the default roles available through the Administrator console:



Roles

Name	Description	ID
Public	Any User	0
CatalogUser	Standard role for reading the catalog	1
Administrator	Administrator	2
Normal	Normal User	3
AlertsUser	Alerts database user	4
AlertsProbe	Probe with insert acccess to alerts database	5
AlertsGateway	Gateway with access to alert database	6
DatabaseAdmin	Database administration	7
AutoAdmin	Automation administration	8
SecurityAdmin	Security Administration	9
ToolsAdmin	Tools administration	10
DesktopAdmin	Desktop Administration	11
ISQL	Read only ISQL access	12
ISQLWrite	Write ISQL access	13
SuperUser	Super User	-1

Rows: 15

Session NCOMS on 192.168.1.39:4100 as root

2.4.6 OMNIbus Server Installation Overview:

UNIX or Linux:

1. Make installation media available to the target server.

2. Set the **$OMNIHOME** variable to the installation directory (default: **/opt/netcool/omnibus**).

3. Set the **$NETCOOL_LICENSE_FILE** variable to the port and host name of a valid license server using the following format:

```
NETCOOL_LICENSE_FILE=27000@<license_server_name>
```

4. Run the **OINSTALL** script

5. Run the **$OMNIHOME/bin/nco_xigen** utility to set the server communication parameters.

6. Initialize the ObjectServer database using the following command:

```
$OMNIHOME/bin/nco_dbinit -server <SERVERNAME>
```

Note: For all commands requiring a **<SERVERNAME>**, remember that **<SERVERNAME>** is unique to an OMNIbus environment and does not automatically correspond to the host name. The default **<SERVERNAME>** is **NCOMS**.

7. Start the ObjectServer with the following command:

```
$OMNIHOME/bin/nco_objserv -name <SERVERNAME> &
```

8. Configure system startup scripts to automatically start the ObjectServer at system start.

9. Start the Conductor console to verify operation:

```
$OMNIHOME/bin/nco
```

Windows:

1. Make installation media available to the target server.

2. Run **setup.exe**.

3. When prompted, enter the license server and ObjectServer information.

4. Initialize the ObjectServer database with the following command:

```
%OMNIHOME%\bin\nco_dbinit -server <SERVERNAME>
default: NCOMS
```

5. Restart the Netcool/OMNIbus ObjectServer service.

6. Verify ObjectServer operation using the **Start > Programs > Netcool Suite > Event List** command.

2.5 Probes

This section compares OMNIbus probes and TEC adapters in terms of:

- general capabilities
- configuration
- installation and distribution
- communication and reliability
- terminology

Probes are documented in the manual *Netcool OMNIbus v7 Probe and Gateway Guide*.

2.5.1 Comparison of General Capabilities

Probes are software programs that collect event information and send it to the ObjectServer. To people familiar with TEC, probes perform the same function as TEC adapters. Probes get their data from devices, log files, data bases, APIs, and CORBA, among others.

A probe consists of the following files:

- **Binary**: The program that retrieves and tokenizes the events from the monitored source.
- **Properties**: A configuration file for runtime settings. This will be explained at a high level in the next section "Customizing the Definition of Duplicate Events" on page 16. To people familiar with TEC, this is similar to the adapter's configuration file.
- **Rules**: Maps tokens into attribute fields in the event. This capability is combined in TEC in the adapter executable code and the format file.

A probe may have additional files as needed but the three mentioned above are common to all probes.

Rules files in OMNIbus are used by probes to create events. Note that the OMNIbus rulesfile is very different from the TEC event server rules files. The OMNIbus rules file combines data parsing and event formatting capabilities in a different way than TEC adapters. Examples of these differences are given in the next two sections.

2.5.2 Customizing the Definition of Duplicate Events

This section shows an example of how TEC and OMNIbus differ in their detection and handling of duplicate events.

In TEC, the **dup_detect** facet in the BAROC file specifies which attributes must match for two events to be considered duplicates. The BAROC file cannot be changed at run time. If the event definitions are changed, the central rule base needs to be recompiled and reloaded and the event server stopped and started. In OMNIbus, the **Identifier** attribute determines which events are to be considered duplicates. The **Identifier** attribute is a concatenation of several other attributes. Care should be taken, as with the definition of the TEC **dup_detect** facet to include attributes which are essential to define duplicates and not to include attributes which are not indicative of the same event. In OMNIbus, the **Identifier** field is assigned in the rules file by the probe using code such as:

```
@Identifier = @Node + @AlertKey + @Summary
```

See the section *1.5 Creating a Unique Identifier* of the *Netcool OMNIbus v7 Probe and Gateway Guide* for more information about creating identifiers.

While the duplicate detection capability is similar, the concatenation processing is done on the distributed side of the architecture. The ObjectServer then simply compares the one **Identifier** attribute to identify duplicates. To customize the definition of what constitutes duplicate events, a programmer can modify the code by adding or removing attributes on the adapter side. This technique is more flexible in that it does not require the ObjectServer to be stopped in order to modify the definition of duplicates for a given probe.

2.5.3 Modifying a Probe's Rules File

OMNIbus probes are implemented in a proprietary interpreted scripting language. TEC adapters can be written in several languages, using the Event Integration Facility (EIF), as long as that language has a call interface to the EIF libraries. This makes OMNIbus probes more flexible than TEC adapters, in that they are easier for an administrator to modify at a customer site. This is because modifying a script is easier than modifying a compiled program using external libraries.

The probe reads its rules file at startup, so if the rules are modified and syntax checked, stop and start the probe to pick up the modifications.

The main function of the rules file is to parse incoming data into tokens and assign the tokens to event attributes. Token names begin with the character $ and attributes begin with @ so for example, a straight-forward mapping is expressed like this:

```
@Node = $Node
```

The equal sign is the assignment operator, not a logical operator.

The probe parsing language has text manipulation capabilities such as string concatenation and a substring extraction function. It is fairly easy to create synthetic, or extrapolated, attributes. For example:

```
@Summary = $Node " has problem " $Summary
```

The language has a table lookup function which allows information to be stored in a separate file. This is useful for dynamic information so that the script does not need to be modified when, for example, a staff member or telephone number changes. Read about the **lookup** function to see how this works.

The language has:

- execution control features:
  - if/else
  - switch
- an input/output capability (table lookup)
- functions:
  - string match
  - regular expression string match (regmatch)
  - utility functions
  - time functions

and more. It is beyond the scope of this paper to detail probe rule writing but this area is significantly different than TEC.

The rules language has a syntax checker invoked by:

```
nco_p_syntax -server <ObjectServer name> -rulesfile <rules
file name>
```

See *Chapter 2, Probe Rules File Syntax* of the *Netcool OMNIbus v7 Probe and Gateway Guide* for more information about language syntax.

2.5.4 An Example Rules File

This is an example of a rules file for the socket probe on Linux. This probe, when started, monitors system communication sockets as defined in the probe invocation. The rules file defines how data received on the socket is parsed.

```
#######################################################
#
# Copyright (C) 1999-2002 Micromuse Ltd. All rights
# reserved.
# All Rights Reserved
#
# RESTRICTED RIGHTS:
#
# This file may have been supplied under a license.
# It may be used, disclosed, and/or copied only as
# permitted under such license agreement.  Any copy must
# contain the above copyright notice and this restricted
# rights notice. Use, copying, and/or disclosure of the
# file is strictly prohibited unless otherwise provided
# in the license agreement.
```

```
#
# Ident: $Id: socket.rules 1.5.1.3 2002/02/27 09:56:53
# daniels Development $
#
############################################################

if( match( @Manager, "ProbeWatch" ) )
{
        switch(@Summary)
        {
        case "Running ...":
                @Severity = 1
                @AlertGroup = "probestat"
                @Type = 2
        case "Going Down ...":
                @Severity = 5
                @AlertGroup = "probestat"
                @Type = 1
        default:
                @Severity = 1
        }
        @AlertKey = @Agent
        @Summary = @Agent + " probe on " + @Node + ": " +
@Summary
}
else
{
      @Identifier = $*
      @Summary = $*
}
```

This is the end of the example rules file.

2.5.5 Probe Configuration

Each probe configuration file, which has the **.prop** extension, contains the runtime environment parameters, such as:

- the ObjectServer to send events
- authentication information
- debugging level

This file is similar to the TEC adapter's .conf file.

Each probe may have properties unique to that probe. The command to list probe properties is:

```
$OMNIHOME/probe/nco_p_<probename> -help or -dumpprops
```

As with parameters in an adapter .conf file, property values in the file may be superceded by values on the command line at invocation. For more detailed information about property files, see section *3.1 Probe Properties and Command Line Options* of the *Netcool OMNIbus v7 Probe and Gateway Guide*.

TEC configuration files have filtering capability to avoid the overhead of creating, sending and processing events that are not of interest. OMNIbus configuration files do not explicitly filter but the rules files could be modified to achieve that objective using the **discard** function to drop unwanted events.

2.5.6 An Example Probe Property Configuration File

This is an example probe property (**.props)** file for the socket probe on Linux. As you can see, default values are provided but commented out and can be modified or enabled using a text editor.

```
#############################################################
#
# Copyright (C) 2002 Micromuse Ltd. All rights reserved.
#
# All Rights Reserved
#
# RESTRICTED RIGHTS:
#
# This file may have been supplied under a license.
# It may be used, disclosed, and/or copied only as permitted
# under such license agreement.  Any copy must contain the
# above copyright notice and this restricted rights notice.
# Use, copying, and/or disclosure of the file is strictly
# prohibited unless otherwise provided in the license
# agreement.
#
#
# Ident: $Id: mkprops 1.21 2003/08/20 15:29:09 csmith
# Development $
#
#############################################################
#############################################################
#
# Property Name          Default
#
# Generic Properties
#
# AuthPassword           : ''
# AuthUserName           : ''
# AutoSAF                : 0
# Buffering              : 0
# BufferSize             : 10
# Help                   : 0
# LookupTableMode        : 3
# Manager               : 'socket'
# MaxLogFileSize        : 1048576
# MaxRawFileSize        : -1
# MaxSAFFileSize        : 1048576
# MessageLevel          : 'warn'
# MessageLog            : '$OMNIHOME/log/socket.log'
# MsgDailyLog           : 0
# MsgTimeLog            : '0000'
# Name                  : 'socket'
# NameToUpper           : 0
# NetworkTimeout        : 0
# PollServer            : 0
# PropsFile             : '$OMNIHOME/probes/<arch>/
socket.props'
# RawCapture            : 0
# RawCaptureFile        : '$OMNIHOME/var/socket.cap'
# RawCaptureFileAppend  : 0
# RawCaptureFileBackup  : 0
# RetryConnectionCount  : 15
# RetryConnectionTimeOut: 30
# RulesFile             : '$OMNIHOME/probes/<arch>/
socket.rules'
# SAFFileName           : ''
# Server                : 'NCOMS'
# ServerBackup          : ''
# StoreAndForward       : 1
# Version               : 0
#
# Specific Properties
#
# Delimiter             : ''
# EventReadString       : ''
# EventTerminator       : '\n\n'
# Footer                : ''
# Header                : ''
# LineTerminator        : '\\n'
# LoginScript           : ''
# MaxEvents             : 10
# NoNameResolution      : 0
# NoTrim                : 0
# ParseAsLines          : 0
# PortNumber            : 4567
# Props.CheckNames      : TRUE
# ReadTimeOut           : 1
# ReportStatus          : 1
# SingleLines           : 0
# SocketBuffer          : 1024
# StreamCapture         : 0
# StreamCaptureFilename : '$OMNIHOME/var/socket.stream'
# StripChars            : ''
############################################################
############################################################
# Add your settings here
#
```

```
################################################################
```

This is the end of the example configuration file.

2.5.7 Probe Installation and Distribution

TEC uses the IBM Tivoli Framework profile distribution capability to distribute adapters and configuration files easily and in large scale. OMNIbus does not have a similar distribution mechanism. Probe files are manually installed on the distributed machines. In addition, the OMNIbus interfaces file needs to be modified to facilitate communication and database access.

2.5.8 Communication between Probes and the ObjectServer

The communication and reliability features of probes are similar to TEC adapters.

– **Protocol**: Probes communicate with the ObjectServer using TCP/IP. For performance reasons, when a probe starts, it establishes a persistent connection with the ObjectServer. By default, TEC adapters use temporary socket communications with the TEC server.

– **Caching:** Probes have the capability to store events locally in the event that they lose communication with the ObjectServer. This is similar to TEC adapter caching.

– **Fail Over:** Probes can be configured to send events to an alternative ObjectServer when the primary event server fails. This is similar to configuring a secondary server using the **ServerLocation** keyword in the **tec_gateway.conf** file on the TEC gateway.

2.5.9 Probe Summary and Terminology Comparison

Netcool OMNIbus probes are similar to TEC adapters in that they are the distributed components that collect, format and send events to the centralized event server.

TEC Component	OMNIbus Equivalent
TEC event server	ObjectServer.
TEC adapters	Probes.

TEC Component	OMNIbus Equivalent
TEC adapter configuration file (**.conf**)	Properties file (**.props**).
dup_detect facet in BAROC modified at the central event server	**Identifier** attribute in adapter rules code. The process of eliminating duplicate events in OMNIbus is referred to as **deduplication**.
Rules file (.rls) used to process events in the event server	ObjectServer SQL triggers and automations. The more advanced functionality of TEC rules would usually be done with other Netcool products, such as RAD and Impact.
TEC adapter format (.fmt) file	Rules file (.rules) maps tokens to attributes at the distributed side (probe). Note this is used with the probe and should not be confused with TEC rules files.
Filter definition in TEC adapter configuration file	**discard** function in probe rules file.

2.6 TEC Gateway

2.6.1 TEC Gateway Overview

Because the ObjectServer is optimized to receive and update events, other products can pull ObjectServer event data to perform higher-level event manipulation. The software programs that pull (and in some cases insert) data from the ObjectServer are called *gateways*. The number and types of available ObjectServer gateways preclude more than a conceptual overview, except for the TEC gateway. This gateway allows unidirectional or bidirectional event communication between a Netcool ObjectServer and a TEC event server.

The currently supported level of integration between TEC and OMNIbus is an OMNIbus gateway. The term *gateway* is different in OMNIbus than it is in the Tivoli Framework. An OMNIbus *gateway* connects the ObjectServer with a peer for the purpose of reading or writing ObjectServer data. The peer might be a backup ObjectServer or, as in this case, a TEC server.

The gateway for TEC allows events to flow from the OMNIbus ObjectServer to the TEC event server. Modifications to events in the OMNIbus ObjectServer are propagated to TEC. The event flow is one way: from OMNIbus ObjectServer to TEC. Modification made to events in TEC are *not* reflected in the OMNIbus ObjectServer.

Refer to the *Netcool/OMNIbus Gateway for TEC Supporting Products* publication dated June 23, 2006.

2.6.2 Installation and Configuration

The TEC gateway is installed as a patch. This section describes a generic installation, configuration and verification on a Linux ObjectServer.

You will need a license for the **nco_g_tec** feature in order to run the TEC gateway. This license is provided with the installation media.

2.6.3 Patch Installation

Two library patches are necessary for the TEC gateway patch to install successfully and they must be installed in a specific order.

The command to install a patch is:

```
$OMNIHOME/install/nco_patch -install <patch-file>
```

Install the patches in the following order:

1. omnibus-7.0-linux2x86-common-libngtk-1_3.tar.Z
2. omnibus-7.0-linux2x86-common-libngobjserv-1_6.tar.Z
3. omnibus-3.6-linux2x86-gateway-nco-g-tec-0.0.tar.Z

2.6.4 Gateway Configuration File

The first configuration step is to copy the gateway configuration file to the correct directory and modify it appropriately. For purposes of demonstrating the OMNIbus side, TEC events were written to a file instead of sending them to TEC. The value of **Gate.TEC.Simulator** was changed from FALSE to TRUE. Instead of sending events to TEC, they were written to a file named **./tec_simulator.txt**.

Other than that change, default configuration values were used. Note that the contents of this file are commented out. Leave the IPC section commented out, but enable the other sections. In early versions of this patch, the **Gate.StartupCmdFile** was incorrectly set. It should be **$OMNIBUS/gates/tec/tec.startup.cmd.**

For a brief description of the properties you can configure, use the command:

```
nco_g_tec -dumpprops
```

This gateway configuration file is not the same as a TEC adapter configuration file. There is a keyword **Gate.TEC.ConfigFile** that is set to the TEC adapter configuration file. By default it is **$OMNIHOME/gates/tec/tec_config**.

2.6.5 Schema Modification

OMNIbus and TEC use different severity values so the OMNIbus ObjectServer database schema is modified to accommodate TEC severity values. A new column **TECFdBk** is added to keep track of which events have been forwarded to TEC.

```
$OMNIHOME/bin/nco_sql < $OMNIHOME/gates/tec/tecgw_setup.sql
```

By default, the password is blank. You do not need to specify a server name if you use the default ObjectServer name of NCOMS.

2.6.6 TEC Configuration File

The TEC configuration should be modified as needed. At a minimum, you should specify the host name for the TEC event server using the **c1ServerLocation** property and the port using the **c1Port** property. For this example, events were written to a file for verification purposes. Because this is the default setting, this file was not modified.

The default location for this file is:

```
$OMNIHOME/gates/tec/tec_config.
```

2.6.6.1 Specify Gateway Interfaces

Specify an interface for **NCO_GATE** if it is not already specified, using the Server Editor on the ObjectServer.

2.6.6.2 Configure the License File

As mentioned in “Installation and Configuration” on page 24, you must have a license to run the TEC gateway. Remember to replace **put_hostname_here** with your server name.

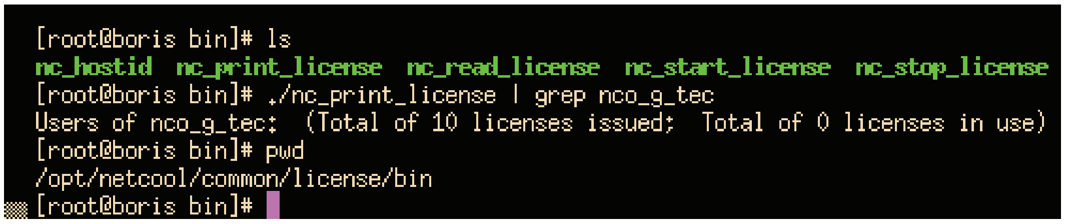
Put the modified license file in the appropriate directory on the license server. On Linux, the default directory is:

```
/opt/netcool/common/license/etc/
```

Either read the new license file or stop and start the license manager. Verify that the new license component has been read. On Linux, do this by issuing the command:

```
/opt/netcool/common/license/bin/nc_print_license | grep
nco_g_tec
```

Shown below is a typical command output:



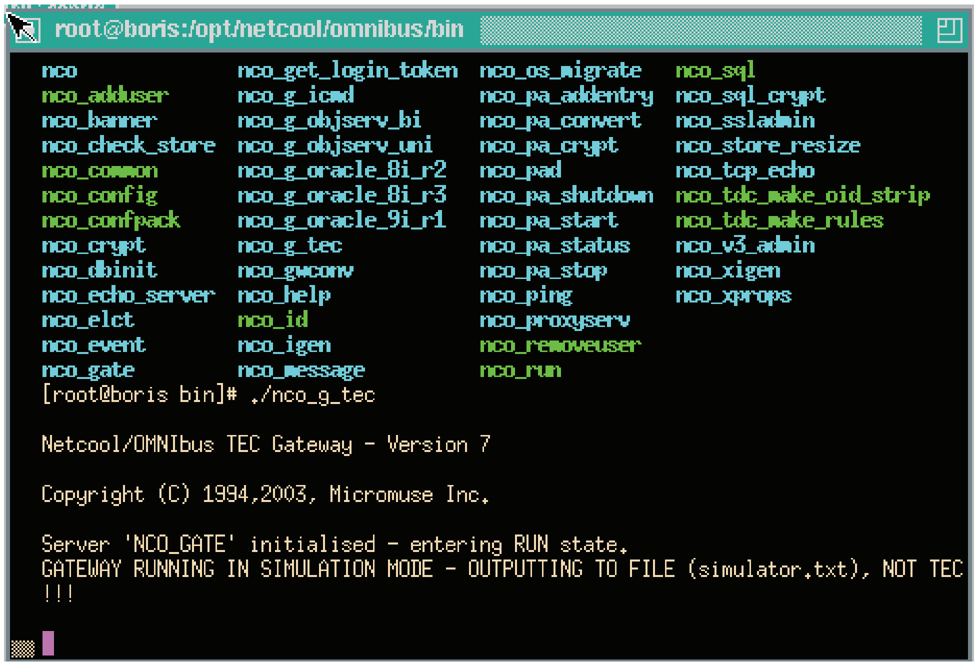
```
[root@boris bin]# ls
nc_hostid  nc_print_license  nc_read_license  nc_start_license  nc_stop_license
[root@boris bin]# ./nc_print_license | grep nco_g_tec
Users of nco_g_tec:  (Total of 10 licenses issued;  Total of 0 licenses in use)
[root@boris bin]# pwd
/opt/netcool/common/license/bin
[root@boris bin]#
```

2.6.7 Starting the Gateway

Start the gateway using the following command:

```
$OMNIHOME/bin/nco_g_tec
```

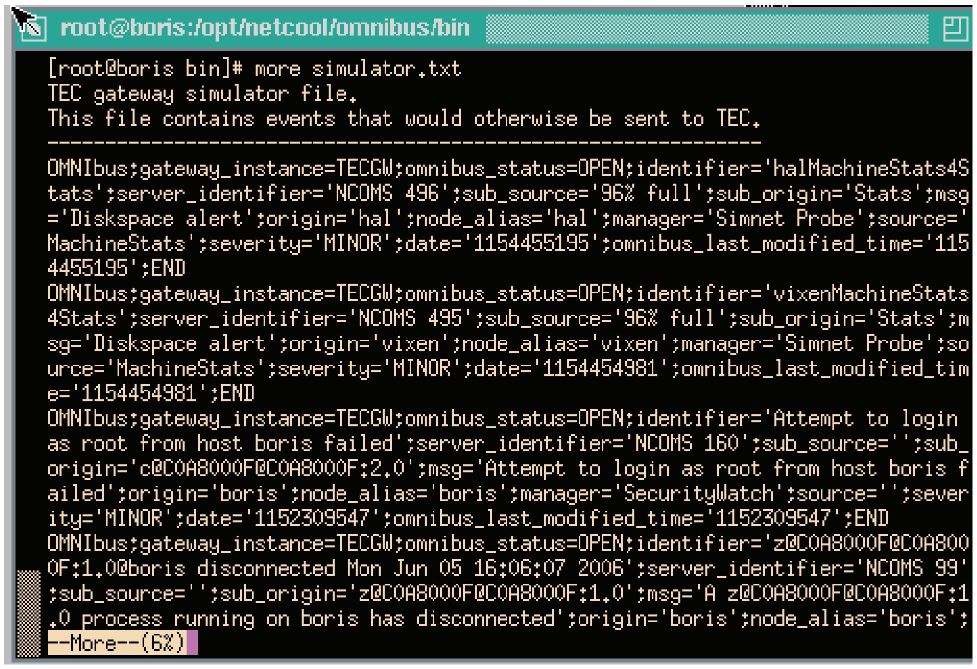
Listed below is an example command output:



```
nco               nco_get_login_token  nco_os_migrate    nco_sql
nco_adduser       nco_g_icmd           nco_pa_addentry   nco_sql_crypt
nco_banner        nco_g_objserv_bi     nco_pa_convert    nco_ssladmin
nco_check_store   nco_g_objserv_uni    nco_pa_crypt      nco_store_resize
nco_common        nco_g_oracle_8i_r2   nco_pad           nco_tcp_echo
nco_config        nco_g_oracle_8i_r3   nco_pa_shutdown   nco_tdc_make_oid_strip
nco_confpack      nco_g_oracle_9i_r1   nco_pa_start      nco_tdc_make_rules
nco_crypt         nco_g_tec            nco_pa_status     nco_v3_admin
nco_dbinit        nco_gwconv           nco_pa_stop       nco_xigen
nco_echo_server   nco_help             nco_ping          nco_xprops
nco_elct          nco_id               nco_proxyserv
nco_event         nco_igen             nco_removeuser
nco_gate          nco_message          nco_run
[root@boris bin]# ./nco_g_tec

Netcool/OMNIbus TEC Gateway - Version 7

Copyright (C) 1994,2003, Micromuse Inc.

Server 'NCO_GATE' initialised - entering RUN state.
GATEWAY RUNNING IN SIMULATION MODE - OUTPUTTING TO FILE (simulator.txt), NOT TEC
!!!
```

If you do not see this, check log files for error messages. In particular, the file **$OMNIHOME/log/NCO_GATE.log** can be helpful in identifying problems.

Verify that events are being written into the **$OMNIHOME/bin/simulator.txt** file.



```
[root@boris bin]# more simulator.txt
TEC gateway simulator file.
This file contains events that would otherwise be sent to TEC.
----------------------------------------------------------------
OMNIbus;gateway_instance=TECGW;omnibus_status=OPEN;identifier='halMachineStats4S
tats';server_identifier='NCOMS 496';sub_source='96% full';sub_origin='Stats';msg
='Diskspace alert';origin='hal';node_alias='hal';manager='Simnet Probe';source='
MachineStats';severity='MINOR';date='1154455195';omnibus_last_modified_time='115
4455195';END
OMNIbus;gateway_instance=TECGW;omnibus_status=OPEN;identifier='vixenMachineStats
4Stats';server_identifier='NCOMS 495';sub_source='96% full';sub_origin='Stats';m
sg='Diskspace alert';origin='vixen';node_alias='vixen';manager='Simnet Probe';so
urce='MachineStats';severity='MINOR';date='1154454981';omnibus_last_modified_tim
e='1154454981';END
OMNIbus;gateway_instance=TECGW;omnibus_status=OPEN;identifier='Attempt to login
as root from host boris failed';server_identifier='NCOMS 160';sub_source='';sub_
origin='c@C0A8000F@C0A8000F:2.0';msg='Attempt to login as root from host boris f
ailed';origin='boris';node_alias='boris';manager='SecurityWatch';source='';sever
ity='MINOR';date='1152309547';omnibus_last_modified_time='1152309547';END
OMNIbus;gateway_instance=TECGW;omnibus_status=OPEN;identifier='z@C0A8000F@C0A800
0F:1.0@boris disconnected Mon Jun 05 16:06:07 2006';server_identifier='NCOMS 99'
;sub_source='';sub_origin='z@C0A8000F@C0A8000F:1.0';msg='A z@C0A8000F@C0A8000F:1
.0 process running on boris has disconnected';origin='boris';node_alias='boris';
--More--(6%)
```

2.7 Webtop

2.7.1 Webtop Overview

Webtop is a program running on the Tomcat 4.1 version of Apache Web server providing thin-client (Web browser) console access to an ObjectServer. The clients connect using either the HTTP or HTTPS protocols. Webtop provides the same functionality as that provided by the TEC UI server and the TEC console program running under IBM WebSphere Application Server. All of the administrative and operational functionality of the desktop console is provided with Webtop. Additionally, Webtop can create views of event data with linked objects called *maps*.

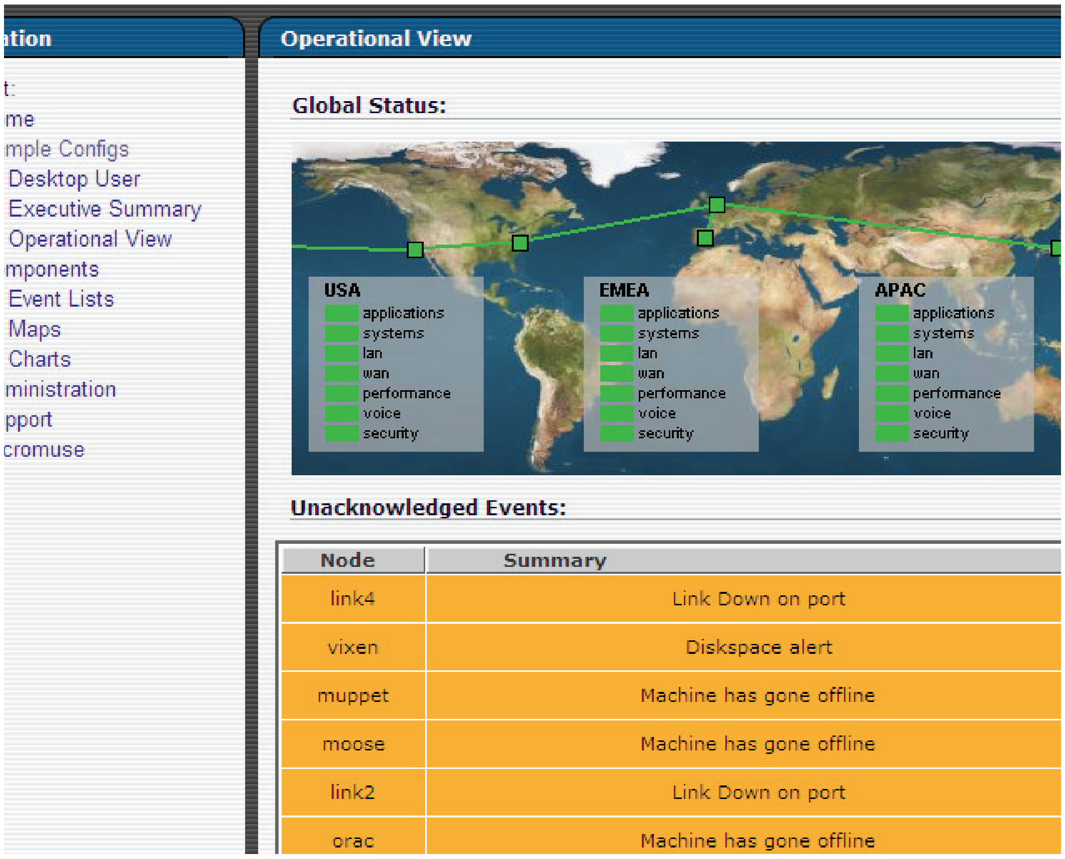
Webtop communicates with an ObjectServer based on a *data source definition*. A data source definition tells the Webtop server the host name, server name, and listening port of the ObjectServer. Each data source connection requires one **cro_webtop** license. The license server information for Webtop is maintained in the **$WEBTOP_HOME/config/ server.init** file, rather than using the **$NETCOOL_LICENSE_FILE** environment variable. The **server.init** file is also used to define other server operational parameters, such as the port number on which to listen for client connections (default port 8080). Web browser clients come in two different configurations, requiring two different client license types, depending on the client requirements:

1. Desktop client: This type of client can modify events. This requires a **cro_webtop_dt** license for every user.

2. Read-only client: This type of client can only view events. This requires a **cro_webtop_ro** license for every read-only client.

2.7.2 Maps and Charts

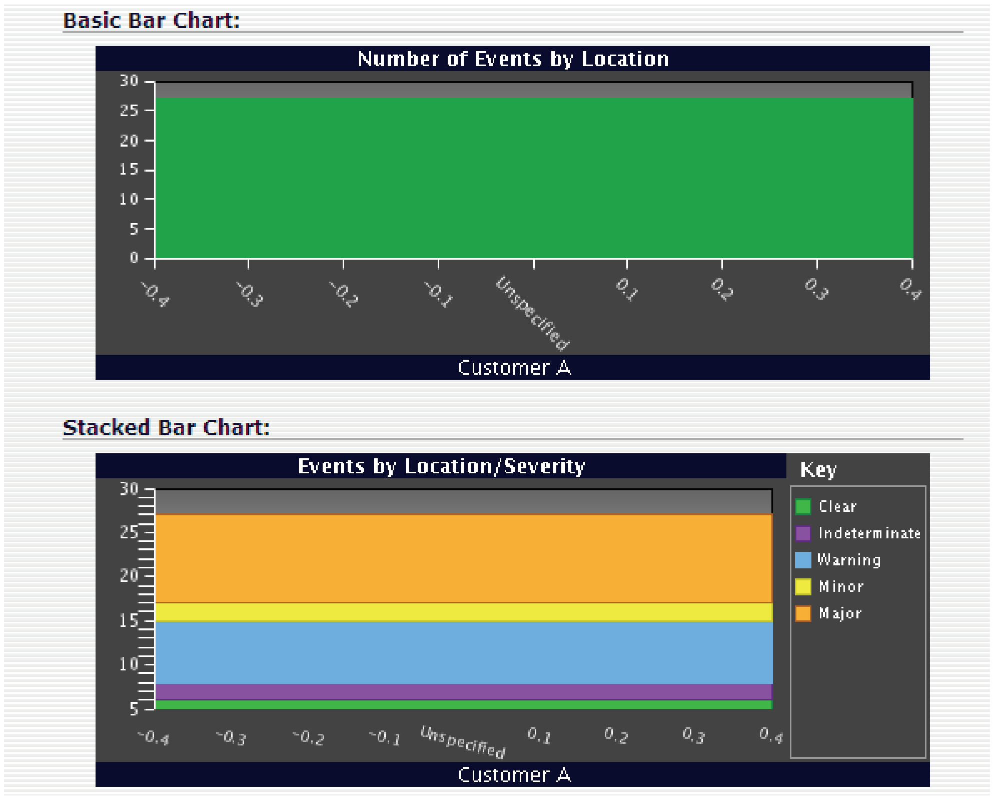
Webtop differs from the native OMNIbus client desktop and TEC in that it provides a feature known as *maps*. Maps allow you to build graphical views of an installation envioronment and assign links to objects on the map. This allows you to create topographical views of an OMNIbus installation, as shown in the screen capture below:



Node	Summary
link4	Link Down on port
vixen	Diskspace alert
muppet	Machine has gone offline
moose	Machine has gone offline
link2	Link Down on port
orac	Machine has gone offline

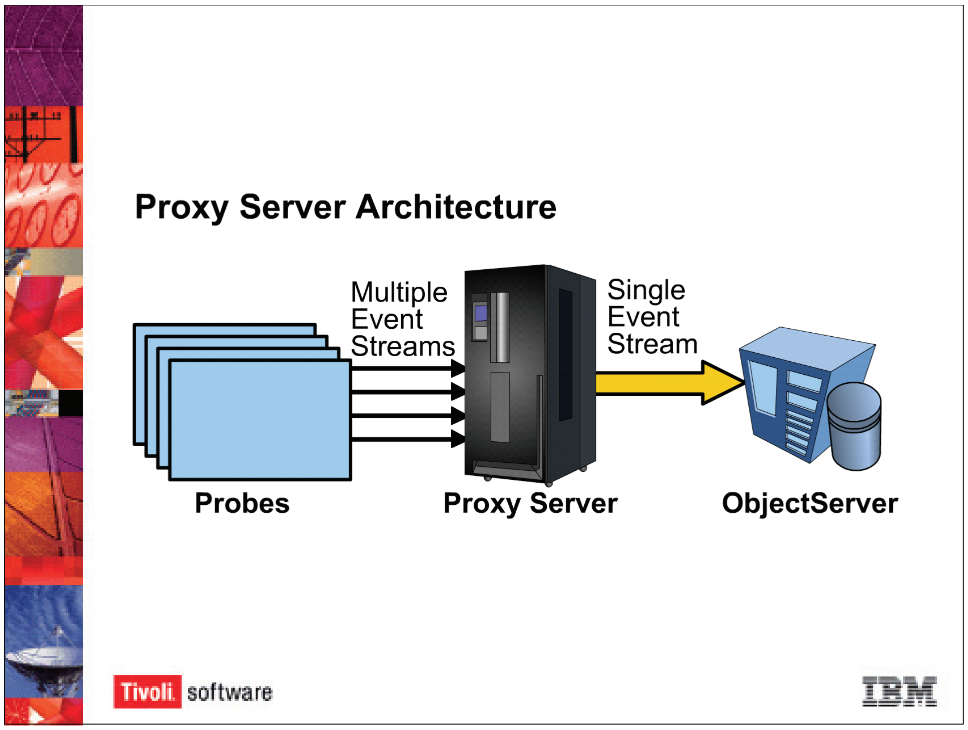
When one of the screen objects is clicked, the view is updated based on the map definition.

Also included in the base installation is the ability to present event data in a variety of chart formats (bar, line, pie, 3-D). An example bar chart is shown in the following screen capture:


Basic Bar Chart:
Number of Events by Location
30
25
20
15
10
5
0
-0.4
-0.3
-0.2
-0.1
Unspecified
0.1
0.2
0.3
0.4
Customer A
Stacked Bar Chart:
Events by Location/Severity
Key
Clear
Indeterminate
Warning
Minor
Major
30
25
20
15
10
5
-0.4
-0.3
-0.2
-0.1
Unspecified
0.1
0.2
0.3
0.4
Customer A

2.7.3 Proxy Server

To reduce the processing overhead on an ObjectServer when a large number of probes are sending event data directly to the ObjectServer, a proxy server is used. The proxy server allows multiple probe data streams to be combined, requiring only a single communication connection to the ObjectServer. A logical diagram of this configuration is shown in the slide below:



2.8 Dedicated Consoles and Command-line Utilities

2.8.1 Conductor

OMNIbus ships with a graphical console known as *Conductor* (**nco**), which provides graphical access to view events and administer an ObjectServer environment. Conductor requires one license per process instance and is installed wherever the user or administrator will be monitoring events. This console performs the same functions as that provided by the TEC Java console.

2.8.2 Administrator

The administrator console (**nco_config** under UNIX or **Linux, Start > Programs > Netcool Suite > Administrator** under Windows) provides a graphical interface to configure ObjectServer behavior. This console is used to manage ObjectServer databases, columns, and tables, configure server operational parameters and manage users, groups, and roles.

2.8.3 Webtop Administration Application Programming Interface

When Webtop is installed, you can choose to also install the Webtop Administration Application Programming Interface (WAAPI). Using appropriate tags in an XML file, the WAAPI program can process the same commands and functions available with the graphical consoles. This provides command-line and script access to the Webtop server.

3 Hardware and Software

Listed in the table below is a comparison of the hardware and software requirements for TEC and OMNIbus running on identical operating systems and servers. Disk space, memory, and processor requirements are estimates based on TEC and OMNIbus installation guides.

Both TEC and OMNIbus are supported on many platforms. This comparison is not intended to be a complete list of supported platforms. To verify the latest supported platforms, review the contents of section 1.2 in the *Netcool/OMNIbus Installation and Deployment Guide v7*.1 and the Tivoli Platform and Support Matrix (URL: *http://www-1.ibm.com/support/docview.wss?rs=203&uid=swg21067036*)

The baseline computer system for both comparisons is listed in the following table:

Baseline hardware for TEC 3.9 + WAS 6.0 and OMNIbus 7.1 + Webtop
Processors: 4 RISC processors, SPECint_rate 2000 of 30 or better
Memory: 4 GB
Disk Space: 8 GB or greater

This comparison assumes that all components required to provide basic event management and Web-based console access are run on a single system, using the baseline specifications above. The operating system used for these estimates assumes AIX 5.2. The software comparison is listed in the following table.

TEC 3.9 Components and Disk Space	OMNIbus 7.1 Components and Disk Space
TEC 3.9 + FP 05: 512 MB	**OMNIbus 7.1**: 380 MB
Tivoli Framework 4.1.1: 512 MB	**N/A**
DB2 8.1 ESE: 800 MB	**N/A**
WebSphere 6.0: 2 GB	**Webtop 1.3.1**: 150 MB
TEC UI Server: 512 MB	**N/A**
N/A	**Flex License Server 9.2**: 5 MB
Total: 4.3 GB	**Total**: 535 MB

Conclusion

Summary

Longtime TEC administrators will find many similarities of function, if not application, with OMNIbus. Many of the differences in application, such as parsing in probes versus TEC adapters, will provide very useful capabilities for an established TEC environment. The TEC-OMNIbus gateway provides a simple, seamless way for integrating the two environments, allowing administrators, consultants, and developers to choose the best features from both products when solving enterprise event management requirements.

Shown in the table below are the summarized comparisons discussed earlier in this white paper.

TEC Component	OMNIbus Equivalent
Framework (Authentication)	User and group information created and maintained in the ObjectServer database.
Framework (Authorization)	User roles defined and maintained in the ObjectServer database.
Framework (RIM Object)	Probes and monitors communicate with an IDUC communication process running on the ObjectServer, which handles database updates. Uses TCP/IP for underlying communication protocol.
Framework (Event Gateways)	None. If necessary, hierarchical ObjectServers could be deployed.
TEC Event Server	ObjectServer.
UI Server and WebSphere TEC Console Server	Webtop.
Java event console	Conductor, Administrator (**nco_config**) consoles.
External event database	SQL database running in server memory, part of ObjectServer installation.
Command-line framework commands	**nco_sql** to access ObjectServer tables, columns, and rows; Webtop Administrative Application Programming Interface (WAAPI).

TEC Component	OMNIbus Equivalent
TEC adapters	Probes.
TEC adapter configuration file (.**config**)	Properties file (.**props**).
dup_detect facet in BAROC modified at the central event server	**Identifier** attribute in adapter rules code. The process of eliminating duplicate events in OMNIbus is referred to as **deduplication**.
Rules file (.**rls**) used to process events in the event server	ObjectServer SQL triggers and automations. The more advanced functionality of TEC rules would usually be done with other Netcool products, such as RAD and Impact.
TEC adapter format (.**fmt**) file	Rules file (.**rules**) maps tokens to attributes at the distributed side (probe). Note this is used with the probe and should not be confused with the TEC rules files.
Filter definition in TEC adapter configuration file	**discard** function in probe rules file.

Resources

Data sheet: Netcool/OMNIbus

White paper: Netcool + Tivoli: delivering service management innovation.

Case study: Customer reference video - Belgacom