

# Best Practices for IBM Tivoli Enterprise Console to Netcool/OMNibus Upgrade



- Integration and upgrade strategies for TEC-based environments

- Provides detailed guidelines for planning an upgrade

- Includes upgrade scenarios and best practice recommendations

Dietger Bahn  
Richard Fowkes  
Raffaella Nicolosi  
Wolfgang Schumacher  
Ilda Yaguinuma





International Technical Support Organization

**Best Practices for IBM Tivoli Enterprise Console to  
Netcool/OMNibus Upgrade**

August 2008

Archived

**Note:** Before using this information and the product it supports, read the information in “Notices” on page xi.

## **First Edition (August 2008)**

This edition applies to IBM Tivoli Netcool/OMNIBus Version 7.2.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	xi
Trademarks .....	xii
<b>Preface</b> .....	xiii
The team that wrote this book .....	xiv
Acknowledgements .....	xv
Become a published author .....	xvi
Comments welcome .....	xvi
<b>Part 1. Overview</b> .....	1
<b>Chapter 1. Introduction</b> .....	3
1.1 IBM Service .....	4
1.2 Netcool/OMNIBus .....	6
1.3 Tivoli Enterprise Console customer choices .....	7
1.4 Product review .....	8
1.4.1 Event management .....	8
1.4.2 Event sources .....	9
1.4.3 Event visualization .....	11
1.4.4 Product administration and configuration .....	15
1.4.5 Integration with other products .....	20
1.4.6 Netcool/OMNIBus .....	22
1.5 Benefits of upgrading to Netcool/OMNIBus .....	23
1.5.1 Scalability and performance .....	23
1.5.2 Consolidation .....	25
1.5.3 Ease of use .....	26
1.5.4 Reliability .....	26
1.5.5 Enhanced event visualization and management .....	28
1.5.6 Enhanced event enrichment .....	32
1.5.7 Security .....	33
1.5.8 IBM product strategy .....	36
<b>Chapter 2. Architecture</b> .....	37
2.1 Tivoli Enterprise Console architecture .....	38
2.1.1 Typical installation .....	38
2.1.2 Describing TEC components .....	41
2.1.3 Complex scenarios .....	50
2.1.4 TEC integration .....	54

2.2 IBM Tivoli Netcool/OMNIBus architecture . . . . .	60
2.2.1 Architecture introduction . . . . .	60
2.2.2 Architecture overview . . . . .	62
2.2.3 Component description . . . . .	62
2.2.4 Probes . . . . .	78
2.2.5 Monitors . . . . .	87
2.2.6 Gateways . . . . .	87
2.2.7 Netcool/Webtop . . . . .	91
2.2.8 Netcool GUI Foundation . . . . .	92
2.2.9 Typical Netcool/OMNIBus deployment . . . . .	94
2.2.10 Two-tiered architecture . . . . .	95
2.2.11 Three-tiered architecture . . . . .	96
2.2.12 Firewall considerations . . . . .	98
2.2.13 Configuring hardware for performance . . . . .	99
2.2.14 Netcool/OMNIBus rules: best practices for performance . . . . .	100
<b>Part 2. Strategies . . . . .</b>	<b>101</b>
<b>Chapter 3. TEC environmental assessment and planning guidelines . . . . .</b>	<b>103</b>
3.1 End-to-end event flow . . . . .	104
3.2 Event source hosts . . . . .	106
3.2.1 Tivoli framework commands . . . . .	106
3.2.2 Non-framework commands . . . . .	107
3.2.3 Other techniques . . . . .	107
3.2.4 Safety net . . . . .	108
3.3 TEC source types . . . . .	108
3.3.1 IBM Tivoli Monitoring event sources . . . . .	109
3.3.2 Omegamon agent sources . . . . .	110
3.3.3 NetView event sources . . . . .	110
3.3.4 NetView forwarding to OMNIBus options . . . . .	110
3.3.5 Windows event log messages . . . . .	113
3.3.6 UNIX and Linux syslog messages . . . . .	114
3.3.7 Logfile messages . . . . .	114
3.3.8 SNMP traps . . . . .	114
3.3.9 AS400 messages . . . . .	114
3.3.10 Command line sources (w)postmsg and (w)postzmsg . . . . .	115
3.3.11 Custom EIF applications . . . . .	115
3.3.12 Tivoli Business Systems Manager (TBSM 3.1) . . . . .	116
3.4 Other planning considerations . . . . .	116
3.4.1 Deployment considerations . . . . .	116
3.4.2 Scoping volumes and throughput of events . . . . .	120
3.4.3 Coping with event storms . . . . .	121
3.4.4 TCP/IP port usage . . . . .	121

3.4.5	EIF probe considerations . . . . .	122
3.4.6	Adapter configuration files and gateway configuration files . . . . .	122
3.5	Distributed event processing . . . . .	123
3.5.1	Formatting . . . . .	123
3.5.2	Filtering . . . . .	125
3.5.3	State correlation engine (SCE) processing . . . . .	125
3.6	BAROC file definitions . . . . .	126
3.7	Central event processing (TEC rules) . . . . .	127
3.7.1	Frequently used rules . . . . .	128
3.7.2	Typical rule types . . . . .	128
3.7.3	Remote procedure execution . . . . .	128
3.8	Some event-processing examples . . . . .	129
3.8.1	General suggestions . . . . .	130
3.8.2	Handling of duplicate events . . . . .	130
3.8.3	Filtering out events with specific content . . . . .	131
3.8.4	Actions for too many events in a defined time frame . . . . .	131
3.8.5	Filling an attribute dependent on another field's content . . . . .	132
3.8.6	Handling of correlations (cause, effect, and clearing events) . . . . .	133
3.8.7	Local and remote script execution . . . . .	133
3.8.8	Escalation of the severity of events . . . . .	133
3.8.9	Forwarding of events . . . . .	134
3.9	TEC outputs . . . . .	134
3.9.1	TEC tasks . . . . .	135
3.9.2	Forwarding to other TEC servers (manager of managers) . . . . .	135
3.9.3	Incident management systems . . . . .	135
3.9.4	Service-level reporting and auditing databases . . . . .	136
3.10	Desktop upgrade (TEC console) . . . . .	136
3.11	Event view customization . . . . .	138
3.11.1	TEC information button . . . . .	138
3.11.2	TEC custom buttons . . . . .	138
3.11.3	Large event messages (greater than 255 characters) . . . . .	139
3.11.4	Operator actions . . . . .	139
3.11.5	Color patterns . . . . .	141
3.12	Resource considerations skills . . . . .	141
3.12.1	Event-processing configuration . . . . .	141
3.12.2	Installation, administration, and operations . . . . .	142

3.13 Checklist .....	143
3.14 Suggested testing plan .....	144
<b>Chapter 4. Upgrade strategies .....</b>	<b>147</b>
4.1 Event flow integration based on TEC .....	149
4.2 Event flow integration based on OMNIBus .....	151
4.3 TEC replacement strategy .....	152
4.3.1 Event flow .....	152
4.3.2 Advantages .....	153
4.3.3 Disadvantages .....	153
4.3.4 Which scenarios this applies to .....	153
4.4 TEC to OMNIBus upgrade .....	154
4.4.1 Event flow .....	154
4.4.2 Advantages .....	155
4.4.3 Disadvantages .....	155
4.4.4 Who this applies to .....	155
4.5 The recommended strategy .....	156
<b>Part 3. Implementation .....</b>	<b>163</b>
<b>Chapter 5. Upgrading to an IBM Tivoli Netcool environment .....</b>	<b>165</b>
5.1 Tivoli Enterprise Console prior to upgrade .....	166
5.1.1 Installed TEC components .....	166
5.1.2 TEC installation and configuration .....	168
5.2 Netcool/OMNIBus lab environment .....	168
5.2.1 AIX lab environment for Netcool/OMNIBus .....	169
5.2.2 Red Hat environment for Netcool/OMNIBus .....	170
5.3 Netcool/OMNIBus installation .....	171
5.4 IBM Tivoli Netcool/OMNIBus configuration .....	171
5.4.1 ObjectServer database initialization .....	171
5.4.2 ObjectServer interfaces omni.dat .....	172
5.4.3 Interfaces file generation .....	173
5.4.4 ObjectServer properties configuration .....	174
5.4.5 Process Automation configuration .....	175
5.4.6 ObjectServer startup .....	179
5.4.7 ObjectServer shutdown .....	180
5.5 IBM Tivoli Netcool probe installation overview .....	181
5.5.1 What you need to know about nco_patch .....	181
5.5.2 Toggle feature for process control .....	181
5.5.3 Installation of probe for Windows NT event logs .....	183
5.5.4 Check the probe installation .....	184
5.5.5 Netcool probe configuration .....	185



5.6 Installing Netcool Security Manager and Netcool Webtop .....	185
<b>Chapter 6. Event processing .....</b>	<b>187</b>
6.1 Differences between TEC and OMNIBus .....	188
6.1.1 Resolving of events .....	188
6.1.2 Processing of events .....	188
6.2 Event processing migration .....	189
6.2.1 General suggestions .....	190
6.2.2 Lab environment .....	191
6.2.3 Handling of duplicate events .....	194
6.2.4 Filtering out events with specific content .....	197
6.2.5 Actions for too many events in a defined time frame .....	206
6.2.6 Filling an attribute dependent on another field's content .....	209
6.2.7 Handling of cause, effect, and clearing events .....	215
6.2.8 Propagating status change from cause to effect events .....	225
6.2.9 Local and remote script execution .....	228
6.2.10 Escalation of event severity .....	235
6.2.11 Forwarding of events .....	238
6.2.12 Use of external information for logic control .....	238
6.3 Probe topics .....	242
6.3.1 Measuring load and numbers of events in a time frame .....	242
6.3.2 Self monitoring .....	243
6.3.3 Parsing failed .....	243
6.3.4 EIF rules file and extended attributes .....	244
6.4 Support of TEC class hierarchy .....	247
6.5 TEC information/URL information for events .....	256
6.5.1 Rule best practices for performance .....	273
6.5.2 Debugging using Netcool IDE .....	273
6.5.3 Netcool Knowledge Library .....	274
<b>Chapter 7. Configuring the event sources .....</b>	<b>277</b>
7.1 Adding a rule to forward raw events to OMNIBus .....	278
7.2 Integration between Netcool/OMNIBus and Tivoli NetView .....	280
7.2.1 Netcool/OMNIBus 7.2 and Tivoli NetView integration overview ..	281
7.2.2 Installing Netcool/OMNIBus probe for Tivoli EIF .....	282
7.2.3 Configuring OMNIBus ObjectServer .....	282
7.2.4 Configuring the Tivoli EIF probe .....	283
7.2.5 Configuring the NetView TEC adapter to send to the EIF probe ..	283
7.2.6 Automatic event management customization .....	284
7.3 Integration between Netcool/OMNIBus and IBM Tivoli Monitoring .....	290
7.3.1 Netcool/OMNIBus 7.2 and IBM Tivoli Monitoring 6.2 integration ..	291
7.3.2 Installing Netcool/OMNIBus probe for Tivoli EIF .....	292
7.3.3 Installing event synchronization .....	292

7.3.4	Configuring the OMNIBus server . . . . .	306
7.3.5	Configuring the monitoring server . . . . .	311
7.4	Deduplication configuration . . . . .	320
7.5	Migrating the TEC Windows event log adapter . . . . .	322
7.5.1	Installing and configuring the Windows NT Event Log probe . . . . .	323
7.5.2	Installing and configuring the process agent on Windows . . . . .	340
7.6	Syslog probe event configuration . . . . .	357
7.7	Completed upgrade . . . . .	360
7.8	Troubleshooting the event flow . . . . .	361
<b>Part 4.</b>	<b>Appendices . . . . .</b>	<b>365</b>
<b>Appendix A.</b>	<b>Lab configuration . . . . .</b>	<b>367</b>
	TEC installation steps . . . . .	368
	TEC event source generation commands and scripts . . . . .	372
	Netcool/OMNIBus directory structure reference . . . . .	374
	IBM Tivoli Netcool default port usage . . . . .	375
	IBM Tivoli Netcool/OMNIBus . . . . .	375
	IBM Tivoli Netcool Security Manager . . . . .	375
	IBM Tivoli Netcool/Webtop . . . . .	376
	IBM Tivoli Netcool probes . . . . .	376
	User profile . . . . .	377
	Netcool Process Automation startup script . . . . .	377
	ObjectServer WEIMAR_PA Process Automation configuration . . . . .	379
	ObjectServer WEIMAR probe nco_p_tivoli_eif rules (TEC, NetView) . . . . .	382
	ObjectServer WEIMAR probe nco_p_tivoli_eif rules (TEC, NetView, ITM) . . . . .	388
<b>Appendix B.</b>	<b>Additional material . . . . .</b>	<b>399</b>
	Locating the Web material . . . . .	399
	Using the Web material . . . . .	400
	How to use the Web material . . . . .	400

<b>Related publications</b> .....	401
IBM Redbooks .....	401
Online resources .....	401
Tivoli Netcool/OMNIBus technical information .....	403
Release notes .....	403
Installation and deployment guide .....	403
Administration guide .....	403
User guide .....	404
Probe and gateway guide .....	404
IBM Tivoli Netcool/Security Manager .....	405
IBM Tivoli Netcool/Webtop .....	405
IBM Tivoli Netcool GUI Foundation .....	405
IBM Tivoli Netcool tools and utilities .....	406
Using IBM Tivoli Include Library .....	406
Global Advanced Technology team tools and utilities .....	406
How to get Redbooks .....	407
Help from IBM .....	408
RSS feed list .....	408
 <b>Index</b> .....	 415

Archived

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information about the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	Micromuse®	Redbooks (logo)  ®
Alerts®	Netcool/OMNibus™	System p™
AS/400®	Netcool®	Tivoli Enterprise Console®
Candle®	NetView®	Tivoli®
DB2®	OS/2®	TME®
IBM®	OS/390®	WebSphere®
Informix®	Proviso®	z/OS®
Maximo®	Redbooks®	

The following terms are trademarks of other companies:

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library, IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Java, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows NT, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Sybase is a trademark of Sybase, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

The acquisition of Micromuse® Inc. brings new opportunities for all involved in IBM® Systems Management discipline, and the development of a new and exciting strategy.

This IBM Redbooks® publication should be used when planning and implementing an integration and upgrade strategy from TEC to OMNIBus. In this book we provide recommended best practices and describe strategies for upgrading existing installations in a way that should best suit the needs of existing TEC-based environments.

The audience for this book is anyone involved in the Systems Management discipline, but it applies primarily to both those with a Tivoli® or Netcool® background, and is aimed at customers with an existing Tivoli Enterprise Console® investment who are looking to evaluate the comparative characteristics of TEC and Netcool/OMNIBus™, so that they can perform a system upgrade.

Depending on the complexity of the existing environment and the depth of the requirements, this upgrade could be a significant project, but we aim with this book to make it as straightforward and as successful as possible.

We have structured the book to first introduce a quick overview of the products, highlighting the key benefits of Netcool/OMNIBus so that both audiences can become familiar with the different concepts. Then the architectures of both products are discussed in more detail, concluding with some typical scenarios.

Part 2, “Strategies” on page 101, reviews planning and strategy. It begins with detailed guidelines on assessing the existing customer environment in order to identify how TEC is currently deployed. We discuss considerations to make and how to plan the activities required to upgrade. Then different upgrade scenarios are presented with a best practice recommendation that the reader can adapt to his environment.

At this point we would like to stress that this book builds on the utilities provided by the *Tivoli and Netcool Integration Event Flow* package (downloadable from the IBM OPAL Web site), and the recommendations from the *Tivoli & Netcool Event Flow Integration* white paper. The main distinction is that they cover various event *integration* scenarios, whereas we, in addition, map out a complete *upgrade path* to OMNIBus in our recommended strategy.

The implementation of the suggested strategy is then covered in Chapter 7, “Configuring the event sources” on page 277. We describe in detail the steps required to achieve the upgrade with the core components, and discuss other tasks to also keep in mind. A wide range of different rule processing examples are provided, giving comparative and practical guidance, providing a valuable asset for the rule programmer.

Finally, additional technical details on configurations and scripts used and other valuable references can be found in Appendix A, “Lab configuration” on page 367.

## The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

**Dietger Bahn** is a Systems Management Specialist at the IBM Software Group, Tivoli Services in Germany, and specializes in IBM Tivoli Netcool product line. His areas of expertise also include a Proof-of-Technology about IBM Tivoli Netcool and integrations at the IBM Technical Marketing Competence Center Europe (TMCC), also used in the IBM Tivoli Integrated Demo Environment (TIDE), with 10 years of experience in Systems Management and technical administration of UNIX® family operating systems including high availability environments. He worked for three years as a Support Specialist at the IBM EMEA Techline pre-sales and post-sales support for Tivoli and Systems Management solutions. Before joining IBM, he worked for seven years as an IT Specialist for UNIX systems administration. Dietger is ITIL® certified and continuative qualified in service-oriented troubleshooting.

**Richard Fowkes** is a Senior accredited Product Services Professional, and Team Leader for the Maintenance and Technical Services Tivoli support team based in the UK, which supports customers in UKISA and the Nordic countries. He has provided technical support for a wide range of Tivoli products at IBM for the last nine years and has over 20 years of experience in the Systems Management field. He holds a degree in business studies and is currently a staff member of the UKISA Technical Council. His areas of expertise include Tivoli Framework, Enterprise Console, NetView® and Workload Scheduler, and generic troubleshooting. He has co-written and taught the Advanced Problem Investigation Principles course.

**Raffaella Nicolosi** works for IBM Italia in Global Response Team EMEA, helping customers resolve critical situations. She earned a degree in Computer Science in 2005 and began working for IBM Italia in 2006. Her areas of expertise in the IBM portfolio include the Netcool suite and its related products such as



OMNibus, Webtop, Impact, Tivoli Business Service Manager 4.1, Tivoli Network Manager IP Edition, and Proviso®. Her areas of expertise include TCP/IP Networks, operating systems, human and computer interaction, and Java™ programming.

**Wolfgang Schumacher** is a Tivoli Certified Consultant and Tivoli Certified Instructor at ACT AG in Niederkassel, Germany. He is also Germany's representative to the Global Tivoli User Group. He has 26 years of experience in the Systems Management field. His experience includes the project management, architecture, implementation, and performance measurement of Systems Management and networking solutions for distributed and mainframe environments using IBM, Tivoli, and OEM products. His areas of expertise include IBM Tivoli Enterprise Console rule writing and developing automation for IBM Netcool OMNibus solutions.

**Ilda Yaguinuma** is an IT Specialist at Server Systems Operations in IBM Brazil. She has been working at IBM since 1993. She supports SNA architecture and NetView family products on the mainframe platform. She has moved to the distributed platform since the Tivoli acquisition. She has worked on several projects, designing and implementing Tivoli core products. She has written about TEC architecture and OMNibus/ITM integration.

***Production of this book was managed by***

**Chris Almond**, an ITSO Project Leader and IT Architect based at the ITSO Center in Austin, Texas. In his current role, he specializes in managing content development projects focused on Linux®, System p™, and AIX®, various IBM Software Group products, and innovation programs. He has a total of 17 years of IT industry experience, including the last 9 with IBM.

## Acknowledgements

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

Special thanks to Elise Kushner of ACT AG, Germany, whose many years of TEC experience are reflected in Chapter 6, “Event processing” on page 187.

Special thanks to Christian Michaelski of ACT AG, Germany, who provided the perl script `tec_help.pl` and the corresponding files.

Special thanks to Fabrizio Salustri of IBM Italia and Ana Paula Godoy of IBM Brazil. They gave us a much help in the development of the integration between OMNibus and ITM, sharing with us their ITM 6.2 lab environment, their best skills, and their friendship.

Chris Almond - our Project Leader, Don Wildman - Product Manager for Netcool/OMNIBus, Tracey McWilliams - Netcool Senior Technical Support Engineer, Kristian Stewart - Engineering Management Netcool/OMNIBus, Stephen Brocklesby - Tivoli Systems Management consultant, Dave Scarr - TEC L2 Support, Jimmy Gholston and the TEC L2 support team in the US, Stephen Cook - Engineering Management Netcool/OMNIBus, Joerg Weikopf - Tivoli Technical Sales, Kai Preuss - Tivoli Netcool Technical Sales, Carsten Otto - Tivoli Netcool Technical sales, Ingo Averdunk - Principle Consultant, and, last but not least, Arzu Gucer, Lupe Brown, and Bill Trimble for smoothing over all of our administrative tasks and making sure that our stay in Austin during this project was most enjoyable.

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

Archived





# Part 1

## Overview

We begin this book with an introductory chapter that provides an overview of the IBM Service Management product portfolio now that it includes the recently acquired IBM Tivoli Netcool monitoring solutions, and in particular, Netcool/OMNIbus.

In this part we also include a chapter that provides a detailed technical comparison between the TEC and OMNIbus product architectures. This part contains the following chapters:

- ▶ Chapter 1, “Introduction” on page 3
- ▶ Chapter 2, “Architecture” on page 37



# Introduction

The IBM acquisition of Micromuse Inc. on February 14, 2006, marked a major milestone for IBM Tivoli software because it significantly strengthened the product offerings in the end-to-end IBM Service Management portfolio. Existing Tivoli Enterprise Console customers can now gain extra benefits by discovering and exploiting the enhanced features of Netcool/OMNIBus and of the other products of the Netcool suite.

This book is intended primarily as a detailed technical guide for customers who want to know more about these benefits, and as a how-to guide for a step-by-step setup of an upgraded environment that fully exploits the many advantages of the IBM Tivoli Netcool monitoring solution.

## 1.1 IBM Service

Today, network and IT operations are under tremendous pressures to deliver new, next-generation services more quickly than ever before. At the same time, lines of business (LOBs) and customers demand more services and service-level agreements (SLAs) to ensure that they receive the service quality that they expect. These challenges are further compounded through increased regulatory and audit requirements that often require budget and labor shifts from more strategic growth initiatives.

By combining the Netcool and Tivoli portfolios, IBM enables customers to take a more comprehensive approach to aligning operations and processes with their organization's business needs—an approach that leverages best practices such as those of the IT Infrastructure Library® (ITIL) and the NGOSS Business Process Framework of the TMForum enhanced Telecom Operations Map (eTOM). IBM calls this approach *IBM Service Management* (Figure 1-1).

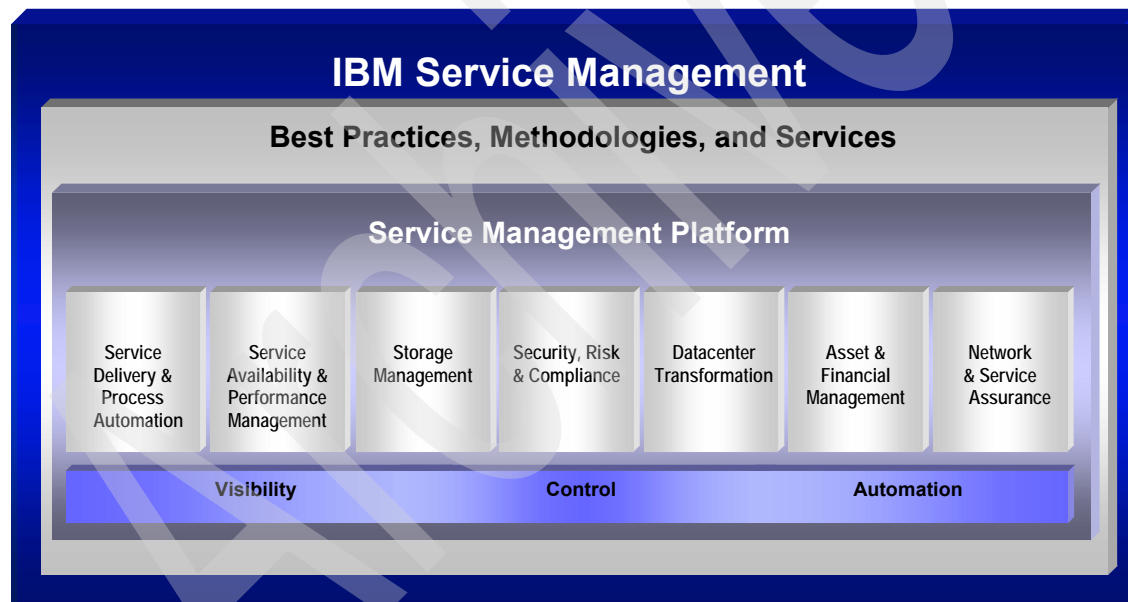


Figure 1-1 IBM Service Management

IBM Service Management includes a uniquely broad and modular set of capabilities that help customers better manage their business.

As organizations evolve from silo-centric to service-centric operations, they need to develop a common *manager-of-managers* view that integrates various monitoring tools across multiple management domains, including distributed data



centers, network operations centers (NOCs), security operations centers (SOCs), and other areas.

With IBM, organizations can gain accurate, timely, and deep visibility across operational silos, tools, and staff—without ripping and replacing existing tools and processes. Even better, IBM Consolidated Operations Management solutions at the heart of a comprehensive Service Management solution can leverage and integrate with hundreds of IBM and third-party tools for discovery, monitoring, event management, provisioning and service desk support, as well as change and configuration management databases (CMDBs).

The result is a powerful yet flexible solution to help quickly and easily identify and resolve problems, streamline operations, and meet the rigorous, complex service demands of today.

By joining the Tivoli leadership and experience managing data center environments with those of Netcool in the network operations center, IBM enables customers to benefit from fully integrated management software that shares event and performance management, visualization, and automated workflow capabilities across the enterprise. The combined Netcool and Tivoli portfolio will help users manage any data related to infrastructure elements such as networks, systems, security devices, storage components, and applications to gain full visibility into the health and performance of infrastructure-dependent services.

IBM is as committed to Netcool customers and products as it is to customers who have invested in Tivoli solutions. The company's strategy is to enable all Netcool and Tivoli users to protect, optimize, and extend their investments in the combined product portfolio.

- ▶ *Protect:* IBM seeks to protect customer investments of not only resources, but also knowledge accumulated over years of building ever more advanced IT operations infrastructures. As the Netcool and Tivoli product portfolios converge, IBM intends to provide smooth upgrade paths that facilitate adoption of the best capabilities across the combined portfolio while preserving and unlocking customers' knowledge investments.
- ▶ *Optimize:* IBM is helping customers leverage expanded capabilities today, even as work progresses toward the converged Tivoli portfolio. In product categories where the combined portfolio capabilities overlap, customers can *trade up* to the more feature-rich product in the category.
- ▶ *Extend:* Whether a customer currently uses Netcool products, Tivoli products, or both, the combined portfolio offers many additional products and capabilities that the organization can leverage.

Specifically, the Netcool portfolio offers Tivoli users a wide range of capabilities for security operations management, performance management, and network management. The Netcool portfolio further extends the Tivoli portfolio with next-generation management solutions for telecommunications infrastructures.

IBM is dedicated to every customer's success. As the company works to deliver a converged portfolio, it is taking numerous steps to enable the investments that customers have made in IBM and Micromuse products over the years to continue to benefit their organizations. Furthermore, the smooth upgrade paths that IBM is putting into place are meant to help customers derive even greater value from these investments moving forward.

## 1.2 Netcool/OMNIBus

IBM Tivoli Netcool/OMNIBus, with its Netcool/Webtop user interface, is the cornerstone of the IBM Tivoli Consolidated Operations Management solution. Netcool/OMNIBus is the heart of the Netcool product suite. It delivers real-time, centralized monitoring of complex networks and IT domains. With event processing scalability that can exceed over 100 million events per day, Netcool/OMNIBus offers round-the-clock management and automation to help customers deliver continuous uptime of business services and applications, optimize operations costs and efficiency, and improve time to market.

Netcool/OMNIBus includes over two hundred out-of-the-box probes (and more than 25 vendor alliances) that enable it to include events from virtually any management system or device in your Network and IT environment.

- ▶ Leader in Gartner Magic Quadrant and OSS Observer 2008 for both Enterprise and Service Provider markets
- ▶ Unique *manager-of-manager* capabilities for reduced operational expense
- ▶ Depth and breadth of event coverage and correlation
- ▶ Software failover for highly available consolidated operations management
- ▶ Event-processing efficiency resulting in cost savings and unmatched scalability
  - A single OMNIBus server can handle the event-processing workload of several servers using competitive offerings.
  - OMNIBus solutions can be scaled to handle greater than 50 million events per day, yet scale down to small and medium business environments.

Leading service providers use Netcool/OMNIBus to manage their complex networks and applications in real time, helping optimize the availability of fixed

and wireless services. The software management helps accelerate time to market of new services and maximize network reliability to enhance customer satisfaction and improve operator efficiency.

Leading enterprises rely on Netcool/OMNIBus to consolidate the management of networks and multiple management systems and tools under a *single pane of glass* view. The software helps make it easier for enterprises to manage problems across large heterogeneous networks and IT silos, and thereby reduce costs and improve overall productivity.

Environments supported by Netcool/OMNIBus software include a vast array of network devices, Internet protocols, systems, business applications, and security devices. Because the software offers breadth of coverage, rapid deployment, ease of use, and exceptional scalability and performance, enterprises and service providers worldwide can leverage the Netcool/OMNIBus suite to manage the world's largest, most complex environments.

Many customers use Netcool/OMNIBus to manage tens of millions of events daily. Furthermore, the software can be deployed in a distributed, parallel, or hierarchical fashion to support complex operations environments that span diverse geographic boundaries. Because it couples scalability with a flexible architecture, the software can deliver robust event management to support environments of any size.

## 1.3 Tivoli Enterprise Console customer choices

While significant focus is being placed on enhancing the ease of installation and use of upcoming versions of Netcool/OMNIBus, IBM will continue to protect our Tivoli Enterprise Console customers' investments and intends to provide a smooth upgrade path to Netcool/OMNIBus. Customers who do not yet need the enhanced event management and enrichment features offered by OMNIBus, and who are concerned about disrupting their environment, can continue to use Tivoli Enterprise Console as the main application for monitoring, and eventually set up an OMNIBus environment for test purposes. In this way they will be able to get acquainted with the enhanced event management and enrichment features, extra scalability, and performance provided by Netcool/OMNIBus. Customers who have an immediate need for these additional capabilities offered by OMNIBus can upgrade immediately.

## 1.4 Product review

In this section we discuss the major features of Tivoli Enterprise Console (TEC) and match them with the equivalent Netcool/OMNIbus features. This will give you a good idea of how your current system management functionality can be provided with Netcool/OMNIbus. The features and capabilities discussed are:

- ▶ Event management
- ▶ Event sources
- ▶ Event visualization
- ▶ Product administration and configuration
- ▶ Integration with other products

### 1.4.1 Event management

This section provides an overview of how events are managed within Tivoli Enterprise Console, followed by a comparison with Netcool/OMNIbus.

The Tivoli Enterprise Console is a rule-based event management application that manages events coming from many different sources. Each source has defined formats and slot definitions that are preloaded into the TEC configuration for use by the reception engine. These formatted events are stored in a relational database table (the reception log). A linear sequence of rules is built to process the incoming event, and these rules are loaded by the TEC start process. The rules can be complex, correlating the upcoming event against many existing cached events. After processing, the event is stored in further TEC database tables. A set of default rulesets is supplied with TEC to aid startup. TEC rules can start external procedures to call resources managed by the Tivoli Framework. The TEC console can be configured to display events logically grouped according to administration needs.

#### **Tivoli Enterprise Console**

In TEC, each source is configured with its own slot format and definition. The slot format and definitions are loaded into the TEC configuration, so this information can be mapped into the TEC environment. Once an event reaches the TEC reception log, this information is stored in the TEC database table. Then the rules engine is able to start processing the event. A linear sequence of rules is structured to handle the event. These rules are loaded in the TEC start process. The rules can be very complex, analyzing the incoming event and comparing it other events already stored. After event processing, the event is stored into another TEC database table.

The correlation of different types of events can be programmed. A set of default rulesets is deployed with TEC (for example, network up/down events that are

automatically closed, and repeat count field updates to register duplicated events).

The rules can start external procedures to run in resources managed by Tivoli Framework. The TEC console can be configured to display events that are logically grouped according to administration needs.

### **Netcool/OMNIBus**

In Netcool/OMNIBus, alerts are collected from event sources in the infrastructure environment, then processed and stored in a high-speed memory resident database called ObjectServer.

Netcool/OMNIBus can receive events from a variety of sources using probes and monitors. Events are sent from probes/monitors to the ObjectServer using a database middleware layer, which runs over TCP/IP. Probes are passive listeners that connect to an event source, detect and acquire event data, and forward them to the ObjectServer as alerts. Probes use the logic specified in a rules file to manipulate the event elements before converting them into fields of an alert in the ObjectServer alerts.status table.

The difference in approach to event processing with TEC is that OMNIBus is designed to automatically detect duplicate and correlated events and only update those affected fields (such as time of delivery, duplicate count, event metrics). These features are provided with an out-of-the-box OMNIBus installation, but can be customized on demand.

## **1.4.2 Event sources**

This section provides an overview of the event sources available for Tivoli Enterprise Console, followed by a comparison with those available for Netcool/OMNIBus.

### **Tivoli Enterprise Console**

An *adapter* is a process that monitors resources so that they can be managed. These monitored resources are called *sources*. A source is an application (for example, a database) or system resource (for example, an NFS server). When an adapter detects an event generated from a source (generally called a raw event), it formats the event and sends it to the event server. The event server then further processes the event.

Adapters can monitor sources in the following ways:

- ▶ An adapter can receive events from any source that actively produces them. For example, SNMP adapters can receive traps sent by the Simple Network Management Protocol (SNMP).
- ▶ An adapter can check an ASCII log file for raw events at configured intervals if the source updates a log file with messages.

Adapters can send events to the event server using a Tivoli interface or a non-Tivoli interface. Both types of interfaces send events using an ordinary TCP/IP channel. The difference between the two interfaces is the method used to establish the connection.

Other event sources for TEC are *IBM Tivoli Monitoring*, *Tivoli NetView*, and *Tivoli Business Manager 3.1*.

## **Netcool/OMNIbus**

Netcool/OMNIbus can receive events from a variety of sources using *probes* and *monitors*. Events are sent from probes/monitors to the ObjectServer using a database middleware layer, which runs over TCP/IP.

Probes are passive listeners that connect to an event source, detect and acquire event data, and forward them to the ObjectServer as alerts. Probes use the logic specified in a rules file to manipulate the event elements before converting them into fields of an alert in the ObjectServer alerts.status table.

Each probe is uniquely designed to acquire event data from a specific source. Netcool/OMNIbus collects data from thousands of device types through a range of generic data collectors (probes) including SOA, SNMP, log, and socket. The wide coverage is augmented by over 200 vendor-specific probes. Probes can acquire data from any stable data source, including devices, databases, and log files.

Netcool/OMNIbus 7.2 probes can communicate with an ObjectServer via IPv6 transmission protocol.

Netcool/OMNIbus also integrates with the Tivoli Composite Application Manager for Internet Service Monitoring product that provides monitors for testing the availability and performance of Internet services (HTTP, HTTPS, DNS, POP, SNMP, FTP, DHCP, IMAP, and so on).

Other event sources for Netcool/OMNIbus are other management applications of the Netcool suite: *Tivoli Business Service Manager 4.1* (formerly known as Netcool/Realtime Active Dashboard) and *Tivoli Network Manager IP Edition* (formerly known as Netcool/Precision for IP Networks).

Other Netcool suite components exist to enrich events received from Netcool/OMNIbus, such as Tivoli Network Manager IP Edition's topology-based Root Cause Analysis (RCA) engine and *Netcool/Impact*.

### 1.4.3 Event visualization

This section provides an overview of how events information can be viewed in Tivoli Enterprise Console, followed by a comparison with the visualization provided by Netcool/OMNIbus.

#### **Tivoli Enterprise Console**

Event consoles provide a GUI that allows the IT staff to view and respond to dispatched events. A senior administrator configures multiple event consoles based on the responsibilities of the IT staff. Users can have independent or shared views of events.

IBM Tivoli Enterprise Console presents two versions of the event console:

- ▶ **Java event console:** The Java version of the event console can be installed on a managed node, an endpoint, or a non-Tivoli host. The Java event console provides a full set of features needed by Tivoli Administrators to perform configuration tasks, start Tivoli NetView functions, run local automated tasks, and manage events.
- ▶ **Web event console:** The Web version of the event console can be used only to manage events from a Web browser. The Java console is still necessary to perform any other tasks available from the event console (that is, configuration tasks).

Figure 1-2 and Figure 1-3 on page 13 show the Summary Chart View and the Event Viewer.

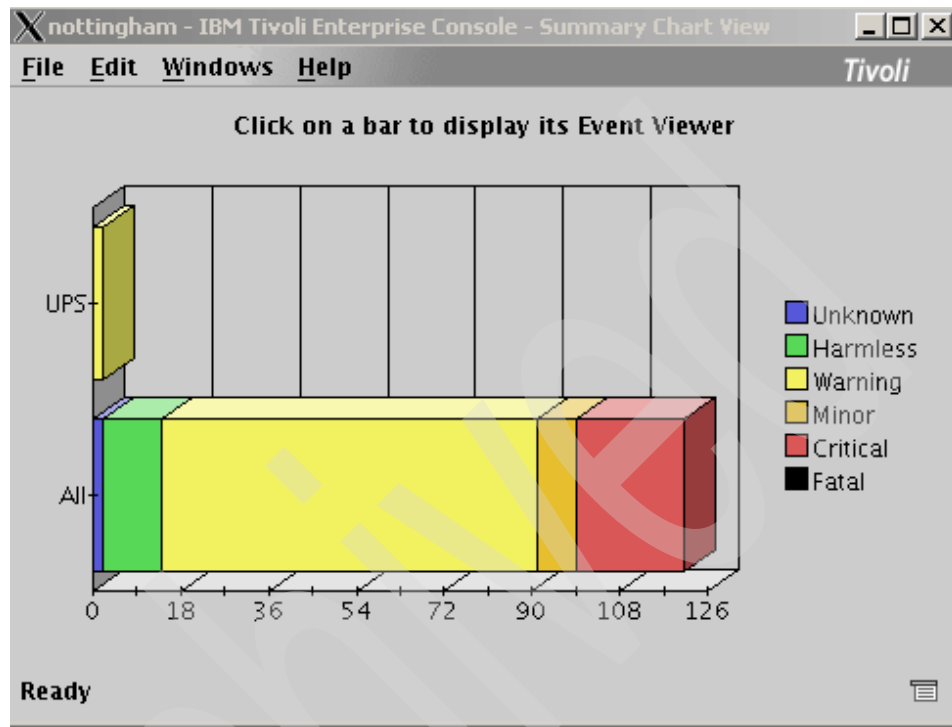


Figure 1-2 Summary Chart View



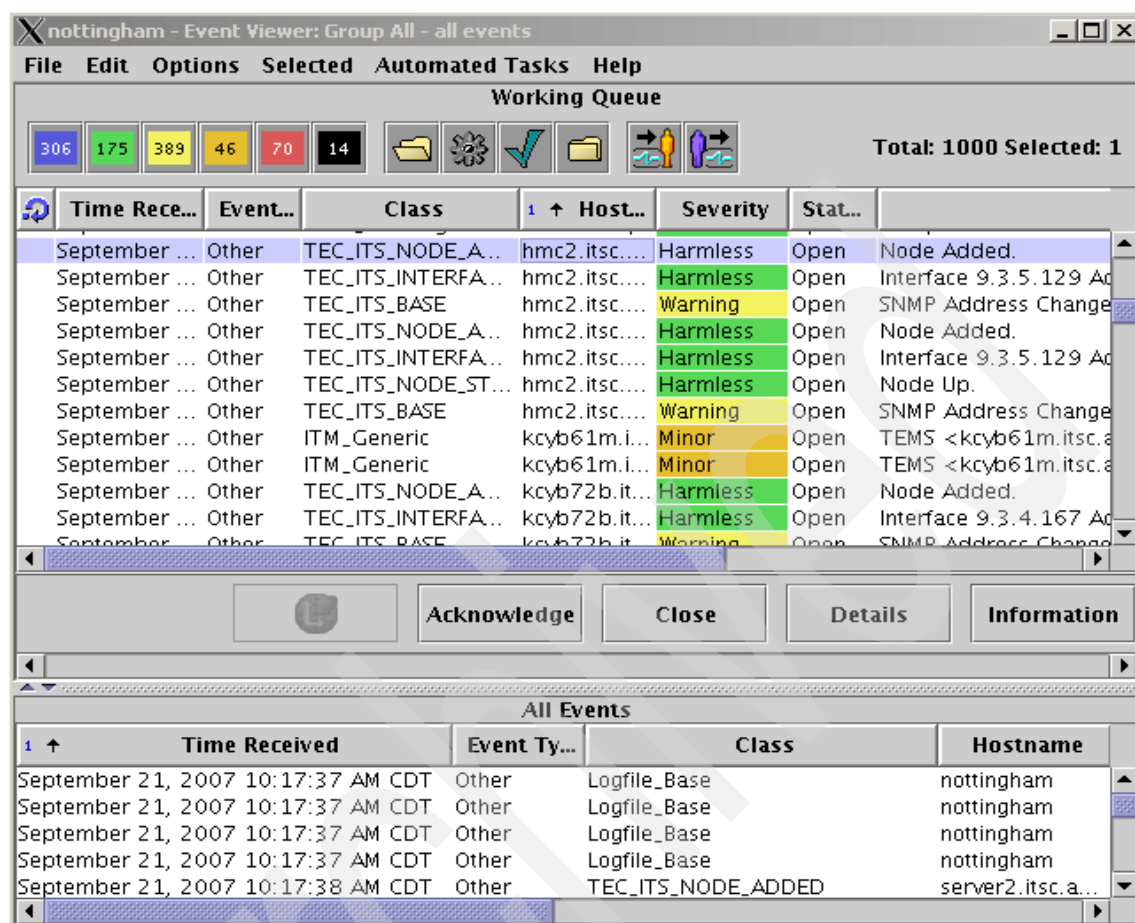


Figure 1-3 Event Viewer

## Netcool/OMNIBus

Like Tivoli Enterprise Console, Netcool/OMNIBus has a native and Web-based Graphical User Interface (GUI).

The native GUI consists in an integrated suite of graphical tools used to view and manage alerts and to configure how alert information is presented. The focal point for the visualization of events information is called *event list*.

Group	Summary	Last Occurrence(+)	Count
	A isql process running on loureed has disconnected	10/24/03 14:05:08	1
	Diskspace alert	10/24/03 14:44:32	1572
	Diskspace alert	10/24/03 14:44:40	1383
	Diskspace alert	10/24/03 14:45:43	4732
	Diskspace alert	10/24/03 14:46:02	5293
	Machine has gone offline	10/24/03 14:50:35	8222
	Machine has gone online	10/24/03 14:50:58	8222
@C0A823A	Attempt to login as admin_12 from host loureed failed	10/24/03 14:51:00	1
@C0A823A	A e@C0A8226C@C0A823A1:0.0 process running on loureed has connected	10/24/03 14:51:04	1
	Link Up on port	10/24/03 14:51:29	201
	Port failure : port reset	10/24/03 14:51:31	1532

Summary bar: 6 (Green), 2 (Purple), 6 (Blue), 2 (Yellow), 12 (Orange), 6 (Red), All Events

Status bar: 0 rows inserted, 21 rows updated and 0 rows deleted | 10/24/03 14:52:44 | root | LON\_PRIM [PRI]

Figure 1-4 The Netcool/OMNIBus event list

The event list can also be accessed from *Netcool/Webtop*. Webtop is the Web-based Netcool/OMNIBus desktop. It supports the same tasks that can be performed through the OMNIBus native GUI, offering an additional range of GUI building blocks that operators can use to enrich the visualization, such as:

- ▶ Maps
- ▶ Charts
- ▶ Tabbed windows
- ▶ Customized pages

Figure 1-5 shows the event list visualization within Netcool/Webtop.

Agent	Serial	Node	Count	LastOccurrence	Summary
	3128	griffint41p	1	16/01/06 09:31:15	A.JavaAdmin process running on griffint41p has disconnected
	840		1	19/05/05 16:37:10	A process running on has disconnected
	3127	griffint41p	2	13/01/06 17:10:00	A.OBJSERVAUTH process running on griffint41p has disconnected
	2920	griffint41p	2	13/10/05 14:49:08	ANT Event List@C0A82275 process running on griffint41p has disconnected
	3193	griffint41p	1	16/01/06 15:36:39	A.PROBE process, simnet, running on griffint41p has disconnected
sla	1014	199.73.184.108	6990	16/01/06 15:36:01	Authentication Failure ( Enterprise = 1.3.6.1.4.1.253.8.62.1.1.21.1.3 )
	3051	griffint41p	2	08/11/05 11:43:48	ANT Event List@AC1995E7 process running on griffint41p has disconnected
sla	1038	213.218.230.35	6105	16/01/06 15:36:26	Authentication Failure ( Enterprise = 1.3.6.1.4.1.253.8.62.1.1.21.1.3 )
	3041		16	13/01/06 17:09:50	Attempt to login as root from host failed
	3059	griffint41p	4	13/01/06 17:09:21	Attempt to login as root from host griffint41p failed
performance	1023	201.41.66.85	6631	16/01/06 15:36:05	System Reboot
performance	1000	35.89.170.188	7051	16/01/06 15:35:54	Cisco Memory Utilization Excessive
fault	1021	106.68.14.249	9896	16/01/06 15:36:12	Device not responding
performance	993	196.206.32.61	9204	16/01/06 15:36:06	10.0.1.20 is the New Root of the Spanning Tree
performance	998	121.83.93.87	10769	16/01/06 15:36:24	Authentication Failure ( Enterprise = 1.3.6.1.4.1.253.8.62.1.1.21.1.3 )
sla	1125	197.160.153.197	4267	16/01/06 15:35:26	Configuration Changed via Command Line: ConfigDataSource = Comm
performance	1003	213.123.206.25	7017	16/01/06 15:36:26	DVC Failed - Connection closed unexpectedly
sla	1065	198.182.113.233	6432	16/01/06 15:36:20	10.0.1.20 is the New Root of the Spanning Tree
fault	1087	68.38.75.44	7743	16/01/06 15:36:39	Interface Status Transitions
sla	989	128.76.240.89	7242	16/01/06 15:36:37	DASD backplane number 1 has reported a fault on hard drive number 0
performance	1079	164.82.186.164	6138	16/01/06 15:36:01	Previously Unavailable Device Now Responding
fault	1128	79.93.252.217	4087	16/01/06 15:36:14	Monitor 'SYS-Volume' Low Error Volume SYS: Space Remaining' has b

Figure 1-5 Event list in Netcool/Webtop

It is important to point out that the Netcool/OMNibus GUI provides a much wider range of visualization and event management features, compared to Tivoli Enterprise Console. These are immediate advantages that Netcool/OMNibus customers can benefit from.

#### 1.4.4 Product administration and configuration

This section provides an overview of how administration and configuration tasks are performed in Tivoli Enterprise Console, followed by a comparison to the corresponding way provided by Netcool/OMNibus.

##### Tivoli Enterprise Console

TEC uses the Tivoli Framework to authenticate the user's access. Each Tivoli Enterprise Console can be configured to assign the groups and operators to design the particular profile. The operator must be defined as a Tivoli

administrator and have the appropriate Tivoli authorization roles to access TEC resources.

Figure 1-6 and Figure 1-7 on page 17 show the Administration panel from the Tivoli Framework Desktop interface and Tivoli Enterprise Console Administration panel, where the association of Tivoli Framework Administration to TEC console should be configured.

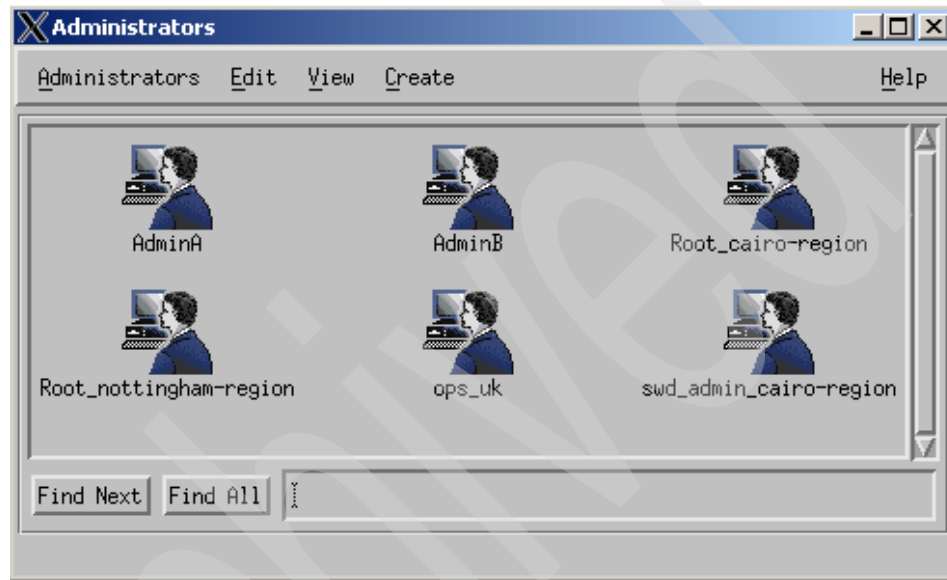


Figure 1-6 Administration in Tivoli Framework Desktop Interface

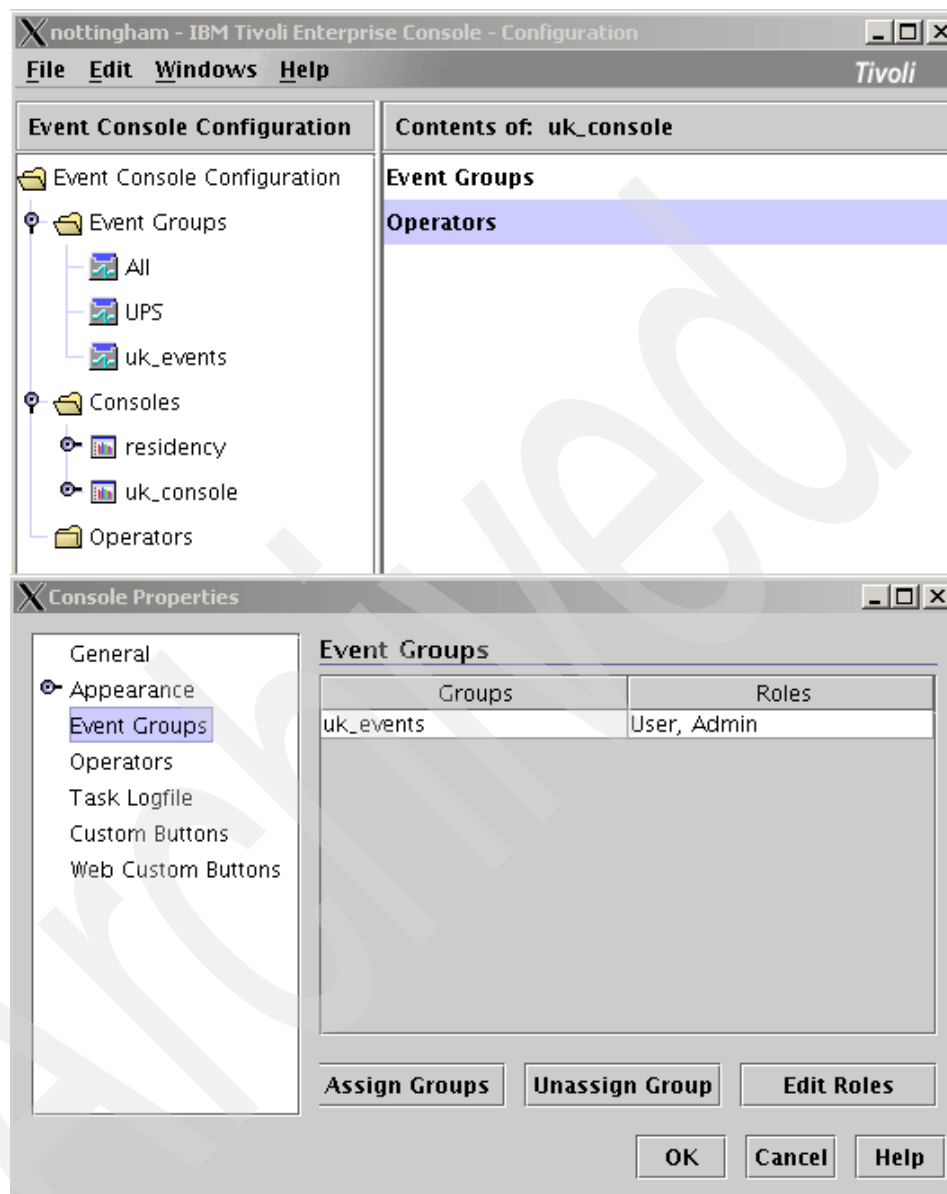


Figure 1-7 TEC Administration Configuration panel

## Netcool/OMNibus

Administration and configuration of Netcool/OMNibus ObjectServer involve the following tasks:

- ▶ Defining and editing users, groups, roles, and restriction filters
- ▶ Editing the structure of the ObjectServer
- ▶ Defining and editing SQL procedures and database triggers

These tasks can be performed in the following ways:

- ▶ Using ObjectServer SQL interactive interface
- ▶ Through the Netcool/OMNibus Administrator GUI

The ObjectServer SQL interactive interface is a command-line utility that permits the execution of SQL queries against the ObjectServer. The user can log in to the ObjectServer with the appropriate credentials and type the SQL queries to be performed.

The administrator GUI provides the ability to accomplish the above tasks in an easier way, simply by clicking the required menu in the Administrator window. In addition, through administrator it is possible to manage multiple ObjectServers and process control agents from a single console.

Figure 1-8 shows the Administrator window.

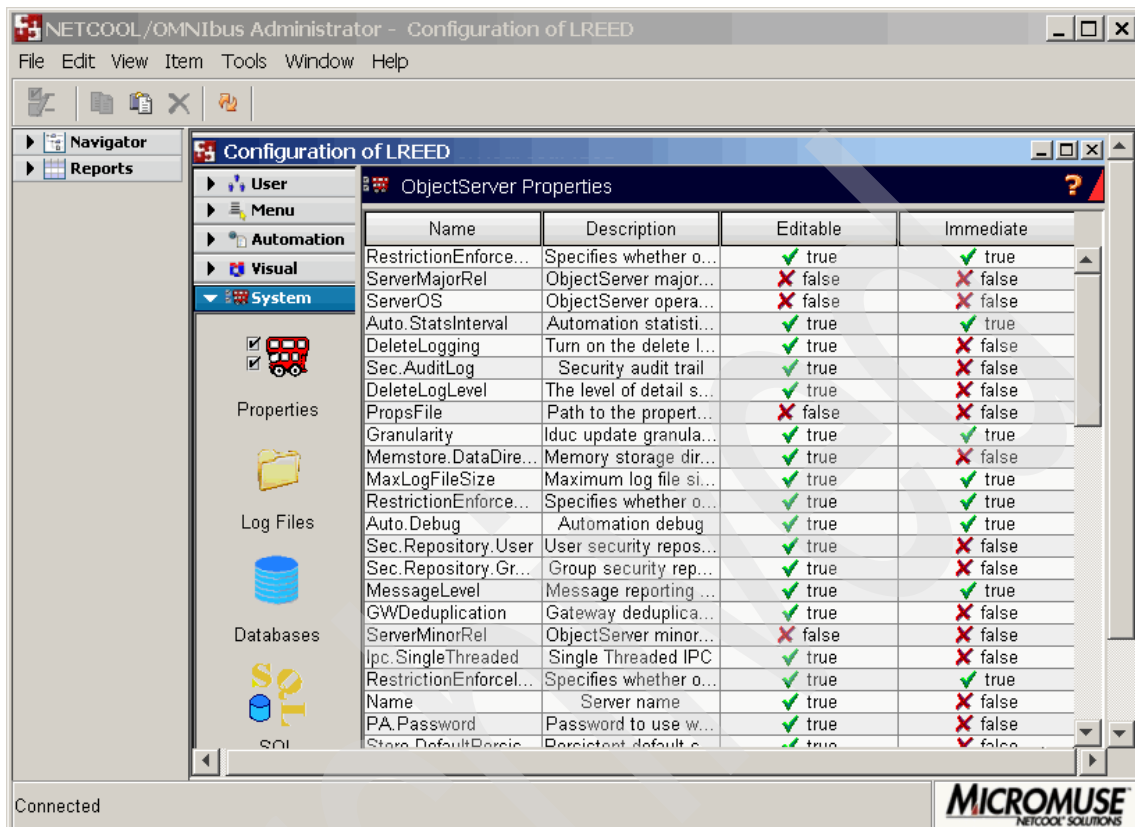


Figure 1-8 Example of Administration window

It is interesting to notice that there is a GUI version of the SQL interactive interface in the Administrator window (Figure 1-9).

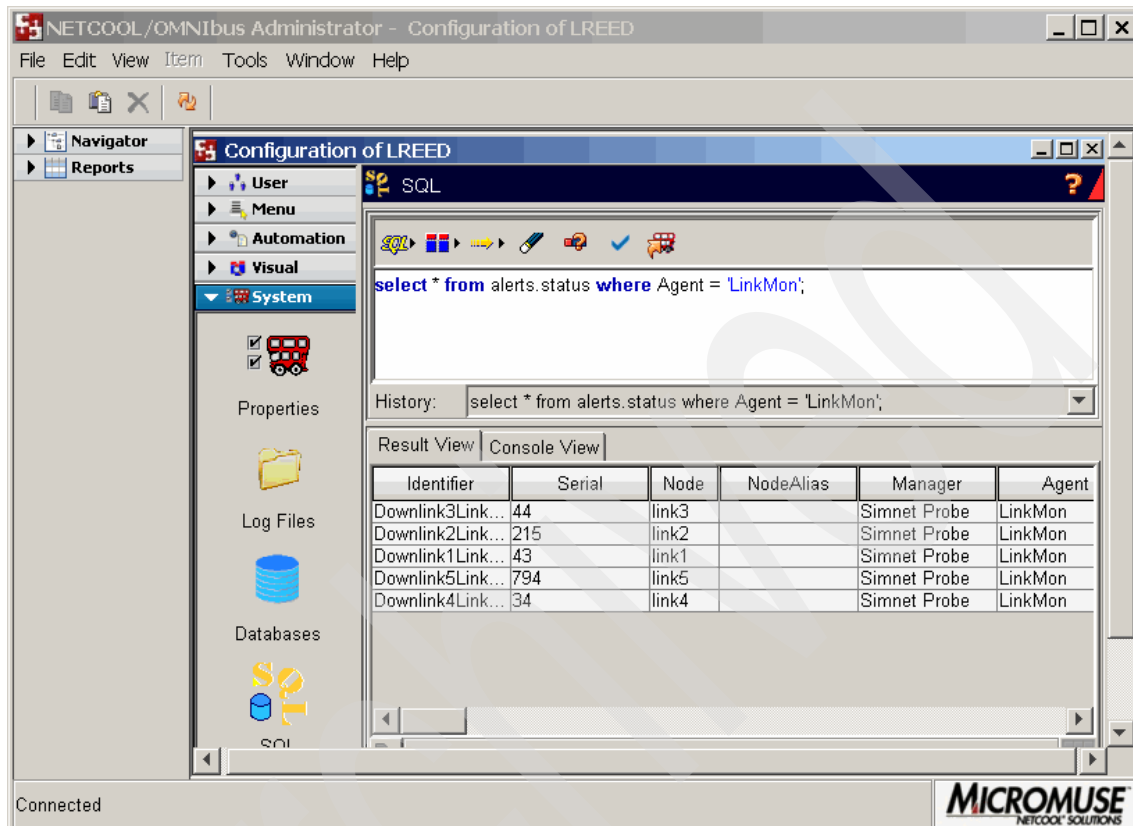


Figure 1-9 SQL interactive interface in the Administrator window

Netcool/OMNIBus user management features are discussed in detail in 1.5.7, “Security” on page 33.

### 1.4.5 Integration with other products

This section describes the typical products that are used with Tivoli Enterprise Console and compares them with Netcool/OMNIBus.

#### Tivoli NetView

Tivoli NetView is a network management product that discovers TCP/IP networks, displays network topologies, correlates and manages events and SNMP traps, monitors network health, and gathers performance data.



In the NetView configuration steps, customers can setup forwarding network events to Tivoli Enterprise Console. NetView provides the necessary definition (BAROC) file to be loaded into the TEC rulebase and a template of predefined rules that correlate events from NetView.

### **IBM Tivoli Monitoring (ITM) V5.1.x**

IBM Tivoli Monitoring allows you to monitor the availability and performance status of resources on your systems to identify bottlenecks and potential resource problems.

This product requires the Tivoli Framework infrastructure to be installed and managed. It follows the Tivoli Framework object concepts: profile managers, profile, subscribers, and endpoints.

ITM 5.1 has in the profile configuration a setup to send the generated events to the event server. The product provides the necessary BAROC file to be loaded into the TEC rulebase.

### **IBM Tivoli Monitoring (ITM) V6.1**

IBM Tivoli Monitoring V6.1 has a different concept of monitoring servers when compared to the previous version V5.1.

It presents different components: Tivoli Enterprise Management Server (TEMS), Tivoli Enterprise Portal (TEP), and Tivoli Enterprise Monitoring Agents (TEMA).

The agent (TEMA) that runs on target machines is completely different from the module that is distributed into the endpoint code.

The product provides the necessary BAROC file to be loaded into the TEC rulebase.

### **IBM Tivoli Monitoring V6.2**

IBM Tivoli Monitoring V6.2 is an upgrade for ITM 6.1.

The Tivoli Enterprise Portal (TEP) has some additional functions to customize situations to send events to TEC, for example:

- ▶ Setting the severity
- ▶ Defining multiple EIF receivers to receive the ITM events
- ▶ Setting up each situation individually to send events to EIF receivers

## 1.4.6 Netcool/OMNIBus

Netcool/OMNIBus *gateways* enable the integration between ObjectServers and complementary third-party applications, such as databases and helpdesk or Customer Relationship Management (CRM) systems.

Gateways are able to send alerts to a variety of targets:

- ▶ A database
- ▶ A helpdesk application
- ▶ Other applications or devices

Database gateways are used to store alerts from an ObjectServer. This is useful to keep a historical record of the alerts forwarded to the ObjectServer.

Helpdesk gateways are used to integrate Netcool/OMNIBus with a range of helpdesk systems. This is useful to correlate the trouble tickets raised by your customers with the networks and systems that you are using to provide their services.

Other gateways are specialized applications that forward ObjectServer alerts to other applications or devices (for example, a flat file or socket).

### Tivoli Network Manager IP Edition

Tivoli Network Manager IP Edition (formerly Netcool/Precision for IP Networks) includes complete layer 2 and event management capabilities out of the box. Like Tivoli NetView, you can customize the console interface to include contextual launch points to third-party products such as element managers and other diagnostic tools from event and topology views. You can also augment the console with CGI-served portlet views by the Netcool GUI Foundation server or from the Internet.

Tivoli Network Manager IP Edition includes a powerful perl API for access to all the data in its various components via *Object Query Language* (OQL) commands. This is widely used as a scripting language for extending the product, such as creating new discovery agents, custom reports, and so forth. Agents and stitchers can be deployed to extend discovery for specialized devices or purposes.

### Netcool/Impact

Impact extends the functionality of the Netcool suite by allowing automation to correlate, calculate, enrich, deliver, notify, escalate, visualize, and perform a wide range of automated actions by accessing data from virtually any source. The key advantage that Impact provides when performing these tasks is that it accesses the data in real time.

## **Tivoli Business Service Manager 4.1**

Tivoli Business Service Manager 4.1 (formerly Netcool/Realtime Active Dashboard) helps business and operations staff understand the complex relationships between business services and supporting technology. It gives organizations advances, and real-time visualization of services and processes in a comprehensive service dependency model.

## **1.5 Benefits of upgrading to Netcool/OMNIBus**

In this section we take a look at some of the extra features that upgrading to Netcool/OMNIBus will bring to existing Tivoli Enterprise Console customers. This is not designed to be an exhaustive list of the capabilities of Netcool/OMNIBus, but instead to demonstrate the flexibility of Netcool/OMNIBus and the Netcool suite. The following topics are included:

- ▶ Scalability and performance
- ▶ Consolidation
- ▶ Ease of use
- ▶ Reliability
- ▶ Enhanced event visualization and management
- ▶ Enhanced event enrichment
- ▶ Security
- ▶ IPv6 support
- ▶ IBM product strategy

### **1.5.1 Scalability and performance**

Netcool/OMNIBus' ultra-scalable event architecture enables growth over time from the minimum configuration through its modular architecture, enabling expansion without re-engineering of the core solution. This enables the customers to optimize investment as service demand increases.

A Netcool/OMNIBus environment is scalable in a variety of ways to meet increased need:

- ▶ Additional and new network equipment: As new network equipment is added to the system, it may be monitored by existing or newly deployed probes. New probes are added to the managed domain without any Netcool/OMNIBus system downtime. The exclusive data paradigm means that if a probe has access to an element, its events will be reported in the ObjectServer and visible on the desktop regardless of whether the element appears within models used by other parts of Netcool. Netcool/OMNIBus has no

application-enforced limit to the number of elements that can be managed because its functionality is not dependent on a finite model.

- ▶ Increased event load: The ObjectServer memory-resident database holds only current events for those elements with active alarms. Historical data is archived through an RDBMS gateway, ensuring that the memory requirements are kept to a minimum. Only in the largest installations does it become necessary to scale the Netcool/OMNIbus architecture across additional tiers of ObjectServers. In a tiered architecture, data collection ObjectServers are deployed, providing an intermediate data handling layer below the aggregation capability.
- ▶ Large User Community: Netcool/OMNIbus can support the majority of operational users within a simple deployment. As user numbers grow, specialized Display ObjectServers may be deployed into a two-tier or three-tier structure to remove the desktop load from the aggregation server.

Designed for ultimate scalability, the high-speed Netcool/OMNIbus ObjectServer collects event information from across the infrastructure, deduplicating and filtering the data so that it can be effectively managed. With event processing support scalable up to the 10–100 million events per day range, Netcool/OMNIbus can be deployed in a distributed, parallel, or hierarchical fashion to support complex operations environments that span diverse geographic boundaries.

The features of Netcool/OMNIbus that contribute to reduced application workload are:

- ▶ Greater scalability through routing of events to multiple tables and ObjectServers: This can be used to help reduce the load on the core of the Netcool/OMNIbus system while ensuring that tables are not overpopulated with inappropriate events.
- ▶ Dynamic configuration and administration: The ObjectServer can be configured while running, reducing downtime.
- ▶ Advanced Profiling: Major improvements to profiling and reporting mechanisms capture resource usage data for all connected clients and for individual trigger processes within the automation system. This can be used to automatically adjust the configuration of the system to ensure that it is optimized.
- ▶ Load balancing: Connection-based load balancing is a part of the desktop ObjectServer architecture. This enables the administrator to define a set of display servers to which desktops can be connected.

Common Event Format and triggers provide a focused structure for consolidated management of disparate entities. Other approaches process an order of

magnitude fewer events, necessitating trade-offs between breadth and speed, which reduces accuracy and time to resolution.

You can use triggers to detect changes in the ObjectServer and execute automated responses to these changes. This enables the ObjectServer to process alerts without requiring an operator to take action.

Netcool/OMNIbus is shipped with a set of standard automations that includes functions like:

- ▶ Backing up the ObjectServer
- ▶ Adding alerts to the ObjectServer
- ▶ Clearing the events on a time basis that can be specified by the operator
- ▶ Correlating the events following a problem/resolution criterion and intercepting the attempt to insert duplicate rows in the database
- ▶ Deleting the clear events after a certain amount of time
- ▶ Inserting journal entries
- ▶ Removing redundant entries from various tables
- ▶ Correlating interconnected events (very useful for root cause analysis of failures)
- ▶ Deleting the less important events, giving operators the opportunity to focus on the critical ones
- ▶ Sending e-mails to operators in case of the occurrence of particular events
- ▶ Creating alerts in response to the occurrence of particular events
- ▶ Automatically acknowledge/deacknowledge events
- ▶ Automatically change the status or the criticality of events

These automations are completely customizable.

## 1.5.2 Consolidation

OMNIbus can receive events from a variety of sources using probes and monitors. In addition, the typical components of the Tivoli Framework (Tivoli Enterprise Console, IBM Tivoli Monitoring, Tivoli NetView) can also act as event sources for OMNIbus. This feature is most useful when applied in complex environments. In fact, OMNIbus can consolidate events from multiple TEC servers, therefore making it easier to manage all the alerts coming from different regions of the infrastructure.

### 1.5.3 Ease of use

OMNIbus is delivered in the Netcool Installer that provides standardized installation and software management methods across the Netcool suite. The installer provides a default GUI mode and two command-line options to allow deployment on servers with limited user display functions or making use of a user prepared set of installation options to run without any user interaction.

A basic configuration can be installed and collecting alarms within a few hours. A fully customized solution including integrations to third-party and other Netcool applications would typically take a few weeks.

Low training costs and ease of use provides immediate return on investment. Off-the-shelf Netcool probes and monitors can be installed quickly and will immediately begin collecting alarms from a wide variety of data sources. With quick installation and setup time, OMNIbus stores alarms, performs basic correlation and deduplication, and then makes the event available to the other components of the Netcool suite.

OMNIbus also has many dynamic features, improving productivity and meaning less interruption to service when compared to TEC. For example, changes to activate, deactivate, or debug automations can be performed with a single click in OMNIbus compared to the process of editing rule files, compiling and loading rulebases, and stopping and starting the event server that we have in TEC.

### 1.5.4 Reliability

The software-based failover architecture of the Netcool/OMNIbus solution provides reliable 24x7 operational cover for many of the world's largest operations centers. The majority of maintenance tasks can be executed in real time without service interruption. Automated backup routines provide the means of archiving the state of the current configuration.

Netcool/OMNIbus supports failover and failback tasks through the setup of a *failover configuration*.

The failover configuration consists of installing two separate ObjectServers that, respectively, act as primary and backup data repositories.

The data flow between the primary and the backup ObjectServer is handled by an *ObjectServer bidirectional gateway*, which allows alerts to flow from a source ObjectServer to a destination ObjectServer. Changes made to the contents of a source ObjectServer are replicated in a destination ObjectServer, and the destination ObjectServer replicates its alerts to the source ObjectServer.

Failover occurs when the component loses its connection to the primary ObjectServer. The component will then connect and forward events to the backup ObjectServer. Failback functionality allows the component to reconnect to the primary ObjectServer when it becomes active again.

**Note:** The gateway is always connected to the backup ObjectServer. On failure of the primary, the gateway loses its connection to the primary, but remains connected to the backup.

In this way, it is possible to keep on working on the backup ObjectServer even if the primary ObjectServer is not available, because all the changes are replicated in it as soon as it is up and running again.

Additionally, Netcool/OMNibus probes can continue to run if the target ObjectServer is down. This prevents the loss of alerts coming from the monitored infrastructure in case of an ObjectServer unavailability.

To accomplish this, when the probe detects that the ObjectServer is not present (usually because it is unable to forward an alert to the ObjectServer), it switches to *store* mode.

In this mode, the probe writes all of the messages it would normally send to the ObjectServer to a file.

When the probe detects that the ObjectServer is back online, it switches to forward mode and sends the alert information held in the file to the ObjectServer. Once all of the alerts in the file have been forwarded, the probe returns to normal operation.

Store and forward functionality is enabled by default, and is activated once a connection to the ObjectServer has been established. If the ObjectServer is not running when the probe starts for the first time, store and forward mode is not triggered and the probe terminates.

**Note:** After installation, a probe needs to learn the ObjectServer schema. This happens on the first connection of the probe to the ObjectServer. Therefore, store and forward mode can be activated, in case of ObjectServer unavailability, only if the probe has successfully connected to the ObjectServer at least one time previously.

In addition to this, two instances of a probe can run simultaneously in a peer-to-peer failover relationship. One instance is designated as the master. The other acts as a subordinate and is on hot standby. If the master instance fails, the subordinate instance is activated. This is called *peer-to-peer failover*.

**Note:** Peer-to-peer failover is not supported for all probes.

### 1.5.5 Enhanced event visualization and management

The Netcool/OMNIbus GUI provides unique flexibility in the visualization and the management of events.

The core component of the Netcool/OMNIbus GUI is the event list, which enables the operator to view and manage alerts.

Information about alerts is displayed in the event list according to filters and views. Filters enable the operator to display a subset of alerts based on specific criteria. Views enable the operator to choose which alert fields to display.

To work on an alert, it has to be selected first. This can be achieved by clicking it. The alert can then be:

- ▶ Acknowledged
- ▶ Deacknowledged
- ▶ Deleted
- ▶ Prioritized
- ▶ Owned
- ▶ Assigned to users
- ▶ Resolved



Figure 1-10 shows how to execute tasks on selected alerts.

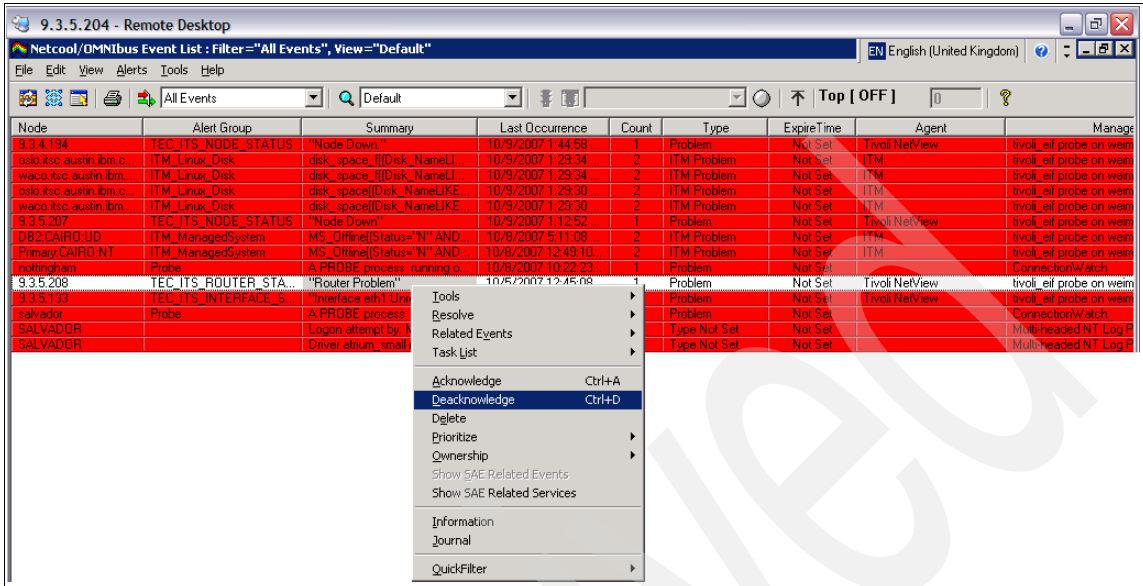


Figure 1-10 Available tasks for selected alert

The range of tasks that can be performed on a selected alert can be enriched with additional tools that can be easily deployed through the administrator console, shown in Figure 1-8 on page 19. In this way, operators can perform some common management operations on the go, simply clicking the events that they want to manage.

Due to the high number of events that OMNIBus can handle, it is useful to be able to filter out the information of interest, and also select what aspects of an alert we are interested in. This can be achieved through the *filter builder* and the *view builder*.

The filter builder is a GUI tool used to define SQL queries against the ObjectServer alerts database tables in order to display custom alert information in monitor boxes and event lists.

Figure 1-11 shows the filter builder tool in UNIX.

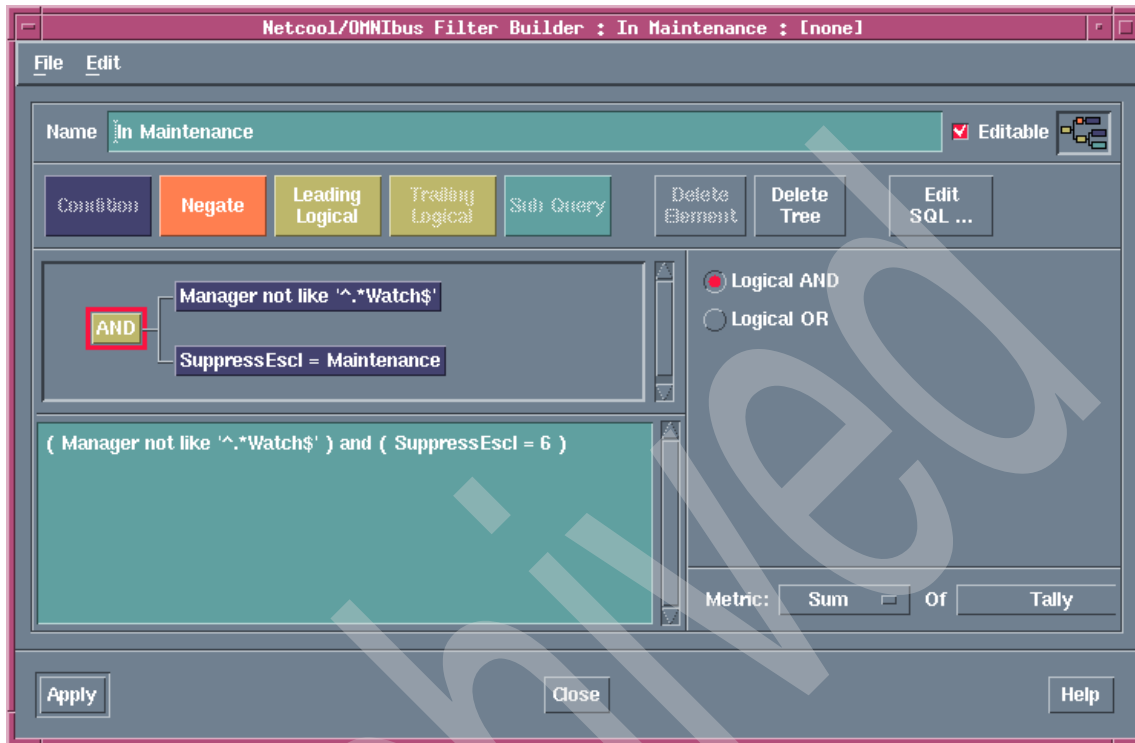


Figure 1-11 Filter builder tool in UNIX

The SQL query can be entered manually in the dedicated space or built through the appropriate SQL building blocks provided within the interface.

### View builder

Views define the type and formatting of information that appears in event lists. For each view, you can specify:

- ▶ Attributes to include in the event list
- ▶ Field titles
- ▶ Field formatting
- ▶ How information is sorted
- ▶ Whether to restrict the number of rows displayed in the event list

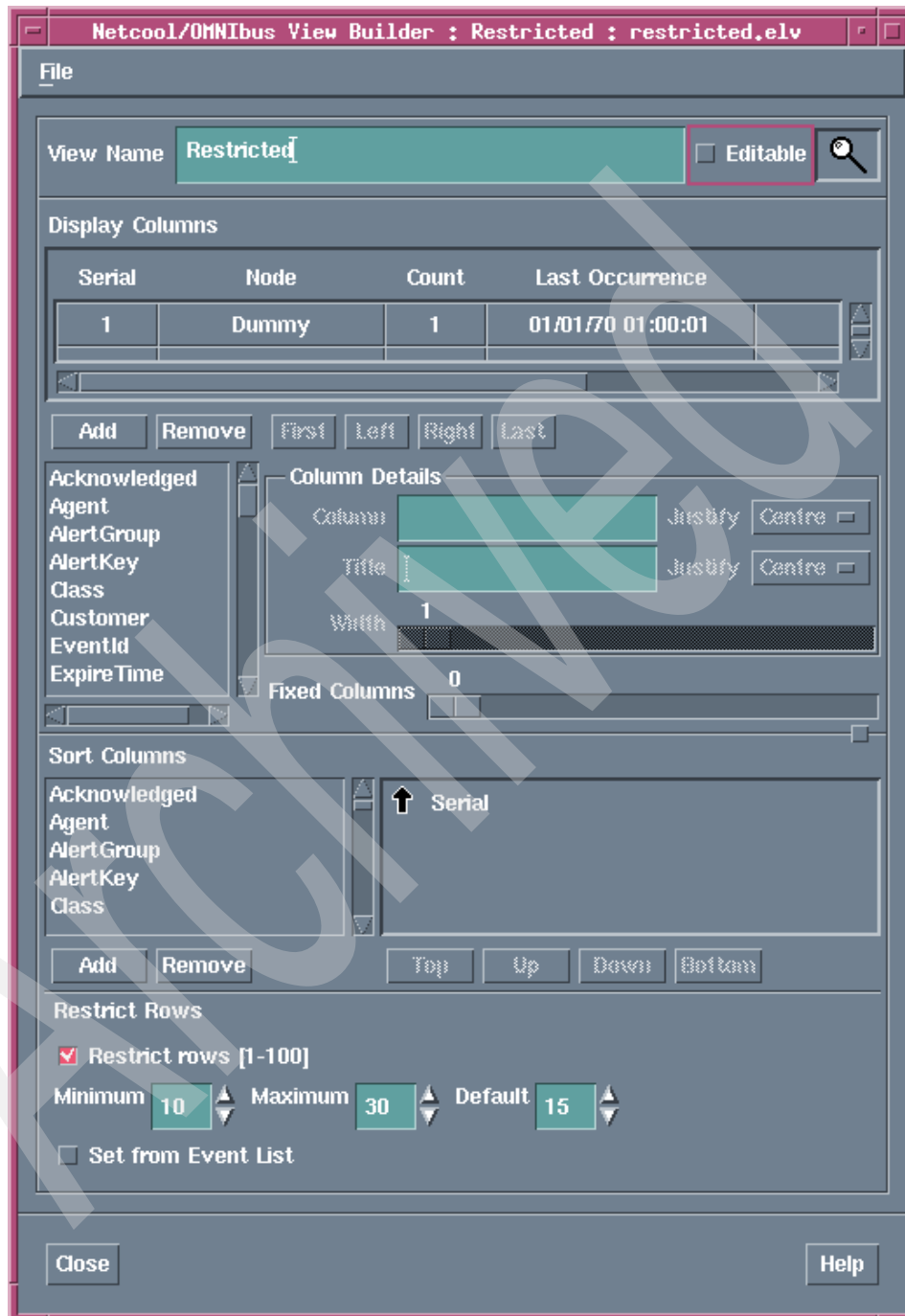


Figure 1-12 View builder

## 1.5.6 Enhanced event enrichment

Netcool/OMNIBus is not only a repository for collected events, but also the central unit for event enrichment.

Additional information to be used to enrich events can be gathered both from OMNIBus internal sources and other products of the Tivoli Netcool suite:

- ▶ Probes lookup tables
- ▶ Netcool/Impact
- ▶ Tivoli Business Service Manager 4.1
- ▶ Tivoli Network Manager IP Edition

### Probes lookup tables

Lookup tables provide a way to add extra information in an event. A lookup table consists of a list of keys and values. It is defined with the table function and accessed using the lookup function.

The lookup function evaluates the expression in the keys of the named table and returns the associated value. If the key is not found, an empty string or the defined default value is returned.

Lookup tables are very useful because they provide a way to enrich the event directly after the collection from the source, and therefore they have no impact on the workload at the ObjectServer level.

### Netcool/Impact

Netcool/Impact complements Netcool/OMNIBus by enriching events with contextual data from almost any source and performing automated actions. Its dynamic real-time data access capabilities deliver a unique and highly scalable approach that facilitates rapid problem resolution.

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.

### Tivoli Business Service Manager 4.1

The Tivoli Business Service Manager 4.1 (TBSM 4.1) product delivers the technology for IT and business users to visualize and assure the health and performance of critical business services.

The TBSM 4.1 tools enable you to build a service model that can be integrated with OMNIBus object server alerts, or optionally with data from an SQL data source. TBSM 4.1 processes the external data based on the service model data

that you create in the TBSM 4.1 database and returns a new or updated TBSM 4.1 service event to the OMNibus object server.

The TBSM server analyzes ObjectServer events or SQL data from other data sources, for matches against the incoming status rules configured for your service models. If matching data changes the service status, the status of the TBSM service model changes accordingly. When a services status changes, TBSM sends corresponding events back to the ObjectServer.

### **Tivoli Network Manager IP Edition**

The addition of Tivoli Network Manager IP Edition to an OMNibus environment extends the network management capabilities to include extensive automated network discovery and best-of-breed topology-based root cause analysis, providing customers with the best possible real-time understanding of their network infrastructures and the fastest possible resolution of network problems.

Network information and relations between network devices discovered by TNM Agents are collected and stored within the TNM database. A graphical network topology (Topoviz) is built based on this information. The graphical network topology together with real time events from network devices provides the information for root cause analysis (RCA). RCA works with OMNibus probe events from network devices (SNMP and Syslog) together with probe events from other sources. These tools can help reduce the time needed to restore network operation and ensure that the network operations staff has meaningful contextual information available.

Within TNM, the topology-based event correlation engine uses the model of the discovered network to understand the relationships between network events based upon the connectivity and containment (various groupings) of network devices. This enables Tivoli Network Manager IP Edition to quickly and accurately identify root cause events (to the node and port level) and their associated symptoms, thereby reducing the time needed to restore the network and ensuring that customer-facing network operations staff has meaningful contextual information at their fingertips.

Integration with Netcool/OMNibus allows the TNM topology-based event correlation engine to process events obtained from both network devices and other management systems using a broad range of available integrations.

## **1.5.7 Security**

Client authorization is defined within the ObjectServer using the administrator client or command-line utility. User authentication may be defined in the ObjectServer where passwords are encrypted using DES encryption by

concatenation of the password and salt data, or by reference to external authentication systems via PAM integration on UNIX platforms. DES is the default encryption method. Is it also possible to enable AES encryption.

**Note:** DES encryption is the default. AES can be enabled by running the ObjectServer with the property PasswordEncryption set to “AES”.

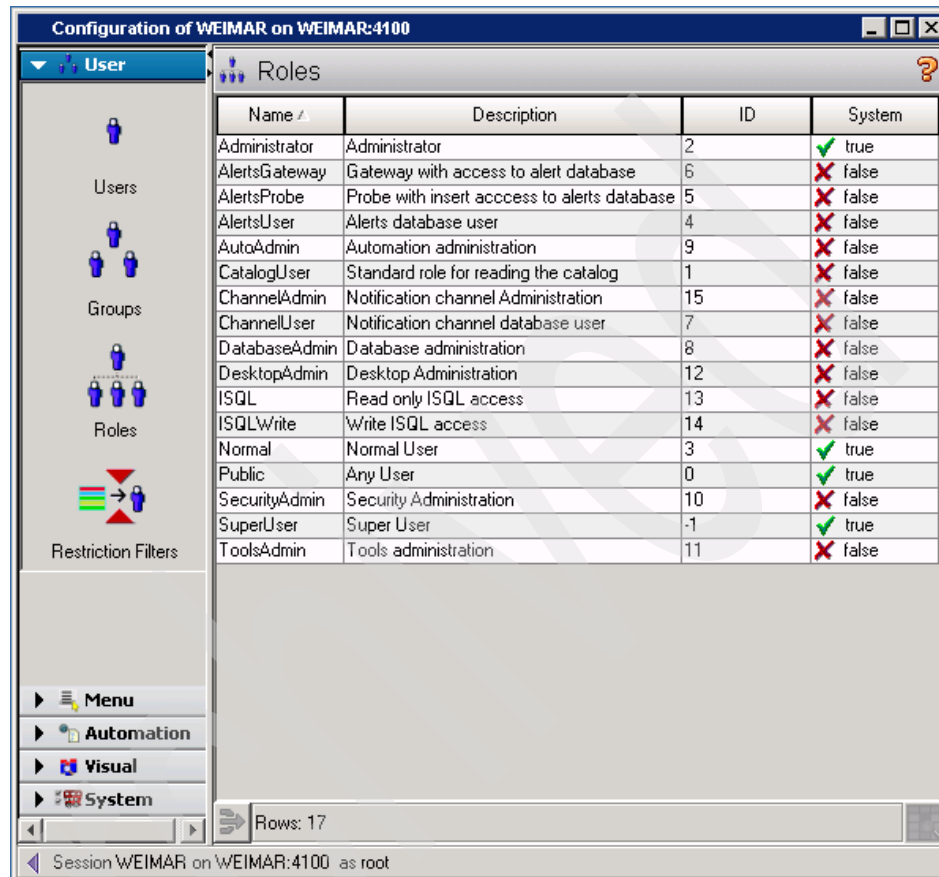
Communication paths between the Netcool/OMNIBus components may optionally be secured by Secure Sockets Layer (SSL) functionality included within each component. SSL/TLS is used for server authentication, data encryption, and data integrity.

Encryption is provided by FIPS 140-2 approved cryptographic providers; IBMJCEFIPS (certificate 376) or IBMJSSEFIPS (certificate 409) and IBM Crypto for C (ICC (certificate 384) for cryptography. The certificates are listed on the NIST Web site at:

<http://csrc.nist.gov/cryptval/140-1/1401val2004.htm>

Netcool/Webtop also maintains a separate list of user definitions, but this information can be automatically migrated from an OMNIBus ObjectServer.

Figure 1-13 shows a view of the default roles available through the administrator console.



Name	Description	ID	System
Administrator	Administrator	2	✓ true
AlertsGateway	Gateway with access to alert database	6	✗ false
AlertsProbe	Probe with insert access to alerts database	5	✗ false
AlertsUser	Alerts database user	4	✗ false
AutoAdmin	Automation administration	9	✗ false
CatalogUser	Standard role for reading the catalog	1	✗ false
ChannelAdmin	Notification channel Administration	15	✗ false
ChannelUser	Notification channel database user	7	✗ false
DatabaseAdmin	Database administration	8	✗ false
DesktopAdmin	Desktop Administration	12	✗ false
ISQL	Read only ISQL access	13	✗ false
ISQLWrite	Write ISQL access	14	✗ false
Normal	Normal User	3	✓ true
Public	Any User	0	✓ true
SecurityAdmin	Security Administration	10	✗ false
SuperUser	Super User	-1	✓ true
ToolsAdmin	Tools administration	11	✗ false

Figure 1-13 User roles management through the administrator console

Moreover, through the Netcool/Webtop console, operators can create customized pages for a single user or groups of users, therefore making it possible to hide or show information on demand, and to create effective operational, executive, and business views.

OMNIbus can support a large number of concurrent users without impacting service performance.

## 1.5.8 IBM product strategy

The key point of IBM product strategy regarding Netcool/OMNIbus and Tivoli Enterprise Console is that IBM supports customers over the coming years to upgrade at an appropriate time.

The general guidelines for the product strategy are as follows:

- ▶ Tivoli Monitoring (TEMA/TEMS/TEPS) is the strategic monitoring platform for Tivoli's product portfolio.
- ▶ Tivoli Enterprise Portal (TEP) and its evolution (Tivoli Integrated Portal, TIP) is the common and strategic console for operations and policy management for Tivoli's product portfolio. The active event list from Webtop will ultimately replace the native desktops.
- ▶ OMNIbus is the base for the convergence of the event infrastructure for Tivoli's product portfolio.

The directions IBM takes into account for this event management unification are:

- ▶ Integrate the positive qualities of both systems into a configurable end-to-end event management solution preserving customers investments in deployed assets.
- ▶ Step-by-step rollout to both product sets that are converging capabilities on normal release cycles to increase value to customers *as we go*.
- ▶ The expectation is that all customers will eventually upgrade to the unified event management platform:
  - Co-existence and interoperation between TEC and OMNIbus will be provided while unification is delivered.
  - Customers can upgrade on their time line.
  - IBM expects the upgrades to occur over a number of years.
  - Tivoli is putting in place mechanisms and capabilities to enable customers to control this upgrade and execute in smooth, nondisruptive steps.
  - No customer will be left behind.



# Architecture

In this chapter we describe first Tivoli Enterprise Console and then Netcool/OMNibus architecture components and how they are distributed in a typical architecture. As we go on with the description, the functionality of the components is compared so that TEC and OMNibus users can consider the differences between them.

## 2.1 Tivoli Enterprise Console architecture

Tivoli Enterprise Console is flexible and scalable to suit different characteristics of a customer's environment. Some of the characteristics to consider are:

- ▶ How big is the environment to be managed, and how are the resources geographically distributed?
- ▶ How many events will be generated and how often?
- ▶ What type of operating system or application will generate events?
- ▶ How many administrators and operators will handle the product?

### 2.1.1 Typical installation

The typical installation scenario is shown in Figure 2-1.

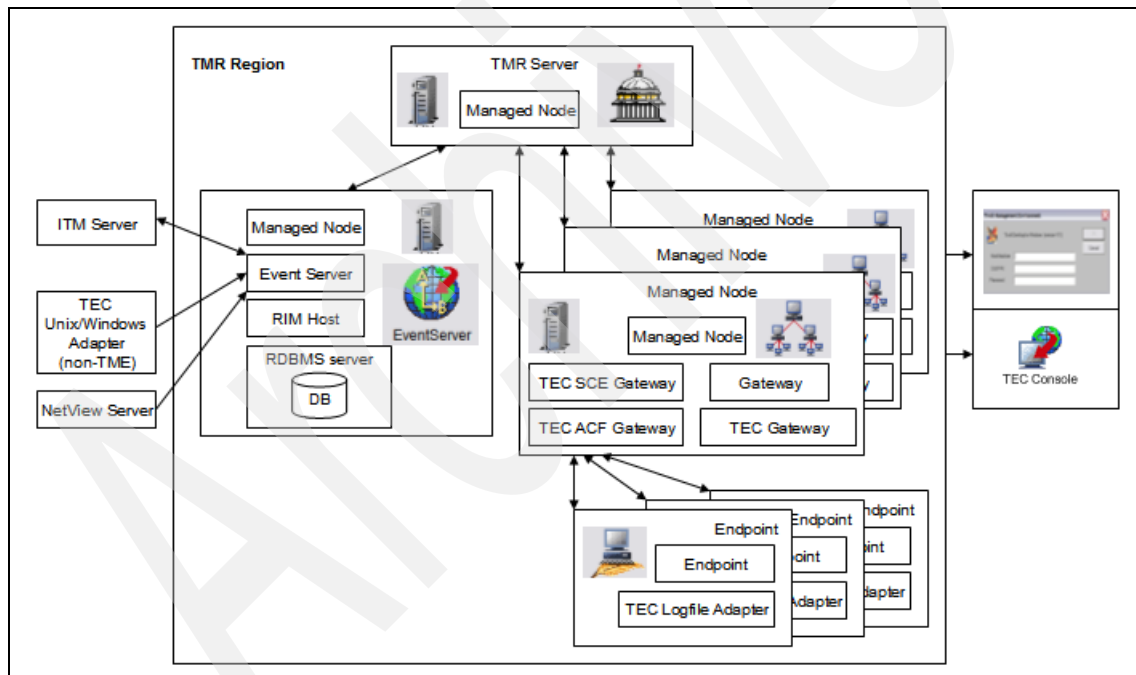


Figure 2-1 Typical Tivoli installation

## Components of the Tivoli Framework architecture

The components of the Tivoli Framework architecture are:

- ▶ TMR region

A Tivoli environment consists of one or more Tivoli regions. Each Tivoli region consists of a TMR server, one or more managed nodes, and multiple endpoints. The Tivoli regions can be interconnected. TMR stands for Tivoli Management Region. A Tivoli region is an entity that contains the Tivoli server and its clients. A Tivoli region contains three tiers of resources: the Tivoli server, managed nodes and gateways, and endpoints.

- ▶ TMR server

This is the main server in one Tivoli region. It can delegate and distribute tasks to managed nodes.

- ▶ Managed node

This is subordinate to a TMR server and receives the binaries to be able to execute almost all of the commands by itself. The TMR server is also a managed node.

- ▶ Gateway

Some managed nodes can be configured as a gateway. One gateway can typically take care of hundreds of endpoints. It receives the binaries to be able to download the code to endpoints. It is often used when customers have distributed sites and one gateway can be placed in each site, for example.

- ▶ Endpoint

This is the agent code that runs at a target machine. Several different flavors of operating system are supported. This agent code is the base for several different Tivoli framework applications such as Enterprise Console adapters, remote control, inventory, and software distribution.

- ▶ RIM host

One managed node in a region is configured to be an RIM host. The RDBMS's client code must be installed at this managed node. The function of this component is be able to connect to a relational database.

- ▶ RDBMS server

Some Tivoli products utilize a database, for example, inventory, TEC, and IBM Tivoli Configuration Manager (ITCM), so the customer has to provide a RDBMS server to implement those products.

- ▶ Event server

This is the main component of the Tivoli Enterprise Console. It has to be installed at a managed node.

- ▶ **TEC console**  
This is the component of the Tivoli Enterprise Console for the visualization of events.
- ▶ **TEC gateway**  
This is the Tivoli Enterprise Console component that provides the function of a gateway between endpoints and the event server. Its code runs on a managed node.
- ▶ **TEC SCE gateway**  
The State Correlation Engine (SCE) is used to provide high-speed event filtering and event collection. It is normally located at a TEC gateway, but can be deployed on an endpoint.
- ▶ **TEC ACF gateway**  
This is the Tivoli Enterprise Console component that has the binaries to distribute Adapter Configuration Facility (ACF) profiles to endpoints.
- ▶ **TEC adapter**  
This is the code that collects the source information such as messages from logfiles (application and system) on targets. There are many available adapters that Tivoli Enterprise Console offers.
- ▶ **TEC adapter (non-TME)**  
It is an adapter that does not require the Tivoli Framework infrastructure to send events. A TCP/IP socket connection will provide the connection from the adapter direct to the TEC server.
- ▶ **Tivoli desktop**  
This is the graphical interface from where Tivoli administrators can manage the resources.

### **Optional source components**

NetView Server is the product that manages network resources. It has its own user and administration interface. It has a strong integration with Tivoli Enterprise Console because, although network operations keep the NetView interface to monitor and control the network events, TEC is the centralized view where events detected by NetView can be correlated with other service events.

IBM Tivoli Monitoring is the product that monitors at a fine level of detail the health of operating systems, databases, and applications. It presents the information in a Web portal interface, called the Tivoli Enterprise Portal (TEP). Again, as operations need a centralized interface to manage events coming from operational systems and applications, the TEC console is typically the central

view to manage these vents. Optionally, the TEC console can be integrated into the ITM TEP view.

## 2.1.2 Describing TEC components

In this section we describe some of the TEC components that have equivalent functions on Netcool/OMNIBus.

### Framework authentication

All Tivoli users have an administrator object associated to them. Each Tivoli administrator is associated to a login and group of the operating system of a managed node. When users accesses the Tivoli desktop, authentication of the user is required. The operating system controls the password changes.

So, for a TEC operator configuration, an association with a Tivoli administrator is required. Then, to access the TEC console, the user has to authenticate with the same login used to access the Tivoli desktop.

### Framework authorization

In a Tivoli environment, to complete an administrator configuration:

1. Define the name of the administrator.
2. Define the login name and group name with which this administrator will be authenticated into the operational system.
3. Define the TMR roles. These are the global TMR roles that the administrator will have in the environment. It will allow him to execute actions in the desktop interface or by command lines.
4. Define the TMR resource roles. These roles are defined if the administrator has to have special roles over specific Tivoli resources (for example, one policy region where a senior authority is required).
5. Define the notice groups. The administrator could have access to one or more groups of notices. Each notices group receives messages about administration actions that have been taken to this group.

On the TEC configuration, some steps should be executed to have the TEC operators configured:

1. TEC operators have to be associated with only one Tivoli administrator.
2. One TEC operator can be associated with just one TEC console.
3. One TEC console can manage one or more TEC groups. For each TEC group, the TEC administrator has to define which roles the TEC operators will

have in the visualization of that event group. The roles are super, senior, admin, and user.

### **Framework (RIM Object)**

The event server stores the events in a relational database. The product interacts with the database server using an internal ODBC connection. This internal ODBC connection receives a special name in the Tivoli environment: RIM object.

The RIM object is configured on one managed node that has the database client or server code.

The main parameters to configure a RIM object are:

- ▶ Database server host name
- ▶ Relational database vendor
- ▶ Database client or server installation path
- ▶ Instance name
- ▶ User and password to access the database

### **Framework (TEC gateways)**

Managed nodes and gateways were created to provide scalability in the Tivoli environment.

The Tivoli architecture is a three-tier hierarchy. The TMR server is on the top level. Managed nodes (with the gateway function) are on the second tier controlled by the TMR server. Endpoints, on the third level, are controlled by the gateways.

Gateways can control hundreds of endpoints. The recommended limitation is 1,500–2,000. If a gateway is unavailable, the other gateway can manage the endpoints that were under control of the unavailable gateway.

The TMR server is the main controller, and policies can guarantee the assignment of endpoints between the gateways.

There are two reasons for configuring the TEC gateway:

- ▶ Creating a gateway receiver component to receive events from non-TME adapters.
- ▶ Adjusting the interval the gateway sends events to the TEC. This time interval can be made dependent on a time interval or on the event cache size.

## TEC event server

The TEC event server is the core component of the TEC environment. The event server architecture consists of the following five processes. The process names are in parentheses.

- ▶ Master process (tec\_server)
- ▶ Reception engine process (tec\_reception)
- ▶ Rule engine process (tec\_rule)
- ▶ Dispatch engine process (tec\_dispatch)
- ▶ Task engine process (tec\_task)

In addition, there is the tec\_ui\_server process, which is discussed later in detail.

Figure 2-2 shows the event flow in TEC.

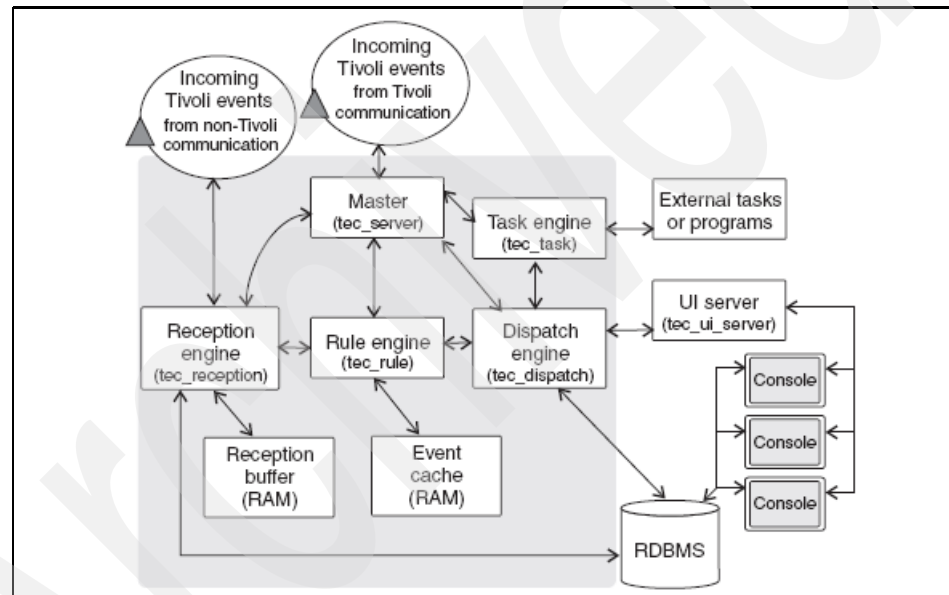


Figure 2-2 TEC event flow

The five processes are:

- ▶ Master process

The master process coordinates all of the other event server processes. The master process also initially receives incoming events sent from Tivoli communication and forwards the events to the reception engine.

- ▶ Reception engine process

The reception engine process receives incoming events and logs them to the reception log. The reception engine pulls the events from the reception buffer to the rule engine for processing.

- ▶ Rule engine process

The rule engine is a rule-based event processor. The event is evaluated against the rules. After rule processing, the event is placed into the event cache of the rule engine. The event in the rule engine currently being evaluated is referred to as the event under analysis. An event that satisfies the specification criteria of a rule causes the rule to run, which means that the actions defined by the rule are performed. This is the first input stream into the rule engine.

- ▶ Dispatch engine process

The dispatch engine communicates with the rule engine and the task engine to know when to update event information. If tasks or programs need to be run for an event, the dispatch engine contacts the task engine for running them. The dispatch engine also manages requests for event changes coming from an event console (through the UI server) and sends them to the rule engine. When the event server is started, the dispatch engine retrieves events from the event database to reload the event cache for the rule engine.

- ▶ Task engine process

The task engine runs programs, tasks, scripts, and commands initiated by rules. The task engine process monitors these running items and can return their exit status to the dispatch engine, which writes the status to the event database. The task engine runs these items as it receives requests to do so. It does not wait for a running item to complete before starting another one.

## **User interface server and WebSphere Console Server**

In this section we discuss the user interface server and the WebSphere® Console Server.

### ***User interface server***

The user interface (UI) server provides communication services between the event consoles and the event server. The UI server communicates with the dispatch engine when it needs to contact the event server. The UI server



provides transaction locking for event console status updates and prevents multiple event consoles from responding to the same event. The UI server also automatically updates the status of events on all event consoles by forwarding the event changes from the event consoles to the dispatch engine, which sends the changes to the event database. For example, when an operator acknowledges an event, the UI server automatically updates the status of the event on each event console that contains the event.

The information about event changes is pulled from the TEC console through its JIRM interface.

### ***Web-based event console***

You can use the Web-based event console to manage events from your Web browser. The Web-based event console includes the following features:

- ▶ Tivoli secure login for added security.
- ▶ Event console definitions, event group definitions, and preferences that an administrator defines using the Java version of the event console take effect for the Web version, so that both the Web version and the Java version of the event console can be administered from a single place.
- ▶ Ability to view additional information about an event in a Web page, which helps an operator determine the actions to perform and whom to contact.
- ▶ Multiple event details with additional contextual information, viewed in a single Web page, improving operator understanding.
- ▶ Multiple operators can be assigned to a single event console definition.
- ▶ Changes to event console definitions are automatically reflected in the event console in the next login session.
- ▶ The event console refresh rate can be changed to update events at an interval that meets your needs.
- ▶ Event viewer data, event summary data, and task information are cached, reducing the load on the event server database and the Tivoli Management Framework. You can configure the timeout intervals for these caches, which helps you to balance your needs for performance and current information.
- ▶ A single installation of the Web version of the event console can be configured to support the installation of the Tivoli Enterprise Console product in multiple Tivoli regions.
- ▶ Ability to run predefined tasks.
- ▶ When there are conflicts between event consoles, such as two operators simultaneously attempting to change the status of an event, they are resolved automatically and operators are notified.

The Web version of the event console organizes the tasks that you can perform in a portfolio, which is titled *my work*. The portfolio contains the following tasks:

- ▶ **Select an Event Group:** lists the event groups that have been assigned to the event console. To manage the events in an event group, select the appropriate event group from the list. An event viewer is displayed, which contains the events in the selected event group.
- ▶ **View Summary of Events:** shows a high-level overview of the health of resources represented by an event group, indicating the number of events for each event severity in each event group and the total number of events for each event severity. You can also display the percentage of events for each event severity. To manage the events in an event group, select the name of the event group.
- ▶ **An event viewer is displayed,** which contains the events in the selected event group.
- ▶ **Run Tasks:** runs predefined tasks from task libraries.
- ▶ **Change User Preferences:** Controls the display of events in the event viewer. For example, you can change the automatic refresh rate and the maximum number of events to display in the event viewer.

### **Java event console**

The Java version of the event console can be installed on a managed node, endpoint, or host in a non-Tivoli environment. The Java version of the event console includes the following features:

- ▶ **Tivoli secure logon** for added security.
- ▶ **Each event console** retrieves event information directly from the database for high performance and scalability.
- ▶ **The event console refresh rate** is configured to allow different event consoles to be given different priorities for event updates.
- ▶ **The ability to run local actions and commands** enables third-party or custom scripts and applications to be run easily from the event console.
- ▶ **The ability to run predefined tasks.** For more information about running tasks from a task library, see the online help for running tasks in the event console.
- ▶ **Automated tasks** can be configured in advance and run when a particular event is received by the event console.
- ▶ **The ability to view more information about an event in a Web page,** which helps an operator to determine the actions to perform and whom to contact. When the sample event information is installed, samples and hooks are provided to help you enable this feature. Customers must provide the additional information to customize their environment.

- ▶ Multiple event details, viewed in a single window with additional contextual information, improving operator understanding.
- ▶ All attributes, including custom attributes, can be displayed and filtered for event groups with SQL operators. For more information about attributes, see the event group entries provided in the online help for the event console.
- ▶ Multiple operators can be assigned to a single event console definition. For more information about defining event consoles and operators, see the online help for the event console.
- ▶ When editing event console definitions, all operators assigned to that event console automatically pick up the changes, thus eliminating the need for scripts to be used to update or create event consoles.
- ▶ When there are conflicts between event consoles, such as two operators simultaneously attempting to change the status of an event, they are resolved automatically and operators are notified.
- ▶ Import and export functions enable the event console and event group definitions to be exported to a file for archiving or for easily migrating from the development environment into production.

The Java version of the event console provides the following views:

- ▶ Configuration view: The Configuration view is used to configure both the Java version and the Web version of the event console. Only administrators have access to this view.
- ▶ Summary Chart view: The Summary Chart view is used to show a high-level overview of the health of resources represented by an event group. Each event group is represented by a single bar, similar to a bar graph, on an operator's event console. To open an event group, click the bar representing that group.
- ▶ Priority view: In the Priority view, event groups are represented by buttons. The buttons representing each event group display the event group name, and the color of the button representing each event group corresponds to the color defined for the highest severity event contained in that group. To open an event group, click the button representing that group.

## External event database

The Tivoli Enterprise Console product uses an external relational database management system (RDBMS) to store the large amount of event data that is received. The RDBMS Interface Module (RIM) component of the Tivoli Management Framework is used to access the event database.

The supported database servers are DB2®, Oracle®, Informix®, MS SQL, and Sybase.

## Command-line framework commands and tasks

Behind all configuration that Tivoli administrators execute by graphical interface, there is at least one command line being executed. So, for automation tasks, command lines used in scripts are very useful and optimize the time for maintenance of Tivoli resources.

It is often convenient or more appropriate to invoke a Tivoli management application operation from the command line than from the graphical user interface for remote access, multiple operations, and scripts. Each command line has one or more authorization roles.

## Framework and TEC tasks

Recognizing the need to put these commands and scripts behind an icon, for operations purposes, there are also several task libraries that have been developed, including several specifically created for TEC operations. Tasks can also be *fired* by TEC rules.

## TEC command-line utilities

Some useful utilities are also provided to help administer a TEC environment, to test communications, or to embed in monitoring scripts, for example:

- ▶ **wtdumpr1** to dump out the contents of the reception log
- ▶ **wtdumper** to dump out the contents of the rule cache
- ▶ **wtdbc1ear** to delete all or a subset of events from the database
- ▶ **wpostemsg** or **wpostzmsg** to send an event to the event server via the Tivoli framework
- ▶ **postemsg** or **postzmsg** to send an event without using the Tivoli framework

## Event sources

The sources that are deployed by Tivoli Enterprise Console out of box are:

- ▶ **AS/400® alert/message adapter**: forwards events from an AS/400 system to the event server.
- ▶ **NetWare adapter**: forwards events from a NetWare server.
- ▶ **OpenView adapter**: forwards events from Hewlett-Packard OpenView to the event server.
- ▶ **OS/2®**: forwards events from an OS/2 system to the event server.
- ▶ **SNMP**: The Simple Network Management Protocol forwards events from SNMP traps to the event server. The SNMP adapter serves the function of collecting SNMP trap messages directly from the SNMP trap socket of a host and translating SNMP traps into appropriate TEC class instances.

- ▶ **UNIX logfile:** The TME® UNIX logfile adapter receives raw log file information from the UNIX syslogd daemon, formats it, and sends it to the TEC gateway. The TEC gateway then sends the information to the event server. The non-TME UNIX logfile adapter sends information directly to the event server.
- ▶ **Windows®:** The adapter for the Microsoft® Windows event log forwards events from a Windows system to the event server. It is registered with the startup configuration of a Windows system so that the adapter is started with all the other applications that are automatically started when the Windows system is started.

## Framework policy-based distribution and subscription

One of the key features of the Tivoli framework is the policy-based architecture, which allows the definition of profile managers, profiles, and subscribers to those profiles. In this way TEC adapter profiles are defined for different event sources and can be distributed in a single step to multiple subscribers or targets in groups of geographical location or department, for example. This distribution, which might be a new adapter, a configuration change, or a new code version, can then take place over the Tivoli framework to hundreds of target endpoints simultaneously.

## Rulesets

TEC provides a default rulebase with a number of rulesets in it. Not all rulesets are initially active, but users can enable them anytime. Usually, an administrator customizes his own rulesets.

Table 2-1 shows a list of the rulesets that are included in the default rulebase.

*Table 2-1 Rule sets included in the default rulebase*

Rule set	Description	Activated	Configuration required
cleanup.rls	Closes old open events	Yes	No
correlation.rls	Event correlation	No	Yes
db_cleanup.rls	Deletes old closed events	No	No
dependency.rls	Defines dependency relationships for e-business rules	Yes	No
ebusiness.rls	Causal analysis of events from e-business applications	Yes	Yes
escalate.rls	Automatic severity escalation	No	No
event_activity.rls	Generation of event activity reports	No	No

Rule set	Description	Activated	Configuration required
event_filtering.rls	Filtering of unwanted events	No	Yes
event_thresholds.rls	Severity escalation based on repeated events	No	Yes
forwarding.rls	Event forwarding	No	Yes
heartbeat.rls	Heartbeat monitoring	Yes	No
maintenance_mode.rls	Maintenance mode support	Yes	No
netview.rls	Clearing and synchronizing of network events	Yes	No
notify.rls	E-mail or pager notification	No	Yes
ov_default.rls	Processing of HP OpenView events	No	No
tecad_nv390fwd.rls	Forwarding of NetView for z/OS® events	No	Yes
tecad_nv390msg.rls	Processing of NetView for OS/390® Message Adapter events	No	No
tecad_snaevent.rls	Processing of SNA alert events	No	No
troubleticket.rls	Integration with trouble ticket systems	No	Yes

### 2.1.3 Complex scenarios

Tivoli Framework can be implemented in a large-scale environment. If the management environment is becoming complex, there are some different architectures that can be chosen to fit the customer needs.

#### Connecting multiple Tivoli regions

To meet the needs and demands of managing thousands of resources that are geographically dispersed across networks, Tivoli Management Framework enables you to logically partition your managed resources into a series of connected Tivoli regions. Each region has its own server for managing local clients and a set of distributed replicated services for performing management operations. Regions can be connected to coordinate activities across the network, enabling large-scale Systems Management and offering the ability for remote site management.

### ***Types of region connections***

With the ability to selectively connect regions, Tivoli administrators can control the scope and effect of local configuration changes. At the same time, Tivoli administrators can enable centralized management and propagation of new policy rules, configuration changes, and operations to all connected regions. Region connections are directed, meaning that they are not necessarily symmetric with respect to the two regions involved. Connections can be either one-way or two-way.

- ▶ **One-way region connections**

In a one-way connection, only one region has knowledge of the other, so information is passed from the managing system only. One-way connections are useful where a central site is responsible for administering several remote sites, but none of the remote sites need to manage resources at the central site or at other remote sites. Each remote site could also have its own local operator who might be responsible for managing day-to-day operations on local resources, while the connection from the central site is used for more global updates across the company, such as a new version of an application. Although one-way connections are feasible, we recommend two-way connections.

- ▶ **Two-way region connections**

Each Tivoli region involved in a two-way connection is aware of the existence of the other. Information exchanges about system resources occur in both directions. Two-way connections are useful in a variety of situations, such as a very large local area network that is logically partitioned. By using two-way connections, the management load is spread across multiple Tivoli servers. In addition, two-way connections are needed to access and manage resources in other regions.

### **Multiple region architectures**

When connecting multiple Tivoli regions, consider the following architectures:

- ▶ Hub and spoke
- ▶ Master-remote
- ▶ Fully interconnected

#### ***Hub and spoke connections***

The hub and spoke architecture improves performance by distributing server load. In this architecture, the Tivoli environment is segmented into several regions, each responsible for directly managing a different physical segment of the enterprise. This architecture supports a centralized management paradigm by having all management operations performed through the hub region. Remote (or spoke) regions are placed and configured to optimize network utilization and system performance and to distribute Tivoli server loads.

The hub region provides a centralized Tivoli server in its own dedicated region. The hub server directly manages a limited number of resources so that it can be dedicated primarily to the maintenance and administration of the Tivoli object database and Tivoli environment. It is also the focal point for hub-wide activities as required. For example, the hub Tivoli server is responsible for configuring and distributing monitoring profiles to any servers in the environment. The hub server is not responsible for directly managing resources. The spoke regions directly control the endpoints in the Tivoli environment. Spoke regions are used to group managed nodes by physical location in the network and to localize Tivoli functions to that physical location, improving network and system performance. Generally, spoke regions are not used as entry points for administrators. With a properly implemented logical design, virtually all Tivoli operations can be performed from the hub region without concern for where the object being manipulated exists in the spoke environment.

### ***Master-remote connections***

The master-remote architecture supports a distributed management structure by having the managed function operations being performed by administrators from their own regions. The different regions are connected to each other through a two-way connection, allowing the resources to be managed by any administrator in the managed environment.

### ***Fully interconnected connections***

The fully interconnected architecture has regions that are created for use within business units or for some subgroup within the enterprise. The regions are connected on an as-needed basis, which often leads to fully interconnected regions.



In Figure 2-3 you can see an example of a large-scale environment with hub-spoke architecture.

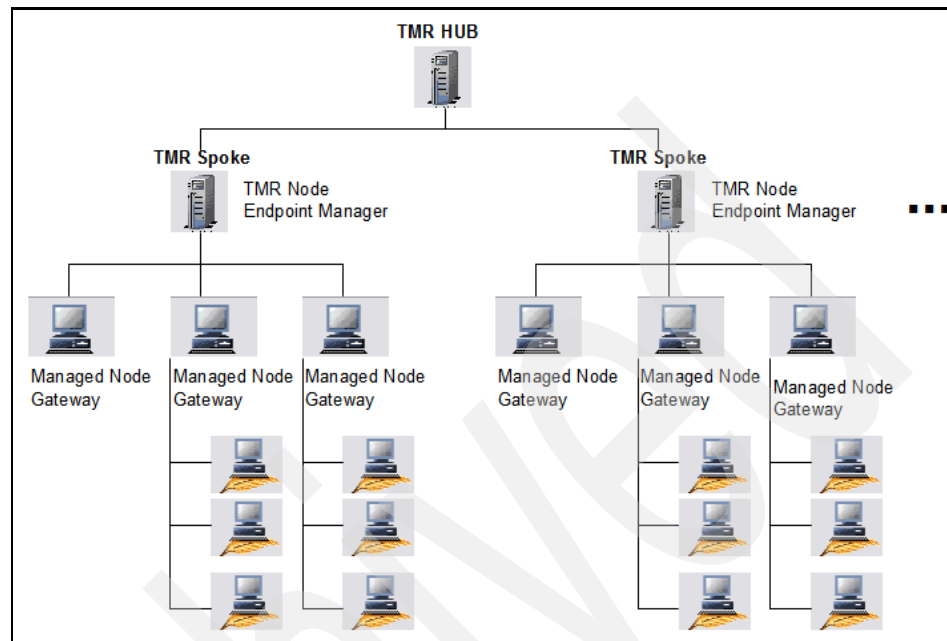


Figure 2-3 Hub-spoke architecture

### TEC servers in a multi-region environment

As noted earlier, there is only one TEC server per TMR region. However, with any of the multi-region configurations above, it is possible for a single TEC server to scale to support multiple TMR environments and, hence, many thousands of event sources.

It is equally possible for each TMR to have its own TEC server, and to have endpoints or gateways sending events to a local TEC server as a primary and a second as a failover. The most usual failover configuration is to use the TEC gateway in this role, but either is possible. If this failover is done, the first communication method is normally the framework and the second is direct socket connection to the remote TEC.

TEC servers can also be configured in a multi-tier architecture of two or even three tiers to suit geographical, operational, or business constraints. Typically, this is configured if a service provider or part of the organization is concentrating events from several customers or other parts of the organization into a central place. This is commonly referred to as a manager of managers role. A multi-tier

configuration can also be implemented for scalability if the forwarding of events from one layer to the next is selective.

## 2.1.4 TEC integration

In this section we describe some integration scenarios between TEC and IBM Tivoli Monitoring.

### IBM Tivoli Monitoring

A simple ITM scenario is shown in Figure 2-4 on page 55, with:

- ▶ TEMS hub: the ITM component responsible for connecting to the agents and integrating with TEPS to publish the collected monitoring data and user access authentication.
- ▶ TEPS: the ITM component responsible for saving configuration information and publishing the monitoring data through the desktop or Web browser interface.
- ▶ TEP client: the ITM user interface, which could be deployed as a desktop console or in Web console mode.
- ▶ ITM agents: represents the targets machines that receive the monitoring agent code and run the code. It connects to the TEMS hub.

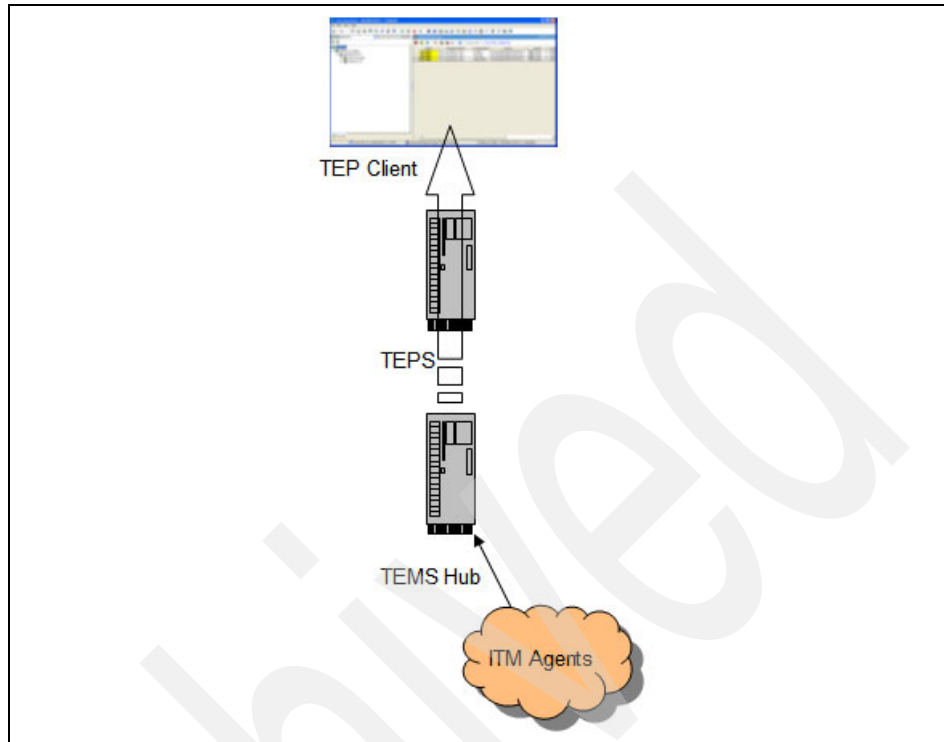


Figure 2-4 ITM basic architecture

### ***Benefits of integration***

Providing the integration, some benefits for the management environment are achieved:

- ▶ TEC is a focal point for enterprise-wide events including ITM.
- ▶ Event correlation capabilities in TEC are not replaced by ITM.
- ▶ Integration facilities are available with ITM code.
- ▶ TEP allows integration with the TEC console.

### TEC - ITM integration architecture

Figure 2-5 shows the TEC-ITM integration architecture. To provide the integration, some components must be known:

- ▶ **Event Forwarder:** This is installed on the TEMS as the default by the base ITM product installer.
- ▶ **TEC event synchronization:** This is installed separately, either by UI or silently. It provides processes, scripts, files, and rules to be installed on the same box of an existing TEC server.
- ▶ **TEC Event Viewer:** This is a feature installed as an option by the base ITM product installer. It is also known as *TEC GUI Integration*.

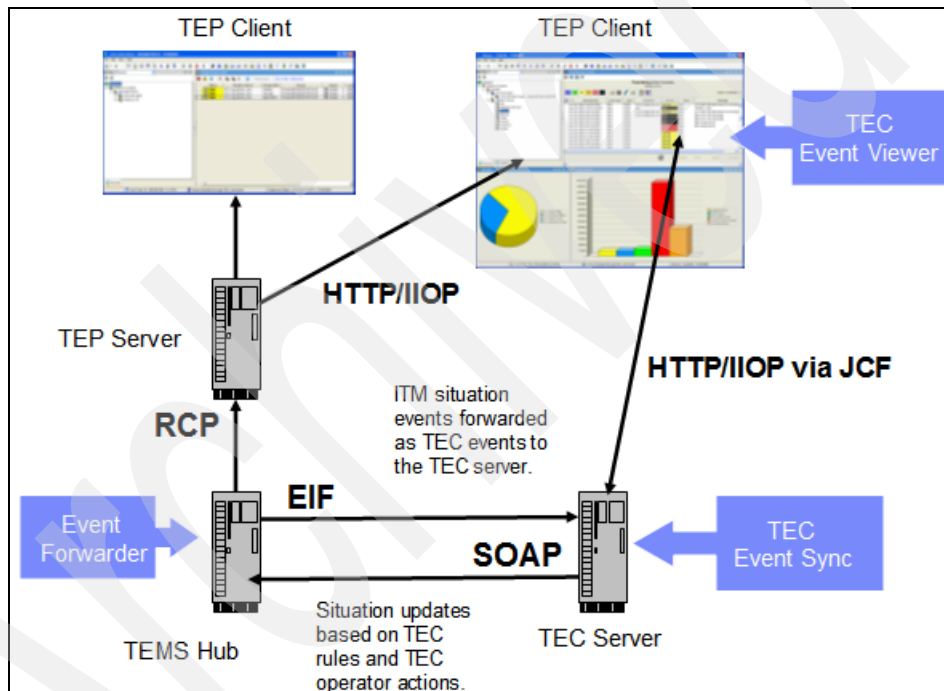


Figure 2-5 TEC-ITM integration components

### ***TEC-ITM integration event flow***

Figure 2-6 shows the sequence of information flow:

1. Agents trigger situation. TEMA code that runs on the target machine is constantly being checked to fire alerts to TEMS.
2. TEMS generates events. The Event Integration Facility function is customized to generate the events following the situations that were configured.
3. Events are received and processed by TEC. The events are processed by rulesets that correlate with other events coming from other sources if required.
4. Status of events sent back by SUF. Situation Update Forwarder is the Java code that is responsible to return information to TEMS.

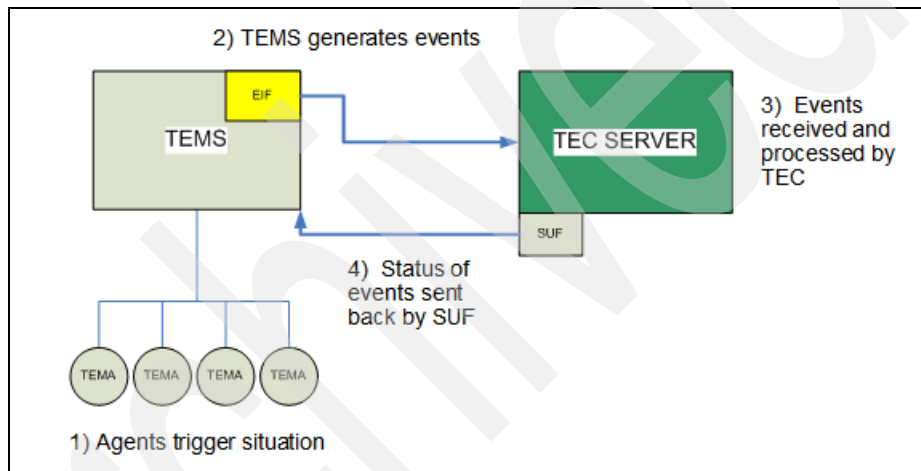


Figure 2-6 TEC-ITM event flow

## Deploying scenarios

There are different scenarios that the TEC-ITM integration can fit. Depending on your management scenario, one of the following configurations could be chosen:

- Multiple HUB TEMS to HUB/Spoke TEC

If your management environment exists on a HUB TEMS and some remote TEMSs, the HUB TEMS can be configured to integrate to a Tivoli architecture of HUB and spoke TEC. Figure 2-7 shows this scenario.

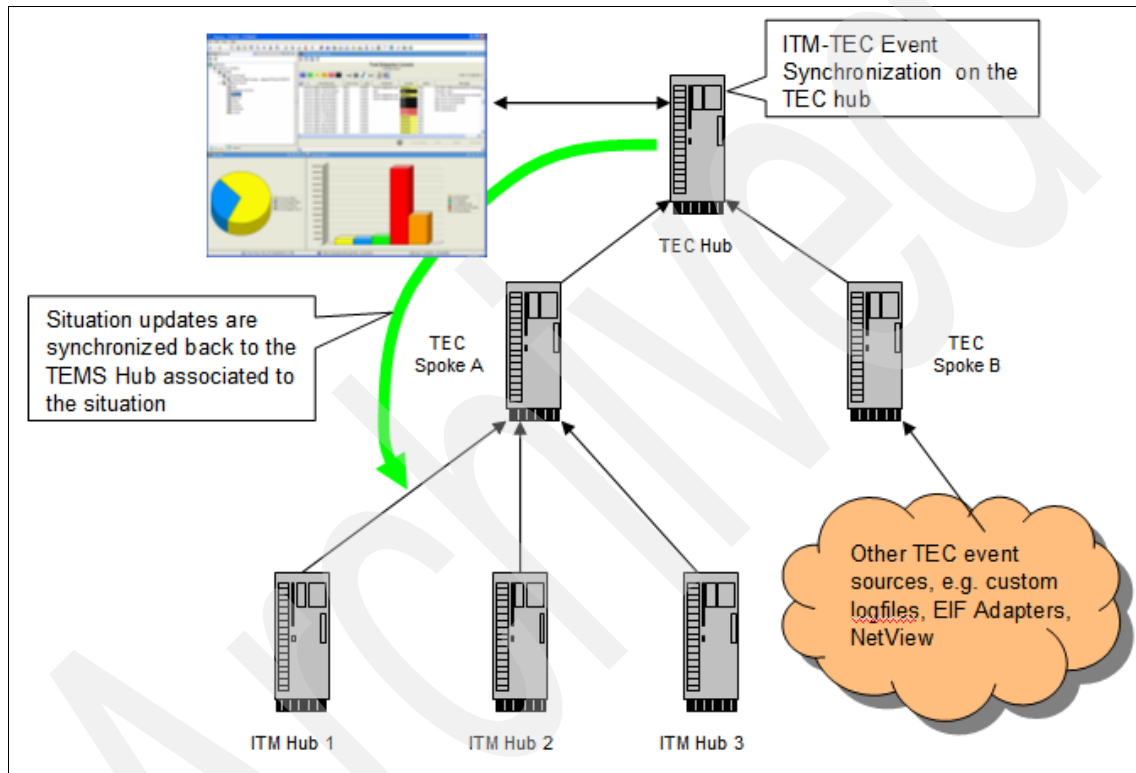


Figure 2-7 Multiples HUB TEMS integration to HUB-Spoke TEC

Here, the customer has the TEMS HUB infrastructure (ITM Hub1, ITM Hub2, ITM Hub3), so these ITM Hubs can integrate into a TEC Spoke.

This TMR Region is structured as HUB/Spoke—one TEC Hub receiving events from other two TEC Spokes (TEC Spoke A and TEC Spoke B).

As ITM events go to TEC Spoke A, the forwarding rules make the TEC Hub receive the events, too. So, the bi-directional function should be deployed at the TEC Hub, so the updates made from TEC Hub operators can be replicated to the TEMS Hub.

Other TEC sources continue to integrate to the TEC Spoke with no interference.

► Multiple HUB TEMS to a single TEC

If your management environment exists on a HUB TEMS and some Remote TEMSs, the HUB TEMS can be configured to integrate to a Tivoli architecture of a single TEC. Figure 2-8 shows this scenario.

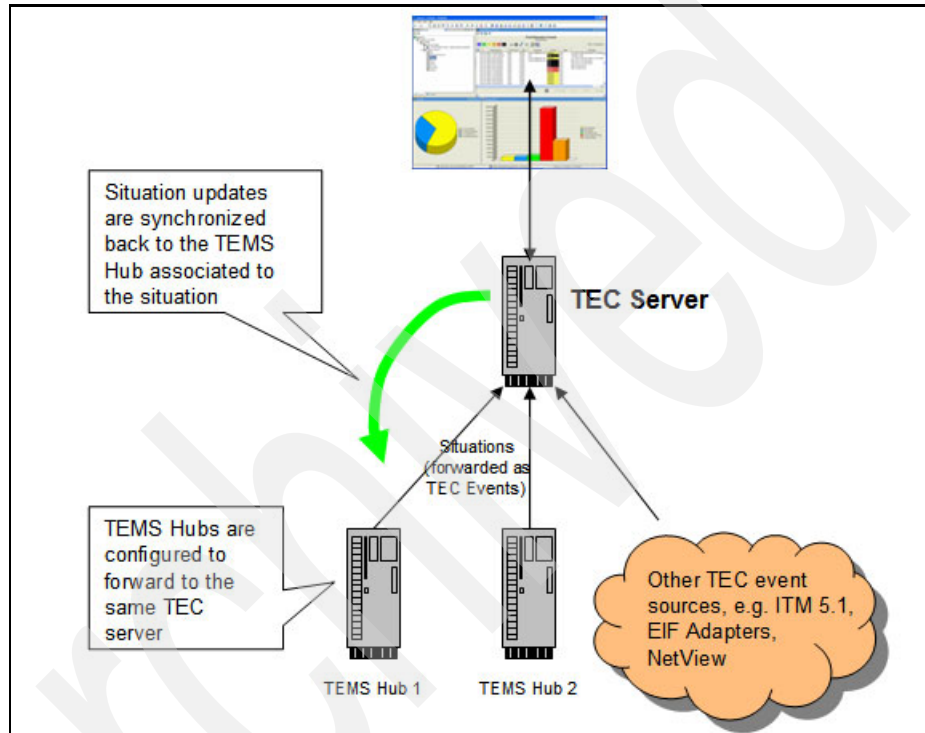


Figure 2-8 Multiple HUB TEMS integrating to TEC

Here, the customer has the TEMS HUB infrastructure (TEMS Hub1 and TEMS Hub2), so these TEMS Hubs can integrate to a TEC server.

As ITM events go to the TEC server, the bi-directional function should be deployed at the TEC server, and then the updates made from TEC server operators can be replicated to the TEMS Hub.

Other TEC sources continue to integrate to the TEC server with no interference.

- Single HUB TEMS to multiple TEC servers (from ITM 6.2)

If your management environment consists of a single HUB TEMS, this can be configured to integrate to a Tivoli architecture of multiple TEC servers. Figure 2-9 shows this scenario.

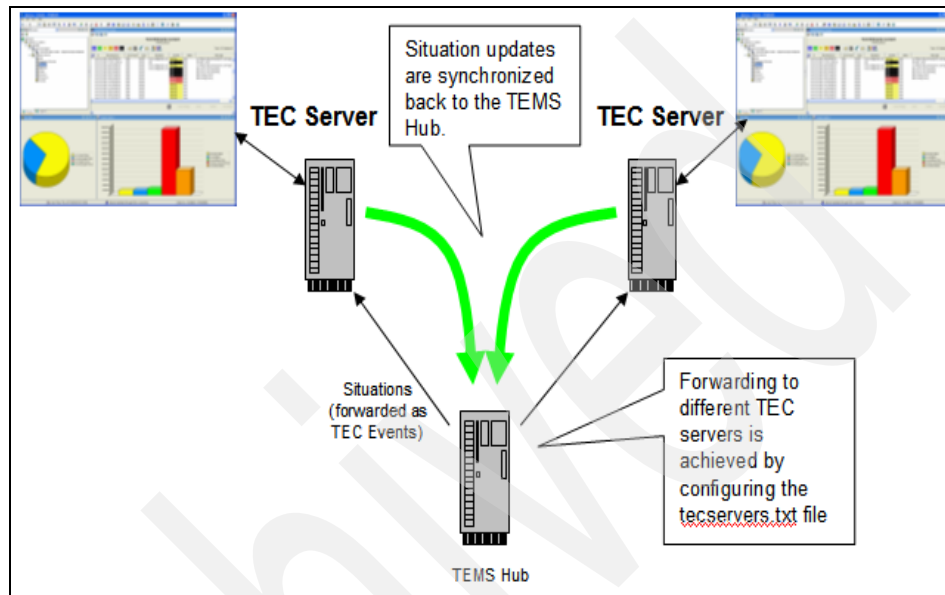


Figure 2-9 HUB TEMS integrating to more than one TEC server

Here, the customer has one TEMS HUB.

From ITM 6.2, a TEMS HUB can be configured to send to more than one EIF receiver, so ITM events can be sent to more than one TEC server.

The bi-directional function should be deployed at all TEC servers. Then the updates made from TEC server's operators can be replicated to TEMS Hub.

## 2.2 IBM Tivoli Netcool/OMNibus architecture

This section gives a detailed overview of IBM Tivoli Netcool/OMNibus architecture and components.

### 2.2.1 Architecture introduction

IBM Tivoli Netcool architecture is based on a layered structure of components, functions, and integrations.



### **Collection layer**

The collection layer with the function of Netcool probes and monitors provides:

- ▶ Lightweight, extensible event and data capture
- ▶ Dynamic real-time data access

### **Consolidate layer**

The consolidate layer with the core components and functionality of Netcool ObjectServer, gateways, and integration of third-party solutions provides:

- ▶ Ultra-scalable event architecture
- ▶ Integrates into performance management

### **Analyze and automate layer**

These ObjectServer functions may involve operator interaction at the native desktop or the active event list (AEL). The analyze and automate layer within the Netcool ObjectServer native desktop provides:

- ▶ Netcool distributed correlation
- ▶ Real-time business metric reporting
- ▶ Netcool/OMNibus automations
- ▶ Integration to Netcool comprehensive discovery components

### **Inform layer**

The inform layer with the Netcool components of the Netcool ObjectServer native active event list and Netcool/Webtop Web-GUI provides:

- ▶ Real-time service dependency visibility
- ▶ Dynamic network views

## 2.2.2 Architecture overview

Figure 2-10 provides an overview of the Netcool/OMNIBus 7.2 architecture.

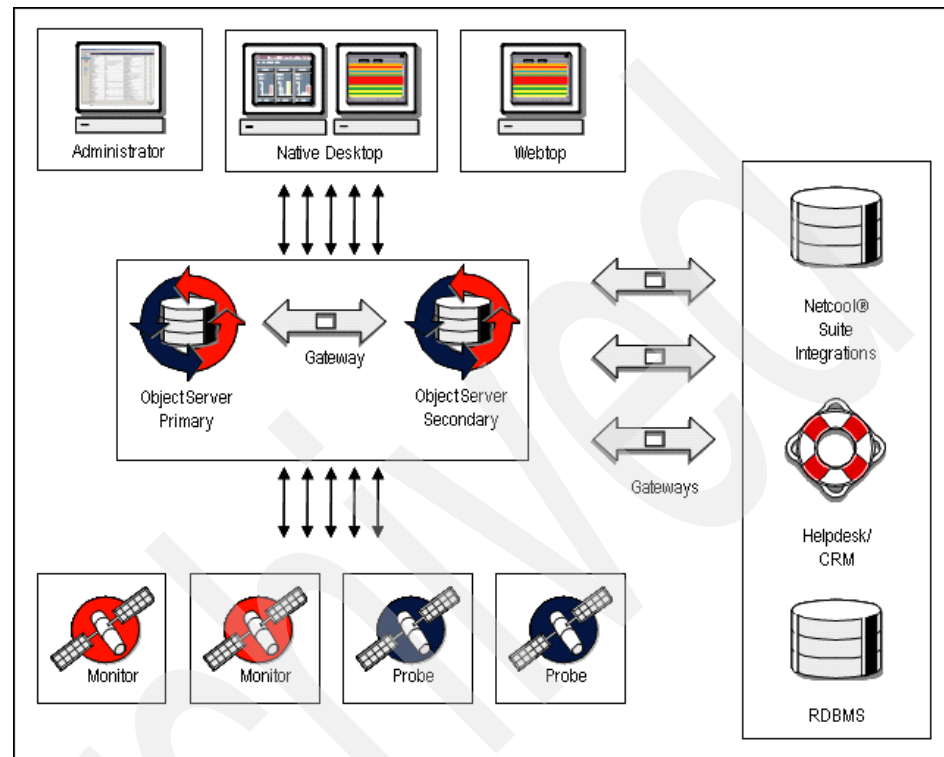


Figure 2-10 Netcool/OMNIBus core architecture

## 2.2.3 Component description

In this section we provide component descriptions.

### ObjectServer



The Netcool/OMNIBus ObjectServer is a Structured Query Language (SQL) database residing completely in memory. It receives events from a variety of monitoring sources or external programs such as probes, monitors, and

gateways. For notifying clients of changes to data within the ObjectServer, insert, delete, update and command (IDUC) protocol is used. Event information is stored and managed in database tables, and displayed in the event list.

Users, groups, and roles are defined within the ObjectServer database. Authentication and access are maintained and controlled within the ObjectServer database. Netcool/OMNIBus V7.2 has enhanced the existing core functions with:

- ▶ Accelerated event notification
- ▶ ObjectServer health and performance agent
- ▶ Out of the box failover configuration for automations
- ▶ Operating in a pure IPv6 or dual-stack environment
- ▶ Class hierarchy and extended attribute support for Tivoli Event Console transition
- ▶ Globalization to support language packs
- ▶ Infrastructure enhancements
- ▶ Enhanced upgrades for earlier OMNIBus versions
- ▶ Enhanced integrations (for example, extended attributes and ObjectServer fields for TEC integration)

Netcool/OMNIBus is multi-threaded and manages threads within the application. Netcool/OMNIBus Version 7 introduced native multi-threading, which provides reduced database lock times. This functionality supports read concurrency within a multi-processor system. This is enabled when the read load within a cycle exceeds the capacity of a single processor. Either a single write operation or multiple read operations can be processed. Read/write concurrency is not possible.

The ObjectServer performs the equivalent 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. TEC connects to an independent and external database as an external component, compared to the ObjectServer in-memory database, which provides much higher performance and availability of event storage.

The event list for the visualization of events is part of the dedicated native desktop console. It consists of an integrated suite of graphical tools used to view and manage events, and to configure how event information is presented. The desktop console, together with the included SQL interactive command-line interface, provides administrative access to the ObjectServer.

**Note:** Netcool/OMNIbus is based on Sybase Open Server technology. Netcool/OMNIbus V7.1 uses Sybase OpenServer Version 12.5.1 and Netcool/OMNIbus V7.2 uses Sybase OpenServer Version 15.

**Note:** Netcool/OMNIbus versions prior to V7.2 also included Flex License Server, which was related to Netcool license management of all components. In all versions covered in this book, the Flex License Server is no longer needed and is therefore not considered.

## Tables

The ObjectServer database contains the following tables: alerts tables, service tables, system catalog tables, statistics tables, client tool support tables, desktop tools tables, desktop ObjectServer tables, and security tables.

### ***alerts.status***

The alerts.status table contains the event information. Each event has 56 standard attributes (the columns) by default in the alerts.status table. Administrators can add additional fields up to a total of 512. Important fields are:

- ▶ Identifier: primary key for the alerts.status database (controls duplication of events).
- ▶ Node: name or IP address of the event originating device.
- ▶ Summary: description of the problem in human-readable form.
- ▶ Severity: specifies event priority level (0–5) and determines background color in the GUI visualization.
- ▶ Acknowledged: indicates whether the alert has been acknowledged.
- ▶ OwnerUID and OwnerGID: restricts the permissions of who can modify an event. These fields are used primarily for auditing purposes. Modification restrictions are a configurable option.
- ▶ FirstOccurrence: date and time during which the event first occurred on the managed device.
- ▶ LastOccurrence: date and time of the last occurrence of the event.
- ▶ AlertKey: the descriptive key that indicates the managed object instance referenced by the alert.
- ▶ AlertGroup: the descriptive name of the type of failure indicated by the alert.
- ▶ Tally: automatically maintained count of the number of inserts and updates of the alert from any source. This count is affected by *deduplication*, which is discussed later in this book.

### ***alerts.details***

In certain circumstances, operators may be interested in the *raw* (that is, before any advanced management) data coming from a probe. When details tracking is enabled, data is stored in the `alerts.details` table as token value pairs. Details may be viewed from the Details tab located on the Information window, accessed using the alerts menu for selected events. Details are linked to their respective events using the identifier field. Details could also be stored in the `ExtendedAttr` field of the event. See “Second solution: extended attributes” on page 213 for a description of this field and the supporting rules file and `ObjectServer` functions.

### ***alerts.journal***

When working with events, users may want to track the history of them. For example:

- ▶ Who has owned it?
- ▶ What severity levels has it passed through?
- ▶ What automations have run on it?

Journals provide this functionality. The journal information is held in the `alerts.journal` table. Journals are linked to their respective events using the `Serial` field, which is a primary key in both `alerts.status` and `alerts.journal`.

## **Communication**

The main communication mechanism is based on Tabular Data Stream (TDS). TDS is licensed from Sybase.

Data flow between `ObjectServer` and the native eventlists and gateways is managed by a second communications link known as Insert Delete Update Control (IDUC). The IDUC channel is used by the `ObjectServer` to periodically notify the clients of the events that have changed since the last notification. On receipt of the IDUC notification the client requests the full data for the updated events via the normal TDS channel. This technique is used to balance the client load over a period of time known as the *granularity period*, which by default is 60 seconds. Each IDUC client is notified on a round-robin basis once in each granularity cycle.

This information is stored in the interfaces file `omni.dat` in the `$NCHOME/etc` directory on UNIX and `%NCHOME%\ini\sql.ini` on Windows. The format of the interfaces file is platform-specific. That is, an interfaces file written on one architecture (for example, Solaris™) cannot be used on another (for example, Linux).

Server components need to access this interfaces file to know how to start up. Client components (desktops and probes) need to access the interfaces file to know where to connect to their named server.

The interfaces file, while text-only, should not be edited directly. Instead, there are both GUI (nco\_xigen) and non-GUI (nco\_igen) utilities that ensure correct formatting of the interfaces file.

Figure 2-11 shows the ObjectServer communication process regarding the interfaces file.

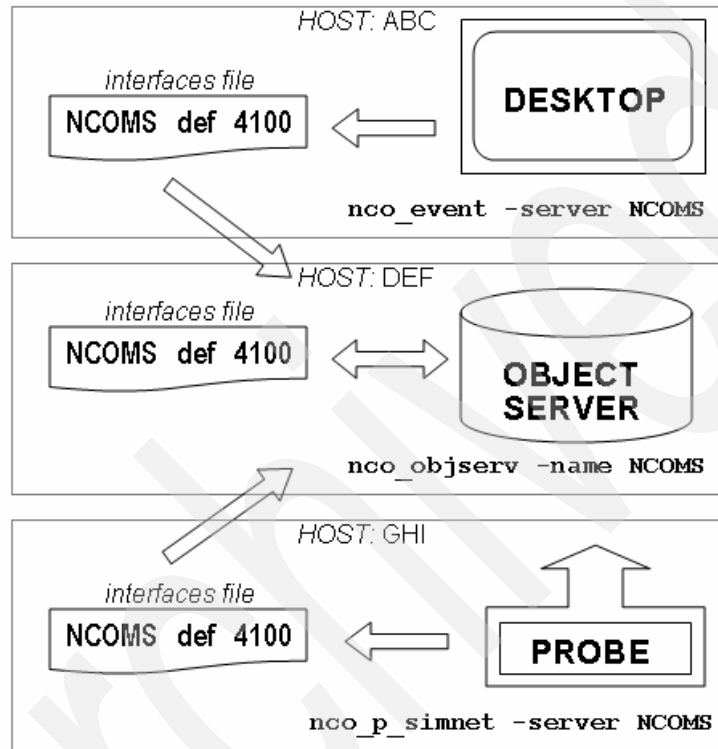


Figure 2-11 ObjectServer communication

## Administration tools

The Netcool administrator configuration application is a stand-alone Java application for configuring one or more ObjectServers.

Within the administrator configuration you can configure a number of settings of Netcool/OMNIbus., which are described in detail on the following pages.

## Automations

Tivoli Netcool/OMNIbus automations provide automatic management of events in the ObjectServer. The automation will detect predefined changes in the ObjectServer and execute automated responses to these changes. This enables

the ObjectServer to process many alerts without requiring an operator to take action. Automations can consist of triggers, procedures, and user-defined signals. Automations can:

- ▶ Perform external commands automatically on the receipt of certain events.
- ▶ Incorporate escalation procedures.
- ▶ Allow correlation of events and automatically manage events.
- ▶ Execute an external command. This gives the ability to launch other applications or run scripts.
- ▶ Execute SQL statements internally on the events stored in the ObjectServer.
- ▶ Use field values from an event.
- ▶ Use variables from the OS environment or those stored in the ObjectServer.
- ▶ Add a journal entry to track changes.

## Triggers

Triggers form the basis of the ObjectServer automation subsystem. Triggers automatically fire (execute a trigger action) when the ObjectServer detects an incident associated with a trigger. In a trigger, you can execute SQL commands and call procedures in response to the change.

Triggers are used to automatically manage events and perform escalation, correlation, and external commands. Trigger groups allow you to manage multiple triggers. Each trigger must belong to only one trigger group, but can be moved between groups. A trigger group can be deleted only if empty.

Triggers can be deployed with the aid of the administrator GUI as well as from a command line using SQL instructions.

There are three types of triggers, as discussed below.

## Temporal trigger

Temporal triggers execute on a time interval. Figure 2-12 shows the configuration window of an temporal trigger.

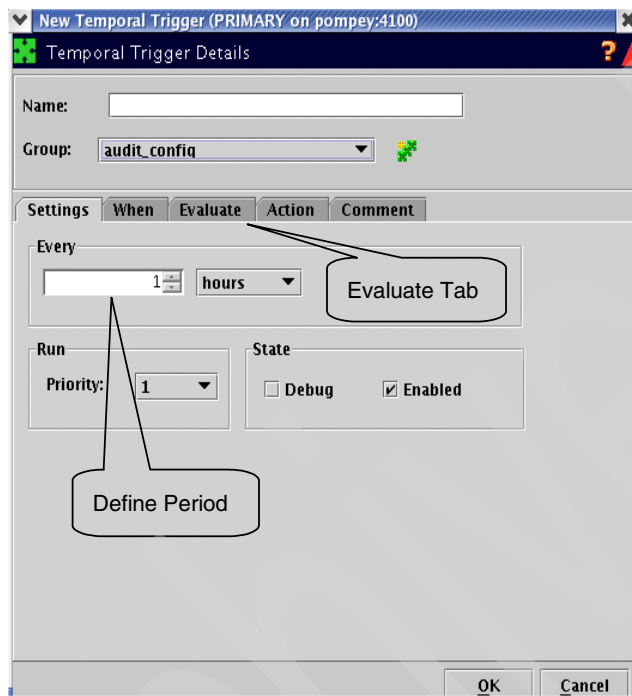


Figure 2-12 Temporal trigger



## Database trigger

Database triggers execute on database conditions. Figure 2-13 shows the configuration window of an database trigger.

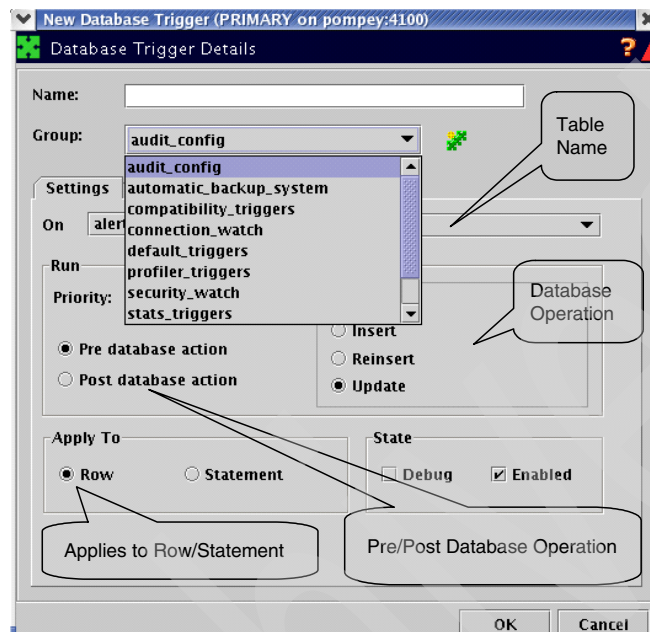


Figure 2-13 Database trigger

## Signal trigger

Signal triggers execute on a system-defined or user-defined signal. Figure 2-14 shows the configuration window of a signal trigger.

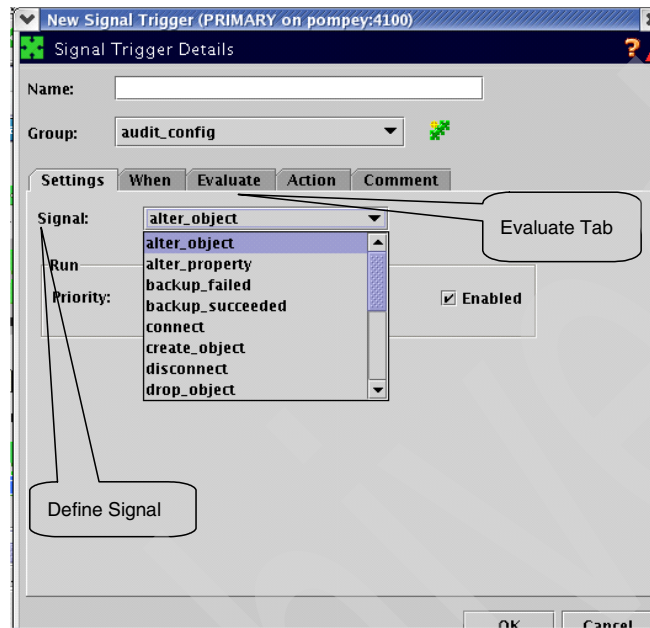


Figure 2-14 Signal trigger

Figure 2-15 shows the configuration window of a user-defined signal.

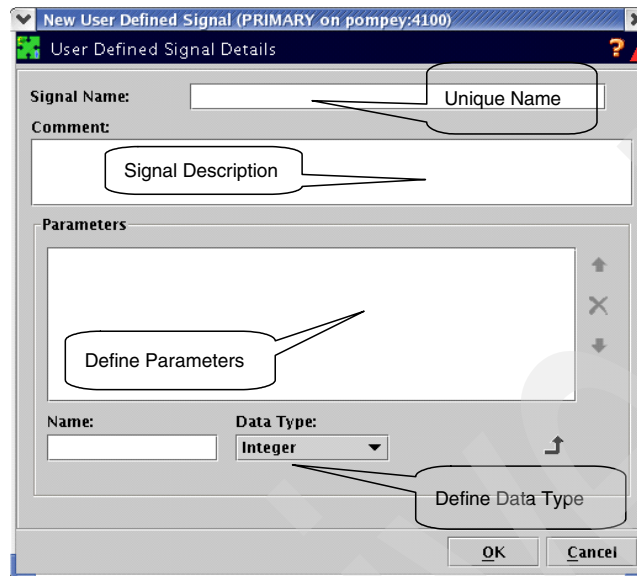


Figure 2-15 User-defined signals

All triggers have a Settings tab to set the conditions under which the trigger executes (time interval, database action, receipt of signal). A When tab allows the operators to test for a particular condition before the action is performed. If the condition is not met, the action is not performed. An Evaluate tab is used to build a read-only temporary table (signal and temporal triggers only). An Action tab determines what tasks have to be performed. A Comment tab is for documentation.

The Action tab can consist of a sequence of SQL instructions as well as a call to a procedure that can be defined in the appropriate panels of the administrator GUI.

OMNIbus is shipped with a set of default automations that help the execution of daily housekeeping activities.

**Note:** Shipped automations can be amended, but changes may affect the running of the ObjectServer.

## Process control

The process control system performs two primary tasks:

- ▶ It runs external procedures that are specified in automations. Automations detect changes in the ObjectServer and run automated responses to those changes.
- ▶ It manages local and remote processes.

**Note:** On Windows systems, process control only runs external procedures that are specified in automations. You cannot use process control to manage OMNIbus processes on Windows. A new version of process control for Windows is planned to provide this functionality.

On UNIX systems, you can use process control to configure remote processes to simplify the management of Netcool/OMNIbus components such as ObjectServers, probes, and gateways. The process control system consists of:

- ▶ Process control agents, which are programs installed on each host for managing processes
- ▶ A set of command-line utilities that provide an interface to process management

Process control can be configured and managed through the administrator console.

See Figure 5-5 on page 174 for an example of the administrator console nco\_xigen from the lab environment.

## Users, roles, and groups

User roles are permission sets that allow access to ObjectServer data.

Predefined roles correspond to:

- ▶ Normal users
- ▶ Administrators
- ▶ Super users

User-settable parameters include:

- ▶ User ID and full name
- ▶ Group membership
- ▶ Restriction filters
- ▶ Password and enable state

Groups have users as members, manage users efficiently, and can have associated roles and restriction filters. There are predefined Netcool/OMNIBus groups.

## Restriction filters

Restriction filters create an SQL filter that can be applied to groups and users. They can filter on any database or database table, restricting the ability to view certain columns of data, for example. Filters are inherited by users in a group.

## Conversions

Conversions substitute integers for character strings. Integers are efficient in the ObjectServer, but character strings are meaningful to humans. These are used for display in the event list and filter builder. Figure 2-16 shows the ObjectServer conversion panel.

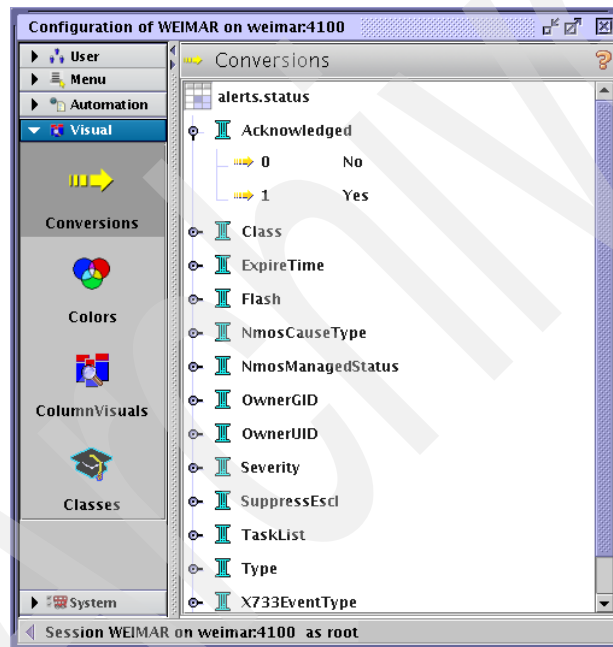


Figure 2-16 Conversion

## Tools

Tools allow the control of alert management functions within Tivoli Netcool/OMNIBus from the event list. Each tool has an associated SQL statement (called an internal effect), an executable (called an external effect), or both. Tools can be grouped in tools menus, which can then be associated with classes of alert. Tools are created independently from menus, so a global set of

tools can be used on different menus. They can be restricted by both class and group. This allows the tools to be event sensitive and user based. Tools can:

- ▶ Execute an external command on UNIX or Windows platforms. This gives the ability to launch other applications or run UNIX commands.
- ▶ Execute SQL statements internally on the events stored in the ObjectServer.
- ▶ Run an executable.
- ▶ Gather more information from the user through the use of prompts.
- ▶ Use field values from an event.
- ▶ Use variables from the OS environment or those stored in the ObjectServer.
- ▶ Add a journal entry to track changes. These can be optional or forced.

Access to tools can be restricted by class and group. A global set of tools may be re-used on different menus. Most standard tools run from the Alerts® menu. Tools can also be anchored to other menus in the GUI.

Sometimes it is necessary to prompt the user for information needed to run the tool. Netcool/OMNIbus provides prompts. Prompts are independent of tools. Define a global set of prompts that can be accessed by any tool. Prompts are integrated into the command syntax of the tool. The location of the prompt within the command is important. It is replaced by user input. There are:

- ▶ Interacting-with-user prompts
- ▶ Fixed-choice prompts
- ▶ Lookup prompts

**Note:** Standard tools are available when Netcool/OMNIbus is installed.

## Administration interface

IBM Tivoli Netcool administrator configuration application `nco_config` is a stand-alone Java application for configuring one or more ObjectServers.

The dedicated native desktop console is an integrated suite of graphical tools used to view and manage events, and to configure how event information is presented. Together with an included Structured Query Language (SQL), the interactive command-line interface provides administrative access to the ObjectServer.

**Note:** `nco_sql` is the SQL interface on UNIX and `isql` on Windows.

Additionally, a complete command-line version of the capabilities provided with the Web console is provided using a Web Administration Application Programming Interface (WAAPI).

The dedicated native desktop console's functionality is equivalent to the Java TEC console. The WAAPI functions are equivalent to Tivoli Framework w-commands, except that WAAPI commands are written and processed in XML formatted files.

ObjectServer severity colors can be edited and added for viewing on Windows desktop and Netcool/Webtop only. Column visuals define default settings for Event List column displays. Most settings can be overridden in the view builder.

### Severities

There are six default levels for each event, as shown in Figure 2-17.



Figure 2-17 Objectserver severities

### Proxy server

The ObjectServer receives alert information from probes. In a standard configuration, alerts are forwarded directly to the ObjectServer. You can configure a proxy server to reduce the number of probe connections to an ObjectServer. Where a large number of probes are forwarding alert information directly to the ObjectServer and a large number of desktop connections are also made to the same ObjectServer, there can be a negative impact on performance.

**Note:** Putting a proxy server in place would not reduce the number of events being forwarded. Its purpose is to reduce the number of connections.

A proxy server provides a buffer to reduce the number of direct connections to the primary ObjectServer. Multiple probe connections made to the proxy server are multiplexed and forwarded through a single connection to the ObjectServer. Figure 2-18 shows how probes communicate with the proxy server.

**Note:** The proxy or *probe consolidation server* was originally introduced to deal with low limits on file handles that restricted connections. The proxy is now used more for easier firewall management. Placing a proxy in the domain where the probes are located requires only one firewall change up to the ObjectServer. Probes can then be added or redeployed without firewall change.

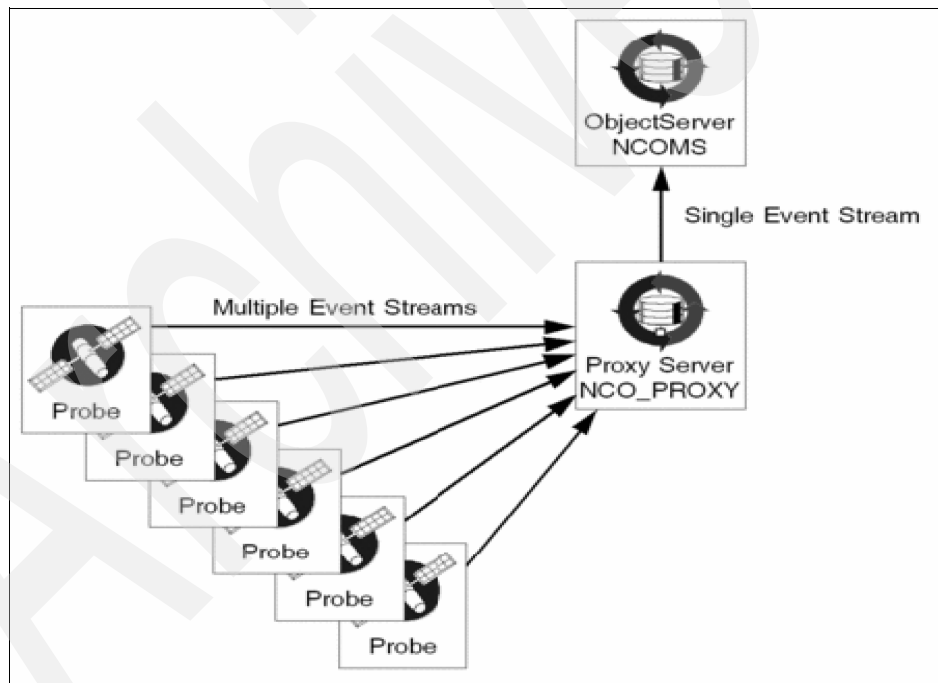


Figure 2-18 Example proxy configuration

### Virtual ObjectServer configuration

A resilient pair configuration consists of two ObjectServers (presumably geographically separated) with a bi-directional gateway serving to keep both in



synchronization. That way, regardless of which ObjectServer receives the event, a desktop connected to either ObjectServer would see the same events.

In the interfaces file, both ObjectServers are configured. Additionally, a third entry is required—a *virtual* entry with the host names and port numbers of both members of the pair.

Clients (desktops and probes) can still connect to the either ObjectServer directly. If, however, they connect to the *virtual* name, the underlying protocol takes care of forwarding traffic to the first entry of the pair.

Figure 2-19 shows the overview of the virtual ObjectServer concept.

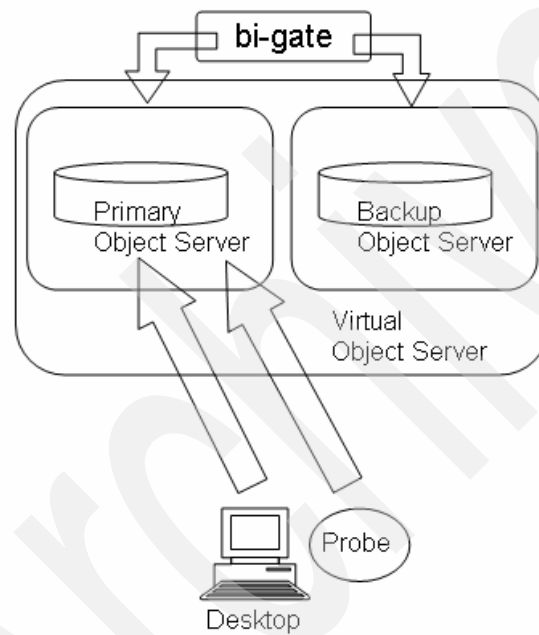


Figure 2-19 Virtual ObjectServer

Should a failure occur, the failover process automatically reroutes traffic to the second entry. From the view of probes, this process is completely transparent. With desktops, a dialog box appears notifying the user that a failure has occurred.

When the first member of a pair recovers, the client will be redirected back to the first interfaces file entry of the virtual server definition through a process known as failback.

Figure 2-20 shows the overview of the virtual ObjectServer concept in failover, failback mode.

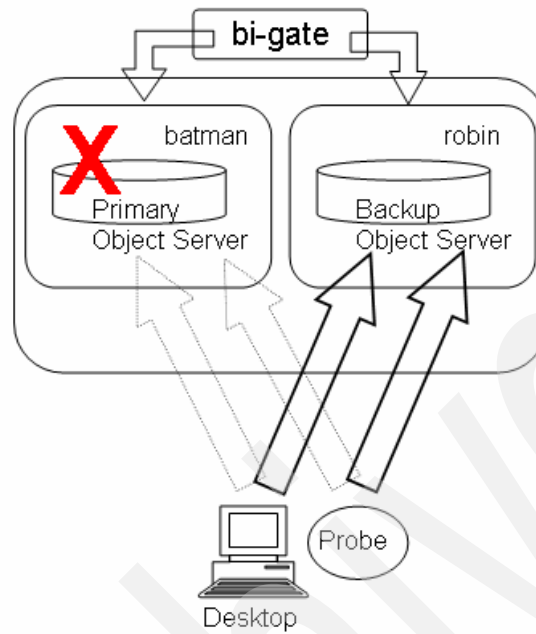


Figure 2-20 Failover and failback

There is no equivalent function in TEC compared to the Netcool/OMNIbus virtual ObjectServer functionality. This is one of the benefits for Netcool/OMNIbus.

## 2.2.4 Probes



Probes are passive and lightweight software components that connect and listen to an event or data source, looking for defined events. Events are formatted and sent to the ObjectServer.

Probes use the logic specified in a rule file to manipulate the elements of an event stream (*tokens*) before converting them into fields of an event in the ObjectServer alerts.status table.

The probes provide similar functionality to that of the TEC adapters, such as pattern matching, assigning variables, and discarding.

Figure 2-21 shows the Netcool/OMNIBus ObjectServer Probe architecture in general.

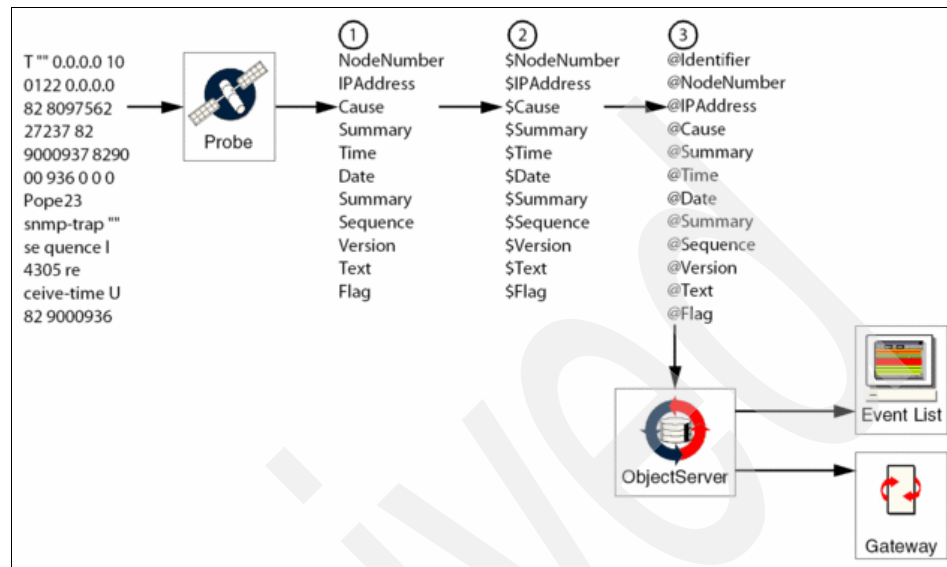


Figure 2-21 Probe architecture

Probes can also be configured to modify and enrich event information via lookup files and rules file includes. Refer to "Lookup tables" on page 87 and 6.5.3, "Netcool Knowledge Library" on page 274 for more information about this topic.

**Note:** The estimated size of an Netcool/OMNIBus event stored in the ObjectServer database is about 10 Kbytes per event, considering a full row and the status, journal, and detail tables. It also includes one update and one delete (for example, 10,000 rows need 100 Mb).

Probes are resilient with a store and forward functionality, automatic fail-over to alternate ObjectServer, and can send to more than one ObjectServer and table.

Each probe is uniquely designed to acquire event data from a specific source. However, probes can be categorized based on how they acquire events. For example, the probe for Oracle obtains event data from a database table, and is therefore classed as a database probe. The types of probes are:

- ▶ Device
- ▶ Log file
- ▶ Database
- ▶ API
- ▶ Miscellaneous

- ▶ CORBA
- ▶ EIF Probe

**Note:** The probe type is determined by the method in which the probe detects events. For example, the probe for Agile ATM Switch Management detects events produced by a device (an ATM switch), but it acquires events from a log file, not directly from the switch. Therefore, this probe is classed as a log file probe and not a device probe.

## Device probes

A device probe acquires events by connecting to a remote device, such as an ATM switch. Device probes often run on a separate machine from the one they are probing and connect to the target machine through a network link, modem, or physical cable. Some device probes can use more than one method to connect to the target machine. Once connected to the target machine, the probe detects events and forwards them to the ObjectServer. Some device probes are passive, waiting to detect an event before forwarding it to the ObjectServer (for example, the probe for Marconi ServiceOn EMOS). Other device probes are more active, issuing commands to the target device in order to acquire events (for example, the TSM for Ericsson AXE10).

## Log file probes

A log file probe acquires events by reading a log file created by the target system. For example, the probe for Heroix RoboMon Element Manager reads the Heroix RoboMon Element Manager event file. Most log file probes run on the machine where the log file resides. This is not necessarily the same machine as the target system. The target system appends events to the log file. Periodically, the probe opens the log file, acquires and processes the events stored in it, and forwards the relevant events to the ObjectServer as alerts. You can configure how often the probe checks the log file for new events and how events are processed.

## Database probes

A database probe acquires events from a single database table, the source table. Depending on the configuration, any change (insert, update, or delete) to a row of the source table can produce an event. For example, the probe for Oracle acquires data from transactions logged in an Oracle database table. When a database probe is started, it creates a temporary logging table and adds a trigger to the source table. When a change is made to the source table, the trigger forwards the event to the logging table. Periodically, the events stored in the logging table are forwarded to the ObjectServer as alerts and the contents of the logging table are discarded. You can configure how often the probe checks the logging table for new events.

**Note:** Existing triggers on the source table may be overwritten when the probe is installed.

Database probes treat each row of the source table as a single entity. Even if only one field of a row in the source table changes, all of the fields of that row are forwarded to the logging table and from there to the ObjectServer. If a row in the source table is deleted, the probe forwards the contents of the row before it was deleted. If a row in the source table is inserted or updated, the probe forwards the contents of the row after the insert or update.

### **API probes**

An API probe acquires events through the API of another application. For example, the probe for Sun™ Management Center uses the Sun Management Center Java API to connect remotely to the Sun Management Center. API probes use specially designed libraries to acquire events from another application or management system. These libraries contain functions that connect to the target system and manage the retrieval of events. The API probes call these functions that connect to the target system and return any events to the probe. The probe processes these events and forwards them to the ObjectServer as alerts.

### **Miscellaneous probes**

All of the miscellaneous probes have characteristics that differentiate them from the other types of probes and from each other. Each of them carries out a specialized task that requires them to work in a unique way. For example, the e-mail probe connects to the mail server, retrieves e-mails, processes them, deletes them, and then disconnects. This is useful on a workstation that does not have sufficient resources to permit an SMTP server and associated local mail delivery system to be kept resident and continuously running. Another example of a probe in the miscellaneous category is the ping probe. It is used for general-purpose applications on UNIX platforms and does not require any special hardware. You can use the ping probe to monitor any device that supports the ICMP protocol, such as switches, routers, PCs, and UNIX hosts.

### **CORBA probes**

Common Object Request Broker Architecture (CORBA) allows distributed systems to be defined independent of a specific programming language. CORBA probes use CORBA interfaces to connect to the data source, usually an Element Management System (EMS). Equipment vendors publish the details of their specific CORBA interface as Interface Definition Language (IDL) files. These IDL files are used to create the CORBA client and server applications. A specific probe is required for each specific CORBA interface. CORBA probes use the

Borland VisiBroker Object Request Broker (ORB) to communicate with other vendor's ORBs. You must obtain this ORB from technical support. Most CORBA probes are written using Java, and require specific Java components to be installed to run the probe, as described in the individual guides for these probes. Probes written in Java use the following additional processes:

- ▶ The probe-nco-p-nonnative probe, which enables probes written in Java to communicate with the standard probe C library (libOpl)
- ▶ Java runtime libraries

## EIF probe

A range of Tivoli products, and all of the Tivoli Enterprise Console adapters, generates events using the Event Integration Facility (EIF). The Netcool/OMNIbus probe for Tivoli EIF can receive EIF events sent from any of these Tivoli applications and forward them to the ObjectServer.

The EIF probe is a key part of the architecture for integration between TEC and OMNIbus. It features in our best practice recommendations using event-forwarding rules for sending events from the TEC server via the EIF probe to the ObjectServer, and also from TEC adapters via the EIF probe to the ObjectServer.

**Note:** We recommend installing the EIF probe at the event source server. One advantage of this is the probes' capability to provide a secure socket layer (SSL) connection between itself and the ObjectServer.

The probe for Tivoli EIF can connect to a remote device. The remote device is specified using the PortNumber property in the properties file.

The probe for Tivoli EIF supports failover configurations where two probes run simultaneously. One probe acts as the master probe, sending events to the ObjectServer, while the other acts as the subordinate probe on standby. If the master probe fails, the subordinate probe activates. Start two instances of the probe, one as master and one as subordinate. While the subordinate probe receives heartbeats from the master probe, it will not forward events to the ObjectServer. If the master shuts down, the subordinate probe will stop receiving heartbeats from the master and any events it receives thereafter will be forwarded to the ObjectServer on behalf of the master probe. When the master is running again, the subordinate will continue to receive events, but will no longer send them to the ObjectServer.

When the probe for Tivoli EIF starts, it monitors the port specified in the properties file, and uses the EIF Java API to receive and parse the events. After this process, the probe generates alerts and sends them to the ObjectServer. The probe comprises two types of threads:

- ▶ One set of threads (the thread pool) to process events read from a client
- ▶ One thread to send the structured events to the ObjectServer

The probe for Tivoli EIF has a timeout facility that allows it to disconnect from the client if it fails to receive the next alarm data within a predefined amount of time. You can specify how long the probe waits before disconnecting using the inactivity property. After this length of time, the probe disconnects from the device and sends a ProbeWatch message to the ObjectServer. We recommend that you set this property to 0, so that the connection never drops.

The probe for Tivoli EIF can capture the data stream sent to it from a TEC adapter or an IBM Tivoli Monitoring (ITM) server. This data is stored in a log file and can be used for debugging purposes, to develop new features for the probe, or to pass to other management systems that require the same data.

All probes support multi-byte character sets. To view the character sets correctly, you must configure the locale settings on the host machine correctly. Each multi-byte character set is configured slightly differently on each platform. The following two sections describe example locale configurations on a UNIX platform and on Windows, respectively. Other character sets on other platforms will be configured in a similar way.

### ***Performance considerations***

The performance can be limited by this mechanism, and customers should be aware of throughput that can be achieved when this integration is enabled. During performance tests, the throughput achieved using this mechanism and the default configuration shipped is approximately 40–50 events per second using the hardware and OS configurations described in the *Tivoli & Netcool Event Flow Integration* white paper.

### **Probe components**

A probe has following primary components:

- ▶ An executable file
- ▶ One properties file
- ▶ One rules file (or more if include statements are counted)

### **Executable file**

The executable file is the core of a probe. It connects to the event source, acquires and processes events, and forwards the events to the ObjectServer as

alerts. Probe executable files are stored in the directory “\$OMNIHOME/probes/arch”, where *arch* is the platform name of the architecture. For example, the executable file for the EIF Probe that runs on AIX is “\$OMNIHOME/probes/aix5/tivoli\_eif.props”.

## Properties file

Probe properties define the environment in which the probe runs. For example, the server property specifies the ObjectServer to which the probe forwards alerts. Probe properties are stored in a properties file in the directory “\$OMNIHOME/probes/arch”. Properties files are identified by the .props file extension. For example, the properties file for the EIF probe that runs on AIX is “\$OMNIHOME/probes/aix5/tivoli\_eif.props”.

## Probe operation

When initializing a probe, it reads its properties file that comes with each probe installation and connects to the ObjectServer. The probe binary retrieves the event stream from its source (for example, an element manager, SNMP traps, log files, APIs, sockets).

Figure 2-22 shows the Netcool/OMNIBus ObjectServer Probe operation in the flow.

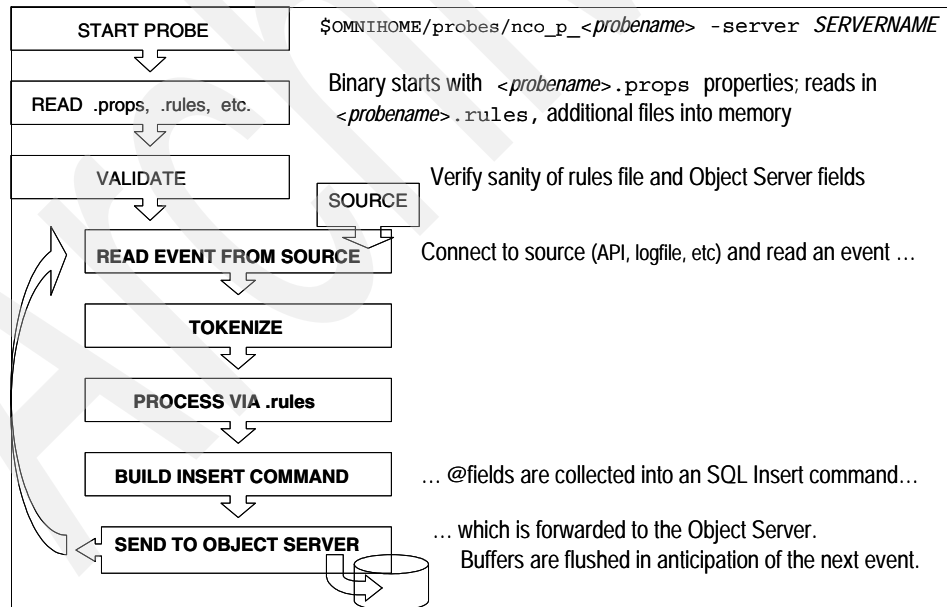


Figure 2-22 shows the ObjectServer probe operation



The probe binary tokenizes the event stream. The interpreted rules assign token elements to ObjectServer fields and add extra information. The event is sent to the ObjectServer and the probe then gets the next event.

Probe properties define a probe environment and behavior. There are two types of probe properties:

- ▶ Generic - All probes have these properties.
- ▶ Probe-specific - additional properties, different for each probe.

### **Probe rules file**

The rules file defines how the probe should process event data it receives in order to create a meaningful Tivoli Netcool/OMNIbus alert.

The goals and objectives of the rules file development are listed below:

- ▶ The rules files created should be of production quality and provide *out-of-the-box* value.
- ▶ The rules files should not require any modifications to the ObjectServer (that is, no additional event fields other than the default fields provided in Tivoli Netcool/OMNIbus).
- ▶ The basic structure of the files should be easy to maintain and easily extendible, enabling the quick addition of event handling for new devices without affecting existing rules.
- ▶ Basic textual-conventions for the rules files should be determined and followed to ensure that rules files created by different persons share a common format.
- ▶ The rules files should be sufficiently documented to allow each event to be understood without additional documentation.
- ▶ The structure of the rules file should be as efficient as possible to maximize throughput of events.
- ▶ The events formatted by the rules files should be *deduplicated* properly by the Tivoli Netcool/OMNIbus ObjectServer.
- ▶ The events formatted by the rules files should be compatible with the "GenericClear" automation whenever possible.

At a minimum, the completed rules files should provide the following basic functionality:

- ▶ Automated deduplication of events and alarms in the Tivoli Netcool/OMNIbus ObjectServer.
- ▶ Automated *Generic Clear* correlation of problem/resolution events.

- ▶ Informative and descriptive event presentation in Tivoli Netcool/OMNIBus ObjectServer.
- ▶ Development of a standalone rules files, which will be the only rules files read by the probe at start time.
- ▶ Development of include rules files that will be added to an already existing library of rules files called the Netcool Knowledge Library (NCKL). At start time, the probe will read the core NCKL rules files, which will *call* (that is, include) the device-specific or application-specific rules files written to process the information received from the products to be integrated.

All probes split the event stream into tokens. Each token is a string containing raw information about the alert. Tokens are identified within the rules file using “\$” (for example, **\$node** is a token holding the node name). The rules file determines how each token is mapped to the corresponding field, for example, @field in the alerts.status table of the ObjectServer to create a Netcool/OMNIBus event. Some or all of the tokens can be used to set the field values of alerts.status.

Example 2-1 shows an example of splitting an event stream into tokens.

---

*Example 2-1 Event stream into tokens example*

---

```
$from = extract($Details, "FTP LOGIN FROM ([^ ]+) .*")
@AlertKey = $from
@AlertGroup = "security"
#$as = extract($Details, ".*, ([a-zA-Z0-9]+)")
$as = extract($Details, ".* ([a-z]+)")
@Summary = "ftp: login from " + $from + ", as " + $as
@Severity = 2
```

---

Example 2-2 shows an example of mapping between event stream tokens and ObjectServer fields in the tivoli\_eif.rules file.

---

*Example 2-2 Token and fields*

---

```
@TECSummary = $msg
@TECHostname = $hostname
@TECFQHostname = $fqhostname
@TECDate = $date
@TECRepeatCount = $repeat_count
@LastOccurrence = getdate
@FirstOccurrence = getdate
@TECStatus = $status
```

---

Nearly all field values of alerts.status may be set through the rules file. Fields that cannot be configured though alerts.status include Serial, ServerSerial, and Tally. A probe can use multiple rules files.

### Lookup tables

Lookup tables are an easy way of adding external information to an event. There are two ways to define a lookup table:

- ▶ Rules table - The table is defined directly in the rules file.
- ▶ File table - The information is stored in a separate file.

The lookup tables provide similar functionality to that of the TEC fact files. The lookup tables may be imbedded in the rules file, but are usually created in a separate file for ease of maintenance. A lookup hash table must contain a set of key and value entries.

Example 2-3 shows a sample of a list of International Mobile Subscriber Identity (IMSI) values mapped to respective customers in a lookup file.

*Example 2-3 Lookup entry example*

---

```
# key <tab separator> value
234011234567890 Corp-2
235150123456789 Netcool
```

---

## 2.2.5 Monitors



Monitors can be compared to the functionality of IBM Tivoli Monitoring. Their function will be integrated into IBM Tivoli Monitoring. Netcool/Internet Service Monitor (Netcool/ISM) is now called IBM Tivoli Composite Application Manager for ISM.

## 2.2.6 Gateways



Netcool Gateways are software applications designed to provide unidirectional or bidirectional communications between a Netcool/OMNIBus ObjectServer and

external programs, complementary third-party applications such as databases, help desk, or Customer Relationship Management (CRM) systems.

Gateways are used to replicate events or to maintain a backup ObjectServer. Application gateways enable the integration of different business functions. The transfer of events is transparent to operators.

Gateways are crucial for the deployment of the Netcool/OMNIbus Virtual ObjectServer configuration for failover and failback.

Figure 2-23 shows the Netcool/OMNIbus gateway architecture.

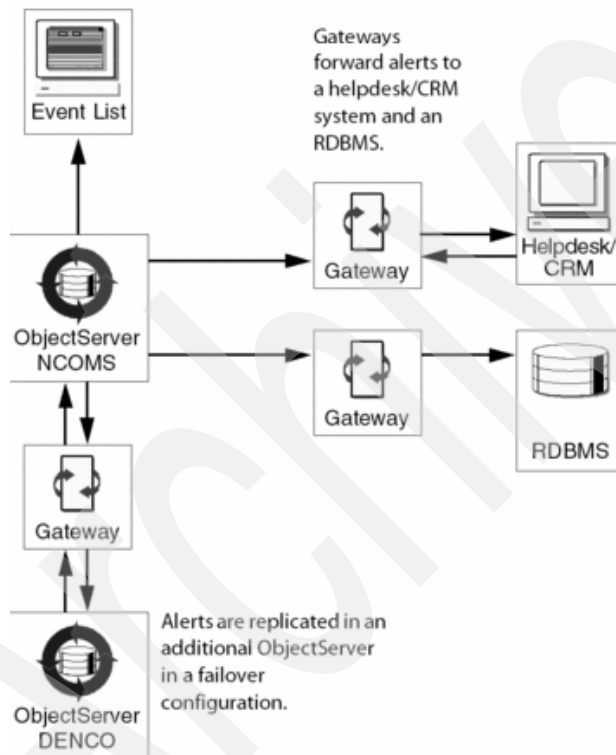


Figure 2-23 Netcool/OMNIbus gateway architecture

The two types of gateways are the *unidirectional* and *bidirectional* types.

Unidirectional gateways allow alerts to flow in only one direction. Changes made in the source ObjectServer are replicated in the destination ObjectServer or application, but changes made in the destination ObjectServer or application are not replicated in the source ObjectServer. Unidirectional gateways can be considered as archiving tools.

Figure 2-24 shows a unidirectional ObjectServer gateway.

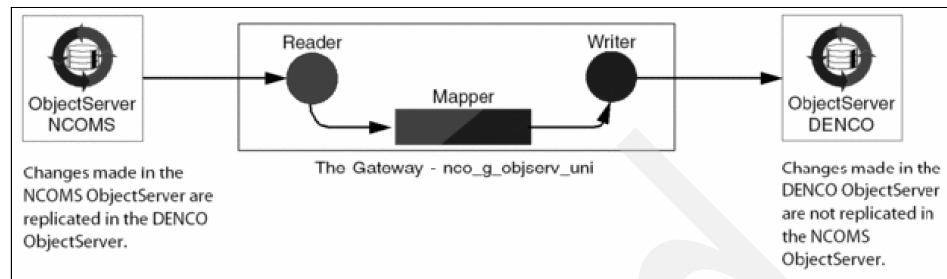


Figure 2-24 Unidirectional gateway

Bidirectional gateways allow alerts to flow from the source ObjectServer to the target ObjectServer or application, and also allow feedback to the source ObjectServer. In a bidirectional gateway configuration, changes made to the contents of a source ObjectServer are replicated in a destination ObjectServer or application, and the destination ObjectServer or application replicates its alerts in the source ObjectServer. Bidirectional gateways can be considered as synchronization tools.

Figure 2-25 shows an example of an bidirectional gateway.

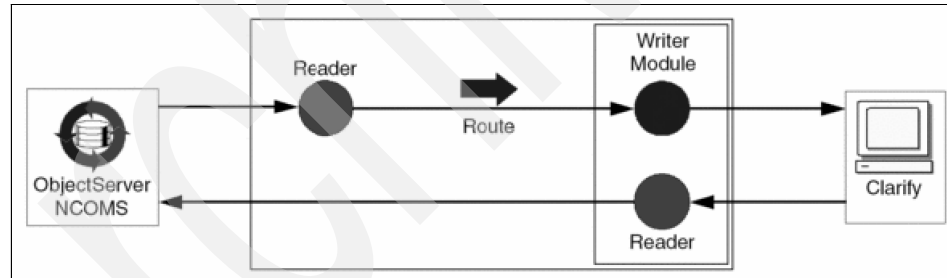


Figure 2-25 Bidirectional gateway

Gateways have reader and writer components. Readers extract alerts from the ObjectServer. There is only one type of reader, the ObjectServer reader.

Writers forward alerts to another ObjectServer or to other gateway-connected applications. Writer modules manage communications between gateways and third-party applications, and format the alert correctly for entry into the application.

Gateways can be configured to operate in secure mode. If the ObjectServer is not running in secure mode, probe, gateway, and proxy server connection requests are not authenticated.

Gateways also operate in store-and-forward mode. If there is a problem with the gateway target, the ObjectServer and database writers can continue to run using store-and-forward mode after switching into store mode. The gateway switches back when it detects the destination server back online again.

Both the unidirectional and bidirectional gateways use the following set of configuration files:

- ▶ Properties file: Define the gateway's operational environment, such as connection details and the location of the other configuration files.
- ▶ Map definition file: The gateway can replicate specific system tables and any user table in the ObjectServer. To do this, the gateway maps data to the appropriate fields in the ObjectServer using a map definition file. This contains mappings that define how the gateways map this data.
- ▶ Startup command file: The startup command file contains a set of commands that the gateway executes each time it starts. These commands allow the gateway to transfer any subsidiary table data to a set of target tables.
- ▶ Table replication file: The gateway can replicate the data in specific system tables and any user table between ObjectServers in a backup pair. Details of the tables to be replicated are stored in the table replication definition file.

Example 2-4 shows syntax examples within the table replication definition file.

*Example 2-4 Example syntax of the table replication file*

---

```
FILTER WITH 'Severity=4 and Severity=5'
FILTER WITH 'Severity=4 and Type=1'
FILTER WITH 'TECGWCtrl=1 and Severity>4'
FILTER WITH 'Summary nmatch "Link Down*" or Summary regmatch "Link
Up*"'
FILTER WITH 'Summary like "Link Down on port"'
FILTER WITH '(Severity = 3 and Node = test) or (Severity = 5)'
FILTER WITH 'Severity = 3 and Node = \'test\' or Class = 11010';
```

---

The parenthesis around “Severity = 5” in Example 2-4 are not strictly necessary. They are added here for readability when showing a complex filter. There is an example of inclusion of single quotation marks in the ObjectServer gateway guide (note that all quotation marks in this example are single ') VIA FILTER 'Name != \'nobody\'".

## 2.2.7 Netcool/Webtop



Netcool/Webtop is a Web-based application that processes network events from one or more ObjectServers and presents the event data to users in various graphical formats. Webtop architecture uses the Apache Tomcat Web server to provide OMNIbus Web console communications with an ObjectServer. 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 a combination of the two.

Figure 2-26 shows the architecture of the Netcool/Webtop Apache Tomcat Web server.

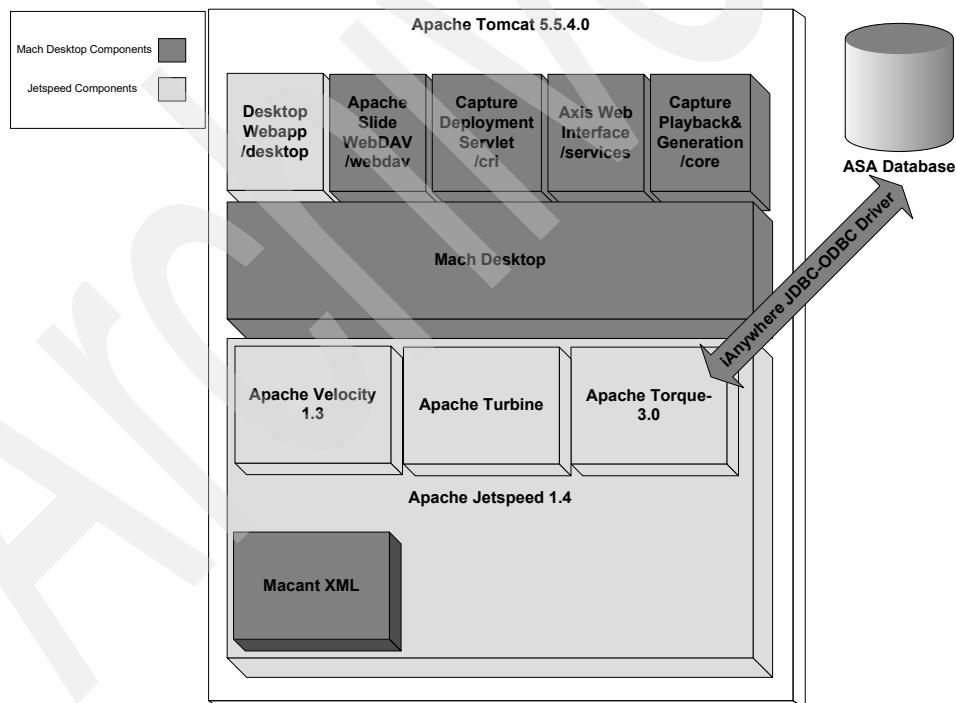


Figure 2-26 Apache Tomcat Web server architecture

Netcool/Webtop displays multiple ObjectServers' information on one Web page.

The main event display components are:

- ▶ The Java-based active event list (AEL) allows clients to run sophisticated native desktop actions such as acknowledging alerts, viewing alert journals, taking ownership of alerts, running tools, and so forth.
- ▶ The dynamic HTML Lightweight Event List (LEL) provides clients with the data filtering, data sorting, and information drill-down capabilities of the AEL.
- ▶ The HTML tableview component provides clients with a static event list in the form of a table showing a defined set of alerts. The non-interactive tableview provides an immediate snapshot of alert status within a monitored system.

The Netcool/Webtop application is equivalent to the TEC UI server and the TEC WebSphere-based GUI console. The Netcool environment can also be configured to be more scalable with a deployment of more than one display ObjectServer and Netcool/Webtop servers.

## 2.2.8 Netcool GUI Foundation

The Netcool GUI Foundation (NGF) is a server application that delivers Web-based IBM Tivoli Netcool products within a single, unified framework. The NGF provides single sign-on, consolidated user management, and a single point of access for integrated Netcool products. It allows you to customize the presentation of content for integrated Netcool products, and control access to content across products. The NGF is not available separately. It is installed automatically with the first Netcool GUI Foundation-integrated product that is installed. Subsequent products may install updated versions of the NGF. The NGF uses Netcool Security Manager for authentication and authorization.

The Netcool GUI Foundation Version 1.1 supports:

- ▶ All external authentication sources supported by IBM Tivoli Netcool Security Manager, and IBM Tivoli Netcool Security Manager failover.
- ▶ Two or more instances of the Netcool GUI Foundation connecting to a single IBM Tivoli Netcool Security Manager server.
- ▶ IBM Tivoli Netcool products including:
  - IBM Tivoli Netcool/Webtop 2.1
  - IBM Tivoli Network Manager IP Edition (Advanced) 3.7
  - IBM Tivoli Network Manager Transmission Edition 5.6
  - IBM Tivoli Business Systems Manager 4.1

The Netcool GUI Foundation supports installation on IPv4 and dual stack IPv4/IPv6 devices.



Figure 2-27 shows an example deployment where three IBM Tivoli Netcool products are installed within the Netcool GUI Foundation.

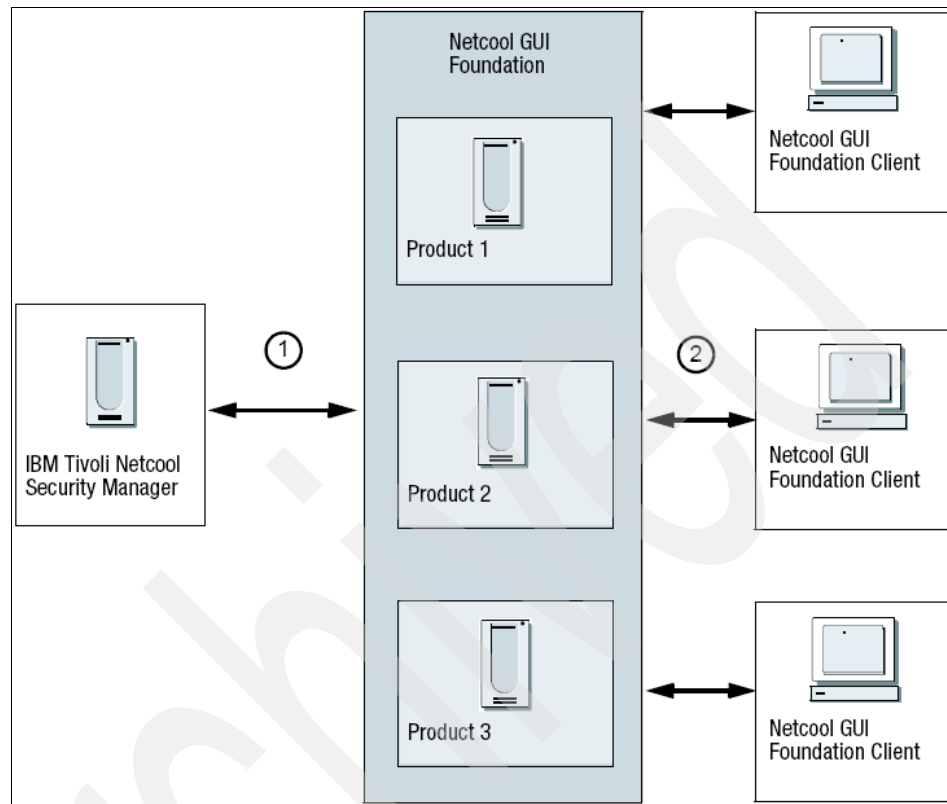


Figure 2-27 Netcool GUI Foundation in the IBM Tivoli Netcool Suite

Figure 2-27 shows that the different components interact in the following ways:

- ▶ The Netcool GUI Foundation uses Netcool Security Manager for user authentication and authorization.
- ▶ Different clients connect to the Netcool GUI Foundation. With the correct product roles, any of these users can access functionality from product 1, product 2, and product 3. In addition, any user with the correct administrative roles can access the administration functions in the NGF.

The Netcool GUI Foundation can be displayed in the following languages:

- ▶ Japanese (JP)
- ▶ Korean (KR)
- ▶ Simplified Chinese (zh-TW)
- ▶ Traditional Chinese (zh-CN, excluding GB 03018)

- ▶ French (FR)
- ▶ Spanish (SP)
- ▶ German (DE)
- ▶ Italian (IT)
- ▶ Portuguese Brazil (pt-BR)

**Note:** Localization support for products within the Netcool GUI Foundation might vary according to product and version. Check that the language that you want to display is supported in both the NGF and the products within it.

If you attempt to display an unsupported language in either the NGF or the products within it, inconsistencies might occur between the language displayed in the NGF and the language displayed in the products.

Accessibility features of the Netcool GUI Foundation are:

- ▶ Keyboard shortcuts
- ▶ The Go button
- ▶ Other accessibility features, alternative text

## 2.2.9 Typical Netcool/OMNIBus deployment

Single-tiered systems are the most basic Netcool/OMNIBus deployment. In single-tiered systems, all Netcool/OMNIBus clients connect to a virtual ObjectServer comprising a failover pair of physical ObjectServers providing collection, aggregation, and display functionality in a single tier. A bidirectional gateway is deployed for synchronization.

Process control agents may be deployed to manage the core processes, ensuring that these are started on system startup, and rapidly restarted in the event of unexpected shutdown.

After initial creation, the ObjectServer is managed through the administrator client where most customization can be completed in real-time without service interruption. The client may be deployed on any supported platform. A single instance of the client may be used to manage all ObjectServers and process control agents. You may also deploy multiple instances of the administrator.

It is not required that all components be deployed on the same hardware platform.

Figure 2-28 shows a typical single-tiered installation.

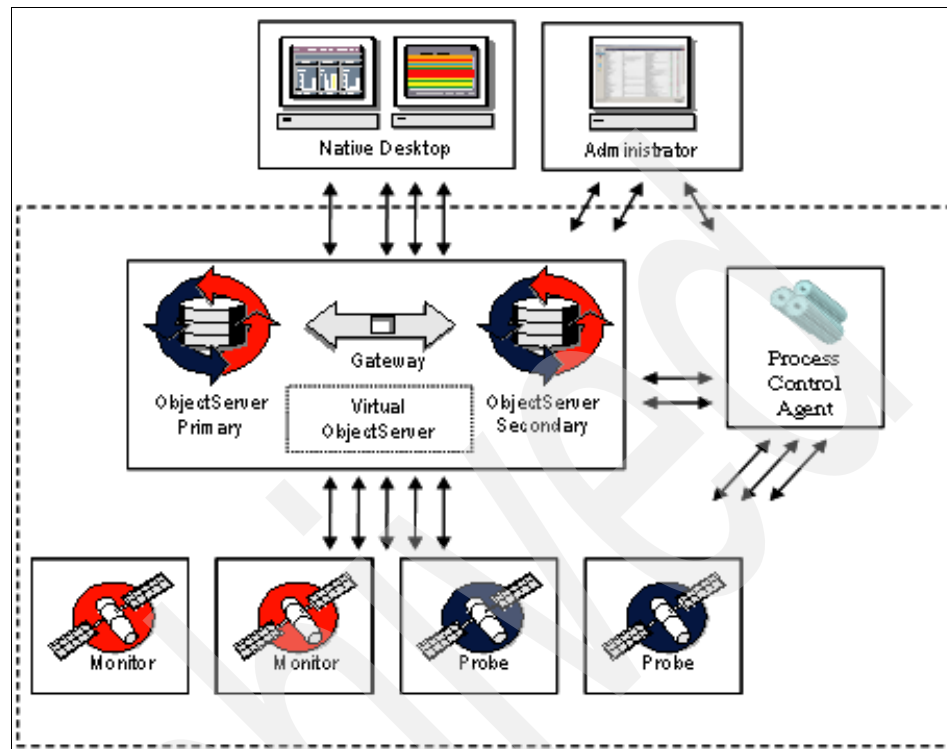


Figure 2-28 Typical Netcool/OMNibus environment

## 2.2.10 Two-tiered architecture

The distributed processing capabilities of Netcool/OMNibus accommodate high event throughput and large user communities. ObjectServers are deployed in tiers to separate the loads of distinct functional capabilities, thereby distributing the total processing load across multiple servers.

A two-tiered system is a combination of aggregation and display ObjectServers with supporting gateway configurations. In a two-tiered system, Native Desktop (and Netcool/Webtop) sessions read from display ObjectServers and write back to both aggregation and display-layer ObjectServers.

Probe clients connect to the aggregation system, as it also functions like a collection layer system. Netcool/Precision IP/TN and other Netcool suite components except Netcool/Webtop connect to the aggregation system.

A two-tier architecture is shown in Figure 2-29.

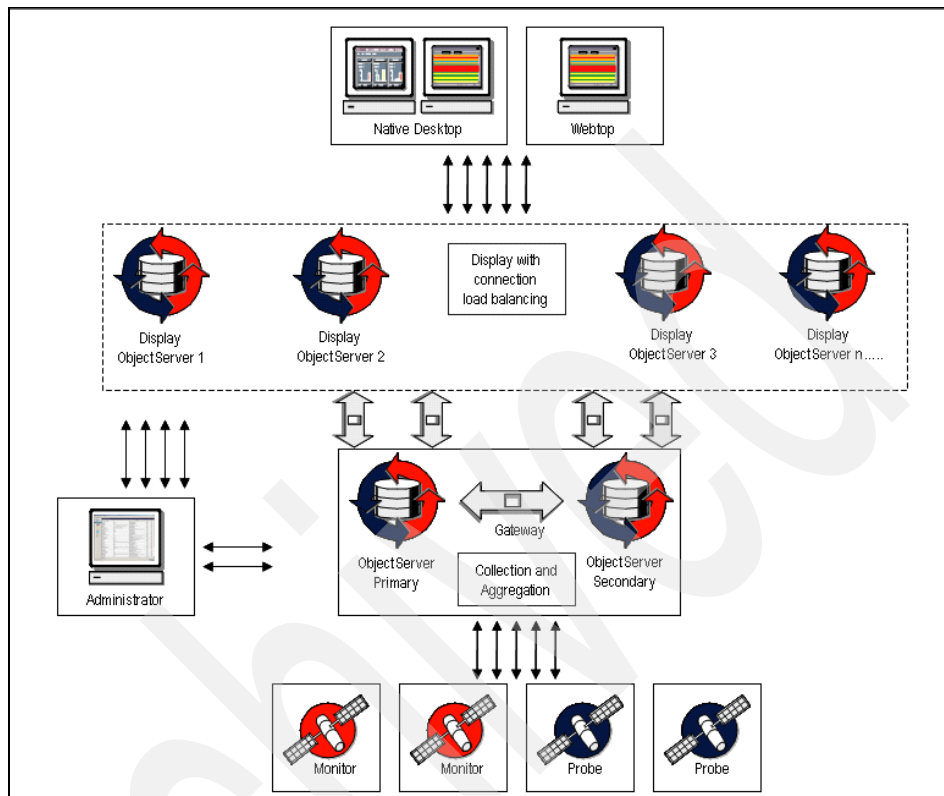


Figure 2-29 Two-tier architecture

### 2.2.11 Three-tiered architecture

A three-tiered system has distinct functions for the three layers of ObjectServers. All probes connect through a collection layer ObjectServer. All alarms are therefore processed first by an ObjectServer at the collection layer. The collection layer exists to buffer the aggregation system from alarm storms and to perform deduplication.

Since rows are deduplicated at the collection layer, the aggregation layer is more likely to scale than if hundreds of probes were directly connected. Put simply, the collection layer exists to process inserts/reinserts. At regularly timed intervals, the deduplicated events are forwarded to the aggregation system via the collection/aggregation routing gateway.

The display layer, just as with a two-tiered system, exists to process read requests from native desktop or Netcool/Webtop sessions.

Lastly, with the desktop read load and insert load removed from the aggregation layer system, the aggregation layer is free to house more data for longer periods of time. Moreover, the aggregation layer can support more complicated downstream integrations with ticketing systems and other Netcool suite components.

The three-tier architecture is shown in Figure 2-30.

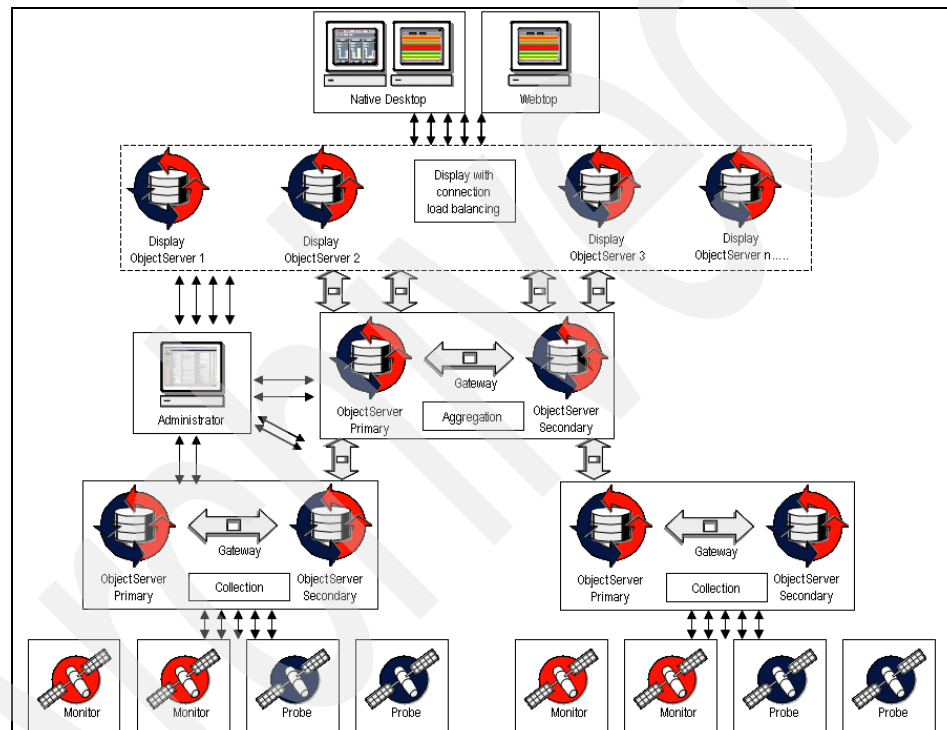


Figure 2-30 Three-tier architecture

**Note:** Large-scale Netcool/OMNIbus environments can be broken into two major categories:

- High event throughput installations
- High desktop load installations

## 2.2.12 Firewall considerations

In this section we discuss firewall considerations.

### Port usage

By default, when an ObjectServer starts, an available port number is chosen for the IDUC connection.

The communication direction between a Netcool probe and its configured Netcool ObjectServer is established by the probe in direction of the Netcool ObjectServer. Probes must use the host name and port combination that is specified in the interface file for the server name. For example, if the host name and port combination is “NCOMS fred 5101”, then the probe will only attempt to connect to the NCOMS instance running on fred on port 5101.

IDUC clients require two connections: first, the connection to the port defined in the interface file for the server. Second, the IDUC port of the ObjectServer, which is automatically allocated by the operating system by default. The second can be hardcoded by setting the Iduc.ListeningPort property. You need to specify the IDUC port when accessing an ObjectServer protected by a firewall.

IDUC is a management channel that notifies the clients as to which events have been inserted, deleted, or updated since the last refresh of that client's data. It is used only by native desktop and gateway client connections.

Refer to the “IBM Tivoli Netcool default port usage” on page 375 for a list of ports.

### Secure Sockets Layer

Netcool/OMNIbus supports the use of the Secure Sockets Layer (SSL) protocol for communications between Netcool/OMNIbus servers and clients. When a client initiates an SSL connection, the server presents the client with a certificate. By reading the certificate, the client can determine whether the server is a trusted source, and then accept (or reject) the connection.

The Netcool/OMNIbus components can also use an existing SSL tunnel environment between the components for their communications.

Netcool/OMNIbus uses SSL for ObjectServer authentication. The certificate is signed by a Certificate Authority (CA) (that is, a trusted party that guarantees the identity of the certificate and its creator). The certificate contains the identity of a server, the public key, and the digital signature of the certificate issuer.

Certificates serve two specific purposes:

- They provide authenticated proof to a client that the server that they connect to is owned by the company or individual who has installed the certificate.

- ▶ They contain the public key that the client uses to establish an encrypted connection to the server.

A set of guidelines is provided for configuring the server editor to use encrypted (SSL) connections, unencrypted (non-SSL) connections, or both. In the server editor, you can enable encrypted connections, unencrypted connections, or both.

## 2.2.13 Configuring hardware for performance

In this section we discuss configuring hardware for performance.

### Platform support

Netcool/OMNIbus may be deployed across mixed hardware environments.

### Hardware sizing

Hardware sizing for Netcool/OMNIbus is dependent on a number of variables, including event throughput, number of active events, number of users, and the complexity of their desktop requirements, as well as integrations with other Netcool and third-party components.

The predominant characteristic of Netcool/OMNIbus is its memory-resident database. While there is no need to access the disk, it still takes time to read through 100 MB of events out of memory (for example, as might happen with a `select * from the alerts table`).

Another key contributor to the ObjectServer performance is the CPU speed. In general, the faster the CPU, the better.

## 2.2.14 Netcool/OMNIbus rules: best practices for performance

There are several features of IBM Tivoli Netcool/OMNIbus and the OMNIbus probe language that can be utilized to handle high event rates in a large production environment. These techniques can be used in high traffic environments, as well as used simply to process event suppression and other event control situations:

- ▶ Using direct advantage of de-duplication to clear events
- ▶ Using the probe to send to tables other than `alerts.status`, or optionally to a table in a different ObjectServer
- ▶ Keeping alarms that must be consolidated out of `alerts.status`
- ▶ Using the load functions within the probe rules

Typically, the Generic Clear automation is used to match a resolution event to a problem event and clear the event in the event list. With more control over de-duplication in v7.x, you can use common identifiers for problem and resolution events, and properly update the problem event to indicate the circumstances of the event resolution. The benefits in using de-duplication for clearing are:

- ▶ Reduce Generic Clear overhead - The Generic Clear automation requires a good deal of processing because of two stages of gathering potentially large tables in memory.
- ▶ Instant Clearing - Clearing by deduplication happens instantly. Generic Clear runs on a timed cycle.
- ▶ Reduce overall number of events - The number of events is reduced by half for all cases implemented.





## Part 2

# Strategies

The second part of this book includes two chapters that focus on TEC environmental assessment and upgrade planning strategies:

- ▶ Chapter 3, “TEC environmental assessment and planning guidelines” on page 103
- ▶ Chapter 4, “Upgrade strategies” on page 147

Archived

## **TEC environmental assessment and planning guidelines**

This aim of this chapter is to take a holistic view of the managed environment and encourage the undertaking of an audit or inventory of the existing event flows in order to understand the information that we need to gather, and understand the considerations and constraints that will have an impact on the upgrade strategy.

We cover the subject in a logical flow starting from the event sources, or inputs to the server, through event processing, and then the outputs from the server. While this is a logical way to conduct the survey, it is not necessarily the same order in which we will recommend implementing the changes.

## 3.1 End-to-end event flow

Our aim is to highlight what needs to be collected to help the reader produce an ordered plan of tasks, as a planning aid to approach the upgrade strategy, which we cover in detail in Chapter 4, “Upgrade strategies” on page 147. We discuss:

- ▶ Event sources (adapters, applications, commands)
- ▶ Event manipulation (filtering and rules processing)
- ▶ Outputs to other systems (databases, trouble ticket systems)
- ▶ Actions via the console (manual and automated alerting)

Figure 3-1 shows the most common input sources and outputs that are typically implemented.

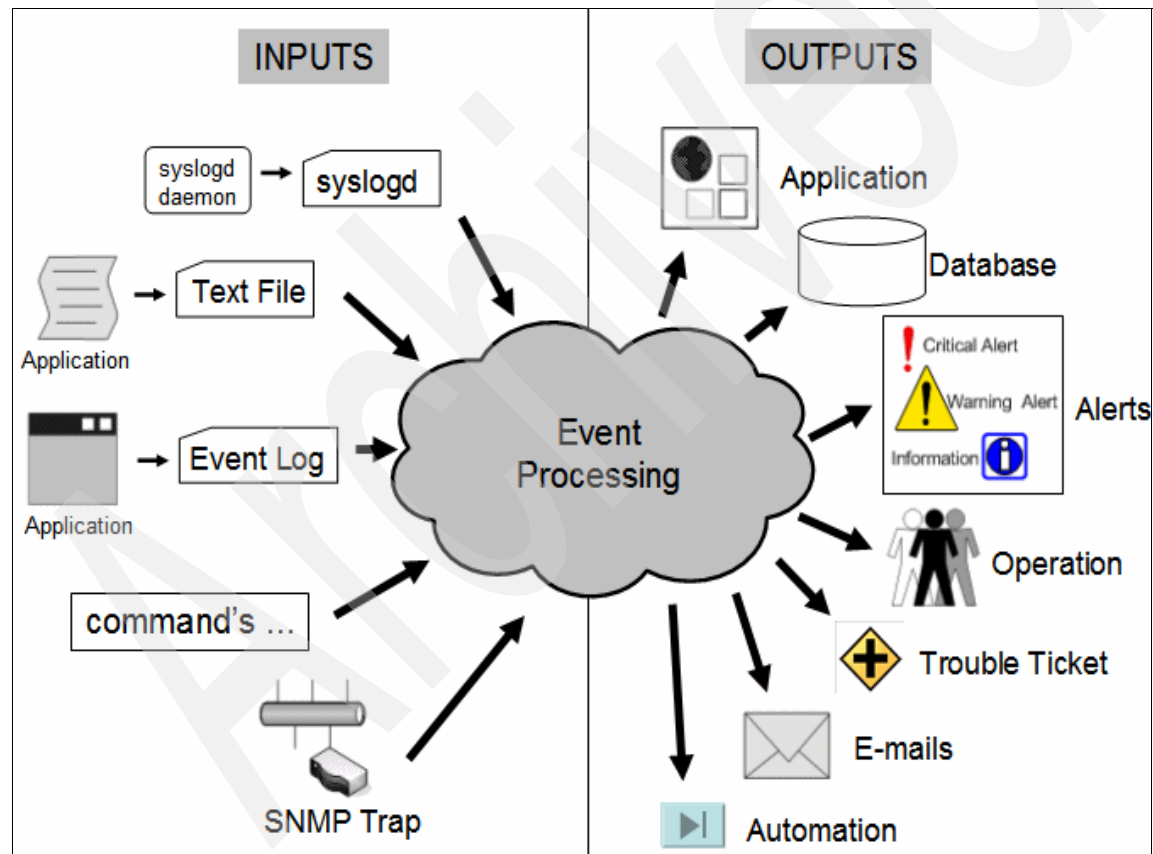


Figure 3-1 Event processing (inputs and outputs)

**Note:** While this will produce a list of tasks, it is not necessarily the intention or the recommendation to produce a like-for-like mapping from how the Tivoli Enterprise Console environment currently works to how Netcool/OMNIbus will work in the future.

Another way of putting this is that the objective of *what* you want to achieve may well be the same, but we suggest, and indeed recommend, that the approach to *how* you achieve this will in fact be different, to best exploit the new features of Netcool/OMNIbus.

From the evidence that we have so far from previous customer experiences we would normally expect the final list of activities to be shorter than the original list of tasks, due to the *out-of-the-box* features of Netcool/OMNIbus.

### **Are all your event management activities really necessary**

It may well be that the 'audit will produce a list of activities (be they filters, pattern matches, rules, or actions) that you now consider are:

- ▶ Obsolete, and can be dropped  
Or, in other words:
  - Why do we still send that alert to do that? Our operational procedures changed months ago.
  - Since our recent network or server upgrade we no longer need to worry about that constraint.
  - And so on.
- ▶ Poorly designed, or have just grown organically
  - Now we have the full view. We can see efficiencies that we can make.
  - Why do we do those two rule processes, when one would achieve the same?
- ▶ Replaced by a new capability of OMNIbus  
We no longer need rules to dup-detect or raise event severities.
- ▶ Still absolutely required in the same format  
This is our most business-critical system and we must retain the same logic.

At the end of the chapter we provide a checklist that has entries for each of the the following sections, and that provides a useful reference for the audit.

## 3.2 Event source hosts

This is a good opportunity to take stock of all the event sources that typically flow into the TEC server, and complete an audit or inventory of the systems that are monitored and the type of sources currently implemented.

Ideally, we should be able to identify and capture all the possible managed systems and event sources that may be out there in the managed environment, but in reality the challenge will be that the inventory may be incomplete or not fully documented. If you are in a position where there is a well documented and fully defined environment then this will speed up the following process, but the hints are intended to apply to all cases.

Most sites will have established methods for this already, so the following is list of suggestions that you can *pick and mix* from, as appropriate, rather than a list of instructions that you need to follow verbatim.

### 3.2.1 Tivoli framework commands

Tivoli framework-related commands can be used to discover the profiles currently deployed and the list of systems subscribing to those adapter profiles.

- Example of commands to list profile managers and ACP profiles:

- **wlookup -ar ProfileManager**

```
ACP_Adapter      1938676155.1.709#TMF_CCMS::ProfileManager#
ACP_EP  1938676155.1.706#TMF_CCMS::ProfileManager#
ACPdefault      1938676155.1.660#TMF_CCMS::ProfileManager#
```

- **wls -l @ProfileManager:ACP\_Adapter**

```
1938676155.1.708#ACP::Prof#      ACP_windows_profile
1938676155.1.714#ACP::Prof#      ACP_UNIX_logfile
```

- Example command to list subscribers to those profiles:

```
wgetsub @ProfileManager:ACP_UNIX_logfile
```

```
mondorf
bonn
```

- Command to dump out profile content details:

```
wlsac -l ACP_AIXlogfile
```

```
$ wlsac -l ACP_AIXLogfile
:::: record 0 ::::
# Tue Oct 9 13:17:05 CDT 2007
```

```
#
# tecad_logfile Configuration
#
TransportList=t1_
t1_Type=LCF
t1_Channels=c1_
c1_ServerLocation=@EventServer
EventMaxSize=4096
BufEvtPath=$TIVOLIHOME/tec/tecad_logfile@${AC_ID}.cache
PollInterval=30
Pre37Server=no
FQDomain=NO
LogSources=/tmp/applications
#
Filter:Class=Logfile_Base;
Filter:Class=Logfile_Sendmail;
Filter:Class=Amd_Unmounted;
Filter:Class=Amd_Mounted;
Filter:Class=TEC_Notice;msg=filter-out;
```

- General topology commands and scripts that can be downloaded from the IBM Tivoli support Web site, such as the following:
  - **epreport.pl**
  - **assess.pl**

These will help aid the planning, as they list the names and numbers of endpoints per managed node gateway and the general health and distribution of endpoints over the Tivoli region.

### 3.2.2 Non-framework commands

For event sources that have not been deployed by the framework, and that may have been distributed by software distribution or manually installed, other means will be needed, such as using Tivoli Configuration Manager, Tivoli Application Dependency Discovery Manager (TADDM), or user-defined scripts.

### 3.2.3 Other techniques

Other means can be deployed at the server side of the event flow to obtain the locations and types of the sources by the events themselves that have been received. While this is likely to be a quick and useful technique, it can only show those that actually get to the TEC server, and so may miss out on infrequently alerting systems that have events filtered out by a state correlation engine at a gateway, for example.

Given that cautionary note, techniques could be to use searches in wtdumpri, use database sql, or use information taken directly from a Tivoli Data Warehouse or other reporting system.

### 3.2.4 Safety net

Having said all the above, we recommend that planning for the upgrade includes a period of time when there is a *safety net* mechanism so that events that come from unexpected sources are not overlooked by the newly employed ObjectServer and console. We describe this upgrade planning in detail in Chapter 4, “Upgrade strategies” on page 147.

Unlike TEC, OMNibus does not reject events that it cannot parse (that is, if it has no baroc file). The probe rules have a catch all mechanism that sends a best effort parse of an unknown event so that it may be investigated.

## 3.3 TEC source types

Table 3-1 shows the existing TEC event sources, the possible routes to get to the ObjectServer, and the type of integration possible. We go through these event sources and explore the options available.

Table 3-1 Event source routes to the ObjectServer

Event source	Via TEC server	Via EIF probe	Via Netcool probe	Synchronization
DM 3.7	Yes <sup>a</sup>	No	No	No
ITM 5.1.x	Yes	No	No	No
ITM 6.1	Yes	Yes	No	One-way
ITM 6.2	Yes	Yes	No	Yes
NetView 7.1.4/5	Yes	Yes	Nv7	Partial
windows adapter	Yes	Yes	NT eventlog	No
unix syslogd	Yes	Yes	Syslogd	No
logfile adapter	Yes	Yes	Logfile/syslog	No
snmp adapter	Yes	No	Snmp	No
as400 adapter	Yes	Yes	No	No
wpostmsg	Yes	No	No	No



Event source	Via TEC server	Via EIF probe	Via Netcool probe	Synchronization
postemsg	Yes	Yes	No	No
wpostzmsg	Yes	No	No	No
postzmsg	Yes	Yes	No <sup>b</sup>	No

- a. The announced IBM Tivoli end of support date for Distributed Monitoring 3.7 was April 2008. For the purposes of this book we assume that plans have been made to migrate monitoring solutions on to ITM 5.x or 6.x for supportability. Sending DM3.7 events was not tested, but in theory the compatibility will be the same as the ITM 5.1.x integration.
- b. Integration could be achieved via a logfile probe if the postzmsg logic is changed to write to a logfile instead of sending a message.

For example:

- ▶ If you are using ITM 5.1.2, integration is possible, but you need to retain the TEC server until ITM is upgraded to 6.1 or 6.2.
- ▶ If you have as400 systems, you can use the TEC server or EIF probe, but not a native Netcool probe.
- ▶ If you need to collect SNMP traps, these can go via the TEC server or use the Netcool SNMP probe.
- ▶ NetView and ITM also have other options on uni-directional or bi-directional event synchronization (see the next two sections).

### 3.3.1 IBM Tivoli Monitoring event sources

With ITM event sources you have a choice regarding the order in which you upgrade ITM compared to diverting event sources from TEC to OMNIBus via the EIF probe. In Table 3-1 on page 108 you can see that this is dependent on the version of ITM.

#### ITM 5.1.x

Since the only available integration is via TEC, ITM must be upgraded before migrating the event source to OMNIBus. We recommend that this is to ITM 6.2.

#### ITM 6.1

You have two choices here, and they should be based on the level of ITM-to-TEC synchronization that you currently perform. If this is simply a one-way flow of events into TEC, then you can divert to the EIF probe straightaway. If, however, you have two-way console synchronization, then be aware that this is not currently provided with the EIF probe and ITM 6.1, so you would be losing functionality. We advise against upgrading to ITM 6.2 first.

## ITM 6.2

Here the functionality is identical, so that two-way functionality is provided with either TEC or the EIF probe, so you can immediately divert the event source. Moving to ITM 6.2 will also give you more integration options, as it allows parallel event flows into both TEC and OMNIBus EIF event destinations. This could be useful for parallel running in an integrated solution or a testing phase. For details on configuring this, refer to 7.3, “Integration between Netcool/OMNIBus and IBM Tivoli Monitoring” on page 290.

**Note:** The best practice for integration with OMNIBus is to upgrade to ITM 6.2 as soon as possible.

### 3.3.2 Omegamon agent sources

We assume that any events coming from an Omegamon (formerly Candle®) agent source will have gone via an ITM 6.1 or 6.2 TEMS first, so they can be considered in the same way as for the ITM 6.1/6.2 sources.

### 3.3.3 NetView event sources

Many existing Tivoli environments will incorporate a NetView server, which will generate network and managed node availability events.

### 3.3.4 NetView forwarding to OMNIBus options

There are currently a few choices here for the way forward. We have outlined the four main ones.

#### **Forwarding via the EIF Probe**

Below we discuss the advantages and disadvantages of forwarding via the EIF probe.

#### **Advantages**

NetView event forwarding can easily be diverted to send to the Netcool EIF probe. The simplest option is to make a single configuration change on the NetView environment to change the configuration file (tecint.conf on UNIX, tecad\_nv6k on Windows) to point to the port of the EIF probe rather than the reception port of the TEC server. NetView events will appear in OMNIBus with a meaningful text, with no other configuration required.

## **Disadvantages**

This will just achieve *one-way* integration. If that is all you are doing currently then that is fine. If, however, there is two-way integration and TEC and NetView event consoles are currently kept synchronized by existing TEC rules, then we need to either keep the TEC server in the event flow or consider some other options, as this synchronization is not currently provided by the NetView to OMNIBus integration package.

Secondly, the EIF probe option does not provide the same functionality when compared to the correlation rules provided by TEC in `netview.rls`, although the default automations in the ObjectServer go some way (for example, if NetView sends a `NodeDown`, then a `NodeUp` for the same host name the former will be *closed* (set to severity 0 and cleared)). This works (as the events are from the same *class* or alert group) in the same way that the `netview.rls` does.

It is worth noting that the reason for this rule in TEC is that a large node or router going down could potentially send large numbers of *interface down* messages and clutter or even fill the console. By default OMNIBus will handle this much better, and de-duplication means only one interface down event, with a tally field of the number of duplicate events.

However, the `netview.rls` also has several other correlation rules, such as an `InterfaceDown` followed by a `NodeDown` or `RouterDown` will close the `InterfaceDown` event (as it was an effect, not the cause). In the EIF probe, or OMNIBus, this will not be matched by default, as they are from different TEC classes (alert groups), so the OMNIBus attribute field will not match. This could be achieved by changing the appropriate settings of the generic fields in the rules, or by adding a trigger rule, as described in the note below.

**Note:** If you do wish to replicate the same behavior, this needs to be handled by an automation, or trigger. In 7.2.6, “Automatic event management customization” on page 284, we provide a sample temporal trigger that can be implemented at the ObjectServer to illustrate how to achieve this correlation. Also note that this is not the entire functionality of the `netview.rls`, just a sample of the most useful rule, *that* can be adapted for other required correlation rules.

You will still need to address upgrading NetView to the strategic IBM Tivoli Network Manager product before end of product support of NetView (currently the same time scale as TEC, in 2012.)

## **Probe for Tivoli NetView**

In this section we discuss the advantages and disadvantages of the probe for Tivoli NetView.

### ***Advantages***

This may be a useful option for a NetView customer who is not a TEC user, and who wants to send snmptraps directly to OMNIBus, and so would not need the EIF probe for any other source. This acquires event data directly from the trapd daemon on a NetView system and captures the raw traps to forward to OMNIBus direct. There is a rules file that can be supplemented with additional Netcool Knowledge Library entries (based on the trap's Enterprise OID information).

### ***Disadvantages***

As with the EIF probe option above, this probe only provides one-way integration, and does not include the correlation rules of TEC's netview.rls. The probe must be installed on the NetView server, and this is additional installation effort, configuration, and overhead, compared to utilizing an EIF probe. It is also the case that the supplied rules require additional knowledge library installation and configuration. Otherwise, the events appear on the OMNIBus event list as unfriendly OIDs, and even simple NodeUp/NodeDown correlation is not performed.

So if you have an existing EIF probe configured, we recommend using that and enhancing the rules as described previously.

You will still need to address upgrading NetView to the strategic IBM Tivoli Network Manager product before end of product support of NetView.

## **Forward via the existing TEC server to OMNIBus**

In this section we discuss the advantages and disadvantages of forwarding via the existing TEC server to OMNIBus.

### ***Advantages***

No configuration changes are required at the NetView server. All existing synchronization with TEC is maintained, and assuming that the TEC/OMNIBus two-way synchronization is implemented via the EIF gateway, changes should be reflected in OMNIBus too. Existing correlation is also performed by the TEC netview.rls. This is the least-effort option. It maybe a valid approach on the path to the next upgrade option.

### ***Disadvantages***

This is only a gradual step forward in upgrading. At some point this may mean that the TEC server is remaining purely to handle network events from NetView. It should be seen as an interim stage only.

You will still need to address upgrading NetView to the strategic IBM Tivoli Network Manager product before end-of-product support of NetView (currently the same time scale as TEC, in 2012.)

## Upgrade NetView to IBM Tivoli Network Manager Entry Edition

In this section we discuss upgrading NetView to IBM Tivoli Network Manager Entry Edition.

### **Advantages**

A new option is available to NetView customers with the product IBM Tivoli Network Manager Entry Edition Version 3.7 (released September 2007). This is the strategic upgrade path for NetView customers and is based on the Netcool Precision/IP technology and therefore allows the network event sources to be greatly enhanced by new capabilities and has very tight integration with OMNIBus, and in fact includes a limited use, embedded ObjectServer.

We assume in this book that this product is not deployed in the existing TEC environment, but it is the recommended strategic way forward to integrate network management events with OMNIBus.

For more information about this topic see *Migrating to Netcool/Precision for IP Networks - Best practices for migrating from IBM Tivoli NetView*, SG24-7375.

### **Disadvantages**

This is obviously an additional upgrade *project* that would be required, and it may or may not make sense to tackle it at the same time as the TEC to OMNIBus upgrade process. This will be a decision based on available resources and business needs, as, technically, upgrading in either order is possible.

## 3.3.5 Windows event log messages

The recommended option to collect messages from the Windows event log in the Netcool product suite is the probe for Windows NT® event log, which performs an almost identical role to the equivalent TEC event log adapter “tecad\_win”. Previous options, such as using Netcool System Service Monitors, which have a different functionality, are now being replaced by the ITM offerings.

The Windows NT event log probe is able to perform the same granularity as pre-filtering by the TEC adapter, in that it can selectively filter in or out security, system, or application messages. Finer granularity is also possible by using rules at the probe level to perform filtering of specific messages or message IDs. Full configuration details are available in 7.5, “Migrating the TEC Windows event log adapter” on page 322.

An interim option for the upgrade strategy here is to route the existing TEC adapter profiles to send messages via the Netcool EIF probe, and on to OMNIBus.

### 3.3.6 UNIX and Linux syslog messages

The equivalent Netcool probe to the TEC logfile adapter reading UNIX and linux syslog messages is the syslog probe. There is a similar offering called the syslogd probe, which provides the syslog daemon functionality as well. Since most production quality operating systems will already have a syslog daemon active and configured, we chose to use the syslog probe, which most closely matches the TEC offering.

The syslog probe works in exactly the same way as the TEC adapter in reading from a named pipe that the operating system's syslog daemon has configured to write specific messages to. In fact, a simple upgrade step is to copy the syntax from the TEC adapter entry and have a parallel output, one read by the adapter and one by the probe. Like all probes, the pattern matching is performed in the probe rules file.

Like the Windows adapter, an interim option for the upgrade strategy here is to route the existing TEC adapter profiles to send messages to the Netcool EIF probe and then on to OMNIBus.

### 3.3.7 Logfile messages

Flat file application log messages can be handled by either the syslog probe mentioned above, which can monitor a flat file for messages, or the generic logfile probe (GLF). Both were installed, tested, and utilized for different scenarios in the lab and used in Chapter 6, "Event processing" on page 187.

As above, an interim option for the upgrade strategy here is to route the existing TEC adapter profiles to send messages to OMNIBus via the Netcool EIF probe.

### 3.3.8 SNMP traps

The TEC SNMP adapter functionality can be replaced one for one with the Netcool SNMP probe. They both require a degree of configuration in converting MIB and OID definitions into something readable and useful, but the functionality is equivalent. The number of vendors MIBs support are enhanced by making use of the Netcool Knowledge Library (NcKL) rules files, which are regularly updated.

### 3.3.9 AS400 messages

The only support for the upgrade strategy here is to route the existing TEC as400 adapter profiles to send messages to the Netcool EIF probe and then on to

OMNibus. However, this assumes that any specific rules logic that you may have developed for as400 alerts will need to be migrated to OMNibus too.

### 3.3.10 Command line sources (w)postemsg and (w)postzmsg

Many customers utilize these command-line binaries in scripts and Tivoli Framework tasks to send an event to the TEC server, for a wide variety of reasons, for example, to update the status of an application, provide a heartbeat mechanism, or alert to a new event to operations.

This is likely to be quite a challenging task to discover and round up all the possible places where one of these commands could be embedded. It is also likely to be a place where the 'ServerLocation' (that is, @EventServer or the -S hostname flag) is used and will need to be re configured.

Since there is no corresponding binary in Netcool, the best option is likely to be to retain the binary (and we recommend using the postzmsg version, as it supports a **-P port** option, so that you do not need to specify an external configuration file) and redirect to the Netcool EIF probe rather than the TEC event server.

Similarly, if the -f flag is used, to read an adapter configuration file, the configuration file will almost certainly need modification to direct events to a different server and port (that is those of the Netcool EIF probe).

**Note:** Care must be taken if you decide to leave this adapter configuration file in place, so it is not deleted should an ACP adapter profile residing on the same system be deleted at a later date.

Another possibility is to use postzmsg with the configuration setting of testmode to write out the messages to a local logfile, which a Netcool logfile probe could then monitor for updates.

Finally, the command could in theory be replaced with the Netcool **nco\_sql** command (or a wrapper script that contains this command), which makes sql updates in the OMNibus ObjectServer itself. This is only likely to be feasible or desirable if the ObjectServer code is already installed locally on that system.

### 3.3.11 Custom EIF applications

In theory, if a custom application uses standard EIF communications to send to TEC at the moment, then the EIF probe should be capable of receiving them

instead. We have not tested any of these for this book, and any of these that do exist should be thoroughly tested by the customer.

### **3.3.12 Tivoli Business Systems Manager (TBSM 3.1)**

If you currently integrate TBSM 3.1 with TEC, then this is going to be a constraint or at least a major consideration on the upgrade strategy. While it may be possible to have events flowing from TBSM 3.1 to the EIF probe, in the same way in which they do to TEC, the integration and synchronization is not two-way with OMNibus. Moreover, most customers forward events in the opposite direction, from TEC to TBSM 3.1. In these scenarios the TEC server will need to remain in the event flow picture until TBSM is upgraded to 4.1.

The Tivoli Business System Manager 4.1 (TBSM 4.1) release is integrated tightly into the Netcool product suite and provides full integration with OMNibus. However, to integrate TBSM 4.1 with TEC, the event flow would have to go via OMNibus first.

## **3.4 Other planning considerations**

Now we have a good idea of the number of systems involved and the application-related integration constraints from the previous section. We are starting to get an idea of those event sources that can be tackled immediately. Similarly, those may need to be deferred until after an upgrade, or will need upgrading prior to an upgrade to OMNibus. This will help the planner build up a time line of dependencies.

To add to this picture of application-related constraints are other site-specific constraints, such as the network topology, the geography, bandwidth constraints, firewalls, and proxies. We start with some deployment considerations.

### **3.4.1 Deployment considerations**

Deploying Netcool components in a distributed environment is more of a challenge than deploying Tivoli framework components. There is no equivalent remote deployment mechanism as with a framework endpoint. The size of the package to install and the configuration effort of a probe is greater than that of an adapter, and there are also limitations on remote process execution and cross-platform functionality, which we discuss later. On the positive side, with OMNibus 7.2, the previous requirement for a license server and licenses has now been removed.



**Note:** Currently, all Netcool products have to be installed via the Netcool installer, which requires the full OMNibus codeset to be downloaded to the target in order to install the Netcool libraries and middleware. This includes the remote execution mechanism process control, which would also mean a full distribution to any target that requires a remote command execution.

Once the OMNibus codeset is installed, the relevant probe software can be downloaded and added. This can mean a relatively large footprint on the remote server of 100 Mb for Windows and 150–175 Mb for Linux/UNIX, depending on the operating system platform. This will require planning and scheduling for the distribution of the code, as well as ensuring that sufficient space is available at the target system.

It is not currently possible to bundle up a smaller software distribution of the binaries required for the probe (which is typically only 300 kb), but this is a consideration for future Netcool development. However, remember that a probe is not necessarily located on every target system and, depending on the type of probe, it may be listening or polling several targets.

There are a few options available to handle these challenges and differences (and to cope with networking and firewall considerations) aimed at minimizing the deployed code, the network connections, and ports required.

- Configure several local *targets* to one probe.

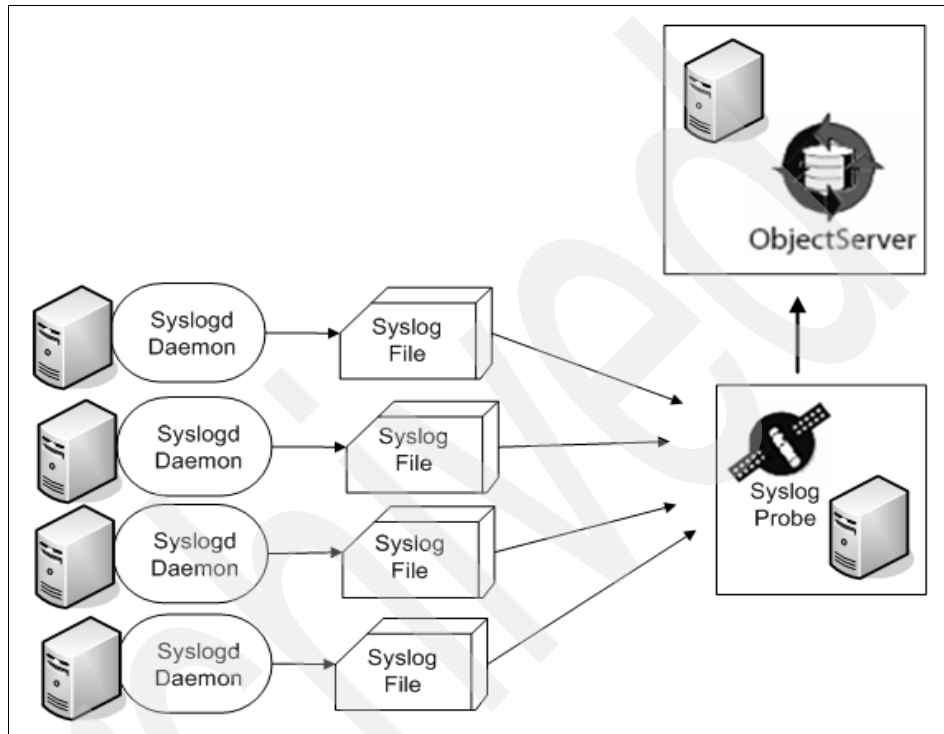


Figure 3-2 Configure several local targets to one probe

As can be seen in Figure 3-2, it is not always necessary to have one probe per target system (for example, the syslog process can be configured to send to a remote system's syslog, so one system can be a syslog host and have the probe on that). This can scale very well, depending on the level of detail required by the syslog messages. We are aware of one customer who has hundreds of systems using one syslog host. It is likely in order to achieve these numbers that the granularity of syslog messages will need to be reduced from the default set that the TEC adapter syslog pipe is configured to listen to.

**Note:** This target-probe communication is via UDP, whereas the rest of the communications from the probe back to the ObjectServer are TCP. For this reason we usually recommend locating the probe close to the target. Also, some probes require installation on the same physical system.

- Concentrate different probes on one system.

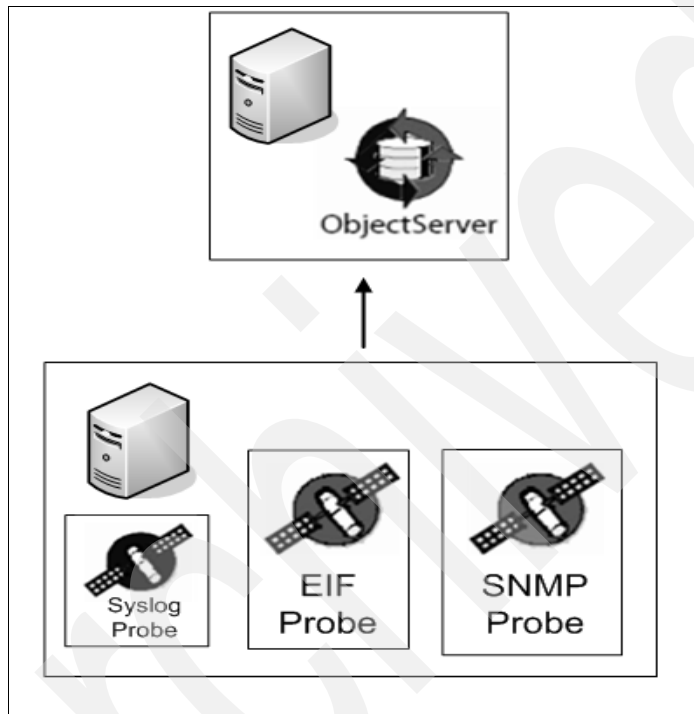


Figure 3-3 Concentrate probes on one system

If several different types of probe are required for a group of servers in one location, all the probes can be physically located on one system, and then only one system needs to communicate back to the ObjectServer, and only one system will need to open up the ports.

- Install a probe consolidation server or proxy server.

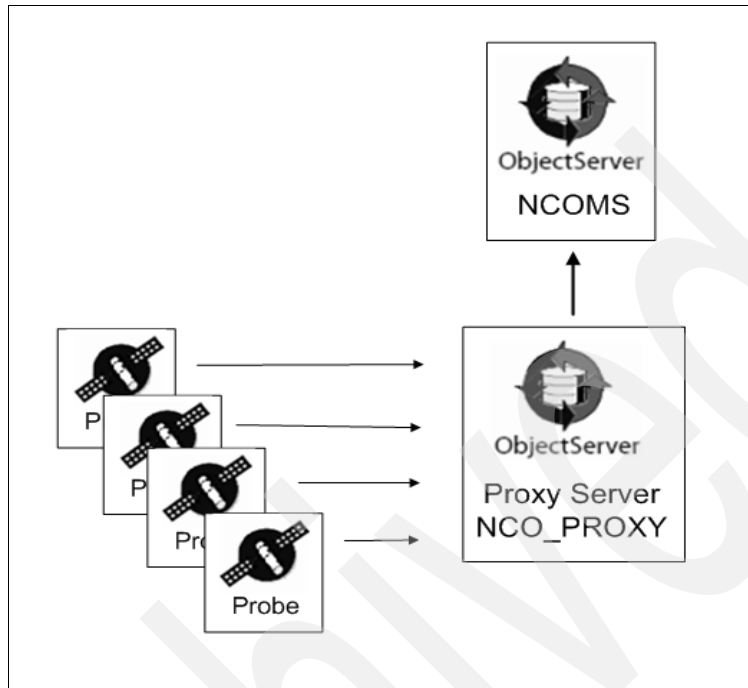


Figure 3-4 Install a probe consolidation server or proxy server

As seen in Figure 3-4, this is a server that allows concentration of communications from several probes into one, for onward forwarding of events to the ObjectServer. This is aimed more at limiting networking traffic and coping with firewall constraints than with deployment issues.

- Install a collection or concentration layer ObjectServer remotely.  
This is similar to the last option but involves putting an ObjectServer on the remote site or the far side of a firewall. As this requires more overhead and administration than installing a proxy server, this is a more complex scenario that would only be recommended if the earlier options are not possible.

### 3.4.2 Scoping volumes and throughput of events

This discussion shows that calculating the exact number of probes required, in relation to the numbers of TEC adapters currently deployed, is not an easy or exact science.

The other major factor is the performance, or throughput, of events that can be handled per probe, and this will be very site specific depending on the type of probes deployed, and the quality and efficiency of the rule programming employed on the probe.

Quoting any hard facts here needs to consider the above disclaimers, but the numbers are likely to be in the hundreds of events per second range.

If the current volumes of events in the TEC environment have been measured or are known, then that should help with the planning, too, but, as mentioned previously, there is not necessarily a like for like comparison.

### 3.4.3 Coping with event storms

In a large TEC environment there are often procedures put in place to prevent event storms from swamping and possibly taking down the TEC server. Most of the configuration work to do this is done at the TEC gateway component. In the Netcool environment the networking consideration options mentioned above will have an impact on where this could be achieved, and provide some tactics to avoid event storms.

However, as we saw in the architecture chapter, the main messages are that this is far less likely to be an issue due to the following two factors:

- ▶ The OMNIBus ObjectServer is designed to handle far more events per second than TEC.
- ▶ If properly configured, the de-duplication rules at the probe level should prevent this from occurring in the first place.

For detailed performance statistics see the *Netcool/OMNIBus v7.0/v7.1 Performance & Scalability* white paper.

### 3.4.4 TCP/IP port usage

Another TCP/IP networking difference is that by default a TEC adapter communicates with a temporary or connection\_less mode to the TEC gateway, which means that a new connection is needed for each event. This can optionally be changed to a permanent connection\_oriented mode, which maintains the link between adapter and gateway until taken down. By default, TEC gateways will communicate to the TEC server by a connection\_oriented mode.

The behavior of the OMNIBus probe to ObjectServer is to use a persistent communication, like connection\_oriented. This can have implications both in the number of ports and sockets in use, and also for firewall considerations (for

example, those that are configured to time out a connection if it has been idle for a number of minutes). Most probes have a heartbeat or *probe watch* back to the ObjectServer that serves to keep this link active.

A list of the default ports used by Netcool components can be found in “IBM Tivoli Netcool default port usage” on page 375.

### 3.4.5 EIF probe considerations

One piece of the architecture to consider and monitor in an integrated TEC/Netcool scenario is the EIF probe component. Previous tests in the *TEC and OMNibus Integration Best Practices* white paper have highlighted that this may be a bottleneck on performance with figures quoted between 50–80 events/second, dependent on operating system and hardware configurations.

In our lab tests we found that both an EIF probe on Red Hat 4.0 on a VMware server and one on AIX 5.3 on a P550 LPAR were able to handle over 1,100 events forwarded from a TEC server in approximately 75 seconds. We used the replay mechanism of `sendEvents.pl`, so maybe our observed rate of 16 events/second was due to real events being used. Another possibility is that that rate was as fast as the TEC server could process them and forward them on to the probe.

To achieve a good throughput we also recommend ensuring the connection is kept maintained (`connection_oriented`) between the TEC and the EIF probe. This will avoid the overhead of setting up and dropping a connection (`connection_less`) and perform much better.

We recommend configuring caching and increasing the default cache size for resilience when forwarding, as it would cause the forwarding TEC server to stop processing eventually if the EIF probe was down for any length of time.

So even though we did not find a bottleneck, this should still be considered when implementing the scenarios in the following chapters. It may be that more than one EIF probe is required on different servers depending on volumes. For example, good practices could be to only handle and migrate one event source type at one time so that this can be measured and controlled, or to implement one EIF probe per event source type (that is, one for TEC events, one for NetView, one for ITM).

### 3.4.6 Adapter configuration files and gateway configuration files

Routing events in the current TEC environment will help with the planning of routes when moving to using the EIF probe or the other Netcool probes. The

gateway configuration can be set to route to a second destination for resilience as follows.

This is how it works now in a TEC 3.9 EIF configuration file. These are the values specified on the adapter (that is the endpoint or the non-framework source):

- ▶ **ServerLocation=@EventServer**  
On receipt of the event, the tec\_gateway will first try to send to the TEC server using TME transport. If the TEC server is down, the tec\_gateway will send to the ServerLocation specified in the tec\_gateway.conf file.
- ▶ **ServerLocation=hostname or IP address (and optionally ServerPort=<port#>)**  
The tec\_gateway will first try to send to the IP address (and <port#> port) specified by the adapter using non-TME transport. If unsuccessful, the tec\_gateway will send to the ServerLocation specified in the tec\_gateway.conf file.
- ▶ **No value specified for ServerLocation at the adapter**  
The tec\_gateway will send to the ServerLocation specified in the tec\_gateway.conf file.

One tactical step on the upgrade path could be to modify the server and port locations to route events to the EIF probe, or even create a second adapter profile pointing to the EIF probe so that there is a parallel stream of events. If you do this, then keep the above facts in mind.

As described earlier, the EIF probe can also have peer-peer failover configured for resilience.

## 3.5 Distributed event processing

Before events are processed centrally in a TEC environment the events are formatted at the adapter or source level. Additionally, filtering out of unwanted events can be achieved at both the adapter and at the TEC gateway via the SCE.

### 3.5.1 Formatting

Part of the audit will be to gather the information about what formatting has been set up in the format (\*.fmt) files in each of the ACP profiles deployed at the TEC adapter, and the underlying logic behind this. In OMNibus, this functionality is handled by the probe rules file, and uses a very similar regular expression pattern-matching process. The full set of regular expressions is available in the

appendix of the Netcool Administration guide. In our testing in the lab, the out-of-the-box solutions handled all the events that we captured from the syslog and flat log files in a very similar way. So the main focus here would be any complex customized format statements.

Figure 3-5 and Figure 3-6 on page 125 compare snippets from the format statements of the adapter and the probe.

```
FORMAT Logfile_Base
%t %s %s*
hostname $2
fqhostname DEFAULT
date $1
origin DEFAULT
msg $3
END
FORMAT Logfile_automountd FOLLOWS Logfile_Base
%t %s automountd[%s]: %s*
pid $3
sub_source automountd
msg $4
END
```

*Figure 3-5 Sample TEC format statements*



```

@Summary = $Details
@Node = $Token4
@Severity = 1
@Agent = $Token5
if(regmatch($Token5, "automountd.*"))

    $agent = "automountd"

    case "automountd":
        if(regmatch($Details, "^server ([^ ]+) not responding"))
        {
            @Severity = 4
            @Type = 1
            @AlertKey = extract($Details, "server ([^ ]+) not
responding")
        }
        {
            @Identifier = "@" + $Token4 + " -> " + $Details
        }
    }
}

```

Figure 3-6 Sample OMNibus probe rules format statements

### 3.5.2 Filtering

The TEC adapter configuration (\*.conf) file in the ACP profile will contain several filter and pre-filter statements that filter out certain unwanted classes or types of events from log files and system error logs.

This can be handled at either the OMNibus probe level or at the ObjectServer. The syslog and windows NT event log probe rules provide many typical filters based on class-like names, out of the box, but there may not be an exact correlation with customer-configured setups. This is likely to be an area where close inspection of the configuration files is required (as detailed earlier, the **wlsac** command will be useful), and there are many template probe rules that can be adapted for use quite easily.

Examples are also provided in 3.8.3, “Filtering out events with specific content” on page 131.

### 3.5.3 State correlation engine (SCE) processing

As we saw in the architecture section, the SCE can be deployed to filter and collect events to reduce the volume of events in a TEC environment dramatically.

The use of the SCE is not particularly widespread, but the most commonly deployed XML rules are the duplicate rules and the collector rules. For this audit you should examine and note down the logic employed in these rules. This file is typically located at the gateway, but can be deployed on a endpoint, so that should be considered, too. The file, by default, is called tecroot.xml.

In the Netcool architecture there is no equivalent middle tier, so we cannot do a simple transformation here. Much of the functionality is provided out of the box by the de-duplication automation. Additional sophistication can be deployed at the probe level, and there are several examples of this in Chapter 6, “Event processing” on page 187. Alternatively, this can be achieved by creating an ObjectServer trigger, as we see in the next section.

## 3.6 BAROC file definitions

Just before we consider event processing currently performed by the TEC rule engine, we need to consider the class definition of the events as defined by the TEC BAROC files.

The audit should also contain a list of all the BAROC files defined and loaded in the customer’s active rulebase, and indeed potential rulebases. There are a few ways in which this can be achieved via command line listing:

- ▶ **wlscurrb** - displays the current rulebase
- ▶ **wrb -lsrb -path <rulebasename>** - shows the path
- ▶ **cat .load\_classes** from the TEC\_CLASSES directory of the rulebase

Again, while the recommended approach here is to keep in mind the entire picture rather than replicate one for one, there is a Netcool Java utility provided located in the \$OMNIHOME/bin directory called nco\_baroc2sql that will make this task much simpler.

What this will do is create sql insert statements to create a class hierarchy in the OMNIbus ObjectServer database. This has two benefits: to allow automations and triggers to be able to be based on a hierarchical parent-child class level, as rules do now in TEC; and also to provide a meaningful name rather than a number in the class field. This can also then let OMNIbus desktop tools take actions on those meaningful class names. We show an example of this in 6.4, “Support of TEC class hierarchy” on page 247.

**Note:** It is not essential to insert these statements for the EIF probe and ObjectServer to process and understand events sent via the EIF probe (that is, you will not get the equivalent of a *parsing failed* message if you do not do this).

To use the utility, do the above commands to get the current rulebase and active classes. You will need to tar up the TEC\_CLASSES directory and move them to the system with the OMNIBus ObjectServer. Then running the following from the same directory as the BAROCs files will create a text file of the classes:

```
$OMNIHOME/bin/nco_baroc2sql -baroc .load_classes -sql inserts.txt
```

Then after reviewing the inserts.txt file, to load these into the ObjectServer run:

```
$OMNIHOME/bin/nco_sql < inserts.txt
```

### 3.7 Central event processing (TEC rules)

This section explains how to manually analyze the TEC rulebase. As mentioned earlier, when considering transforming the rules, there are distinct advantages to taking a holistic approach, rather than attempting some form of one-to-one mapping. Therefore, a utility would not be appropriate, and we would need to use a manual process to go through the TEC rules in the current rulebase, analyzing the common event processing tasks or rule types.

Once we have this we can transform this specification to the OMNIBus environment, which will include moving the logic to both the ObjectServer and to the probes, to create a new rule specification.

Again, there are command-line techniques using file listings and the **wrb** command:

- ▶ **wlscurrb** - displays the current rulebase
- ▶ **wrb -lsrb -path <rulebasename>** - shows the path

The rules are held in the TEC\_RULES directory, and the list of active rules in the rule\_sets\_EventServer file. Of course, it is quite possible that a customer may have a number of rulebases in production, or development that should be included.

### 3.7.1 Frequently used rules

If you have ITM 6.1 or 6.2 you can utilize the information from the TEC health and performance agent, which will show you the most commonly used rules in your TEC environment. This might be a useful indicator of the priority of the rules that you should examine for migration and also an idea of those that you can really best leave behind.

### 3.7.2 Typical rule types

The TEC ruleset reference guide refers to five types of rules:

- ▶ Simple rules - Do something simple when an event comes in.
- ▶ Plain rule - Do something when an event arrives. Can use predicates.
- ▶ Change rule - Change event details. Take actions dependent on an operator.
- ▶ Timer rule - Do something when a timer expires, or wait before doing something.
- ▶ Correlation rule - Establish a causal relationship between events.

As a starting point, it is likely that the simple and plain rules may be covered by probe rules, whereas timer, change, and correlation rules are more likely to be triggers and automations in the ObjectServer. This is not a hard and fast rule, and there are exceptions.

Converting or coding the logic for actions taken on the first instance or on all instances of a particular event are quite straightforward, as is clearing duplicates of a particular event, or modifying an event's attribute. More challenging are likely to be the timer and more complex correlation style rules, which you will need to be more creative about.

We recommend looking at the existing TEC rules, but in the context of the new capabilities, and constraints of the new OMNIBus sql-based language and not to try to replicate the function of every TEC rule predicate one for one.

### 3.7.3 Remote procedure execution

There is one exception to the above statement, where you should go through and search for a particular predicate, and that is with any that deal with remote execution of processes (exec\_program) or remote execution of framework tasks (exec\_task) currently. It is important for planning the upgrade process to understand if you do this currently in a TEC rule, so that the OMNIBus Process Automation component (process control) can be installed and configured on those hosts where running a remote process is required.

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.

The ObjectServer only communicates with one PA (usually the one that started it). Where processes are required to run on a different host to the ObjectServer, it is the local PA that the ObjectServer communicates with. The local PA then communicates with the remote PA.

**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. An update to the Windows process agent is planned that will remove this restriction.

Due to this architectural limitation, there may need to be some workarounds deployed, especially for Windows systems, and it may be necessary to deploy more than one ObjectServer.

### 3.8 Some event-processing examples

Previous surveys and investigations of many TEC rulebases from user groups and customer implementations have found that they broadly fit into a small number of rule types, listed below:

- ▶ Handling of duplicate events (simple and time or volume related)
- ▶ Filtering out events with specific content
- ▶ Enrichment of attributes in an event
- ▶ Correlation (cause and effect)
- ▶ Execution of an alert, task, or process
- ▶ Escalation of severity
- ▶ Forwarding events
- ▶ Heartbeat rules for monitoring system health
- ▶ Clearing (housekeeping and maintenance-based rules)

**Note:** We provide some non-technical examples of these later in this chapter, and then for the rules programmer, fully detailed worked examples are provided in 6.2, “Event processing migration” on page 189.

### 3.8.1 General suggestions

Here are some general suggestions for event handling:

- ▶ Try to handle event processing as close to the event source as possible.
- ▶ Before developing, check the default supplied triggers and automations.
- ▶ Install the Netcool Knowledge Library for best practice rules files.
- ▶ For event enrichment use probe lookup files (similar to TEC fact files).
- ▶ Enrich the OMNIBus database schema *only* for the most commonly used attributes.

Let us look at the typical event handling scenarios starting with handling of duplicate events. Each of the following sections will have a common description of the event scenario before a description of the implementation in TEC or SCE (if appropriate) and then the OMNIBus deployment. This should help you gauge the amount of effort required for each type of rule.

### 3.8.2 Handling of duplicate events

In this section we discuss the handling of duplicate events.

#### **Common description of the scenario**

Here duplicate events are sent repeatedly from an event source. Only a single event containing the newest event information should be seen on the event console.

#### **TEC implementation**

A baroc class with the dup\_detect modifier for specific slots must be defined first. Then we can have two possible solutions:

- ▶ Create a specific rule for this event class or a generic rule for all event classes. Define a reception action that updates the older event with current information from the new duplicate event, and then drops the newly arrived event.
- ▶ Create a specific rule for this event class or a generic rule for all event classes. Define a reception action that closes the older duplicate event and leaves the newly arrived event untouched.

#### **Implementation with the SCE**

Define a threshold or collector rule with an appropriate time interval.

### OMNIbus implementation

The standard deduplication can be used for the first solution, but for the second one the standard deduplication needs some modification.

## 3.8.3 Filtering out events with specific content

In this section we discuss filtering out events with specific content.

### Common description of the scenario

An event coming from server xyz should be shown on the console, except when the msg is equal to "mmm".

### TEC implementation

There are two options:

- ▶ Create a filter for the logfile adapter. Define the appropriate class and compare the source for the msg field.
- ▶ Create a specific rule for this event source. Define a reception action that compares the host name and the msg. If both are true, then the incoming event is dropped.

### Implementation with the SCE

Define a match rule with an appropriate comparison to the content of the host name and the message.

### OMNIbus implementation

There are two options:

- ▶ Create a rule file for the probe. Compare the msg field if it is true, then discard the event.
- ▶ Put a check in a pre-insert trigger such as the new\_row trigger, and if the field matches "mmm", cancel the operation. For example:

```
if ( new.msg = 'mmm' ) then
  cancel;
end if;
```

## 3.8.4 Actions for too many events in a defined time frame

In this section we discuss actions for too many events in a defined time frame.

### **Common description of the scenario**

If one event type arrives x times in y minutes then raise the severity to critical.

### **TEC implementation**

Create a specific rule for this event class. A reception action, triggered by the new duplicate event, searches for an older duplicate event and examines its arrival time and repeat count.

### **Implementation with the SCE**

Define a threshold rule with an appropriate time interval.

### **OMNIBus implementation**

For the OMNIBus implementation, two supplied probe functions, 'updateload' and 'geteventcount', can be used to measure the number of occurrences of a specified event x in y minutes. This can be used to create a procedure to raise the event severity.

It is also possible to create a timed array window at the probe level to handle this situation.

## **3.8.5 Filling an attribute dependent on another field's content**

In this section we discuss filling an attribute dependent on another field's content.

### **Common description of the scenario**

Based on the naming conventions of servers, the platform field is filled with the platform type.

### **TEC implementation**

A generic rule is triggered by all arriving events. In a reception action, the host name is parsed. Based on that key, a platform field is filled with the value *Windows*. The use of fact files is a common implementation here.

### **OMNIBus implementation**

Use the following steps:

1. Use lookup tables at the probe level to resolve. We recommend this for performance.
2. Add the column "platform" to the ObjectServer. Define a database trigger that populates the OMNIBus platform field by parsing the information from the node attribute.



### 3.8.6 Handling of correlations (cause, effect, and clearing events)

In this section we discuss handling of correlations (cause, effect, and clearing events).

#### **Common description of the scenario**

At the source, the following events are raised: cause, effect, and clearing. The operator should see only the cause event and no effect event. After the clearing event arrives, the cause event should no longer be shown on the event console.

#### **TEC implementation**

Create specific rules for these event classes. Link the effect events to the cause event at arrival, and close the effect events. Close the cause event when the clearing event arrives and discard the clearing event.

#### **OMNIbus implementation**

Use the default correlation behavior of the generic\_clear automation for simple correlations. For more complex correlations, the user should take the existing generic\_clear automation as a starting point and create new triggers as appropriate.

### 3.8.7 Local and remote script execution

In this section we discuss local and remote script execution.

#### **Common description of the scenario**

When an event arrives, a specific program has to be executed locally or remotely under a specific GID/UID.

#### **TEC implementation**

Create a specific rule for this event class. In a reception action call a local script or start the execution of a TME task on a remote endpoint.

#### **OMNIbus implementation**

Create a database trigger that starts an external procedure, locally or remotely, through process control.

### 3.8.8 Escalation of the severity of events

In this section we discuss escalation of the severity of events.

### **Common description of the scenario**

If an event is not closed 10 minutes after arriving, set its severity to critical.

### **TEC implementation**

Create a specific rule for this event class. Set a timer for 10 minutes in a reception action. If it is still open, set the severity to critical.

### **OMNibus implementation**

Create a temporal trigger that sets to 5 the severity of events that have not been acknowledged as arrived in the last 10 minutes. There is a default flash\_not\_ack trigger that provides something similar, and operators can easily change event severity via the console.

## **3.8.9 Forwarding of events**

In this section we discuss the forwarding of events.

### **Common description of the scenario**

Send an incoming event to a trouble ticketing system.

### **TEC implementation**

Create a specific rule for this event class. Call a script that has access to the contents of the event and passes these contents to the trouble ticketing software.

### **OMNibus implementation**

Execute an external procedure, or install a gateway for a trouble ticketing system or for an external database.

**Note:** Again, refer to Chapter 6, “Event processing” on page 187, for fully worked examples of the above and more scenarios. The aim of the aim of the above is to give an idea of the required changes.

## **3.9 TEC outputs**

Here we consider the third phase of the end-to-end event flow: the outputs from the central rules engine to other interested parts of the organization, be it for help desks, event notification, automation, links to other applications, or historical reporting.

### 3.9.1 TEC tasks

TEC tasks are often used within TEC rules. With these you can execute scripts and programs on the TEC server, on Tivoli managed nodes, or on endpoints to restart a process or to alert someone via SMS or e-mail.

As mentioned, to execute programs by OMNibus on a local or remote site you must set up a process control agent to control the targets first (this is equivalent to having the Tivoli endpoints installed to run the task on), so you should audit the number and type of tasks in the current implementation.

Many tasks are specific to TEC, and not relevant to OMNibus, but others do have equivalents. For example:

- ▶ The Popup\_Message task in TEC can be replaced with an event notification in OMNibus.
- ▶ The Send\_Email task in TEC can be replaced with a procedure called send\_email, fired by a mail\_on\_critical trigger.

In general, OMNibus is very function rich in the areas of notifications and ease of customization of the operator console.

### 3.9.2 Forwarding to other TEC servers (manager of managers)

If you currently have a hierarchy of TEC servers or two TEC servers are forwarded to for resilience, then you will be using the tec\_forward or re\_send\_event\_conf predicates in TEC rules. As we will see in the upgrade scenarios, all of these combinations are technically possible, so for planning purposes this forwarding should not be an issue.

Tests that we performed showed that the TEC rule, rather than the receiving probe or ObjectServer, was the limiting factor. We recommend that caching of the forwarding events is properly configured to cope with any short-term bottlenecks or server outages. The size of this will depend on the event rates and average event sizes.

### 3.9.3 Incident management systems

If specific rules are in place to send alerts or messages to other applications, such as trouble ticketing systems like Remedy ARS or IBM Maximo®, then this will require careful planning and coordination for these to be migrated. In many cases OMNibus can provide a bi-directional or uni-directional gateway application between the ObjectServer and the application, and it is likely that the

integration will be more straightforward than with TEC (although we did not implement this during the book).

For a complete and up-to-date list of the available gateways see the *IBM Netcool/OMNIBus Probe and Gateway 7.2*, SC23-6373.

### 3.9.4 Service-level reporting and auditing databases

Many customers will capture details from the event database for use by service-level and other reporting systems. Here again, OMNIBus gateways are available for most applications. If it is an in-house database that is the backend, then an OMNIBus ODBC gateway to a relational database can be implemented to do this.

At the time of writing, a Tivoli Data Warehouse gateway integration is not available, but this is planned for release in early 2008.

Collecting data for auditing purposes can also be done in OMNIBus.

Finally, there is also the option of using other tools such as Netcool/Reporter or Netcool/Provisio for performance and trend analysis, which are designed for these functions and tightly integrated with the ObjectServer.

## 3.10 Desktop upgrade (TEC console)

It is likely that most customers will want or need to keep the same operational structure of event groups and views, and also preserve the same roles and authorization of the existing users. It is possible to export the current TEC console settings, but this is only designed for import by another TEC server. It would be desirable to dump this out for import into the ObjectServer desktop and Netcool/Webtop.

It is beyond the scope of this book to provide a comprehensive script to output all the TEC operator roles and event group filters, as that is part of the serialized TEC database object and requires a Java program to access it. While in theory it would be possible to produce a tool or utility to convert TEC operator information into OMNIBus user information (and this is currently being evaluated by development), there is extra functionality in OMNIBus that would need to be added, so this is probably best done manually if there is a small list.

If, however, there is a large number of users or groups to add, then we recommend building templates and utilizing the `$OMNIHOME/etc/security.sql` script to add the details about what permissions are granted to each role and

group and then using the lists created from the TEC console sql commands in Figure 3-7 in a loop as input into a script to add the remaining roles, groups, and users.

```
su - db2inst1
connect to tec
select * from tec_t_assign_op > file_op_IDs_names
select * from tec_t_assign_eg > file_con_ID_group_ID
select group_id,name from tec_t_event_groups > file_group_ID_names
select console_id,name from tec_t_consoles >file_con_ID_op_name
```

Figure 3-7 Suggested commands to extract TEC console information

See the *Netcool/OMNibus Installation and Deployment Guide* for a full discussion, as the level of functionality is slightly different from the TEC model in concept, but here are some suggested nco\_sql statements.

```
First create the roles...:
CREATE ROLE 'role_name'
[ ID identifier ]
[ COMMENT 'comment_string' ];
then create the groups....:
CREATE GROUP 'group_name'
[ ID identifier ]
[ COMMENT 'comment_string' ]
[ MEMBERS 'user_name', ... ] ;
Then finally the users...:
CREATE USER 'user_name'
[ ID identifier ]
FULL NAME 'full_user_name'
[ PASSWORD 'password' [ ENCRYPTED] ]
[ PAM { TRUE | FALSE } ]
```

Figure 3-8 Suggested nco\_sql statements

**Note:** The above are suggestions for importing into the OMNibus user tables for native desktop use only. For import into the Netcool/Security Manager model and therefore use by Netcool/Webtop, an additional export and import stage is required.

In addition to the above, it is possible to set a *restriction filter* to limit the rows and tables that a user or group of users can see. There are also OMNibus View

builder and Filter builder desktop tools that can be applied on a user-by-user basis, and even customized canned views can also be created with a transient event list, which references a filter file.

### **Desktop scalability**

The lists created from the TEC database in the last section can also be used to help plan the number of additional display ObjectServers that are required to cope with very large numbers of concurrent operators. If the loading of the desktop gets too high, then additional ObjectServers can be configured. Previous studies have shown this may be in the region of 30–40 active users, but we did not verify this in our lab setup.

## **3.11 Event view customization**

Upgrade considerations for event view customizations.

### **3.11.1 TEC information button**

Some customers utilize the feature in the TEC console to create context-related help or operating instructions in a set of html formatted Web pages, accessed via the information button. The information is based on event class type, allowing the customer to build a context-related online knowledge system for troubleshooting. The pages can be located on the same server or on a remote Web server.

The same functionality can be developed in the OMNIBus desktop. By default, there is a column in the alerts.status database called URL that can be utilized and a procedure called openurl. It is also possible to select an event, right-click, and select the **Open URL** tool.

This together with the ability to easily import the BAROC class information with the baroc2sql tool should mean that this data migration is a straightforward process.

We provide an example in 6.5, “TEC information/URL information for events” on page 256.

### **3.11.2 TEC custom buttons**

It is possible for TEC operators to have up to three action buttons per console, which can be configured to perform *canned* tasks or actions. The benefit is typically for ease of operations, rather than having to use a command line. The OMNIBus native desktop does not have exactly the same feature, but the same

functionality can be provided by creating a drop-down list of tools from the menu. It should be possible to create a larger set of options if required.

### 3.11.3 Large event messages (greater than 255 characters)

Currently, in TEC the msg field slot maximum size is 255 characters, and there is a long msg slot for an additional 4,000 characters (with a total limit of 4,096). If you have these large messages in your event domain (and ITM 6.1 is a common source of large messages), then by default we have observed that the OMNibus field that holds the tec messages when they are passed across (called *summary*) truncates the message to 255 characters.

There are a few straightforward ways in OMNibus to handle this. The Summary column could be extended, but as it is a default field, it might be better to create a new column in the ObjectServer. This could have a definition of varchar 4000 in the table, and additionally, a mapping in the probe rules would be required to add this extra information.

Another option could be to use the extended attribute feature of OMNibus to concatenate the two TEC attribute fields together (up to a maximum of 4,096). An example of the use of this appears in 6.3.4, “EIF rules file and extended attributes” on page 244.

**Note:** The OMNibus desktop GUIs can only display a maximum of 255 characters, so the only way to access the extra information would be via command-line sql.

### 3.11.4 Operator actions

The operator actions in TEC vary slightly depending on whether the Java console or Web console is used. This is comparable with the OMNibus native GUI and the NetCool/Webtop.

Java console:

- ▶ General preferences: refresh time, maximum age of closed events to display, maximum number of events, and display trouble ticket success messages.
- ▶ Working queue: Classify the order of the appearance of columns (Time Received, Class, Event Type, Severity, Hostname, Status, Sub-source, Message) and the sorting preference (ascending or descending order).
- ▶ All events: Classify the order of the appearance of columns (Time Received, Class, Event Type, Severity, Hostname, Status, Sub-source, Message) and the sorting preference (ascending or descending order).

- ▶ Filtering events: You can filter the events in the working queue based on severity, status, and operator ownership.

Web console:

- ▶ Change user preferences: refresh rate, maximum number of events to show on each page, severity counts and banner, and show the event severity as (show the severity types).
- ▶ Sorting events: By default, events are sorted in the event viewer by the date received field. You can sort events based on up to three event fields.
- ▶ Filtering events: You can filter the events in the working queue based on severity, status, and operator ownership.

## **Netcool/OMNIbus implementation**

Some preferences can be customized on the event viewer:

- ▶ Monitors tab: Show severity border, show count, show highest severity, show lowest severity, show metric, show lavalamp, and show histogram.
- ▶ Refresh tab: timed refresh.
- ▶ Notification tab: Notify when iconized, when (new, change, delete events), and how (ring bell, alert icon, open window, run external command).
- ▶ Flashing tab: Enable flashing.
- ▶ Misc tab: Show event list colors, show distribution summary, show toolbars, jump button on by default, sort information details.
- ▶ Appearance of the event list: resizing columns, sorting columns.

Comparing the two sets of consoles, it appears that OMNIbus is more flexible, more dynamic, and more customizable than TEC, and has some useful additional features such as the ability to alter the severity of an event, take ownership or delegate ownership, or escalate an event.

The performance of the desktops in loading and navigating menus also appears to be far quicker on some comparisons we performed, giving an all-around better user experience.



### 3.11.5 Color patterns

Each product has a color pattern scheme to represent event severities. They are almost equivalent. Figure 3-9 shows the colors and severity for TEC and OMNIbus.

TEC severity	Color	OMNIbus severity	OMNIbus severity number
FATAL	black	n/a	
CRITICAL	red	CRITICAL	5
MINOR	orange	MAJOR	4
WARNING	yellow	MINOR	3
UNKNOWN	blue	WARNING	2
n/a	purple	INDETERMINATED	1
HARMLESS	green	CLEAR	0

Figure 3-9 TEC and OMNIbus severity colors

It is possible to alter these colors for all users by changing the X11 app-default configuration files, and it is possible to set up private color maps for use by color-blind operators.

There are also good console integration options with ITM 6.1 and ITM 6.2 TEPS for both TEC and OMNIbus consoles. ITM 6.2 has introduced the Common Event Console (CEC). CEC provides an alternative method for integrating TEC and OMNIbus console into the ITM situation alert view on the TEPS. The CEC is not intended to provide all of the functionality of the native desktops of TEC or OMNIbus. It provides a subset of acknowledging or closing style actions. Therefore, the CEC is not designed as a strategic single console, but should be seen as a proof of concept and aid for integration and upgrade.

## 3.12 Resource considerations skills

As with any new technology, there will be new skills required and a learning curve. For some key individuals this may well be quite a steep curve, but for the majority of operations staff OMNIbus is quite intuitive and straightforward to use.

### 3.12.1 Event-processing configuration

It is very much the case that some form of fast tracking is advisable here to use IBM consultancy services and skills transfer. The main skill required for the

administrator is standard sql knowledge, and the better this knowledge is the more sophisticated the rules can become. This skill is likely to be the most important resource required.

### **3.12.2 Installation, administration, and operations**

Installing and administering OMNIbus requires a very methodical approach. We show in Chapter 4, “Upgrade strategies” on page 147, where we recommend changes to the default values for improved stability or ease of use. We also recommend that plenty of time is given to get an environment stable and bedded in and administrators familiar with the new concepts. Operationally, there are a few subtle differences, and there is be a minimal amount of education for orientation and the new features (particularly the escalation features) and tools.

### 3.13 Checklist

The checklists in Figure 3-10 and Figure 3-11 on page 144 can be used or modified to suit your requirements to assist in the taking of the audit.

		Checklist			
Checkbox		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Identify Sources	TEC Sources	Profile(s)	Endpoint(s)	Filter(s)	Format(s)
	(TME) Unix Logfile Adapter	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	(TME) Unix Syslogd	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	(TME) Windows Logfile Adapter	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	(TME) Windows Event Log	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	SCE Gateway	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	SNMP Adapter	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	AS/400 Adapter	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	non-TME Sources		Hostname(s)	Filter(s)	Format(s)
	(non-TME) Unix Logfile Adapter		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	(non-TME) Windows Logfile Adapter		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	(TME) Customized Adapters		Endpoint(s)	Format(s)	Rule(s)
	wpostmsg		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	wpostzmsg		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	(non-TME) Customized Adapters		Hostname(s)	Format(s)	Rule(s)
	postmsg		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	postzmsg		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Other Applications				Rule(s)
	DM 3.7				<input type="checkbox"/>
	ITM 5.1.x				<input type="checkbox"/>
	ITM 6.1				<input type="checkbox"/>
	ITM 6.2				<input type="checkbox"/>
	Omegamon Agents				<input type="checkbox"/>
Custom EIF applications				<input type="checkbox"/>	
NetView 7.1.4/5				<input type="checkbox"/>	
TBSM 3.1				<input type="checkbox"/>	

Figure 3-10 Checklist - identify sources

Checklist			
Identify Output	Checkbox	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
			Rule(s)
	Trouble Ticket		<input type="checkbox"/>
	Databases		<input type="checkbox"/>
	e-mails		<input type="checkbox"/>
Console Interface	pager		<input type="checkbox"/>
		Operators	Targets
	Custom buttons	<input type="checkbox"/>	<input type="checkbox"/>
	Information buttons	<input type="checkbox"/>	<input type="checkbox"/>
	Automated Tasks	<input type="checkbox"/>	<input type="checkbox"/>
	Color customization	<input type="checkbox"/>	<input type="checkbox"/>

Figure 3-11 Checklist - identify output and console interface

## 3.14 Suggested testing plan

**Note:** In an ideal case, it would seem that the obvious order in which to make changes would be to first move the event sources, then the processing, and then the desktop and the other outputs. When you are testing and building the new environment this is probably the way it is approached. However, when considering how the migration might have to be done in a live scenario, unless you are fortunate enough to have the chance to change everything at once, we recommend that the best practice is to take the reverse approach. This approach has two advantages: first, it allows the main benefits of OMNIbus to be realized faster, which are primarily at the back end rather than at the event sources. Secondly, it allows a gradual phased migration of event sources, which will be particularly useful in a very large environment. So this plan is in the order that replicates how you may upgrade the live system.

Below is a list of suggested steps to take when building a test environment:

1. Follow the guidelines of this chapter and conduct the survey or audit and be aware of the constraints and limitations of your live environment and differences in functionality of Netcool. Then use the checklist to try to ensure that all areas are covered.
2. Collect some real events from your environment via wtdump.pl, parseEvents.pl, and sendEvents.pl.
3. Utilize an existing TEC or build a test TEC to run these events through. You can optionally follow the steps in 5.1.2, "TEC installation and configuration" on page 168.

4. Build the OMNibus environment, as detailed in 5.2, “Netcool/OMNibus lab environment” on page 168.
5. Add a forwarding rule as described in 7.1, “Adding a rule to forward raw events to OMNibus” on page 278, and send the TEC events via the `re_send_event.conf` predicate and the EIF probe to the ObjectServer.
6. Add the user-related customizations:
  - a. Add the required roles, groups, and users, optionally using methods described in “Desktop upgrade (TEC console)” on page 136, to assist you.
  - b. Build filters and views for operators to match TEC event groups.
  - c. Optionally, deploy the security manager and Webtop if you have not already done so and import the users into there.

For the reason in the above note, the following steps are in the order in which you might need to approach them during the upgrade, but can be followed in any order for test purposes. All steps are dependent on whether you have the event source.

7. Implement the event processing:
  - a. Automations, triggers, and probe rules, referring to Chapter 6, “Event processing” on page 187. At this point you should have the OMNibus event list desktop looking how you want it.
  - b. Create desktop tools configured like the TEC information button and custom buttons.

You should now be able to handle the events that you are going to have flowing through the EIF probe in the long term.

8. Connect to any backend outputs. (You might need to simulate some of these systems.)

Add gateways to test versions of backend databases, other TEC event servers (managers of managers), and trouble ticketing systems, like Remedy or Maximo.

9. Deploy test ITM sources to match your live environment. As mentioned, ITM6.2 can be configured to send to parallel destinations, so this could be utilized. If you are not planning to upgrade to ITM 6.2 first, then you will also need test ITM 6.1 events, and if still on 5.1.1, they need to go via the test TEC.
10. Implement the EIF probe rules as detailed in 7.3, “Integration between Netcool/OMNibus and IBM Tivoli Monitoring” on page 290.
  - a. Add the triggers for synchronization for ITM.
  - b. Add the Situation Update Forwarder (SUF).
11. Deploy a test NetView system (either monitor a live or test network).

12. Implement the EIF probe rules for NetView, as described in 7.2, “Integration between Netcool/OMNIBus and Tivoli NetView” on page 280.
  - a. In addition, add the temporal trigger example for NetView detailed in 7.2.6, “Automatic event management customization” on page 284.
  - b. Develop any additional NetView triggers required based on this template.
13. Deploy and test other event sources. (In addition to using test systems, depending on your networking, it may be possible to install a second TEC adapter with the configuration pointing to the test EIF probe. It is also possible to configure two syslog pipes on one system to get events in parallel, for example.)
  - Probes for syslog
  - General logfile prob
  - Probe for Windows NT event logs, as described in 7.5, “Migrating the TEC Windows event log adapter” on page 322
  - Sources from (w)postmsg and (w)postzmsg commands
14. Deploy Process Automation (PA) if you have not already done so.

Activate the triggers to do remote process execution logic. Remember that currently the Windows PA and UNIX PA cannot be mixed.
15. Add and test any additional configuration, as described in Chapter 2, “Architecture” on page 37. Also see the *Netcool Installation And Deployment Guide* for more details.
  - Failover and resilience
  - Proxies
  - Firewalls
16. You should now have a system ready for the following:
  - Performance testing
  - User acceptance testing
  - Skills transfer or training

After some additional fine tuning from user feedback you will be ready to deploy in production.

## Upgrade project plan

The above steps could now form the broad headings for a project plan, and depending on your environment, available resources, and skills, you may now be able to put some time estimates to each step to start to form an upgrade plan.

## Upgrade strategies

There are a number of options available when designing an upgrade to OMNibus. In this chapter we focus on two main approaches:

- ▶ Parallel installation, with all-at-once switchover
- ▶ Installation in phases, with coexistence during gradual switchover

For each approach, we have detailed the event flow, the advantages and disadvantages, and made some suggestions regarding the typical customer profile that this might fit.

We conclude this chapter by going into more detail regarding the phased upgrade approach. We have used this strategy as the main focus of this book to illustrate the way in which a mature TEC environment will most likely be upgraded. It should be noted that we are not advocating that a phased approach is superior (actually, the more straightforward that you can make the replacement the better), but this strategy seems to better suit the potential for different situational upgrade requirements that you may encounter.

Before we proceed to discuss the full upgrade strategies, we reference the existing integration approaches that have been the subject of previous IBM white papers (*IBM Tivoli and Netcool Event Flow Integration*), and have been widely distributed as best practices. These have focussed on integrating the two event management technologies together, in a manager of managers approach. They detail how to do this with either TEC or OMNibus as the master. It must be made clear that these should be seen as interim approaches to combining the

technologies, rather than a strategic *final picture* of the event management environment. Furthermore, these white papers do not describe how to upgrade fully to OMNibus to facilitate removal of TEC. That is the goal of this book.



## 4.1 Event flow integration based on TEC

Figure 4-1 shows an overview of the event flow in a deployment where Netcool/OMNIBus is added to provide TEC with additional event information. Events are consolidated and managed from TEC. This strategy is described in the *Tivoli and Netcool Event Flow Integration*<sup>1</sup> white paper.

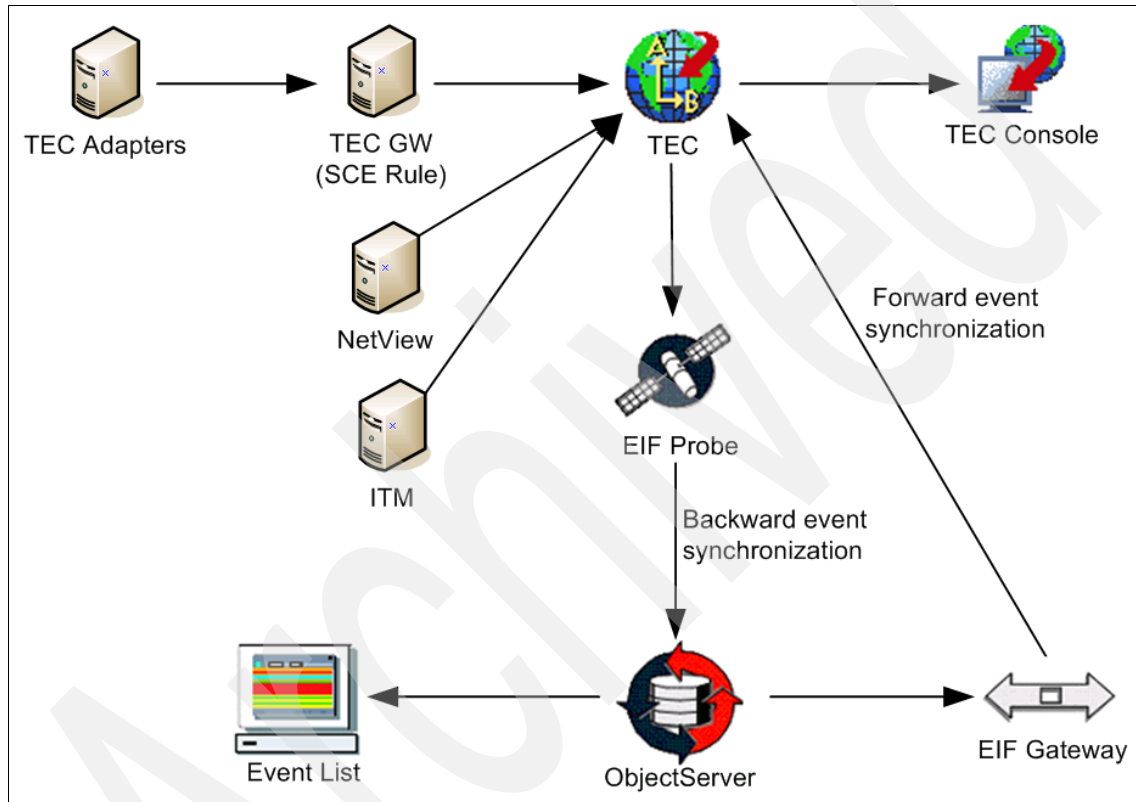


Figure 4-1 TEC as the manager of managers

As OMNIBus is added to this existing TEC environment, it can be configured to send events to TEC through the installation of the Netcool/OMNIBus EIF gateway. The gateway retrieves events from the OMNIBus server and sends them to the TEC server using the EIF interface. Forward event synchronization uses same mechanism to send to TEC the events that have been updated in OMNIBus. To achieve the backward event synchronization (events that are updated in TEC are sent to OMNIBus), the EIF probe has to be installed and configured in addition.

<sup>1</sup> <http://catalog.lotus.com/wps/portal/topal/details?NavCode=1TW10EC01>

While this solution allows the customer to enrich the TEC environment with new probe sources and help become familiar with other new features from the Netcool suite, it does not progress very far along the upgrade route, and also has the following disadvantages:

- ▶ **Performance:** The Netcool/OMNIBus EIF gateway is used in this integration flow. The performance could be limited by the EIF mechanism if the volume of events from OMNIBus is high. Secondly, if additional Netcool probe sources are introduced, then they may overload an already stressed TEC server.
- ▶ **Duplicate event consoles:** There is no longer a single focal point for the administration of events, but there are two valid event consoles, and a choice about which one will be used is delegated to the customer.

## 4.2 Event flow integration based on OMNibus

Figure 4-2 shows a typical event flow for OMNibus. This strategy is described in the *Tivoli and Netcool Event Flow Integration*<sup>2</sup> white paper.

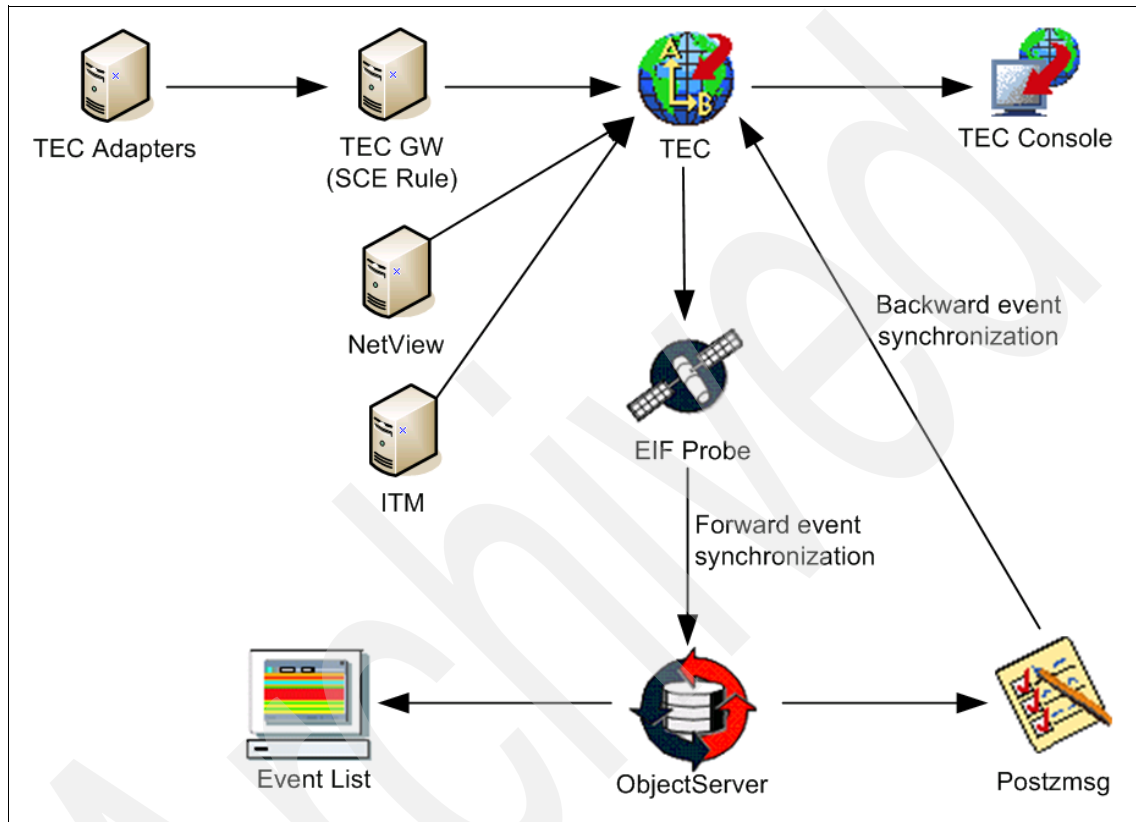


Figure 4-2 OMNibus as the manager of manager

TEC forwards events to Netcool/OMNibus using event forwarding rules via the Netcool/OMNibus EIF probe. The forward event synchronization is implemented so that status and severity changes in TEC are updated in OMNibus. No other event updates on the TEC server are synchronized with Netcool/OMNibus. The backward synchronization is implemented so that status and severity changes in OMNibus are updated in TEC.

This method has similar limitations on performance and potential duplication of consoles, but is a preferable option to that described in 4.1, “Event flow integration based on TEC” on page 149, as at least it provides a step toward the

<sup>2</sup> <http://catalog.lotus.com/wps/portal/topal/details?NavCode=1TW10EC01>

strategic goal of having an event management solution based on Netcool technologies.

## 4.3 TEC replacement strategy

This strategy is based on building a stand-alone OMNIBus environment in parallel to the existing TEC system, and switching over in a single phase. There is no integration with any Tivoli components.

### 4.3.1 Event flow

4.3, “TEC replacement strategy” on page 152 shows a typical event flow for this solution.

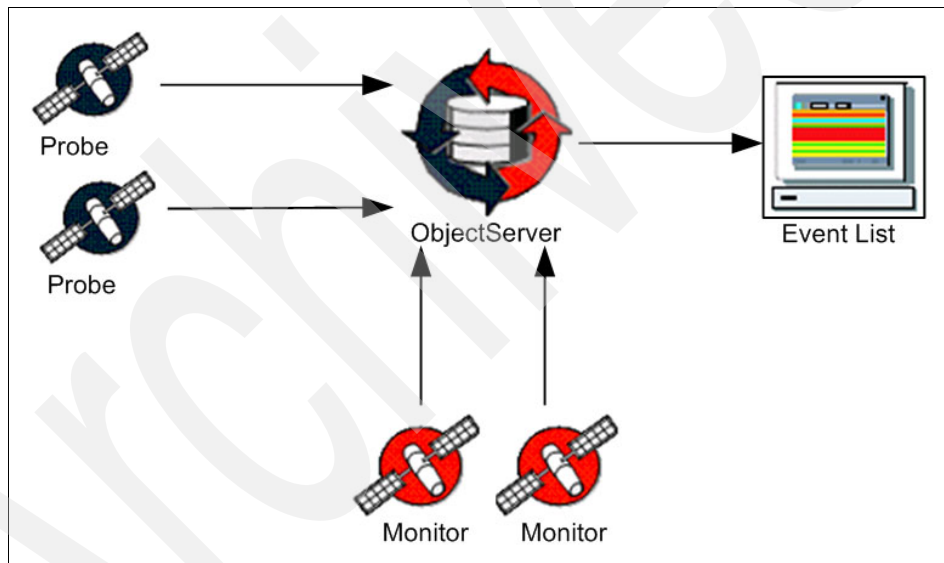


Figure 4-3 TEC replacement strategy

This solution is basically the one provided by a typical Netcool/OMNIBus installation. Events are collected from the underlying infrastructure by probes and monitors and sent to the ObjectServer for processing. The event list is the console for event visualization and management.

Other event sources can be added to this architecture, such as Tivoli Network Manager IP Edition and Tivoli Business Service Manager 4.1. The role of TNM is

the same of NetView for TEC, that is, network discovery and network failures root cause analysis.

Further event enrichment features could be provided by Netcool/Impact.

### 4.3.2 Advantages

The advantages are:

- ▶ Ease of deployment: To install this kind of solution the customer can follow the installation instructions provided step by step. The deployment, as pointed out before, is quite straightforward, and the out-of-the-box features are enough to represent a good starting point for monitoring.
- ▶ OMNibus single console for monitoring: OMNibus is the event manager of the future, therefore this strategy is proactive and in line with IBM product strategy.
- ▶ A fresh start redesign: In reality this option is the one most likely to completely exploit all the best features of an OMNibus design, without any constraints from the earlier environment.

### 4.3.3 Disadvantages

The disadvantages are:

- ▶ No integration with the existing Tivoli environment: If the customer is satisfactorily monitoring his infrastructure with the products of the Tivoli Framework, there will be a *big bang* cut over to the OMNibus environment abruptly without migrating his rules, automations, and event sources.
- ▶ Possible loss of investment: This approach is probably only suitable for those who have not made a large investment in configuring and programming TEC rules, or who are prepared to redevelop.

### 4.3.4 Which scenarios this applies to

This solution is most suitable for customers who do not have any need to maintain their existing environment, and wish to start with a fresh approach. They do not have any requirements to integrate their Tivoli environment with OMNibus, and of course for those who have no Tivoli environment at all. They will typically be customers with a small deployment of event sources, a straightforward or even default rulebase setup. This can also be a good option for a customer who no longer has, or never had, the skills in the TEC prolog rules, for example.

## 4.4 TEC to OMNibus upgrade

This strategy aims to install an up-and-running OMNibus infrastructure that is completely integrated with an existing Tivoli environment in terms of event sources and event management. The difference from the previously discussed strategy lies in the fact that this solution takes into account the existing Tivoli investment and allows a more gradual switchover process to OMNibus.

### 4.4.1 Event flow

Figure 4-4 shows the required event flow in a TEC to OMNibus upgrade solution.

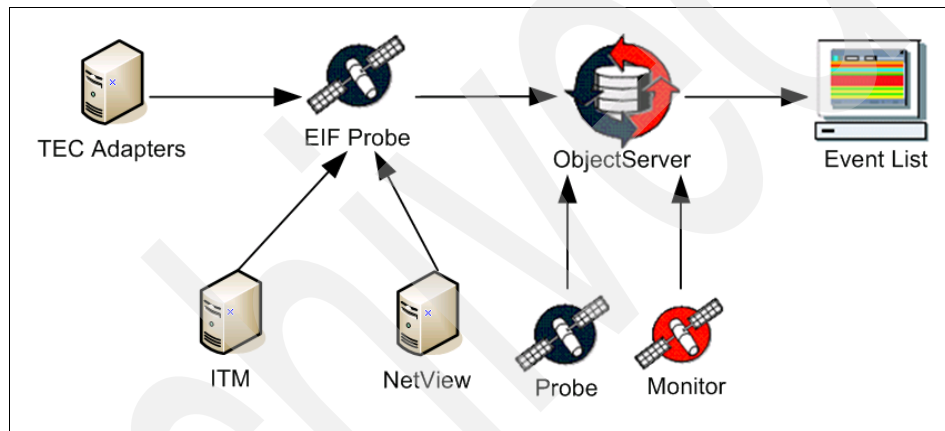


Figure 4-4 TEC to OMNibus upgrade

When a TEC and an OMNibus environment are integrated, event information is collected in a wide range of ways. Where possible, the approach consists of replacing the TEC Adapter with Netcool probes and monitors, applying the appropriate configuration required to replicate the original behavior of TEC adapters.

Where this cannot be achieved, TEC adapters are not removed from the system, but instead they are reconfigured to send events to the EIF probe instead of the TEC server. A similar procedure is followed with ITM and NetView, but with more integration options. In the future, on the basis of customers' needs, NetView can be replaced with Tivoli Network Manager IP Edition, which will send events directly to OMNibus.

## 4.4.2 Advantages

The advantages are:

- ▶ No customer left behind: This solution is the best for customers who want to upgrade from TEC to OMNibus in a phased approach. They will have an integrated new event management console that benefits both from the old TEC automations and the new features, and they will keep on receiving alerts from every source on their infrastructure without leaving anything uncovered.
- ▶ OMNibus and Tivoli together: This solution allows an immediate take up of the benefits of the new features of OMNibus and combines the existing Tivoli configuration, giving the chance to upgrade to a newer monitoring infrastructure without losing the configurations previously used.

## 4.4.3 Disadvantages

The disadvantage is deployment. The deployment of this solution is not as straightforward as the previous ones, but this is due to the fact that it covers the evolution of the entire existing infrastructure. The main goal of this book is to assist and guide customers through the entire process, hopefully, making it much easier to tackle the upgrade process.

## 4.4.4 Who this applies to

This solution is designed for all those customers who wish to upgrade to the benefits of OMNibus, protecting and migrating as much of their TEC investment as is appropriate.

After a deep analysis of the available strategies for the upgrade from TEC to OMNibus, we recommend that this solution can be applied to the widest range of customers, and it is therefore the one that has been deployed in our lab environment. It is appropriate for large and small implementations, and those with quite sophisticated rulebases.

The next section discusses in more detail a description of the tasks that will lead to setting up this strategic TEC to OMNibus upgrade.

## 4.5 The recommended strategy

When starting to plan an upgrade from TEC to OMNibus, we imagined a typical Tivoli Framework customer environment, as Figure 4-5 suggests.

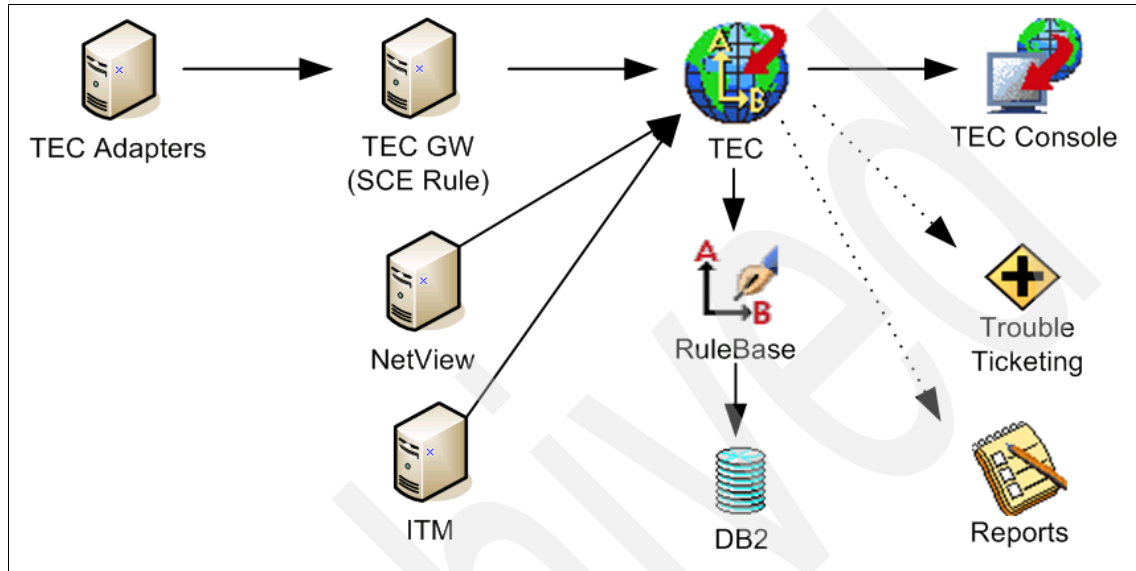


Figure 4-5 Initial Tivoli customer environment

In this hypothetical customer environment, raw events are collected from the following sources:

- ▶ TEC adapters (UNIX/Windows system log files, SNMP)
- ▶ ITM
- ▶ NetView

Events are sent through the TEC gateway to the TEC server, in which they are managed through the rulebase and then stored in the DB2 database that is installed with TEC. Operators use the TEC console as the event viewpoint, but they could also use Tivoli Enterprise Portal. Trouble ticketing and reporting systems are optional components of this architecture.



The first step to go through on the path that leads to the upgrade, which consists of installing an independent OMNIBus environment. At the end of this step, the customer environment should look like Figure 4-6.

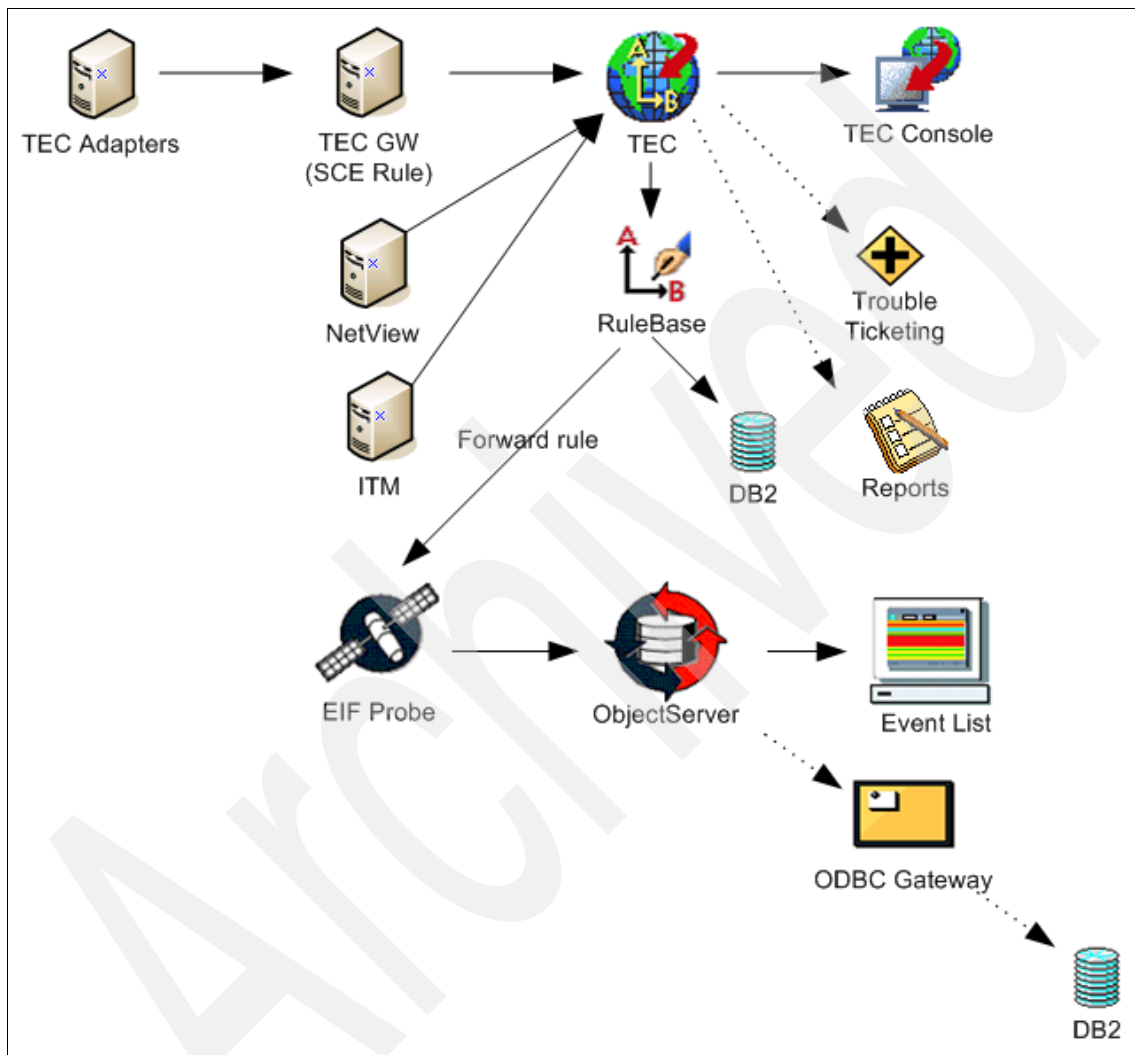


Figure 4-6 Customer environment at the end of the first step

At the end of the first step, the TEC environment works as normal, but an additional forward rule added as the first rule in the rulebase sends events to the EIF probe. The forward rule sends to the EIF probe the events collected from the Tivoli sources before they are processed by the other rules of the rulebase.

Therefore, all the events that are collected in the TEC server should be seen also in the ObjectServer, but in their *raw* form.

This is *not* the same forwarding mechanism as is used in the Tivoli and Netcool Integration package (TEC\_OMNibus.tar).

For the next steps we need to consider again the three main components of the environment, as described in Chapter 3, “TEC environmental assessment and planning guidelines” on page 103: the inputs, the processing, and the outputs.

The order in which you upgrade these components will depend on the complexity of the environment, as discussed in the following sections.

### **Ideal (simple) case**

In an ideal case, make changes to first move the event sources, then the event processing, then the desktop and the other outputs.

When you are testing and building the new environment this is probably the way it is approached. However, when considering how the migration might have to be done in a live scenario, unless you are fortunate enough to have the chance to change everything in one go, we recommend that the most realistic and therefore best practice is to take the reverse approach.

### **Realistic (more complex) case**

In a realistic case, migrate the event outputs and desktops, then the event processing, and then gradually migrate the event sources.

This approach has two advantages: first, it allows the main benefits of OMNibus to be realized faster, which are primarily at the backend rather than at the event sources. Secondly, it allows a gradual phased migration of event sources, which will be particularly useful in a very large environment.

So first you need to consider all the outputs from the system and work on migrating the interfaces (Netcool Gateways) to those applications such as trouble ticketing, reporting, ODBC gateways for DB2 and other relational databases. Similarly, make operational help desk related changes such as e-mail and SMS, and optionally make the OMNibus desktop the main focal point for operations. Optionally, Webtop can be used for the visualization of events.

This goes *hand in hand* with the relevant event processing for those changes, and result the process shown in Figure 4-7.

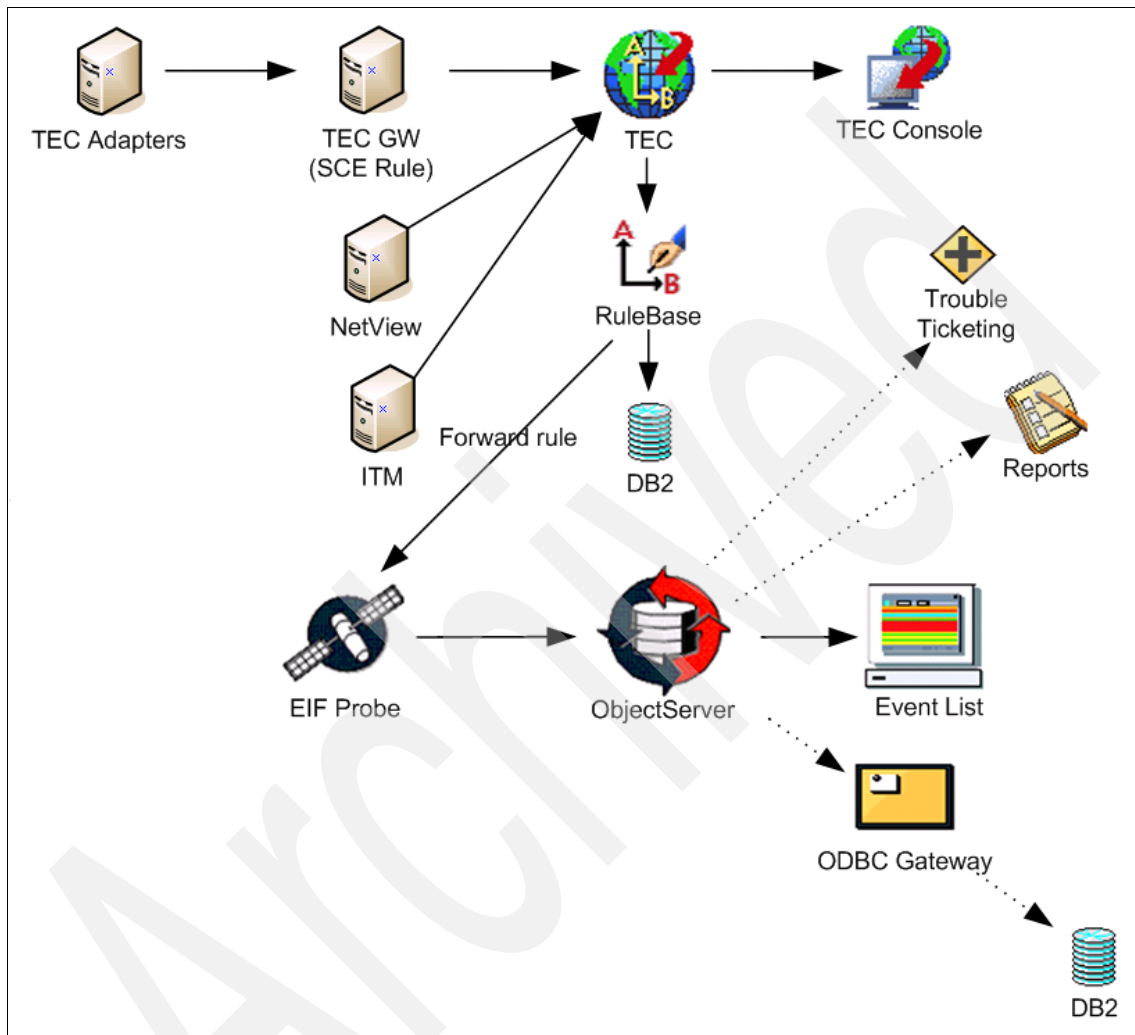


Figure 4-7 Customer environment at the end of the second stage

Further configuration using automations or probe rules can now be implemented in addition to the features OMNibus provides out of the box, in preparation for moving the event sources.

The initial step that can be performed easily when migrating the event sources is to change the *destination* of the events coming from ITM and NetView from the TEC server to the EIF probe. The configuration files for this integration are

provided in Chapter 5, “Upgrading to an IBM Tivoli Netcool environment” on page 165, together with the appropriate instructions. At the end of this step, the architecture should look like Figure 4-8.

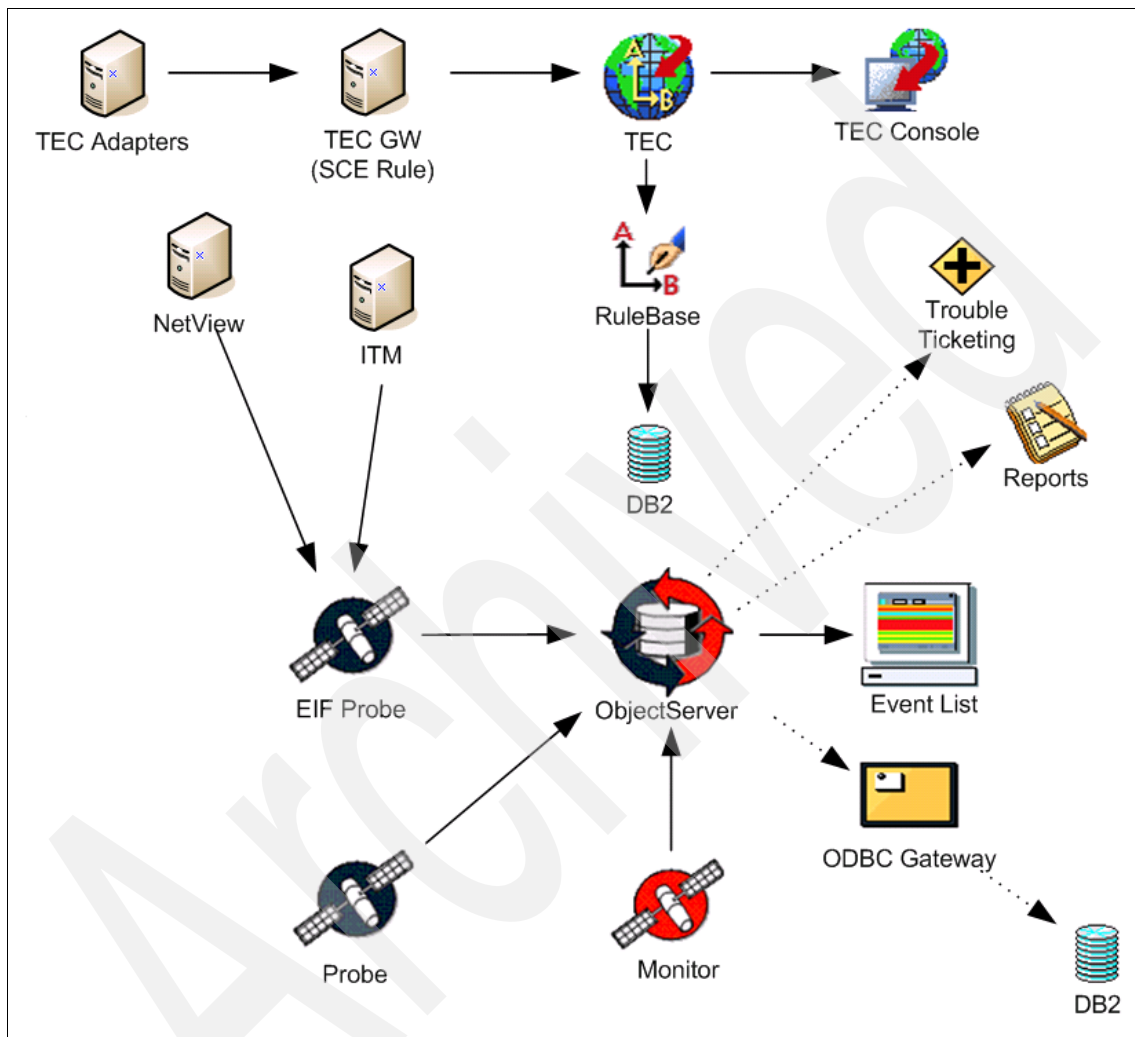


Figure 4-8 Customer environment at the end of the third stage

Next, Netcool probes and Process Automation can be deployed into the infrastructure. After evaluating whether the Netcool event collectors (probes and monitors) can gather events from the same sources as TEC adapters did (with either out-of-the-box features or with some rule enhancement), we can decide to route the source through the EIF probe or use the native Netcool probe.

In this phase TEC adapters can be quickly reconfigure to send to the EIF probe, as an interim phase, and indeed some TEC sources will need to remain on there, if it is their only means of integration. For other sources we can deploy the relevant Netcool probes, such as the syslog, Windows NT Event Log, and SNMP probe, and reconfigure to go directly to the ObjectServer.

Once all events are correctly processed by OMNibus, and if there are no earlier requirements for TEC, we can disable the forward rule from the TEC server to OMNibus.

At this point, the upgrade is almost complete. Now OMNibus should handle everything TEC did and maybe something more in terms of effectiveness, efficiency, and coverage of the infrastructure.

Once we are happy with the result achieved, we can disable the TEC server and enjoy our brand new OMNibus environment (Figure 4-9).

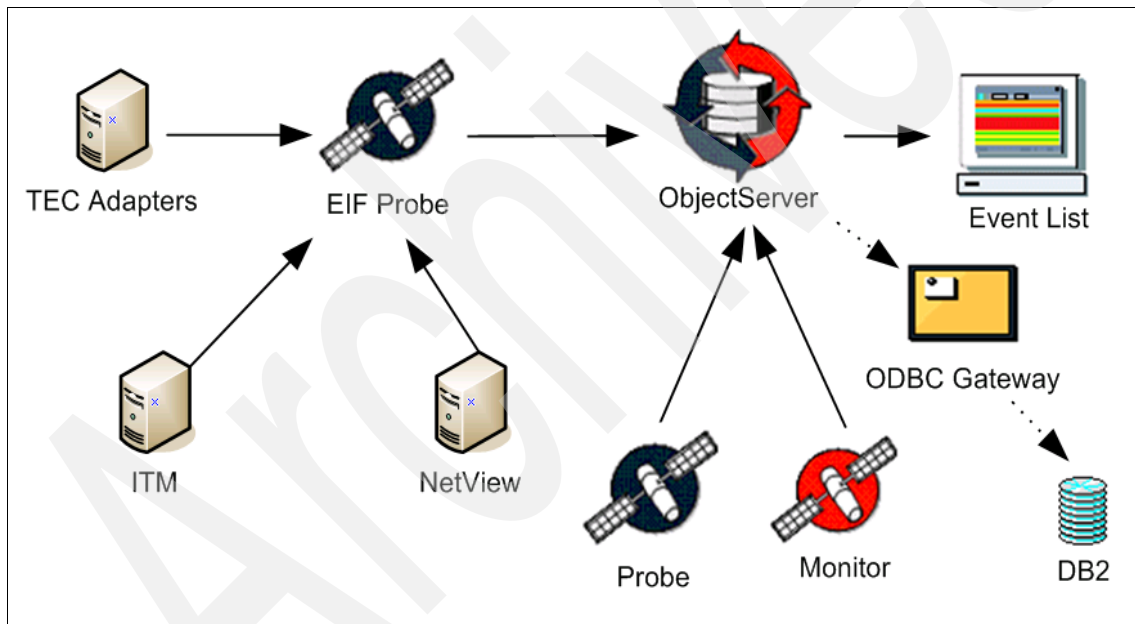


Figure 4-9 Desired final configuration of the customer's environment





## Part 3

# Implementation

In the final part of this book we provide three chapters that focus on providing detailed guidance, examples, and how-to for system architects, developers, and system administrators who need to plan and perform an upgrade from TEC to Netcool/OMNIbus. The chapters are:

- ▶ Chapter 5, “Upgrading to an IBM Tivoli Netcool environment” on page 165
- ▶ Chapter 6, “Event processing” on page 187
- ▶ Chapter 7, “Configuring the event sources” on page 277.





## Upgrading to an IBM Tivoli Netcool environment

This chapter describes upgrading from TEC to OMNIbus via the recommended upgrade scenario, as outlined in Chapter 4, “Upgrade strategies” on page 147.

This first section of this chapter describes the lab that was set up to represent a typical Tivoli event management environment, prior to testing and demonstrating the upgrade scenarios.

The second and subsequent sections focus on some key technical considerations when installing, configuring, and deploying a typical Netcool/OMNIbus environment, with references to the official installation and administration guides where appropriate.

# 5.1 Tivoli Enterprise Console prior to upgrade

The lab was set up to try to emulate a typical customer environment, with integration from typical event sources. For clarity of explanation and diagrams we have limited the scope to one per type of event source to demonstrate functionality. This is illustrated in Figure 5-1.

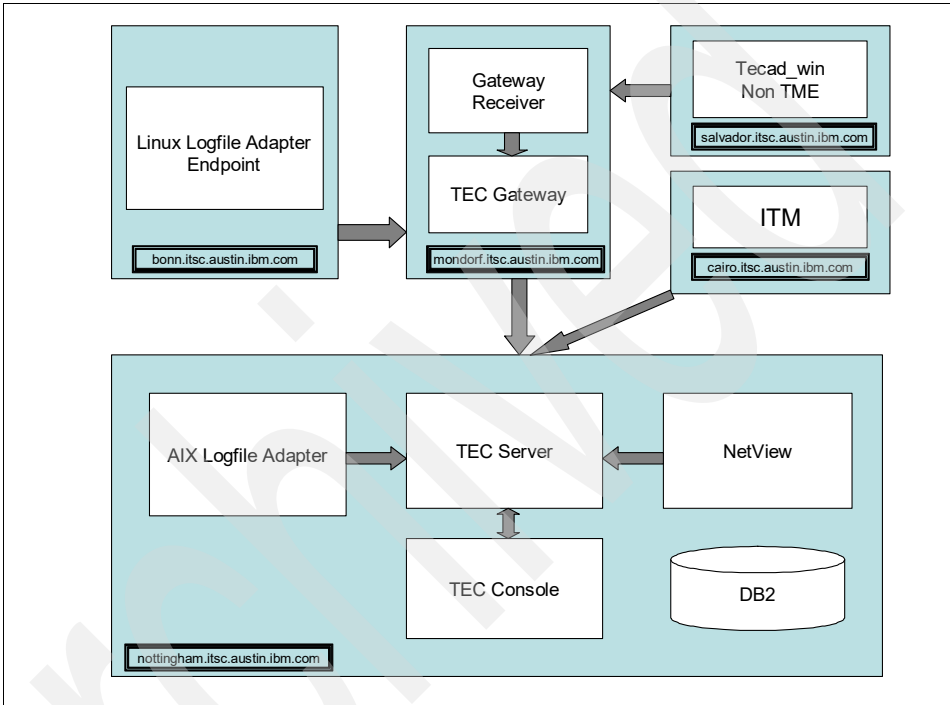


Figure 5-1 TEC event flows as set up in the lab

## 5.1.1 Installed TEC components

In this section we provide a series of tables that show a summary of the key components installed on each lab server, to build the typical environment prior to upgrade to OMNIbus.

Table 5-1 Host : nottingham.itsc.austin.ibm.com

Tivoli TMR server	AIX 5.3
Tivoli Framework	4.1.1 fix pack 07
TEC server, UI, Console, ACF	3.9.0 fix pack 06

<b>Tivoli TMR server</b>	<b>AIX 5.3</b>
TEC Logfile adapter for AIX	3.9.0 fix pack 06
DB2	8.2 fix pack 07a
NetView	7.1.5

*Table 5-2 Host : mondorf.itsc.austin.ibm.com*

<b>Tivoli managed node/gateway</b>	<b>Red Hat 4.0</b>
Tivoli Framework	4.1.1 fix pack 07
TEC ACF, Gateway	3.9.0 fix pack 06
TEC Gateway Receiver	3.9.0 fix pack 06

*Table 5-3 Host : bonn.itsc.austin.ibm.com*

<b>Tivoli endpoint</b>	<b>Red Hat 4.0</b>
Tivoli Framework	4.1.1 fix pack 07
TEC Logfile adapter for linux	3.9.0 fix pack 06

*Table 5-4 Host : salvador.itsc.austin.ibm.com*

<b>Non-Tivoli system</b>	<b>Windows 2003 server</b>
TEC Non-TME windows adapter	3.9.0 fix pack 06
TEC SNMP Adapter	3.9.0 fix pack 06

*Table 5-5 Host : cairo.itsc.austin.ibm.com (and server2.itsc.austin.ibm.com)*

<b>Server</b>	<b>AIX 5.3</b>
IBM Tivoli Monitoring	5.1.2, 6.1 and 6.2
EIF integration	6.1 and 6.2

The full lab environment is shown in Figure 5-2, with TEC, ITM, and OMNibus servers in place.

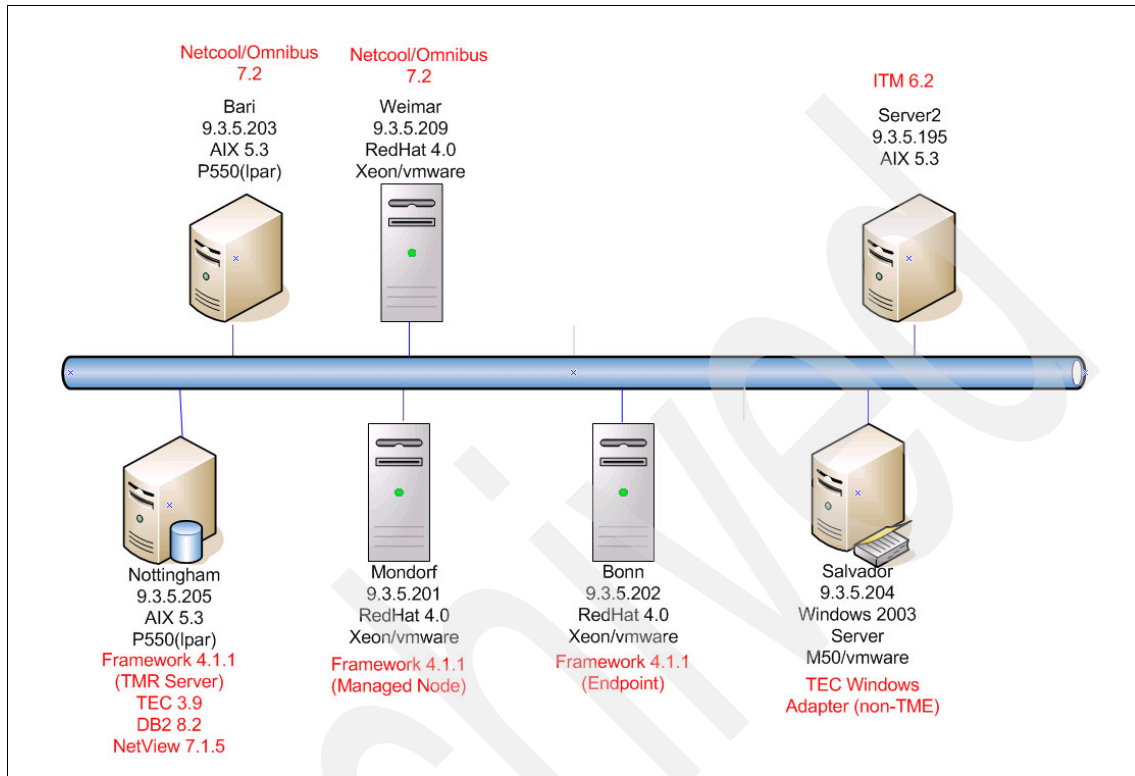


Figure 5-2 Lab environment

## 5.1.2 TEC installation and configuration

We assume that the reader is familiar with setting up a TEC environment, so we have not detailed those installation steps here. If you wish to refer to the process used and configuration files for reference purposes, these can be found in “TEC installation steps” on page 368 and “TEC event source generation commands and scripts” on page 372.

## 5.2 Netcool/OMNibus lab environment

In this section we describe our Netcool/OMNibus lab environment setup.

## 5.2.1 AIX lab environment for Netcool/OMNIbus

Figure 5-3 shows the lab environment for Netcool/OMNIbus on AIX.

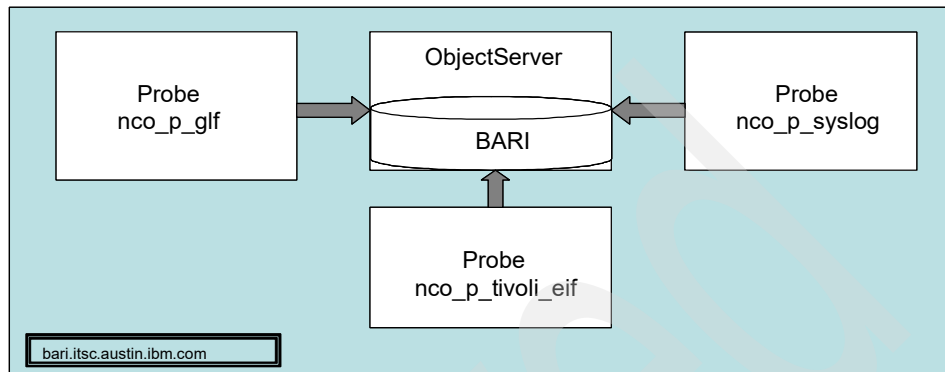


Figure 5-3 Lab environment for Netcool/OMNIbus on AIX

Table 5-6 Host : bari.itsc.austin.ibm.com

Netcool server 2	AIX 5.3
OMNIbus ObjectServer	7.2
non-native probe	2.0
EIF probe	2.1
syslog probe	4.3
Knowledge library	1.3b

## 5.2.2 Red Hat environment for Netcool/OMNIBus

Figure 5-4 shows the lab environment for Netcool/OMNibus on Red Hat.

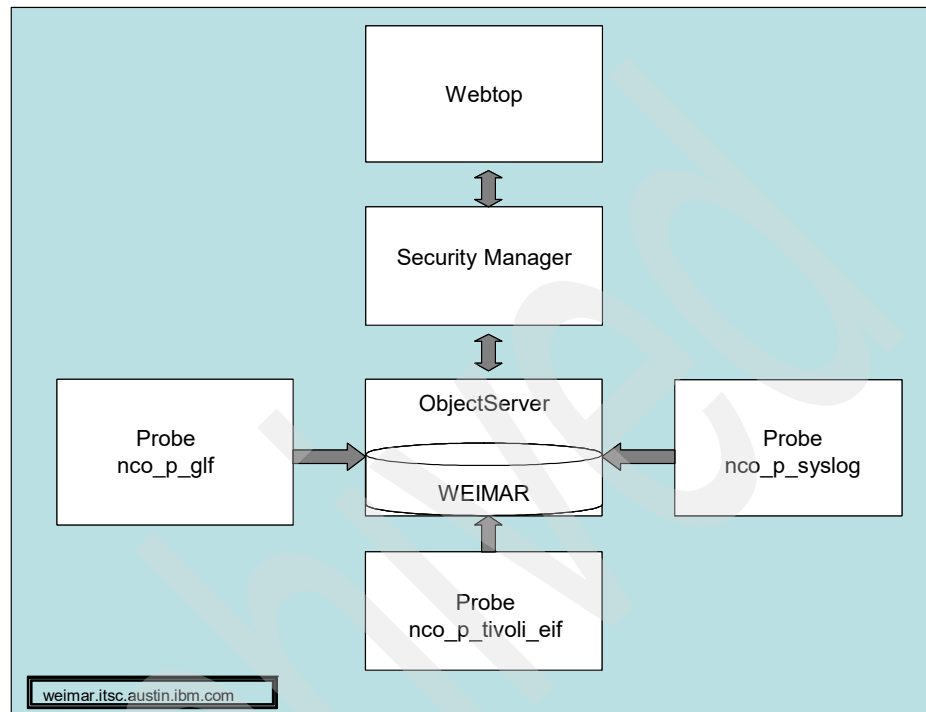


Figure 5-4 Lab environment for Netcool/OMNibus on Red Hat

Host information for the lab environment is shown in Table 5-7 and Table 5-8 on page 171.

**Note:** With Netcool/OMNIBus V7.2, Netcool/Security Manager V1.3, and Netcool/Webtop V2.1, the license server is no longer required. It is an ongoing process to remove licenses in the family of Netcool probes as well. Check that you have the very latest version available. The installation steps described in this chapter do not consider Flex licensing.

*Table 5-7 Host : weimar.itsc.austin.ibm.com*

<b>Netcool server 1</b>	<b>Red Hat 4.0</b>
OMNIBus ObjectServer	7.2
Non-native probe	2.0

Netcool server 1	Red Hat 4.0
EIF probe	2.1
Security manager	1.3
Webtop	2.1

Table 5-8 Host : salvador.itsc.austin.ibm.com

Netcool probe	Windows 2003 server
NT event log probe	7.2

## 5.3 Netcool/OMNIBus installation

Rather than repeat the information in the existing documentation, as this is liable to change with new version updates, we advise you to follow the core product documentation to acquire, install, configure, and deploy the Netcool OMNIBus products. You can begin this process here with the *Quick Start Guide for OMNIBus 7.2*, available online at the following location:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc\\_7.2.0/C145GEN.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc_7.2.0/C145GEN.pdf)

All of the latest product documentation can be found at the following location:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool\\_OMNIBus.doc\\_7.2.0/welcome.htm](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_OMNIBus.doc_7.2.0/welcome.htm)

## 5.4 IBM Tivoli Netcool/OMNIBus configuration

In this section we briefly describe the configuration of the OMNIBus ObjectServer performed on our AIX-based lab system.

### 5.4.1 ObjectServer database initialization

The “**nco\_dbinit**” command creates an ObjectServer database. ObjectServer databases are located under the /opt/netcool/omnibus/db directory. Without any parameters there will be an ObjectServer called NCOMS created. We recommend creating your own defined ObjectServer database name instead of using the default name NCOMS. The default name will be created automatically if no specific parameters are used with the command “**nco\_dbinit**”.

**Note:** ObjectServer names will be automatically truncated to 11 characters. Use the “**nco\_igen -notrunc**” command option for more than 11 characters.

Example 5-1 shows how to set up an ObjectServer with name BARI.

*Example 5-1 \$OMNIHOME/bin/nco\_dbinit*

---

```
# cd $OMNIHOME/bin/  
# ./nco_dbinit -server BARI
```

---

## 5.4.2 ObjectServer interfaces omni.dat

The file “\$NCHOME/etc/omni.dat” holds the base configuration for generating the interfaces file with the command “**\$NCHOME/bin/nco\_igen**”.

The connections data file is used to create the interfaces file for Netcool/OMNIbus ObjectServer communications. There might be occasions when you need to edit the connections file directly, for example, on UNIX systems that do not have a graphical interface.

Example 5-2 shows the interface configuration file omni.dat as an example of our lab environment. This example also shows a high-availability configuration for one primary and backup ObjectServer.

*Example 5-2 \$NCHOME/etc/omni.dat*

---

```
#  
# omni.dat file as prototype for interfaces file  
#  
# Ident: $Id: omni.dat 1.5 1999/07/13 09:34:20 chris Development $  
#  
[BARI]  
{  
    Primary:      bari 4100  
    Backup:       bari 4500  
}  
  
[BARI_GATE]  
{  
    Primary:      bari 4300  
}  
  
[BARI_PA]  
{  
    Primary:      bari 4200  
}
```



```
[BARI_PROXY]
{
    Primary:      bari 4400
}

```

---

### 5.4.3 Interfaces file generation

The command “**nco\_igen**” in Example 5-3 generates the interfaces file based on the configuration in the “omni.dat” described in Example 5-2 on page 172.

*Example 5-3* \$NCHOME/etc/omni.dat

---

```
# cd $NCHOME/bin
# ./nco_igen -arch aix5 -in /opt/netcool/etc/omni.dat

```

---

Figure 5-5 shows configuring and generating the interface file over the native GUI with the command “nco\_xigen”.

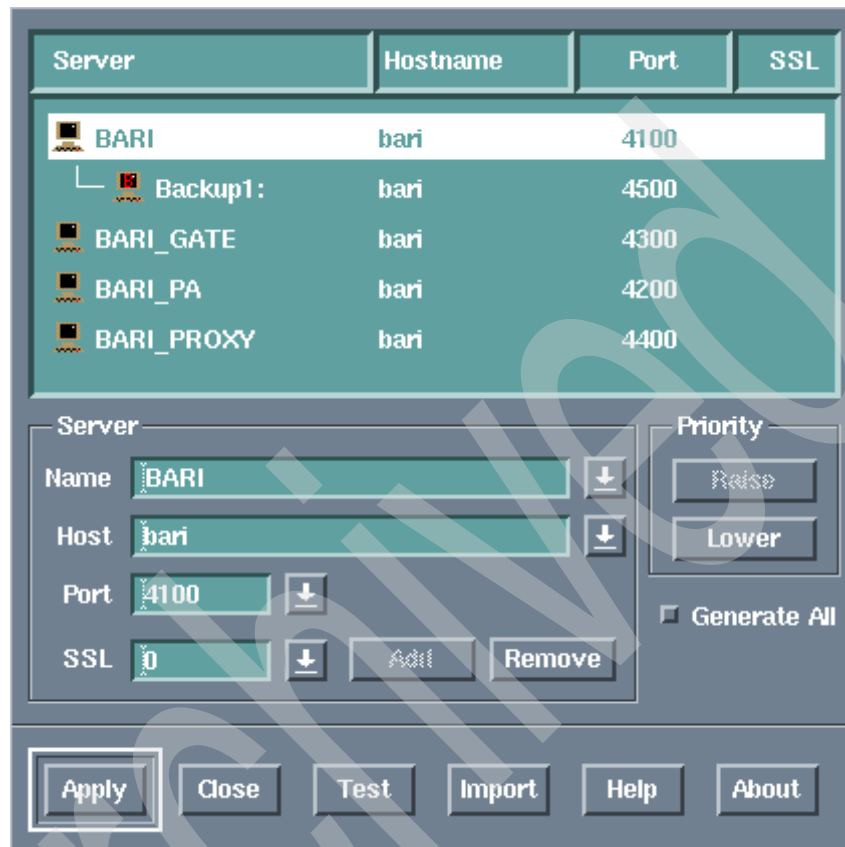


Figure 5-5 \$NCHOME/bin/nco\_xigen

#### 5.4.4 ObjectServer properties configuration

Configure the following parameters in /opt/netcool/omnibus/etc/BARI.props, as shown in Example 5-4, for the created ObjectServer called BARI.

Example 5-4 shows the ObjectServer configuration from the lab environment.

Example 5-4 \$OMNIHOME/etc/BARI.props

```
...
AlertSecurityModel: 0
AllowConnections: TRUE
AllowISQL: TRUE
AllowISQLWrite: TRUE
```

```
AllowTimedRefresh: FALSE
Auto.Debug: FALSE
Auto.Enabled: TRUE
Auto.StatsInterval: 60
BackupObjectServer: FALSE
Connections: 30
DTMaxTopRows: 100
DeleteLogging: FALSE
DeleteLogLevel: 0
DeleteLogSize: 1024
GWDeduplication: 0
Granularity: 60
Iduc.ListeningPort: 0
Ipc.SSLCertificate: ''
Ipc.SSLEnable: FALSE
Ipc.SSLPrivateKeyPassword: ''
MaxLogFileSize: 1024
Memstore.DataDirectory: '$OMNIHOME/db'
MessageLevel: 'debug'
MessageLog: '$OMNIHOME/log/BARI.log'
Name: 'BARI'
PA.Name: 'BARI_PA'
PA.Password: '6wJpmTtVj8G'
Profile: FALSE
ProfileStatsInterval: 60
PropsFile: '$OMNIHOME/etc/BARI.props'
RestrictPasswords: FALSE
RestrictProxySQL: FALSE
RestrictionUpdateCheck: TRUE
Sec.AuditLevel: 'warn'
UniqueLog: FALSE
...
```

---

### 5.4.5 Process Automation configuration

Netcool/OMNIBus process control agents may be deployed to manage the running of the ObjectServer, gateways, and probes. Processes may be started automatically on system startup. The agents will restart any process that stops unexpectedly. System administrators may stop and start the processes via the administrator client or from the command line.

**Note:** All processes configured under Process Automation should be controlled by process control (for example, "nco\_pa\_stop" and "nco\_pa\_start"). Process Automation has no control if processes are stopped and started manually, even if they are already configured within Process Automation.

### **Adding startscript “/etc/rc.nco\_pa” and entry in “/etc/inittab”**

The routine \$OMNIHOME/install/startup/aix5install generates the startscript “/etc/rc.nco\_pa” and adds the entry “nco:2:once:/etc/rc.nco\_pa > /dev/console 2>&1 # Start Netcool/OMNIbus in /etc/inittab”.

**Note:** Netcool/OMNIbus Process Automation is designed for controlling the components of ObjectServer, probes, and gateways.

Example 5-5 shows how to generate the “/etc/rc.nco\_pa startup” script for the Process Automation daemon and add an entry for the general process dispatcher for AIX in “/etc/inittab”.

*Example 5-5 Add the /etc/rc.nco\_pa and entry in /etc/inittab*

---

```
# cd $OMNIHOME/install/startup
# chmod 750 aix5install
# ./aix5install
```

---

**Note:** Consider whether the ObjectServer should run under a specific user (for example, Netcool). It needs to set up the Netcool environment and path variables in the startscript “rc.nco\_pa”.

Refer to “Netcool Process Automation startup script” on page 377, which shows the related entries for the “rc.nco\_pa” script in the “/etc/inittab” directory.

### **Configuring Process Automation**

Edit the Process Automation properties file for configuring process that need to be controlled by the Process Automation.

Example 5-6 shows the Process Automation configuration file.

*Example 5-6 \$OMNIBUS/etc/BARI\_PA.conf*

---

```
#NCO_PA3
#
# Process Agent Daemon Configuration File 1.1
#
```

---

```

# Ident: $Id: nco_pa.conf 1.3 2002/05/21 15:28:10 renate Development $
#

#
# List of processes
#
nco_process 'MasterObjectServer'
{
    Command '$OMNIHOME/bin/nco_objserv -name BARI -pa BARI_PA -propsfile
$OMNIHOME/etc/BARI.props' run as 10000
    Host = 'bari'
    Managed = True
    RestartMsg = '${NAME} running as ${EUID} has been restored
on ${HOST}.'
    AlertMsg = '${NAME} running as ${EUID} has died on
${HOST}.'
    RetryCount = 0
    ProcessType = PaPA_AWARE
}

nco_process 'nco_p_tivoli_eif'
{
    Command '$OMNIHOME/probes/nco_p_tivoli_eif -propsfile
$OMNIHOME/probes/aix5/tivoli_eif.props' run as 10000
    Host = 'bari'
    Managed = True
    RestartMsg = '${NAME} running as ${EUID} has been restored
on ${HOST}.'
    AlertMsg = '${NAME} running as ${EUID} has died on
${HOST}.'
    RetryCount = 0
    ProcessType = PaPA_AWARE
}

nco_process 'nco_p_syslog'
{
    Command '$OMNIHOME/probes/nco_p_syslog -propsfile
$OMNIHOME/probes/aix5/syslog.props' run as 10000
    Host = 'bari'
    Managed = True
    RestartMsg = '${NAME} running as ${EUID} has been restored
on ${HOST}.'
    AlertMsg = '${NAME} running as ${EUID} has died on
${HOST}.'
    RetryCount = 0
    ProcessType = PaPA_AWARE
}

nco_process 'nco_p_glf'

```

```

{
    Command '$OMNIHOME/probes/nco_p_glf -propsfile
$OMNIHOME/probes/aix5/glf.props' run as 10000
    Host = 'bari'
    Managed = True
    RestartMsg = '${NAME} running as ${EUID} has been restored
on ${HOST}.'
    AlertMsg = '${NAME} running as ${EUID} has died on
${HOST}.'
    RetryCount = 0
    ProcessType = PaPA_AWARE
}

#
# List of Services
#
nco_service 'Core'
{
    ServiceType = Master
    ServiceStart = Auto
    process 'MasterObjectServer' NONE
    process 'nco_p_tivoli_eif' 5
    process 'nco_p_syslog' 5
    process 'nco_p_glf' 5
}

#
# This service should be used to store processs that you want to temporarily
# disable. Do not change the ServiceType or ServiceStart settings of this
# process.
#
nco_service 'InactiveProcesses'
{
    ServiceType = Non-Master
    ServiceStart = Non-Auto
}

#
# ROUTING TABLE
#
# 'user' - (optional) only required for secure mode PAD on target host
# 'user' must be member of UNIX group 'ncoadmin'
# 'password' - (optional) only required for secure mode PAD on target host
# use nco_pa_crypt to encrypt.
nco_routing
{
    host 'bari' 'BARI_PA'
    host 'weimar' 'WEIMAR_PA'
}

```

```
}
```

---

## Process Automation remote execution configuration

For connecting remote Process Automation daemons you have to configure the configuration file in the section “nco\_routing”. Specify the remote host name and the name of the remote Process Automation. Consider that you have already configured the ObjectServer properties with the parameters “PA.NAME” and “PA.PASSWORD”.

### 5.4.6 ObjectServer startup

Start up the Netcool/OMNIBus ObjectServer over the system startup script and the Process Automation daemon. The ObjectServer also can be started manually on the command line.

Example 5-7 shows how to start the Process Automation daemon over the startup script.

*Example 5-7 Starting up ObjectServer over Process Automation*

---

```
# cd /etc
# ./rc.nco_pad
```

---

If you do not want to let the Netcool Process Automation start up and control the ObjectServer, just start it manually with the command **nco\_objserv**.

Example 5-8 shows how to start the ObjectServer manually.

*Example 5-8 Starting up Netcool/OMNIBus ObjectServer manually*

---

```
# cd $OMNIBUS/bin
# ./nco_objserv -name BARI -propsfile $OMNIBUS/etc/BARI.props &
```

---

Example 5-9 shows the processes of an up-and-running ObjectServer with all of its components. This example especially shows these processes up and running:

- ▶ “nco” native event GUI process
- ▶ “nco\_objserv” ObjectServer process
- ▶ “nco\_config” process for one administrative login
- ▶ “nco\_event” process for one up-and-running active event list native GUI

*Example 5-9 ps -ef|grep nco*

---

```
# ps -ef|grep nco
root 311326 520314 0 08:33:12 pts/2 0:00
/opt/netcool/omnibus/platform/aix5/bin/nco
```

```

root 323654 503886 1 12:07:15 pts/4 2:43
/opt/netcool/omnibus/platform/aix5/bin/nco_objserv -name BARI
root 352382 311326 2 08:33:21 pts/2 0:01
/opt/netcool/omnibus/platform/aix5/bin/nco_event
root 360592 520314 0 Oct 06 pts/2 8:50
/opt/netcool/platform/aix5/jre_1.5.4/jre/bin/java -classpath
/opt/netcool/omnibus/java/jars/oem_administrator.jar:/opt/netcool/omnibus/java/
jars/ControlTower.jar:/opt/netcool/omnibus/java/jars/hsqldb.jar:/opt/netcool/om
nibus/java/jars/jms.jar:/opt/netcool/omnibus/java/jars/log4j-1.2.8.jar:/opt/net
cool/omnibus/java/jars/jconn2.jar
-Djava.rmi.server.codebase=file:///opt/netcool/omnibus/java/jars/ControlTower.j
ar -Djava.security.policy=file:///opt/netcool/omnibus/etc/admin.policy
-Dnc.home=/opt/netcool -Domni.home=/opt/netcool/omnibus
-Domni.arch.dir=/opt/netcool/omnibus/platform/aix5
-Dtrusted.cert.file=/opt/netcool/platform/aix5/config/trusted.txt -Xms64m
-Xmx512m com.micromuse.centralconfig.LaunchApplication

```

---

### 5.4.7 ObjectServer shutdown

To shut down and stop the ObjectServer manually, on the command line, at any time, use the following described commands if you do not control the ObjectServer process automatically with Process Automation.

**Note:** To stop the ObjectServer again at any time use the “`nco_sql`” command.

Example 5-10 shows how to shut down and stop the ObjectServer process.

*Example 5-10 nco\_sql -server BARI -user root*

```

# cd $OMNIBUS/bin
# ./nco_sql -server BARI -user root
Password:
1> alter system shutdown
2> go
(0 rows affected)
1> quit

```

---

**Note:** Shutting down the ObjectServer does not cause the clients to exit. The client GUI processes are still shown in the process list on the ObjectServer.



## 5.5 IBM Tivoli Netcool probe installation overview

As with the ObjectServer, the detailed instructions to install the many different Netcool/OMNIBus probes available are best referred to via the IBM Information Center link provided in 5.3, “Netcool/OMNIBus installation” on page 171.

Probes connect to an event source, detect and acquire event data, and forward the data to the ObjectServer as alerts. Probes use the logic specified in a rules file to manipulate the event elements before converting them into fields of an alert in the ObjectServer alerts.status table.

**Note:** With the latest available versions of Netcool probes, the flex licensing is no longer required. Probes are installable without any prerequisite Netcool license server or any required license files.

### 5.5.1 What you need to know about nco\_patch

As a Netcool/OMNIBus ObjectServer component, Netcool probes will be installed as a patch over the “nco\_patch” mechanism of Netcool/OMNIBus. This mechanism requires the Netcool common installer to be used to first install the Netcool *middleware* and supporting libraries. This is an equivalent concept to the Tivoli framework, and is required once on each system that will have a Netcool component installed on it.

**Note:** An Netcool/OMNIBus base installation is a prerequisite for any probe installation, selecting no options. Just select **Process Control** if you need to be able to use process control and remote execution automation.

### 5.5.2 Toggle feature for process control

We recommend selecting and installing just the process control. It provides the function to configure Netcool-related components (for example, probes and gateways under Process Automation). Example 5-11 shows how to toggle just the feature process control.

*Example 5-11 Toggle feature process control*

---

Product: Netcool/OMNIBus

Select    Feature

-----

- [ ]    1) Desktop - Desktop GUI Applications
- [ ]    2) Gateways - ObjectServer Gateways

[I] 3) Process Control - Process control and remote execution support.  
 [ ] 4) Servers - ObjectServer and Proxy Server  
 [ ] 5) Confpack - Confpack configuration backup and transfer tool  
 [ ] 6) Administrator - Administrator configuration GUI  
 [ ] 7) AEN Client - Accelerated Event Notification Client  
 [ ] 8) Local Help System - Local On-line Help System. To use Standalone mode on-line help or start a Infocentre server, this feature must be installed.

Select:

- 1-8) Toggle feature
- s) Select all features
- u) Unselect all features
- i) Install selected features
- n) Next page (properties configuration)
- q) Quit

Option [i]: 3

Figure 5-6 show how to select just the feature process control in Installer wizard mode.

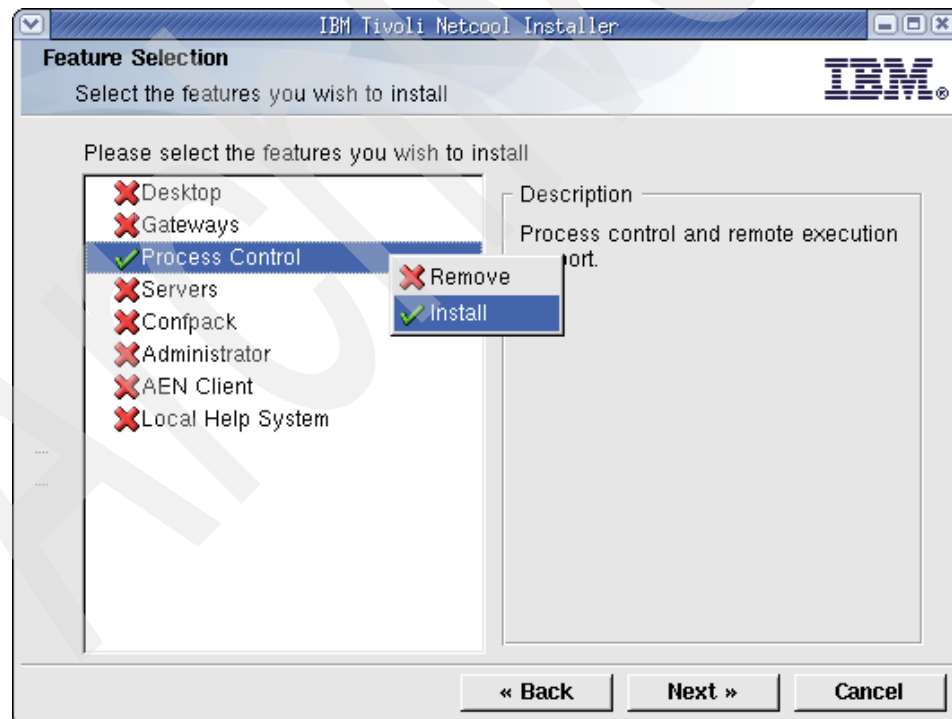


Figure 5-6 Wizard mode: toggle feature for process control

**Note:** If you do not want to install just process control, leave all features unselected here and go forward through the entire installation process. This way only the base components of Netcool/OMNibus will be installed.

Example 5-12 shows the next step in the text installer for installing just the process control.

*Example 5-12 Feature selection*

---

Product: Netcool/OMNibus

Select    Feature

-----

[ ]    1) Desktop - Desktop GUI Applications  
[ ]    2) Gateways - ObjectServer Gateways  
[I]    3) Process Control - Process control and remote execution support.  
[ ]    4) Servers - ObjectServer and Proxy Server  
[ ]    5) Confpack - Confpack configuration backup and transfer tool  
[ ]    6) Administrator - Administrator configuration GUI  
[ ]    7) AEN Client - Accelerated Event Notification Client  
[ ]    8) Local Help System - Local On-line Help System. To use Standalone mode on-line help or start a Infocentre server, this feature must be installed.

Select:

1-8) Toggle feature  
s) Select all features  
u) Unselect all features  
i) Install selected features  
n) Next page (properties configuration)  
q) Quit

Option [i]: n

---

### 5.5.3 Installation of probe for Windows NT event logs

The installation and configuration of this probe is a slightly different process and is covered in 7.5.1, “Installing and configuring the Windows NT Event Log probe” on page 323, which also covers other Windows issues and event source migration.

## 5.5.4 Check the probe installation

Example 5-13 shows how to check probe installation and the patch level.

*Example 5-13 \$OMNIHOME/install/nco\_patch -print*

```
# cd $OMNIHOME/install
# ./nco_patch -print
-----
INSTALLED PATCHES
-----
Patch probe-nco-p-glf-5
-----
    Short Description : Netcool/OMNIBus Generic Log File Probe
        Revision : 0
        Requires : probe-compatibility-3.x
        Obsoletes : probe-nco-p-glf probe-nco-p-glf-4
    Installation Date : Wed Oct 3 21:35:37 CEST 2007
-----
Patch probe-nco-p-syslog-4
-----
    Short Description : Syslog probe update
        Revision : 3
        Requires : probe-compatibility-3.x
        Obsoletes : probe-nco-p-syslog probe-nco-p-syslog-1
probe-nco-p-syslog-2 probe-nco-p-syslog-3
    Installation Date : Wed Oct 3 21:49:53 CEST 2007
-----
Patch probe-nco-p-tivoli-eif-2
-----
    Short Description : Netcool/OMNIBus Tivoli EIF probe
        Revision : 1
        Requires : probe-compatibility-3.x probe-nonnative-base-1
        Obsoletes : probe-nco-p-tme10tecad probe-nco-p-tme10tecad-2
probe-nco-p-tme10tecad-3 probe-nco-p-tme10tecad-4 probe-nco-p-tivoli-eif-1
    Installation Date : Wed Oct 3 21:59:11 CEST 2007
-----
Patch probe-nonnative-base-2
-----
    Short Description : Nonnative probe server
        Revision : 1
        Requires : probe-compatibility-3.x
        Obsoletes : probe-nco-p-nonnative probe-nco-p-nonnative-1
probe-nco-p-nonnative-2 probe-nco-p-nonnative-3 probe-nco-p-nonnative-4
probe-nco-p-nonnative-5 probe-nco-p-nonnative-6 probe-nco-p-nonnative-java-2
probe-nco-p-nonnative-java-1 probe-nco-p-nonnative-java
probe-nco-p-nonnative-scripts probe-nonnative-base-0 probe-nonnative-base-1
    Installation Date : Wed Oct 3 21:54:20 CEST 2007
```

---

## 5.5.5 Netcool probe configuration

To configure the Netcool probes it is necessary to edit the properties file of each probe. The default values are stored in the first section of the properties file. Append your own settings at the prepared section “Add your settings here” at the bottom of the file. It is also possible to start the probes by specifying required parameters at startup, without any stored configuration in a properties file.

At a minimum, the name of the ObjectServer needs to be configured (field name 'Server') if there is no default ObjectServer name NCOMS used. You can see all parameters available for a certain probe by using the -help function of the probe.

Example 5-14 shows sample configuration options that we appended to the properties file for the EIF probe, nco\_p\_tivoli\_eif.

*Example 5-14 \$OMNIHOME/probes/aix5/nco\_p\_tivoli\_eif -help*

---

```
MessageLevel : 'debug'
Server : 'BARI'
RawCapture : 1
Inactivity : 0      - specific to EIF probe , highly recommended for reliable
connectivity (rather than the default of 600)
```

---

## 5.6 Installing Netcool Security Manager and Netcool Webtop

Optionally, you can now also install Netcool Security Manager and Netcool Webtop, referring to the details provided in the online documentation:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool\\_OMNIBus.doc\\_7.2.0/welcome.htm](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_OMNIBus.doc_7.2.0/welcome.htm)

### Installation check

To check the installation packages in detail run the following command:

```
$NCHOME/install/ncpkg -query -audit
```

The output from this command will show the installed components and revision levels.

Archived

# Event processing

This chapter describes typical event-processing scenarios and discusses their implementation in both the TEC and the OMNibus environments. The following scenarios are covered:

- ▶ Handling duplicate events
- ▶ Filtering out events with specific content
- ▶ Actions for too many events in a defined time frame
- ▶ Filling an attribute dependent on another field's content
- ▶ Handling of cause, effect, and clearing events
- ▶ Propagating status change from cause to effect events
- ▶ Local and remote script execution
- ▶ Escalation of event severity
- ▶ Forwarding of events

We also discuss:

- ▶ Lookup files
- ▶ Extended attributes
- ▶ TEC class hierarchy support
- ▶ TEC Information button/ URL translation

## 6.1 Differences between TEC and OMNibus

Before we illustrate the migration of TEC rules to OMNibus automation, we must clarify the following topics.

### 6.1.1 Resolving of events

First of all, there are differences in the way TEC and OMNibus handle events. The main difference is in the way that events are resolved.

TEC resolves events by closing them. Closed events can then be further processed by the TEC maintenance tasks `Clear_Closed_Events` and `Clear_Reception_Log`, which delete old closed events from the event repository and the reception log.

The OMNibus philosophy for resolving events is to change the severity to clear. The terminology used is *the event is cleared*. After clearing, the standard maintenance trigger `generic_clears` deletes the events from the database.

### 6.1.2 Processing of events

Another important difference is the way events are processed. In TEC, the event arrives and is processed by the rules engine. The rule flow within the rulebase can be described as procedural processing.

Event processing in OMNibus is completely different, because OMNibus is based on Structured Query Language (SQL) that, as an example of Data Manipulation Language (DML), is non-procedural.



## 6.2 Event processing migration

Event processing migration is the main part of the upgrade process. The current TEC rulebase will need to be analyzed and then evaluated against the OMNibus implementation methods and tools as described Chapter 3, “TEC environmental assessment and planning guidelines” on page 103, and 3.1, “End-to-end event flow” on page 104. In general, we recommend that you follow a two-step process, as follows:

1. Look at the TEC rules in the current rulebase, and analyze and document the common event processing task.
2. Transform this specification to the OMNibus environment, create a definition, and deploy it.

Starting with 6.2.3, “Handling of duplicate events” on page 194, we provide details for selected migration scenarios, using the following analytical sequence:

- ▶ Common description of event processing task
- ▶ TEC implementation
- ▶ SCE implementation
- ▶ OMNibus implementation

In Figure 6-1 we present a comparative overview of the keywords used in TEC and the OMNIbus implementations. You can see that some logic from the TEC rules has moved to the probe rules, because OMNIbus has more capabilities in that area.

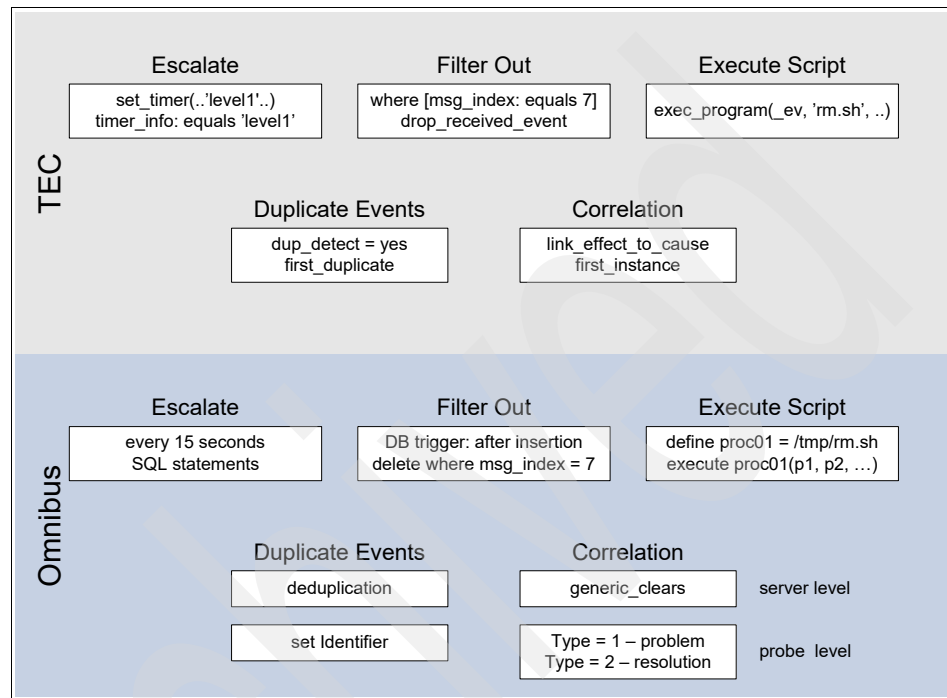


Figure 6-1 Rule processing comparison

Before we go into more detail let us consider some general suggestions.

## 6.2.1 General suggestions

Here are some general suggestions for Netcool OMNIbus event management:

- ▶ Try to handle your event management as close to the event source as possible.
- ▶ Use probe lookup files for event enrichment.
- ▶ Enrich the OMNIbus database schema *only* for the most commonly used attributes.
- ▶ Use the field ExtendedAttr in conjunction with the nvp\_\* functions to integrate specific TEC class attributes into one OMNIbus field.

- ▶ Pay attention to the field Identifier, which is used by the deduplication automation, when you transform the dup\_detect attributes from the TEC classes to the ObjectServer database schema.
- ▶ Keep the generic\_clears automation in mind when you migrate the TEC class attributes.

Let us look at the typical event management scenarios, starting with handling of duplicate events. Each one of the following chapters has a common description of the required event management and discusses the implementations in TEC, SCE (where possible), and OMNibus. We assume that OMNibus 7.2 is installed.

## 6.2.2 Lab environment

In this section we discuss the lab environment.

### TEC

For the creation of events we use either the TEC logfile adapter with a configured LogSources environment entry or the **wpostmsg** command. The LogSources entry points to the file /tmp/applications, whose content is now listed in Figure 6-2.

```
date time hostname applicationname errortype message
2007-09-27 09:01:02 host01 appl01 WARNING message01
2007-09-27 09:01:03 host01 appl01 MINOR message01
2007-09-27 09:01:04 host01 appl01 ERROR message01
```

Figure 6-2 Application log file example /tmp/applications

This application log file will be parsed with the format statements shown in Figure 6-3, which are used in the distribution section of a corresponding ACP profile.

```
FORMAT TEC_Notice
%s %s %s %s ERROR %s
date $1
sub_origin $2
hostname $3
sub_source $4
severity CRITICAL
msg $5
END

FORMAT TEC_Notice
%s %s %s %s MINOR %s
date $1
sub_origin $2
hostname $3
sub_source $4
severity MINOR
msg $5
END

FORMAT TEC_Notice
%s %s %s %s WARNING %s
date $1
sub_origin $2
hostname $3
sub_source $4
severity WARNING
msg $5
END
```

*Figure 6-3 Format file application.fmt*

In addition to the TEC\_Notice class, we use our own class file ups.baroc, which represents events coming from the monitoring of an uninterruptible power supply (UPS).

We assume that the reader is familiar with the configuration and distribution of ACP profiles, so we do not explain this in detail.

```
#-----#
# UPS.BAROC
#-----#
TEC_CLASS :
  UPS_ISA_EVENT
  DEFINES
  {
    source: default=SNMP;
  };
END

TEC_CLASS :
  UPS_Fan_Down ISA UPS
  DEFINES
  {
    severity: default=WARNING;
    hostname: dup_detect=yes;
  };
END

TEC_CLASS :
  UPS_Temp_Degraded ISA UPS
  DEFINES
  {
    severity: default=WARNING;
    hostname: dup_detect=yes;
  };
END

TEC_CLASS :
  UPS_Fan_Up ISA UPS
  DEFINES
  {
    severity: default=HARMLESS;
  };
END
```

Figure 6-4 BAROC class file ups.baroc

The classes UPS\_Fan\_Down and UPS\_Temp\_Degraded have the facet dup\_detect turned on so that the slot host name in these classes can be used to detect duplicate events.

## SCE

Rules for the SCE are sent to the endpoint of a TEC gateway with the distribution of a tec\_gateway\_sce ACP profile. In this profile you must configure the configuration parameters:

UseStateCorrelation=YES

StateCorrelationConfigURL=file:\$TIVOLIHOME/tec/tecroot.xml

Also, adjust the distribution location of the tecroot.xml file.

## OMNibus

In OMNibus we are using the same application log file that we used with the TEC, but this time in conjunction with the generic logfile probe (glf). In addition, we use the **nco\_sql** command to send events manually to the ObjectServer.

The configuration files we used are located in the directory /opt/netcool/omnibus/probes/linux2x86 and are named glf.props and glf.rules. They are discussed later in detail. To start the glf probe you can use **nco\_p\_glf** from the directory /opt/netcool/omnibus/probes. To stop the probe you can kill the process.

In the following scenarios you will find various configurations of the generic logfile probe rules that process our logfile.

### 6.2.3 Handling of duplicate events

In this section we discuss the handling of duplicate events.

#### Common description of the scenario

Here duplicate events are sent repeatedly from an event source. Only a single event containing the newest event information should be seen on the event console.

#### TEC implementation

A BAROC class with the dup\_detect modifier for specific slots must be defined first. Below are two possible solutions:

- Create a specific rule for this event class or a generic rule for all event classes. Define a reception action that updates the older event with current

information from the new duplicate event, and then drops the newly arrived event.

- Create a specific rule for this event class or a generic rule for all event classes. Define a reception action that closes the older duplicate event and leaves the newly arrived event untouched.

### First solution

In Figure 6-5 we developed the following rule.

```
rule: filter_duplicate_ups:
(
  description: 'Filter duplicates for UPS events',
  event:_event of_class 'UPS'
    where [ msg: _new_msg ],
  action: filter:
    (
      first_duplicate(_event, event: _dup_ev
        where
          [
            status: outside ['CLOSED']
          ],
          _event - 600 - 600
        ),
      add_to_repeat_count(_dup_ev, 1),
      set_event_message(_dup_ev, '%s', [_new_msg]),
      drop_received_event
    )
).
```

Figure 6-5 TEC rule file lab.rls for duplicate event processing

For testing purposes we use the following commands:

```
wpostemsg -m test1a hostname=mondorf UPS_Fan_Down TEC
wpostemsg -m test1b hostname=mondorf UPS_Fan_Down TEC
```

After this we can see the result in the TEC console in Figure 6-6.

	Time Re...	Event Type	Class	Hostname	Severity	Status	Message	Repeat count
1	10/12/07 09:42	Other	UPS_Fan_Down	mondorf	Warning	Open	test1b	1

Figure 6-6 TEC console with duplicated event

### **Second solution**

This solution is not illustrated here because it does not offer any additional information.

### **Implementation with the SCE**

Define a duplicate rule with an appropriate time interval.

#### **Solution**

In Figure 6-7 you can see a possible solution for our issue.

```
<rule id="itso.rule01">
  <eventType>UPS_Fan_Down</eventType>
  <duplicate timeInterval="15000">
    <predicate>
      <![CDATA[
        true
      ]]>
    </predicate>
  </duplicate>
  <triggerActions>
    <action function="TECSummary" singleInstance="false"/>
  </triggerActions>
</rule>
```

*Figure 6-7 SCE duplicate XML rule*

In this example the first event is forwarded and the following events are discarded until the time frame is reached. Then the counting will start again.

We tested this rule with the same events with which we tested the TEC rule. The difference in this test is that we use the event flow through the SCE. You do this on the TEC server by activating the LCF environment with **lcf\_env.sh** and by using the endpoint **wpostemsg** command, which is located in **\$LCF\_BINDIR/../bin**.

### **OMNIBus implementation**

The standard deduplication should be used.

#### **Solution**

We created the following SQL statement and executed it with **nco\_sql** three times:

```
insert into alerts.status (Identifier, Summary, Node, Agent, Type,
Severity, AlertKey, AlertGroup, Manager, LastOccurrence) values
```



```
('itso-corr-01', 'Fan down on UPS ups_atl', 'mondorf', 'Agent', 1, 2,
'UPS', 'SNMP', 'nco_sql', getdate());
```

The result of the execution is shown in Figure 6-8.

Node	Alert Group	Summary	Last Occurrence	Count	Type	ExpireTime	Agent	Manager
mondorf	SNMP	Fan down on UPS ups_atl	9/28/2007 8:58:05 AM	3	Problem	Not Set	Agent	nco_sql

Figure 6-8 OMNibus event list for duplicate events

The processing for duplicate events is done by the default deduplication database trigger, which is listed in Figure 6-9.

```
begin
    set old.Tally = old.Tally + 1;
    set old.LastOccurrence = new.LastOccurrence;
    set old.StateChange = getdate();
    set old.InternalLast = getdate();
    set old.Summary = new.Summary;
    set old.AlertKey = new.AlertKey;
    if (( old.Severity = 0) and (new.Severity > 0))
    then
        set old.Severity = new.Severity;
    end if;
end;
```

Figure 6-9 Standard OMNibus deduplication database trigger

This trigger fires when an event with the same identifier content is to be reinserted. This automation updates several attributes of the existing event, either with information from the new event or with system information.

## 6.2.4 Filtering out events with specific content

In this section we discuss filtering out events with specific content.

### Common description of the scenario

An event coming from server mondorf should be shown on the console, except when the msg is equal to “filter-out”.

## TEC implementation

Two different solutions are possible:

- ▶ Create a filter for the logfile adapter. Define the appropriate class and compare the source for the msg field to “filter-out”.
- ▶ Create a specific rule for this event source. Define a reception action that compares the host name to mondorf and the msg to “filter-out”. If both are true, then the incoming event is dropped.

### **First solution**

Only a filter has to be defined in the ACP profile and distributed to the server named mondorf. See Figure 6-10.

```
Filter:Class=TEC_Notice;msg=filter-out;
```

Figure 6-10 *tecad\_logfile.conf* - filter statement

### **Second solution**

Figure 6-11 shows the solution for this event management.

```
rule: filter_out:
(
  description: 'Filter special event out',
  event:_event of_class 'UPS'
  where
  [
    hostname: equals 'mondorf',
    msg: equals 'filter-out'
  ],
  reception_action: filter:
  (
    drop_received_event
  )
).
```

Figure 6-11 *TEC rule for filtering out special events*

The test was done with the following **wpostemsg** command:

```
wpostemsg -m filter-out hostname=mondorf UPS_Fan_Down TEC
```

Because the event does not arrive at the event console, we cannot provide a meaningful graphic here.

## Implementation with the SCE

Define a match rule with an appropriate comparison to the content of the host name and msg.

### **Solution**

To filter out specific events at the SCE we provide Figure 6-12.

```
<rule id="itso.rule02">
  <eventType>UPS_Temp_Degraded</eventType>
  <match>
    <predicate>
      <![CDATA[
        &hostname == "mondorf" &&
        &msg == "filter-out"
      ]]>
    </predicate>
  </match>
  <triggerActions>
    <action function="Discard" singleInstance="true"/>
  </triggerActions>
</rule>
```

Figure 6-12 SCE rule for filtering specific events

Here all events from the TEC Class UPS\_Temp\_Degraded that have “mondorf” in the host name and “filter-out” in the msg slot are discarded.

## OMNIbus implementation

To implement OMNIbus:

1. Create a rules file for the probe. Compare the host name to “mondorf” and the msg field to “filter-out”. If it is true, then discard the event.
2. Create a database trigger that cancels the event insertion if the msg attribute is equal to "filter2b" and if the host name is equal to "mondorf".
3. If you want to document all events in a reporting database and that non-wanted events will automatically deleted after they have arrived, fill the field ExpireTime at the probe level for the unwanted ones. These events will automatically be cleared after the defined time frame with the standard expire automation. To prevent the event from being deleted before the gateway has stored it in the reporting database, modify the expire automation. First mark the event in the gateway AFTER IDUC command that the gateway has stored

the event and then delete only those events in the expire automation that has been marked.

4. In the case that the event should go to a log file, write an audit log at the ObjectServer and cancel the event insertion.

### ***First solution***

Create a rules file for the probe. Compare the host name to “mondorf” and the msg field to “filter-out”. If it is true, then discard the event.

This solution concerns the glf.props and the glf.rules files. Both files are responsible for controlling the behavior of the probe. Let us look at the props file (Figure 6-13).

Server	:	"WEIMAR"
LogFileName	:	"/tmp/applications"

*Figure 6-13 Used configuration of the generic logfile probe - glf.props*

Here we listed only the name of the ObjectServer and the location of the logfile that should be observed. All other standard fields we left untouched. The more interesting file is glf.rules, where the rules of the probe are located. We did not modify the standard section. Our modifications are shown in Figure 6-14.

```
@AlertGroup = "applications"
@Agent = "logfile"
@Node = $FieldVal03
@NodeAlias = "itso"
@Manager = "glf"
@Summary = $Details
@Type = 9
@Identifier = @Manager + @AlertGroup + $FieldVal02
#
# defining the Type and Severity
#
if( match( $FieldVal04, "appl01" ) )
{
    switch($FieldVal05)
    {
        case "ERROR" : @Severity = 5
                        @Type = 1
        case "MINOR" : @Severity = 4
                        @Type = 1
        case "WARNING": @Severity = 3
                        @Type = 2
        default:       @Severity = 1
                        @Type = 2
    }
}
#
# Filter out special events
#
if(match($FieldVal06, "filter-out") and match($FieldVal03,
"mondorf"))
{
    discard
}
```

Figure 6-14 Probe rule - glf.rules

In the upper part we define how the fields of the ObjectServer should be filled with constants or with the parsed input fields of the observed logfile. The content of field four controls the setting of the severity (5 - critical, 4 - major, or 3 - minor).

The content of the five field sets the event type (1 - problem or 2 - resolution). The last part checks, depending on the content of six field, whether the event should be discarded. To test this behavior, we append the next line to /tmp/applications:

```
2006-12-10 17:01:14 mondorf appl01 ERROR filter-out
```

The probe processed this event and discarded it.

### **Second solution**

Create a database trigger that deletes the event after insertion if the msg attribute is equal to “filter-out” and if the host name is equal to “mondorf”.

This solution is located at the ObjectServer. Here we had to create a new database trigger with the following settings:

- ▶ Priority = 1
- ▶ Pre database action
- ▶ Apply to row
- ▶ Fire on insert
- ▶ Action:  

```
if (new.Node = 'mondorf' and new.Location = 'filter2b') then cancel;  
endif;
```

To execute this SQL statement, we had to extend the file glf.rules with the assignment of \$FieldVal06 to Location if the content of \$FieldVal06 is equal to filter2b. Figure 6-15 shows the appropriate part of the rules file.

```
if(match($FieldVal06, "filter2b"))  
{  
  @Location = $FieldVal06  
}
```

*Figure 6-15 Code extract filter2b from glf.rules*

To test this part of the probe rule and the ObjectServer automation described above we appended the three lines shown in Figure 6-16 to the file /tmp/applications.

```
2007-09-28 11:01:16 mondorf appl01 MINOR filter-in
2007-09-28 11:01:17 mondorf appl01 MINOR filter2b
2007-09-28 11:01:18 mondorf appl01 MINOR filter-in
```

Figure 6-16 Application logfile lines for the automation itso\_filter2b

After this we see the event view shown in Figure 6-17.

Node	Alert Group	Summary	Last Occurrence	Count
mondorf	applications	2007-09-28 11:01:16 mondorf appl01 MINOR filter-in	9/28/2007 10:25:26 AM	1
mondorf	applications	2007-09-28 11:01:18 mondorf appl01 MINOR filter-in	9/28/2007 10:25:26 AM	1

Figure 6-17 Result of the filter automation

Here we can see that the event with the time stamp 11:01:17 did not show up at the event console.

### Third solution

If you want to document all events in a reporting database and have non-wanted events automatically deleted after they have arrived, fill the ExpireTime field at the probe level for the unwanted ones. These events will automatically be cleared after the defined time frame with the standard expire automation. To prevent the event from being deleted before the gateway has stored it in the reporting database, modify the expire automation. First mark the event in the gateway AFTER IDUC command in which the gateway has stored the event and then delete only those events in the expire automation that have been marked.

For this solution we extend our glf.rules file such that we insert a condition based on the six field of our application log file. If this contains filter-expire, then the Expire field of the event should be filled with 30. Figure 6-18 shows this part of code.

```
if(match($FieldVal06, "filter-expire"))
{
  @ExpireTime = 30
}
```

Figure 6-18 Code extract expire from glf.rules

Every minute the ObjectServer runs the expire automation, which updates the severity of the event if the lifetime of the events has expired. You can see this function in Figure 6-19. The function getdate() gets the actual system time.

```
for each row expire in expires
begin
update alerts.status via expire.Identifier set Severity = 0 where
LastOccurrence < (getdate() - expire.ExpireTime) and Location =
'marked';
end;
```

Figure 6-19 Standard automation expire

This function has an evaluation statement that looks like Figure 6-20.

```
select Identifier, ExpireTime from alerts.status where ExpireTime >
0 and Severity > 0
```

Figure 6-20 Selection statement for the expire trigger

To test this, we appended the following line in our applications file:

2007-09-28 13:01:18 mondorf appl01 MINOR filter-expire

The results can be seen on the event console, as shown in Figure 6-21.

Node	Alert Group	Summary	Last Occurrence	Count	Type	Location	ExpireTime
mondorf	applications	2007-12-12 07:42:09 mondorf appl01 MINOR filter-expire	12/11/2007 10:05:04 AM	1	Problem	marked	30

Figure 6-21 Result of the expire event

To simulate the gateway functionality we provide an additional menu with the tool contents shown in Example 6-1.

Example 6-1 Tool contents of additional menu to simulate gateway

```
update alerts.status set Location = 'marked' where Serial in (
$selectd_rows.Serial );flush iduc;
```

After 60 seconds the severity of this event changes to 0 - clear (color green).

**Fourth solution**

In the case that the event should go to a log file, write an audit log at the ObjectServer and cancel the event insertion.



This is similar to the second solution. We only had to add the instructions to write to a predefined logfile. This logfile is defined in Figure 6-22.

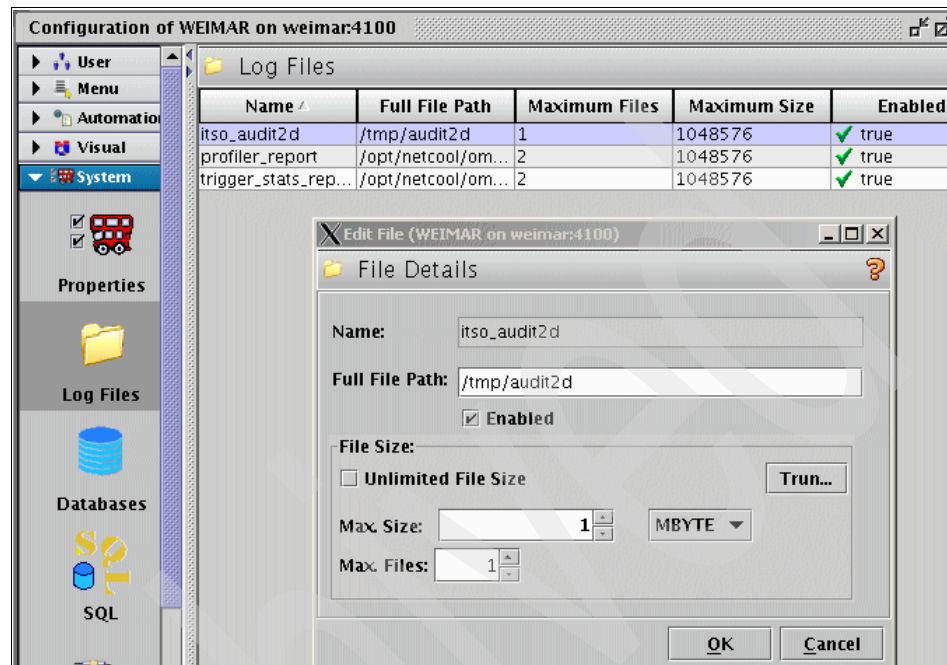


Figure 6-22 Definition of a ObjectServer logfile

After this definition is entered the ObjectServer has to be recycled. Now we can use the logfile in our new automation, shown in Figure 6-23.

```
if ((new.Node = 'mondorf') AND (new.Location = 'filter2d'))
THEN write into itso_audit2d values ('audit-content=', new.Summary);
cancel;
end if;
```

Figure 6-23 ObjectServer automation itso\_filter2d

To store the word filter2d in the Location field we had to extend the glf.rules file as shown in Figure 6-24.

```
if(match($FieldVal06, "filter2b") OR match($FieldVal06, "filter2d"))
{
  @Location = $FieldVal06
}
```

Figure 6-24 Code extract of glf.rules

To test the extended glf.rules file and the automation itso\_filter2d we appended the following line to our application logfile /tmp/applications:

```
2007-09-28 15:01:17 mondorf appl01 MINOR filter2d
```

We do not see any additional event in the event console because the incoming event is deleted after it is stored in the file /tmp/audit2d1. This file contains the line:

```
audit-content= 2007-09-28 15:01:17 mondorf appl01 MINOR filter2d
```

Now we have considered all four solutions for filtering out events with a specific content.

## 6.2.5 Actions for too many events in a defined time frame

In this section we discuss actions for too many events in a defined time frame.

### Common description of the scenario

If one event type arrives five times in 3 minutes, then raise the severity to critical.

### TEC implementation

Create a specific rule for this event class. A reception action, triggered by the new duplicate event, searches for an older duplicate event and examines its arrival time and repeat count. If the older event is still newer than 3 minutes and the repeat count is 3 (because there were already an original event plus 3 duplicates), then set the severity of the older event to critical and increase its repeat\_count by one. Drop the new duplicate event.

If the repeat\_count of the older event is not 3, then add one to repeat\_count. Drop the new duplicate event.

If the older event is older than 3 minutes, then close the older event and leave the new event untouched.

## Solution

Here we find the solution shown in Figure 6-25 using a TEC rule.

```
rule: dup_escalate_5_in_3_minutes:
(
  event: _event of_class within [ 'UPS' ],
  reception_action:
  (
    first_duplicate( _event,
                    event: _dup_event
                    where [
                        status: outside ['CLOSED'],
                        repeat_count: _rc
                    ],
                    _event - 180 - 0),
    add_to_repeat_count(_dup_event, 1),
    drop_received_event,
    _rc == 3,
    set_event_severity(_dup_event, 'CRITICAL')
  )
).
```

Figure 6-25 TEC rule for too many events in a defined time frame

This rule is very similar to the one in the duplicate section. The main differences are the two lines containing the word `repeat_count` and the line that raises the event severity. To be sure that this rule will work properly in the content of the entire rulebase, we must replace the old duplication rule in the rulebase with this new one.

The first `repeat_count` line stores the `repeat_count` in the variable `_rc`, and the second line compares the content of this variable to 3. Only if this comparison is true will the rest of the rule (here the raising of the event's severity) be executed. If it fails, the rule will terminate.

To test this rule we send the following **wpostmsg** commands:

```
wpostmsg -m event#1 hostname=mondorf UPS_Fan_Down TEC
wpostmsg -m event#2 hostname=mondorf UPS_Fan_Down TEC
wpostmsg -m event#3 hostname=mondorf UPS_Fan_Down TEC
wpostmsg -m event#4 hostname=mondorf UPS_Fan_Down TEC
wpostmsg -m event#5 hostname=mondorf UPS_Fan_Down TEC
```

After the last event arrives, the severity is set to critical, as you can see in Figure 6-26.

	Time Received	Event Type	Class	Hostname	Severity	Status	Message	Repeat count
	9/28/07 16:00	Other	UPS_Fan_Down	mondorf	Critical	Open	event#1	4

Figure 6-26 TEC console for too many events in a defined time frame

## Implementation with the SCE

Define a threshold rule with an appropriate time interval.

### Solution

Here we developed a SCE rule that fires against the TEC class UPS\_Temp\_Degraded, collects all events in the defined time frame, and at the end of the time frame sends an event with the filled attribute repeat\_count.

```
<rule id="itso.rule03">
  <eventType>UPS_Temp_Degraded</eventType>
  <threshold thresholdCount="5" timeInterval="180000"
triggerMode="allEvents">
    <predicate>
      <![CDATA[true]]>
    </predicate>
  </threshold>
  <triggerActions>
    <action function="TECSummary" singleInstance="false"/>
  </triggerActions>
</rule>
```

Figure 6-27 SCE threshold XML rule

We tested this rule with the wpostmsg of the endpoint as in the filtering section. The TEC event console looks similar to Figure 6-26 except for the TEC class name.

## OMNIbus implementation

Use the function's updateload and geteventcount at the probe level to develop a standard *X in Y* solution.

**Solution**

In this solution we use an array to solve this requirement. Let us look at the developed code before we explain it.

```
%time_window = 60
%event_buffer_size = 5
if(match(loads[@Node], "" ))
{
    loads[@Node] = %time_window + "." + %event_buffer_size
}
loads[@Node] = updateload(loads[@Node])
%myeventcount = geteventcount( loads[@Node] )
if (int(%myeventcount) >= 3)
{
    @Summary = @Summary + ": Occurred " + %myeventcount + " times in "
+ %time_window + " seconds"
}
```

Figure 6-28 Probe X in Y solution with updateload and geteventcount

The first six lines define the sliding time window and the maximum of events. For a detailed description see 6.3.1, “Measuring load and numbers of events in a time frame” on page 242.

The updateload line refreshes the counting and the eventcount extracts the account number. If the number of events is greater than three, the Summary field is updated.

We tested this function by appending five lines to our application log file.

On the OMNibus event console the result looks like Figure 6-29.

Node	Alert Gro...	Summary	Last Occurrence ▾
Mondorf	Linux	2007-10-09 13:44:28 Mondorf appl01 MINOR test:33: Occurred 5 times in 60 sec...	10/9/2007 12:44:11 PM

Figure 6-29 OMNibus event list for X in Y solution

**6.2.6 Filling an attribute dependent on another field’s content**

In this section we discuss filling an attribute dependent on another field’s content.

**Common description of the scenario**

Based on the naming conventions of servers, the field sub\_source/Location is filled with the operating system name.

## TEC implementation

A generic rule is triggered by all arriving events. In a reception action, the host name is parsed to find the portion that provides a key to the operating system (for example, S12345678 is a Windows system). Based on that key, the field sub\_source is filled with the value “Windows”.

### Solution

To develop a simple rule without extending our BAROC classes, we fill the field sub\_source with the string “Windows” if the first letter of the host name is “M” (for the Mondorf machine) or we fill it with “Linux” if the first letter is “S” (for the machine Salvador). In Figure 6-30 you can find the corresponding rules code.

```
rule: parse_hostname_set_sub_source:
(
  event: _event of_class _class
    where [
      hostname: _hostname outside ['']
    ],
  reception_action:
  (
    atompart(_hostname,_hostbegin,1,1),
    (
      _hostbegin == 'M' ->
      bo_set_slotval(_event,sub_source,'Linux');
      (
        _hostbegin == 'S' ->
        bo_set_slotval(_event,sub_source,'Windows')
      )
    )
  )
).
```

Figure 6-30 TEC rule - filling fields

For testing this rule we used the following **wpostemsg** commands:

```
wpostemsg -m test-4 hostname=Mondorf UPS_Temp_Degraded TEC
wpostemsg -m test-4 hostname=Salvador UPS_Temp_Degraded TEC
```

In the event console we can see the result of our rule.


	Time Received	Event Type	Class	Hostname	Severity	Status	Message	Sub-source
	10/1/07 15:39	Other	UPS_Temp_Degraded	Mondorf	Warning	Open	test-4	Linux
	10/1/07 15:39	Other	UPS_Temp_Degraded	Salvador	Warning	Open	test-4	Windows

Figure 6-31 TEC console - filled fields

## OMNibus implementation

To implement:

1. Use lookup tables at the probe level to resolve this issue.
2. Use the extended attributes if you do not want to extend the database schema of the ObjectServer.

### First solution: lookup tables

Here we use lookup tables at the probe level to resolve this issue.

Lookup tables can be defined either inline or externally in flat files. In Figure 6-32 you can see both types.

```
table machine_table = {"Mondorf", "Linux"},
                      {"Salvador", "Windows"}}
table machine_txt = "/tmp/machine.txt"
...
@Agent=lookup(@Node,machine_table)
@Manager=lookup(@Node,machine_txt)
```

Figure 6-32 Probe rule - lookup tables

Here the lookup table consists of two lines and two columns. Each line defines a machine name and the corresponding operating system. The two items are separated with a tab. To access this table you use the lookup keyword. The first parameter of this function contains the search argument and the second the table name. If the search is successful the result is placed as an assignment to the left variable. The referenced file looks like Figure 6-33.

```
Mondorf LINUX
Salvador    WINDOWS
```

Figure 6-33 Probe rule - lookup table file /tmp/machine.txt

Note that there are tabs between the machine name and the operating system type.

**Note:** To improve the look of the events in the EventList, customers often use lookup tables, putting the machine host name into the field NodeAlias depending on the IP address in the Node field. A common way to achieve this is to use a recently refreshed copy of the file /etc/hosts.

Alternatively, you can fill the NodeAlias field *dynamically* with a DNS lookup, but be sure that this is very fast and reliable, or probe performance will suffer dramatically.

Another way of implementing this request is the use of regular expressions. These are shown in Figure 6-34.

```
if (regmatch(@Node,"^Mondorf"))
{
  @AlertGroup = "Linux"
}
if (regmatch(@Node,"^Salvador"))
{
  @AlertGroup = "Windows"
}
```

Figure 6-34 Probe rule - use of regular expressions

To test all these rules types, we appended the two lines shown in Figure 6-35 to our /tmp/applications file.

```
2007-10-01 15:01:03 Mondorf appl01 MINOR test-4
2007-10-01 15:02:03 Salvador appl01 MINOR test-4
```

Figure 6-35 Logfile - /tmp/applications

In Figure 6-36 we see the OMNibus event list.

Node	Alert Group	Agent	Manager	Summary	First Occurrence
Mondorf	Linux	Linux	LINUX	2007-10-01 15:01:03 Mondorf appl01 MINOR test-4	10/1/2007 3:47:08 ...
Salvador	Windows	Windows	WINDOWS	2007-10-01 15:02:03 Salvador appl01 MINOR test-4	10/1/2007 3:47:08 ...

Figure 6-36 OMNibus event console with lookup table results

**Note:** Remember that we have reused existing ObjectServer fields in this example. That is *not* standard. You have to consider whether you want to extend the OMNibus database schema or use extended attributes.



## ***Second solution: extended attributes***

Use the extended attributes if you do not want to extend the database schema of the ObjectServer.

This solution makes use of the extended attributes, which are new since OMNIBus 7.2. The database schema of OMNIBus 7.2 was extended by the field ExtendedAttr, which is a varchar field of 4,096 bytes. It was added to contain name-value pairs for TEC attributes names and TEC attribute contents. In addition to this field itself, there are corresponding functions to access the field:

<b>nvp_add</b>	This function is used at the probe level to concentrate different probe fields to the ExtendedAttr field. The first parameter of this function defines the name of the container to which the variable-value pair is transported before it is assigned to the left side of the function. Then the variable-value pairs follow. Here is an example: " <code>@ExtendedAttr=nvp_add(@ExtendedAttr, 'HostName', 'Mondorf', 'HostType', 'Linux')</code> ". 'Mondorf' is the first variable name and 'Linux' is the first value content.
<b>nvp_remove</b>	At the probe level <code>nvp_remove</code> removes keys from a previously defined list. We did not test this function.
<b>nvp_exists</b>	At the ObjectServer level this functions checks whether a variable-value pair exists in the ObjectServer field ExtendedAttr. Therefore, we provide the example: <code>If ( nvp_exists(new.ExtendedAttr, 'HostName') = TRUE ) THEN update alerts.status via new.Identifier set Location = nvp_get(new.ExtendedAttr, 'HostName');</code> end if;.
<b>nvp_get</b>	With <code>nvp_get</code> you can extract the value content from a name-value pair. For example, " <code>set Location = nvp_get(new.ExtendedAttr, 'Mondorf')</code> ". Here the ObjectServer field Location is filled with the value that is referenced by the value name 'Mondorf'.
<b>nvp_set</b>	The <code>nvp_set</code> command adds or replaces keys from a name-value pair. We did not test this function.

In our environment we established the commands at the probe level and at the ObjectServer level. Figure 6-37 shows the probe level.

```
$f1-descr ="Mondorf"
$f1-content ="Linux"
@ExtendedAttr = nvp_add(@ExtendedAttr, "HostName", $f1-descr,
"HostType", $f1-content )
```

Figure 6-37 Extended attributes at probe level in glf.rules

Instead of using strings in the nvp\_add command as shown in the definition of the nvp-functions above, we used the command the normal way with probe variables. Therefore, we defined temporary fields and filled them with strings.

To create events, we used our normal approach and extended our logfile with the following line:

2007-10-02 16:45:25 Mondorf app101 MINOR test-9

This results in the OMNibus events list, as shown in Figure 6-38.

Node	Location	Alert Gro...	Summary	Last Occurrence	Count	Type	ExpireTime	Agent	Manager
Mondorf	Linux	Linux	2007-10-02 16:45:25 Mondorf app101...	10/2/2007 3:40:20 PM	1	Problem	Not Set	Linux	LINUX

Figure 6-38 OMNibus eventlist for extended attributes

The content of the extended attributes is shown in Figure 6-39.

```
ExtendedAttr: HostName="Mondorf";HostType="Linux"
```

Figure 6-39 Event content of extended attributes

The Location field of the event is automatically filled as a result of the following automation (Figure 6-40), which uses the new extended attribute commands in a database insertion trigger.

```
if ( nvp_exists(new.ExtendedAttr, 'HostName') = TRUE )
  THEN
    update alerts.status via new.Identifier
    set Location = nvp_get(new.ExtendedAttr, 'HostName');
end if;
if ( nvp_exists(new.ExtendedAttr, 'HostType') = TRUE )
  THEN
    update alerts.status via new.Identifier
    set Location = Location + ' - ' + nvp_get(new.ExtendedAttr,
'HostType');
end if;
```

Figure 6-40 Database trigger for extended attributes

After testing this automation, the Location field has the content shown in Figure 6-41.



Figure 6-41 Location field after test

## 6.2.7 Handling of cause, effect, and clearing events

In this section we discuss the handling of cause, effect, and clearing events.

### Common description of the scenario

At the source, the following events are raised:

- ▶ Cause event
- ▶ Effect event
- ▶ Clearing event

The operator should see only the cause event and no effect event. After the clearing event arrives, the cause event should no longer be shown on the event console.

### TEC implementation

Create specific rules for these event classes. Link the effect events to the cause event at arrival, and propagate the status from the UPS\_Fan\_Down event to the

UPS\_Temp\_Degraded event. Close the cause event when the clearing event arrives and discard the clearing event.

## Solution

Typically, TEC rules event processing is based on special BAROC classes in which cause, effect, and clearing events are defined. For these events we find a corresponding rule set. In this rule set there are rules that use the `link_effect_to_cause` predicate to link two event instances together, so that when the clearing event arrives, another rule can search for the linked events with the `all_instances` or `first_instance` predicate. Figure 6-42 shows us these files.

```
#-----#
# UPS.BAROC #
#-----#
TEC_CLASS :
  UPS_ISA_EVENT
  DEFINES
  {
    source: default=SNMP;
  };
END

TEC_CLASS :
  UPS_Fan_Down ISA UPS
  DEFINES
  {
    severity: default=WARNING;
    hostname: dup_detect=yes;
  };
END

TEC_CLASS :
  UPS_Temp_Degraded ISA UPS
  DEFINES
  {
    severity: default=WARNING;
    hostname: dup_detect=yes;
  };
END

TEC_CLASS :
  UPS_Fan_Up ISA UPS
  DEFINES
  {
    severity: default=HARMLESS;
  };
END
```

Figure 6-42 BAROC class `ups.baroc`

Here the definition of the TEC class UPS again. It is a child of the base class EVENT. This class has three child classes: UPS\_Fan\_Down (cause event), UPS\_Temp\_Degraded (effect event), and the clearing event UPS\_Fan\_Up.

```
rule: link_temp_to_fan:
(
  description: 'Link the UPS_Temp_Degraded events to the
UPS_Fan_Down events',

  event: _event of_class 'UPS_Temp_Degraded'
  where [
    status: _status outside ['CLOSED'],
    hostname: _hostname
  ],

  action: link_to_fan_down :
  (
    first_instance(event: _fan_down_ev of_class 'UPS_Fan_Down'
      where [
        status: _status_fan_down outside ['CLOSED'],
        hostname: equals _hostname
      ]
    ),
    link_effect_to_cause(_event, _fan_down_ev),
    set_event_status(_event, _status_fan_down)
  )
).
```

Figure 6-43 TEC rule link\_temp\_to\_fan

Here we find the linking between the cause and the effect event. They are not just linked, but the status of the UPS\_Fan\_Down event is propagated to the newly arrived UPS\_Temp\_Degraded event, so after this rule processing both have the same status.

Later on when the clearing event arrives, the rule in Figure 6-44 will be fired.

```
rule: auto_close_when_fan_up:
(
  description: 'Automatically close the UPS_Fan_Down event and the
linked \
                UPS_Temp_Degraded when we receive UPS_Fan_Up',
  event: _event of_class 'UPS_Fan_Up'
  where [
    status: _status outside ['CLOSED'],
    hostname: _hostname
  ],
  reception_action: find_and_close:
  (
    all_instances(event: _ev_fan_down of_class 'UPS_Fan_Down'
      where
      [
        status      : outside ['CLOSED'],
        hostname    : equals _hostname,
        date_reception : _date,
        event_handle : _ev_handle
      ]
    ),
    set_event_status(_ev_fan_down, 'CLOSED'),
    all_instances(event: _ev_temp_degraded of_class
'UPS_Temp_Degraded'
      where
      [
        status      : outside ['CLOSED'],
        cause_date_reception : equals _date,
        cause_event_handle  : equals _ev_handle
      ]
    ),
    set_event_status(_ev_temp_degraded, 'CLOSED')
  ),
  action : drop :
  (
    drop_received_event
  )
).
```

Figure 6-44 TEC rule *auto\_close\_when\_fan\_up*

When the UPS\_Fan\_Up (clearing) event is received, the action find\_and\_close looks for all UPS\_Fan\_Down (cause) events from the same host and searches for all the corresponding UPS\_Temp\_Degraded events that are linked to the cause event. The cause and the effect event will be closed. After this, the arrived clearing event is dropped.

To test these rules we provided the following events. We sent them with the **tec\_agent\_demo** tool. The **tec\_agent\_demo** command is located in the directory \$BINDIR/TME/TEC and is used via \$TEC\_BIN\_DIR/tec\_agent\_demo -a data /directory/event-filename. TEC\_BIN\_DIR must be exported first as TEC\_BIN\_DIR=\$BINDIR/TME/TEC. The file events-filename only comprises file names that are located in the same directory. These files contains the real event content, as shown in Figure 6-45, Figure 6-46, and Figure 6-47.

```
UPS_Fan_Down;  
    origin='9.36.11.3';  
    hostname=ups_atl;  
    msg='Fan down on UPS ups_atl';  
    adapter_host=nfs_server;  
END
```

*Figure 6-45 UPS\_Fan\_Down event*

```
UPS_Temp_Degraded;  
    origin='9.36.11.3';  
    hostname=ups_atl;  
    msg='Temperature on UPS ups_atl is too high';  
    adapter_host=nfs_server;  
END
```

*Figure 6-46 UPS\_Temp\_Degraded event*

```
UPS_Fan_Up;  
    origin='9.36.11.3';  
    hostname=ups_atl;  
    msg='Fan on UPS ups_atl is up again';  
    adapter_host=nfs_server;  
END
```

*Figure 6-47 UPS\_Fan\_Up event*



After sending the events we have the output on the TEC console shown in Figure 6-48.

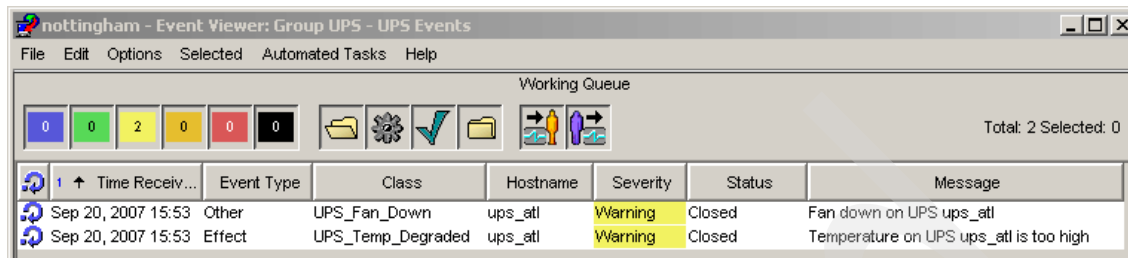


Figure 6-48 TEC console with correlated events

We see that the rule `auto_close_when_fan_up` closes the cause and effect event and drops the `UPS_Fan_Up` event.

## OMNibus implementation

Check whether the event probe sets the event type correctly and use the default correlation behavior of the `generic_clear` automation.

### First solution

In OMNibus the type of event (cause, effect, or clearing) is defined in the Type field at the probe level. The following values are possible:

- ▶ 0 - Type not set
- ▶ 1 - Problem
- ▶ 2 - Resolution
- ▶ 3 - Netcool/visionary problem
- ▶ 4 - Netcool/visionary resolution
- ▶ 7 - Netcool/ISMs new alarm
- ▶ 8 - Netcool/ISMs old alarm
- ▶ 11 - More severe
- ▶ 12 - Less severe
- ▶ 13 - Information

The ObjectServer automations provide two standard database triggers for this task. The first one is the `generic_clear` temporal trigger, which runs every 5 seconds, and the other is the temporal trigger `delete_clears`, which runs every minute. To achieve a better understanding of these triggers, let us take a deeper look at the OMNibus attributes that are used in this automation:

### Agent

The Agent field contains a descriptive name of the sub-manager that generated the alert (for example, `manager = MTTTrapd Probe`, `agent = IETF-BRIDGE-MIB`).

<b>AlertGroup</b>	The AlertGroup field contains a descriptive name of the type of failure indicated by the alarm (for example, interface status, CPU utilization, and so on).
<b>AlertKey</b>	The AlertKey field contains a descriptive key that indicates the object instance referenced by the alarm (for example, the disk partition indicated by a <i>file system full</i> alarm, or the switch port indicated by the utilization alarm).
<b>LastOccurrence</b>	Time in seconds since midnight (GMT), Jan. 1. 1970. when this alert was last updated at the probe.
<b>Node</b>	The Node field is used to identify the managed entity from which the alarm originated. This could be a host/device name, service name, customer, or other entity.
<b>Manager</b>	The Manager field contains a descriptive name of the probe that collected and forwarded the alarm to the ObjectServer (for example, MTTTrapd Probe, HP OpenView NNM, and so on).
<b>Severity</b>	The Severity field can indicate any of six defined severity levels, which provide an indication of how it is perceived that the capability of the managed object has been affected.

With the understanding of the contents of these OMNibus attributes we can now have a closer look at the automation. Let us first look at the generic\_clear code (Figure 6-49).

```
begin
- 1 - Populate a table with Type 1 events corresponding to any uncleared
Type 2 events
  for each row problem in alerts.status where
    problem.Type = 1 and problem.Severity > 0 and
    (problem.Node + problem.AlertKey + problem.AlertGroup + problem.Manager)
  in
    ( select Node + AlertKey + AlertGroup + Manager from alerts.status where
      Severity > 0 and Type = 2 )
  begin
    insert into alerts.problem_events values ( problem.Identifier,
    problem.LastOccurrence,
    problem.AlertKey, problem.AlertGroup,
    problem.Node, problem.Manager, false );
  end;

- 2 - For each resolution event, mark the corresponding problem_events
entry as resolved
-- and clear the resolution
  for each row resolution in alerts.status where resolution.Type = 2 and
  resolution.Severity > 0
  begin
    set resolution.Severity = 0;
    update alerts.problem_events set Resolved = true where
    LastOccurrence < resolution.LastOccurrence and
    Manager = resolution.Manager and Node = resolution.Node and
    AlertKey = resolution.AlertKey and AlertGroup = resolution.AlertGroup ;
  end;

- 3 - Clear the resolved events
  for each row problem in alerts.problem_events where problem.Resolved = true
  begin
    update alerts.status via problem.Identifier set Severity = 0;
  end;

- 4 - Remove all entries from the problems table
  delete from alerts.problem_events;
end
```

Figure 6-49 OMNibus standard automation generic\_clear

This automation is divided into four parts:

1. The first part populates the table `alerts.problem_events` with type 1 events from the `alerts.status` table corresponding to any uncleared type 2 events.
2. In the second step, each resolution event in the `alerts.status` table is in the second step marked as cleared, and the corresponding entry in `alerts.problem_events` is marked as resolved.
3. All the resolved marked events in `alerts.problem_events` and the corresponding events in `alerts.status` are cleared (set severity = 0).
4. The last step deletes all events from the temporary table `alerts.problem_events`.

Now let us look at the `delete_clears` automation (Figure 6-50).

```
begin
delete from alerts.status where Severity = 0 and StateChange <
(getdate() - 120);
end
```

*Figure 6-50 OMNIBus standard automation delete\_clears*

The `delete_clears` automation will delete the cleared events from the database as long as they are older than two minutes.

To test this scenario we execute the following SQL statements with `nco_sql` (Figure 6-51). Here look at the numbers previous to the word UPS. The type and the severity are defined.

```
insert into alerts.status (Identifier, Summary, Node, Agent, Type,
Severity, AlertKey, AlertGroup, Manager, LastOccurrence) values
('itso-corr-01', 'Fan down on UPS ups_atl', 'ups_atl', 'Agent', 1,
2, 'UPS', 'SNMP', 'nco_sql', getdate());

insert into alerts.status (Identifier, Summary, Node, Agent, Type,
Severity, AlertKey, AlertGroup, Manager, LastOccurrence) values
('itso-corr-02', 'Temperature on UPS ups_atl is too high',
'ups_atl', 'Agent', 1, 3, 'UPS', 'SNMP', 'nco_sql', getdate());

insert into alerts.status (Identifier, Summary, Node, Agent, Type,
Severity, AlertKey, AlertGroup, Manager, LastOccurrence) values
('itso-corr-03', 'Fan on UPS ups_atl is up again', 'ups_atl',
'Agent', 2, 1, 'UPS', 'SNMP', 'nco_sql', getdate());
```

*Figure 6-51 OMNIBus events for nco\_sql execution*

After the events have been sent, we have the output in the OMNibus event list shown in Figure 6-52.

Node	Alert Group	Summary	Last Occurrence	Count	Type	ExpireTime	Agent	Manager
ups_atl	SNMP	Fan on UPS ups_atl is up again	9/20/2007 4:42:44 PM	1	Resolution	Not Set	Agent	nco_sq
ups_atl	SNMP	Temperature on UPS ups_atl is too high	9/20/2007 4:42:38 PM	1	Problem	Not Set	Agent	nco_sq
ups_atl	SNMP	Fan down on UPS ups_atl	9/20/2007 4:42:31 PM	1	Problem	Not Set	Agent	nco_sq

Summary: 3 (green), 0 (purple), 0 (blue), 0 (yellow), 0 (orange), 0 (red)

0 rows selected | 9/20/2007 4:45:19 PM | root | NCOMS [PRI]

Figure 6-52 OMNibus event list with correlated events

**Note:** For another worked example of a correlation (cause/effect) rule scenario, see 7.2.6, “Automatic event management customization” on page 284, where the equivalent of the TEC netview.rls is implemented in OMNibus.

## 6.2.8 Propagating status change from cause to effect events

In this section we discuss propagating status change from cause to effect events.

### Common description of the scenario

If the operator acknowledges the cause event, the status change should be propagated to the effect event.

### TEC implementation

Create specific rules for these event classes. Link the effect events to the cause event at arrival. When the status of the cause event is changed to acknowledged, also change the status of all the linked effect events to acknowledged.

### **Solution**

The linking of the effect to the cause event was previously done in Figure 6-43 on page 218. Therefore, we only have to add the change rule for the acknowledgements. The next rule provides this functionality. See Figure 6-53.

```
change_rule: close_linked_auto:
(
  description: 'When an administrator acks or closes a UPS_Fan_Down
we \
              want to ack or close the linked UPS_Temp_Degraded',
  event: _event of_class 'UPS_Fan_Down'
  where
  [
    date_reception: _date,
    event_handle: _ev_handle
  ],
  slot: status set_to _status within['ACK', 'CLOSED'],
  action: search_temp_degraded_and_close:
  (
    all_instances(event: _ev_temp_degraded of_class
'UPS_Temp_Degraded'
              where
              [
                status          : outside ['CLOSED'],
                cause_date_reception : equals _date,
                cause_event_handle  : equals _ev_handle
              ]
    ),
    set_event_status(_ev_temp_degraded, _status)
  )
).
```

*Figure 6-53 TEC change rule for status propagation for linked events*

When the status of an UPS\_Fan\_Down event is changed to ACK or CLOSED, then the event cache is searched for corresponding UPS\_Temp\_Degraded events. If there are such events, they will get the same status.

We tested this rule similarly to the test in the correlation section. The difference here is that after sending the following two events the administrator acknowledges the cause event. This results in the propagation of the status and in the view shown in Figure 6-56 on the event console.

```
UPS_Fan_Down;
    origin='9.36.11.3';
    hostname=mondorf;
    msg='Fan down on UPS mondorf';
    adapter_host=nfs_server;

END
```

Figure 6-54 Cause event UPS\_Fan\_Down

```
UPS_Temp_Degraded;
    origin='9.36.11.3';
    hostname=mondorf;
    msg='Temperature on UPS mondorf is too high';
    adapter_host=nfs_server;

END
```

Figure 6-55 Effect event UPS\_Temp\_Degraded

	Time Received	Event Type	Class	Hostname	Severity	Status	Message
	10/3/07 11:49	Other	UPS_Fan_Down	mondorf	Warning	Acknowledged	Fan down on UPS mondorf
	10/3/07 11:49	Effect	UPS_Temp_Degraded	mondorf	Warning	Acknowledged	Temperature on UPS mondorf is too high

Figure 6-56 TEC event console after status propagation

## OMNIbus implementation

In OMNIbus there is no difference between cause and effect events. They are both problem events (type=1), but there are similar tasks to be handled. Assume that a network management software monitors switches and sends events that signal the failure of interfaces and nodes (switches). For example, see Figure 6-57.

Node	Alert Group	Summary	Last Occurrence	Count	Type	ExpireTime	Agent	Manager
borrn	SNMP	interface down	9/28/2007 12:23:50 PM	1	Problem	Not Set	Agent	nco_sql
austin	SNMP	node down	9/28/2007 12:23:47 PM	1	Problem	Not Set	Agent	nco_sql
austin	SNMP	interface down	9/28/2007 12:23:45 PM	1	Problem	Not Set	Agent	nco_sql
mondorf	SNMP	node down	9/28/2007 12:23:42 PM	1	Problem	Not Set	Agent	nco_sql
mondorf	SNMP	interface down	9/28/2007 12:22:20 PM	2	Problem	Not Set	Agent	nco_sql
mondorf	SNMP	interface down	9/28/2007 12:22:14 PM	1	Problem	Not Set	Agent	nco_sql

Figure 6-57 Events for OMNIbus correlation

The task is now to write a temporal trigger that clears all interface down events for nodes (switches) that have corresponding node down events. In this example, the two interface down events from the switch mondorf and the event from the switch austin should be cleared, but not the one from the switch bonn, because this event does not have a corresponding node down event.

### Solution

To fulfill this requirement we provide an SQL statement. This SQL statement should set the severity set to 0 (clear) for interface down events, but only for those events whose own node is equal to an event with a corresponding node down. Figure 6-58 shows the corresponding SQL code.

```
update alerts.status set Severity = 0 where Summary = 'interface
down' and Node in (select Node from alerts.status where Summary =
'node down')
```

Figure 6-58 SQL correlation code example

When we put it in a temporal trigger that runs every minute, it produces the result shown in Figure 6-59.

Node	Alert Group	Summary	Last Occurrence ▾	Count	Type	ExpireTime	Agent	Manager
bonn	SNMP	interface down	10/3/2007 1:10:07 PM	1	Problem	Not Set	Agent	nco_sql
austin	SNMP	node down	10/3/2007 1:10:05 PM	1	Problem	Not Set	Agent	nco_sql
austin	SNMP	interface down	10/3/2007 1:10:03 PM	1	Problem	Not Set	Agent	nco_sql
mondorf	SNMP	node down	10/3/2007 1:10:01 PM	1	Problem	Not Set	Agent	nco_sql
mondorf	SNMP	interface down	10/3/2007 1:09:58 PM	1	Problem	Not Set	Agent	nco_sql
mondorf	SNMP	interface down	10/3/2007 1:09:55 PM	1	Problem	Not Set	Agent	nco_sql

Figure 6-59 OMNibus event console with result of the SQL correlation statement

## 6.2.9 Local and remote script execution

In this section we discuss local and remote script execution.

### Common description of the scenario

When an event arrives, a specific program should be executed locally or remotely. In addition, the execution should take place under a specific GID/UID.

### TEC implementation

Create a specific rule for this event class. In a reception action call a local script or start the execution of a TME task on a remote endpoint.



### **Solution**

Before we execute a script or task in a TEC rule, we should test whether these are working at the OS level. Here we show the script `broadcast.sh`.

```
#!/bin/sh
text=$@
wbroadcast "$text"
exit 0
```

*Figure 6-60 Test script broadcast.sh*

This will send a message to the Tivoli administrator's desktop. The next script will provide an echo to stdout and additionally to a file. Its contents are shown in Figure 6-61.

```
#!/bin/sh
echo "Hello from ACT - $1"
echo "Hello from ACT - $1" > /tmp/hello.out
```

*Figure 6-61 Test script hello.sh*

The purpose of this script is to be executed by a Tivoli task. The Tivoli task should be run under a specific user, for example, `db2inst1`. When the script is executed the output file `hello.out` should contain the UID of the user `db1inst` in the file attributes in at OS level.

To run both we developed the TEC rules shown in Figure 6-62 after we successfully tested the scripts and the task at OS level.

```
rule: exec_fan_down:
(
  description: 'Executes the broadcast.sh script when we receive a \
                UPS_Fan_Down',
  event: _event of_class 'UPS_Fan_Down'
  where
  [
    hostname: _hostname
  ],
  reception_action: exec_fan_script:
  (
    exec_program(_event,
                  'scripts/broadcast.sh',
                  'Please check the fan of %s',
                  [_hostname],
                  'YES'
                )
  ),
  reception_action: exec_hello_task:
  (
    exec_task(_event,
              'hello.ta',
              '-l "ACT.tl" -h "nottingham" -a "%s"',
              ['Wolfgang'],
              'YES'
            )
  )
).
```

Figure 6-62 TEC rule *exec\_fan\_down*

Here you can see the predicates `exec_program` and `exec_task`, which execute the corresponding scripts. We tested this by sending the following command.

```
wpostmsg -m test5 hostname=mondorf UPS_Fan_Down TEC
```

This produces a broadcast pop-up on the Tivoli administrators desktop, as shown in Figure 6-63.

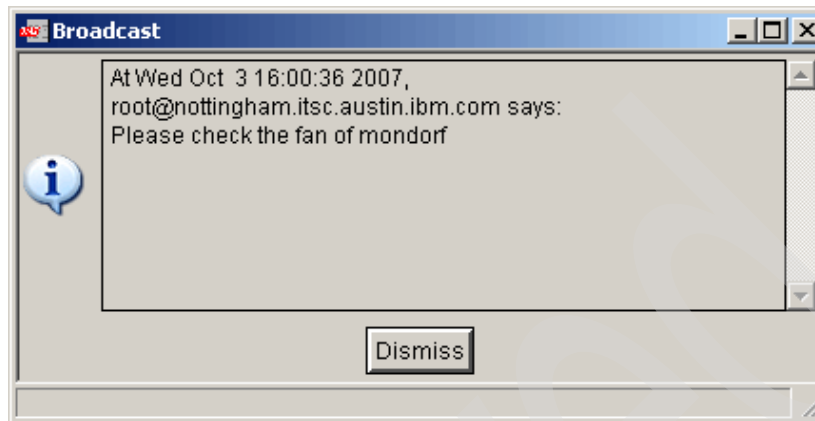


Figure 6-63 Output of script broadcast.sh

After this the TEC event console shows the content shown in Figure 6-64.

	Time Received	Event Type	Class	Hostname	Severity	Status	Message
	10/3/07 16:00	Other	UPS_Fan_Down	mondorf	Warning	Open	test5

Figure 6-64 Event list for executing scripts

Figure 6-65 shows the task output that is provided by the TEC console.

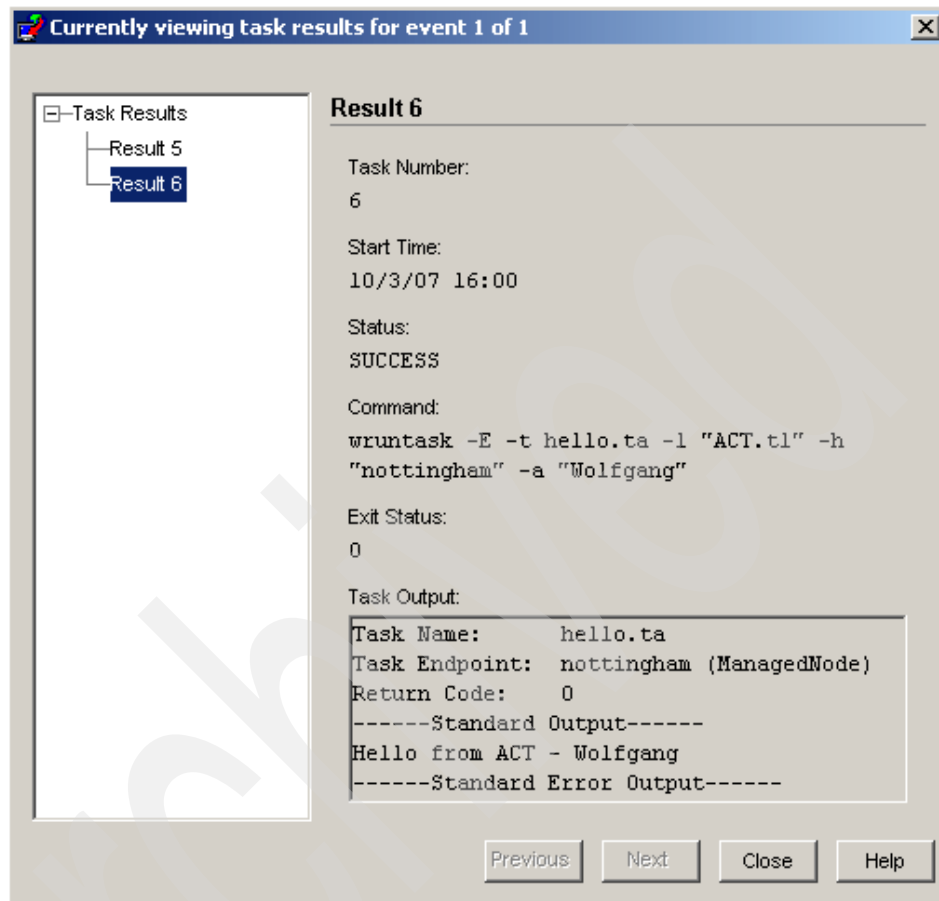


Figure 6-65 TEC task output

Here we can see the echo to standard output of our script hello.sh.

And lastly we must look at the attributes of the output file in the file system (Figure 6-66).

-rw-r--r--	1	db2inst1	nobody	26 Oct 03 15:42	/tmp/hello.out
-rwxr-xr-x	1	root	system	83 Sep 21 10:58	/tmp/hello.sh

Figure 6-66 Attributes of /tmp/hello.\*

## OMNIbus implementation

**Note:** Process control has to be installed and configured on every target machine on which you want to execute the procedure.

Create a database trigger that starts an external procedure locally or remotely through process control.

Archived

## Solution

When process control is installed and working properly, you must test your script in the operating system environment. Then you can create an external procedure (for example, named `itso_7`) in the OMNibus administrator configuration dialog, as in Figure 6-67.

The screenshot shows a window titled "Edit Procedure (WEIMAR on weimar:4100)". Inside, there's a section "External Procedure Details". The "Name:" field contains "itso\_7". Below it is a "Parameters" section with an empty list box and navigation arrows. Underneath are fields for "In/Out:" (set to "in"), "Name:" (empty), "Data Type:" (set to "Integer"), and "Array:" (unchecked). The "Executable:" field contains "/tmp/hello.sh" with a "Browse..." button next to it. Below that is an "Arguments:" section with an empty list box. The "Host:" field contains "weimar". At the bottom, "User ID:" is set to "2" and "Group ID:" is set to "0". "OK" and "Cancel" buttons are at the bottom right.

Figure 6-67 Creation of an external OMNibus procedure

You must provide the script name `/tmp/hello.sh` and the target machine. Optionally, you can set the UID (for example, 2 for the daemon user). After saving this procedure, you can execute it via **execute procedure itso\_7** in the

nco\_sql interface. After the execution, you can see the following file attribute information in the /tmp directory of the machine weimar:

```
-rw-r--r-- 1 daemon root 6 Oct 4 11:20 hello.out
```

A database trigger that calls this procedure could also be used.

## 6.2.10 Escalation of event severity

In this section we discuss escalation of event severity.

### Common description of the scenario

If an event is not closed 1 minute after arriving, set its severity to critical.

### TEC implementation

Create a specific rule for this event class. Sets a timer for one minute in a reception action. When the timer pops, examine the event's status. If it is still open, set the severity to critical.

## Solution

The following rule is separated into two parts. The first part is an action triggered by reception of the event class UPS\_Fan\_Down. Here the timer level 1 is set with a duration of 60 seconds. In the second part, the timer rule escalate\_1 is executed after 60 seconds and sets the severity to CRITICAL unless the status has the content OPEN. You can see the code of both rules in Figure 6-68.

```
rule: exec_fan_down:
(
  description: 'Executes the broadcast.sh script when we receive a \
                UPS_Fan_Down',
  event: _event of_class 'UPS_Fan_Down'
  where
  [
    hostname: _hostname
  ],
  action: set_timer:
  (
    set_timer(_event, 60, 'level1')
  )
).

timer_rule: escalate_1:
(
  description: 'When a UPS_Fan_Down stays open for more than 1
                minute we want\
                to increase the severity to CRITICAL',
  event: _event of_class 'UPS_Fan_Down'
  where [
    status: equals 'OPEN'
  ],
  timer_info: equals 'level1',
  action: set_timer_and_escalate:
  (
    set_event_severity(_event, 'CRITICAL')
  )
).
```

Figure 6-68 TEC rule timer setting and timer rule for escalation

We test this rule with the following **wpostemsg** command:

```
wpostemsg -m test-8 hostname=mondorf UPS_Fan_Down TEC
```



This command produces the next TEC console output (Figure 6-69).

	Time Received	Event Type	Class	Hostname	Severity	Status	Message	Sub-source
	10/4/07 09:05	Other	UPS_Fan_Down	mondorf	Warning	Open	test-8	

Figure 6-69 TEC console for before escalation

After 60 seconds the TEC console looks like Figure 6-70.

	Time Received	Event Type	Class	Hostname	Severity	Status	Message	Sub-source
	10/4/07 09:05	Other	UPS_Fan_Down	mondorf	Critical	Open	test-8	

Figure 6-70 TEC console after escalation

## OMNibus implementation

Create a temporal trigger that sets to 5 the severity of events that have the Acknowledged field set to zero and the LastOccurrence field within the last minute.

### Solution

This request is similar to the default temporal trigger `flash_not_ack`. Therefore, we can easily adapt this. Figure 6-71 shows the code.

```
update alerts.status set Severity = 5 where Acknowledged = 0 and
LastOccurrence > (getdate - 60);
```

Figure 6-71 OMNibus temporal trigger for escalation

We tested this trigger by appending the next line to our applications log file:

```
2007-10-04 09:30:25 Mondorf appl01 MINOR test-8.
```

After adding this, the event list has the content shown in Figure 6-72.

Last Occurrence	Seve...	Node	Summary	Count	Alert Group	Type
10/12/2007 9:46:43 AM	Major	Mondorf	2007-10-04 09:30:25 Mondorf appl01 MINOR test-8	1	Linux	Problem

Figure 6-72 OMNibus event list before escalation

After the duration of 60 seconds the list looks like Figure 6-73.

Last Occurrence	Seve...	Node	Summary	Count	Alert Group	Type
10/12/2007 9:46:43 AM	Critical	Mondorf	2007-10-04 09:30:25 Mondorf appl01 MINOR test-8	1	Linux	Problem

Figure 6-73 OMNibus event list after escalation

## 6.2.11 Forwarding of events

In this section we discuss forwarding events.

### **Common description of the scenario**

Send an incoming event to a trouble ticketing system.

### **TEC implementation**

Create a specific rule for this event class. Call a script that has access to the contents of the event and passes these contents to the trouble ticketing software.

#### ***First solution***

The implementation of this request is identical to running a local script. This was covered in 6.2.9, “Local and remote script execution” on page 228.

### **OMNIbus implementation**

To implement:

1. Execute an external procedure.
2. Install a gateway for a trouble ticketing system or for an external database.

#### ***First solution***

You can find this kind of a solution in 6.2.9, “Local and remote script execution” on page 228.

#### ***Second solution***

In our lab environment we do not have any gateways installed. Normally, you find the following gateways installed:

- ▶ Gateway for HP Service Desk
- ▶ Gateway for Remedy ARS

## 6.2.12 Use of external information for logic control

In this section we discuss use of external information for logic control.

### **Common description of the scenario**

Depending on the information that is stored outside the event sever, the logical control of the event processing should be influenced.

## TEC implementation

Create a fact file for the rulebase. Compile and consult it in an action in a rule file. Later on, use it to control the logic of your rule.

### ***Solution***

Fact files contain prolog statements that are used in rules. Figure 6-74 shows a fact file.

```
tec_type('nottingham', 'MASTER').  
tec_type('mondorf', 'SLAVE').
```

*Figure 6-74 TEC fact file tec\_r.tec\_servers*

Here you can see that the machine 'nottingham' is defined as a MASTER. Later in the rule this statement will be dynamically created. The dynamic part is the host name that will be extracted from the event attribute host name. The extracted host name will be inserted here, so that this code line can have different appearances. The corresponding code content is shown in Figure 6-75.

```
rule: lab_init:
(
  event: _event of_class 'TEC_Start'
    where [
      hostname: _hostname
    ],
  reception_action:
  (
    set_global_var('This TEC', 'Hostname', _hostname)
  ),
  reception_action:
  (
    compile('/entw/tec_r.tec_servers'),
    consult('/entw/tec_servers')
  )
).

rule: lab_use :
(
  event: _event of_class UPS,
  action:
  (
    get_global_var('This TEC', 'Hostname', _tec_name, ''),
    tec_type(_tec_name, 'MASTER'),
    bo_set_slotval(_event,sub_source,'fact-file OK');
  )
).
```

*Figure 6-75 TEC rule for the use of global variables and fact files*

Whenever the TEC starts, the host name will be put in a global variable named 'This TEC'. With the compile and consult predicates, the content of the fact file is compiled and loaded into the TEC knowledge base.

In the lab\_use rule, the global variable 'This TEC' is read and the value assigned to it is stored in the variable \_tec\_name. This variable is now the variable part of our prolog statement. If this statement is identical to one line of our fact file, the condition (statement) is true, and the execution of the rest of the rule actions are

executed, here the change of the sub\_source attribute. If the fact file does not contains this line, the execution of this statement fails.

To test this rule we send the next **wpostemsg** command:

```
wpostemsg -m test-fact-file hostname=nottingham TEC_Start TEC
```

After the execution of this event the TEC console looks like Figure 6-76.

	Time Received	Event Type	Class	Hostname	Severity	Status	Message	Sub-source
	10/4/07 15:36	Other	TEC_Start	nottingham	Harmless	Open	test-fact-file	fact-file OK

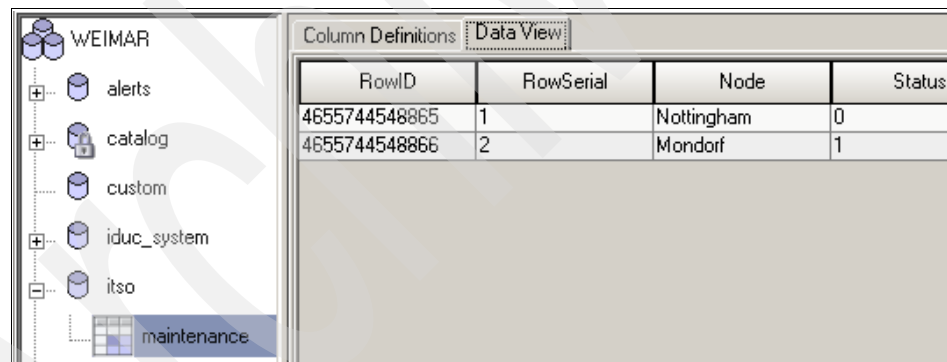
Figure 6-76 TEC console after execution of a fact file

## OMNibus implementation

To influence the logic of OMNibus triggers you must read data from an external device.

### Solution

A database table could be an external device, for example. We created the database shown in Figure 6-77 called *itso* and the table named *maintenance*.



RowID	RowSerial	Node	Status
4655744548865	1	Nottingham	0
4655744548866	2	Mondorf	1

Figure 6-77 OMNibus database *itso.maintenance*

In this database we created two columns. The *Node* column contains the host name and the *Status* column contains either 0 for production or 1 for maintenance.

The next step in controlling our logic with the content of this data is to create a temporal trigger that, every 15 seconds, clears all the events from all nodes that have the status 1 (here only Mondorf).

The SQL statement looks like:

```
update alerts.status set Severity = 0 where Node in (select Node from
itso.maintenance where Status = 1)
```

The test of this trigger is done by adding one line to our application log:

2007-10-05 15:08:28 Mondorf appl01 MINOR test-21

After 15 seconds the event console has the appearance shown in Figure 6-78.

Last Occurrence	Severity	Node	Summary	Count	Alert Group
10/12/2007 9:55:06 AM	Clear	Mondorf	2007-10-05 15:08:28 Mondorf appl01 MINOR test-21	1	Linux

Figure 6-78 OMNibus console after own logic control

## 6.3 Probe topics

In this section we describe other probe rules that may be helpful in some environments. We also introduce methods for the self-monitoring of probes.

### 6.3.1 Measuring load and numbers of events in a time frame

The requirement is to measure how many events the probe processes in a defined time frame.

#### Solution

The code shown in Figure 6-79 provides the solution.

```
array loads
...
if (match(loads[@Node], ""))
{
    loads[@Node]="60.50"
}
loads[@Node]=updateLoad(loads[@Node])
%current_load=getLoad(loads[@Node])
@Location=%current_load
@AlertGroup = geteventcount(loads[@Node])
```

Figure 6-79 Example of updateLoad, getLoad, and geteventcount functions

The first line defines a one-dimensional array. The condition is used to define the initial load/count measurement. Here the interval is set to 60 seconds, and a

boundary is set to a maximum of 50 events. If there are more than 50 events within 60 seconds, then the oldest will be discarded. The updateload line updates the measurement, and the getload line stores the current load in a variable. And, finally, the number of events in the time frame is assigned to the Field AlertGroup.

To test this function we add several lines to our application file:

```
2007-10-05 11:07:25 Mondorf appl01 MINOR test-9
2007-10-05 11:07:26 Mondorf appl01 MINOR test-9
2007-10-05 11:07:27 Mondorf appl01 MINOR test-9
2007-10-05 11:07:27 Mondorf appl01 MINOR test-9
2007-10-05 11:07:28 Mondorf appl01 MINOR test-9
```

This results in the display on event console, shown in Figure 6-80.

Node	Alert Group	Location	Summary ▾	Last Occurrence	Count
Mondorf	6	0.035621	2007-10-05 11:07:28 ...	10/5/2007 10:42:17 AM	1
Mondorf	4	0.023753	2007-10-05 11:07:27 ...	10/5/2007 10:42:17 AM	2
Mondorf	3	0.022681	2007-10-05 11:07:26 ...	10/5/2007 10:41:40 AM	1
Mondorf	2	0.062267	2007-10-05 11:07:25 ...	10/5/2007 10:41:28 AM	30

Figure 6-80 OMNibus event console with counting and load information

On the event console you can see the counting information in the field AlertGroup and the load of the probe in the Location field. This information can be used in other triggers.

## 6.3.2 Self monitoring

Most of the probes have a self-monitoring capability implemented, so they will send an event in the following cases:

- ▶ Going down
- ▶ Running
- ▶ Unable to get events

## 6.3.3 Parsing failed

In a TEC environment events can be sent with class or attribute definitions that are not defined at the server level. These events are documented as PARSING FAILED. In the OMNibus environment the corresponding parsing failed is checked at the startup of a probe. Here all assignments are checked against the ObjectServer if the fields exist in the database schema. If one or more fields are not defined, the probe does not start. This situation is logged in the logfile \$OMNIHOME/log/probe-name.log.

### 6.3.4 EIF rules file and extended attributes

Referring to our recommended upgrade strategy, there will be a situation where you switch the TEC adapter to the OMNibus environment. The events will be delivered through the EIF probe to the OMNibus ObjectServer. This section shows you the flow and the access to custom-developed TEC class attributes.

The rules file `tivoli_eif.rules` that is delivered with the EIF probe is the best location in which to move custom-developed TEC class attributes content to the new ObjectServer field `ExtendedAttr`.

When we look at the file `tecad_logfile.baroc`, for example, we see the classes shown in Figure 6-81.

```
TEC_CLASS :
    Logfile_Su ISA Logfile_Base
    DEFINES {
        from_user: STRING, dup_detect = yes;
        to_user: STRING, dup_detect = yes;
        on_tty: STRING, dup_detect = yes;
        severity: default = WARNING;
    };
END
TEC_CLASS :
    Su_Success ISA Logfile_Su;
END
TEC_CLASS :
    Su_Failure ISA Logfile_Su;
END
```

Figure 6-81 Extract from `tecad_logfile.baroc`

Here we can find typical *custom* attributes (for example, `from_user`, `to_user`, or `on_tty`). Now we send the next **wpostemsg** to the TEC server:

```
wpostemsg -r MINOR -m test41 hostname=mondorf from_user=db2inst1
to_user=root on_tty=/dev/pts/4 Su_Failure TEC
```



After receiving this event at the TEC server we can see its content, as shown in Figure 6-82.

class	Su_Failure
credibility	1
date_event	Oct 9 14:49:23 2007
date_reception	1191959363
duration	0
event_hdl	1
from_user	db2inst1
hostname	mondorf
last_modified_time	1191959363
msg	test41
msg_catalog	none
msg_index	0
num_actions	0
on_tty	/dev/pts/4

Figure 6-82 Extract of a *SU\_Failure* event

When the event is forwarded through the EIF probe, the content of this must be assigned to ObjectServer fields. To avoid extending the database schema of the ObjectServer for every class attribute, the new field ExtendedAttr should be used instead. Now we must develop for every TEC class, which we want to use unmodified at the ObjectServer, an assignment in the EIF probe rules file like in Figure 6-83. Here we identified the TEC class with an IF statement. When many different classes must go through the EIF probe rule, a SWITCH statement should be used instead.

```
@ExtendedAttr=nvp_add(@ExtendedAttr, "Su_Failure",
"any-text","from_user", $from_user, "to_user", $to_user, "on_tty",
$on_tty)
```

Figure 6-83 Assignment from TEC class attributes to ObjectServer field ExtendedAttr

We can see the result of this code extract in the event details in the ObjectServer event list, as shown in Figure 6-84.

ExtendedAttr	Su_Failure="any-text";from_user="db2inst1";on_tty="/dev/pts/4";to_user="root"
--------------	---

Figure 6-84 Content of the ExtendedAttr field

On the basis of Figure 6-84 we can try to explain the use of the parameters of the **nvp\_add** command in Figure 6-82. We used the *Su\_Failure* string, because we wanted to search in the OMNibus automation in the ExtendedAttr field for a

variable name, rather than the content of the variable. Variable names in our example are Su\_Failure, from\_user, to\_user, and on\_tty (all variables that do not show up with quotation marks).

If we want to access the content of the field ExtendedAttr in an OMNibus automation we must develop, for example, the corresponding trigger, as in Figure 6-85.

```
begin
  if ( nvp_exists(new.ExtendedAttr, 'Su_Failure') = TRUE )
    THEN
      update alerts.status via new.Identifier
      set Location = nvp_get(new.ExtendedAttr, 'from_user') + '-' +
                    nvp_get(new.ExtendedAttr, 'to_user') + '-' +
                    nvp_get(new.ExtendedAttr, 'on_tty');
    end if;
end
```

*Figure 6-85 OMNibus database trigger on insertion for TEC attributes*

After implementing this automation, the wpostmsg event originated at the TEC server, forwarded and processed through the EIF probe (triggered by the above database trigger), and arrived on the OMNibus event list, as shown in Figure 6-86.

This code example shows only the technique to access the extended attribute. The design of this solution would normally be done at the probe level.

Node	Alert Group	Summary	Last Occurrence	Count	Type	Agent	Location
9.3.5.2...	Su_Failure	test41	10/9/2007 5:21:27 PM	1	Problem	TEC	db2inst1-root-/dev/pts/4

*Figure 6-86 TEC event arrived at OMNibus event list*

## 6.4 Support of TEC class hierarchy

The support of the TEC class hierarchy is based on the fact that there are TEC rules that are defined against parent classes, as in Figure 6-87.

```
rule: support_hierarchy:
(
  description: 'This event will be excuted later by a procedure at
the OMNibus server',
  event:_event of_class 'UPS'
  where
  [
    sub_origin: equals 'hierarchy'
  ],
  action: filter:
  (
    bo_set_slotval(_event,sub_source,'support hierarchy')
  )
).
```

Figure 6-87 TEC rule against a parent class UPS

This rule fires against the leaf TEC classes UPS\_Fan\_Down, UPS\_Temp\_Degraded, and UPS\_Fan\_Up. These classes are defined in the baroc file ups.baroc, which is listed in Figure 6-4 on page 193.

We test the above rule with the following **wpostemsg** commands:

```
wpostemsg -m test-hierarchy-01 hostname=mondorf sub_origin=hierarchy
UPS_Fan_Down TEC
```

```
wpostemsg -m test-hierarchy-02 hostname=mondorf sub_origin=hierarchy
UPS_Temp_Degraded TEC
```

On the TEC event console it looks like Figure 6-88.

	Time Received	Event Type	Class	Hostname	Severity	Status	Message	Sub-source
	10/8/07 16:27	Other	UPS_Fan_Down	mondorf	Warning	Open	test-hierarchy-01	support hierarchy
	10/8/07 16:27	Effect	UPS_Temp_Degraded	mondorf	Warning	Open	test-hierarchy-02	support hierarchy

Figure 6-88 TEC console for a hierarchy event

When this event is forwarded from the TEC server to the EIF probe, the field AlertGroup is filled with the TEC class name, and here we have added the assignment @Summary = \$msg to the standard EIF rules file tivoli\_eif.rules to fill the summary field. The details of this event at the OMNibus server are shown in Figure 6-89.

Event Information: Alert Status for Serial Number 3282	
Alert Fields   Alert Details   Journal	
Field	Value
Node Alias	9.3.5.201
TECEventHandle	1
Process Required	0
Prec. Entity ID	0
First Occurrence	10/8/2007 2:49:52 PM
Rem. Sec Obj.	
Port	0
Suppr./Escl.	Normal
Rem. Root Obj.	
ExtendedAttr	
Slot	0
Local Node Alias	
Managed Status	Managed
Alert Group	UPS_Temp_Degraded
Class	TME10tecad
Flash	No
URL	
Summary	test-hierarchy-02

Figure 6-89 Forwarded TEC event through EIF probe

**Note:** The EIF probe and the first integration packages (NetView and ITM) that use the EIF probe were developed before OMNibus 7.2 went GA, so they do not yet support the TEC class hierarchy.

Today there is no automated method for generating probe statements to parse the TEC class name (%ClassName) and assign the correct TEC class number in the ObjectServer class field. A switch statement needs to be manually created to perform this assignment based on values extracted from the output of the nco\_baroc2sql script. A revised version of this script is planned that will provide the required probe rule data.

To perform similar event processing as the above TEC rule we must execute some tasks first.

First we must copy the rulebase from the TEC server to the OMNIbus server to make the TEC classes available here. We have access to /tmp-ws/TEC\_CLASSES. There we find the file .load\_classes, illustrated in Figure 6-90.

```
root.baroc  
tec.baroc  
ups.baroc
```

*Figure 6-90 File .load\_classes*

Now we used the new command **nco\_baroc2sql** with the following syntax:

```
nco_baroc2sql -baroc /tmp-ws/TEC_CLASSES/.load_classes -sql insert.sql
```

It produced the output shown in Figure 6-91.

```
Info - Processing file /tmp-ws/TEC_CLASSES/root.baroc  
Info - Processing file /tmp-ws/TEC_CLASSES/tec.baroc  
Info - Processing file /tmp-ws/TEC_CLASSES/ups.baroc  
Info - Processed 27 BAROC classes.
```

*Figure 6-91 Output of nco\_baroc2sql*

The created file insert.sql contains two parts, as shown in Figure 6-92.

```
insert into master.class_membership (Class, ClassName, Parent)
values ( 76013, 'DB_Cleanup_event', 76000);
( 76010, 'TEC_Maintenance', 76000);
( 76006, 'TEC_Stop', 76000);
( 76008, 'TEC_Heartbeat', 76000);
( 76015, 'TEC_Tick', 76000);
( 76017, 'TEC_GWR_Event', 76000);
( 76005, 'TEC_Start', 76000);
( 76018, 'TEC_GWR_Start', 76017);
( 76002, 'TASK_COMPLETE', 76000);
( 76022, 'TEC_GWR_Error', 76017);
( 76007, 'TEC_DB', 76000);
( 76014, 'TEC_Generic', 76000);
( 76012, 'Escalate_event', 76000);
( 76021, 'TEC_GWR_Notice', 76017);
( 76011, 'TEC_Cleanup_event', 76000);
( 76024, 'UPS_Fan_Down', 76023);
( 76000, 'EVENT', -1 );
( 76009, 'TEC_Heartbeat_missed', 76000);
( 76001, 'TASK', -1 );
( 76025, 'UPS_Temp_Degraded', 76023);
( 76026, 'UPS_Fan_Up', 76023);
( 76003, 'TEC_Error', 76000);
( 76023, 'UPS', 76000);
( 76004, 'TEC_Notice', 76000);
( 76019, 'TEC_GWR_Stop', 76017);
( 76020, 'TEC_GWR_ReStart', 76017);
insert into master.class_membership (Class, ClassName, Parent)
values ( 76016, 'TEC_LOGGING_BASE', 76000);
go
```

Figure 6-92 File insert.sql - part one

Note that in this part we omitted the following prefix from the second line to the last, but one, line so that it can be easier read:

```
insert into master.class_membership (Class, ClassName, Parent) values
```

These SQL statements implement the class hierarchy in the ObjectServer. Now let us look at the second part (Figure 6-93).

```
insert into alerts.conversions values ( 'Class76013', 'Class',
76013, 'DB_Cleanup_event');
( 'Class76010', 'Class', 76010, 'TEC_Maintenance');
( 'Class76006', 'Class', 76006, 'TEC_Stop');
( 'Class76008', 'Class', 76008, 'TEC_Heartbeat');
( 'Class76015', 'Class', 76015, 'TEC_Tick');
( 'Class76017', 'Class', 76017, 'TEC_GWR_Event');
( 'Class76005', 'Class', 76005, 'TEC_Start');
( 'Class76018', 'Class', 76018, 'TEC_GWR_Start');
( 'Class76002', 'Class', 76002, 'TASK_COMPLETE');
( 'Class76022', 'Class', 76022, 'TEC_GWR_Error');
( 'Class76007', 'Class', 76007, 'TEC_DB');
( 'Class76014', 'Class', 76014, 'TEC_Generic');
( 'Class76012', 'Class', 76012, 'Escalate_event');
( 'Class76021', 'Class', 76021, 'TEC_GWR_Notice');
( 'Class76011', 'Class', 76011, 'TEC_Cleanup_event');
( 'Class76024', 'Class', 76024, 'UPS_Fan_Down');
( 'Class76000', 'Class', 76000, 'EVENT');
( 'Class76009', 'Class', 76009, 'TEC_Heartbeat_missed');
( 'Class76001', 'Class', 76001, 'TASK');
( 'Class76025', 'Class', 76025, 'UPS_Temp_Degraded');
( 'Class76026', 'Class', 76026, 'UPS_Fan_Up');
( 'Class76003', 'Class', 76003, 'TEC_Error');
( 'Class76023', 'Class', 76023, 'UPS');
( 'Class76004', 'Class', 76004, 'TEC_Notice');
( 'Class76019', 'Class', 76019, 'TEC_GWR_Stop');
( 'Class76020', 'Class', 76020, 'TEC_GWR_ReStart');
insert into alerts.conversions values ( 'Class76016', 'Class',
76016, 'TEC_LOGGING_BASE');
go
```

Figure 6-93 File *insert.sql* - part two

Note that for reading purposes we have omitted text here, the same omitting as in part one.

This part extends the conversion table of the OMNibus server. We executed both parts with the following **nco\_sql** command:

```
nco_sql -user root -server WEIMAR -password "" < insert.sql
```

The result of this command is listed in Figure 6-94 and Figure 6-95 on page 253.

RowID	RowSerial	Class #	Parent	ClassName
4565550235665	17	76000	-1	EVENT
4565550235667	19	76001	-1	TASK
4565550235657	9	76002	76000	TASK_COMPLETE
4565550235670	22	76003	76000	TEC_Error
4565550235672	24	76004	76000	TEC_Notice
4565550235655	7	76005	76000	TEC_Start
4565550235651	3	76006	76000	TEC_Stop
4565550235659	11	76007	76000	TEC_DB
4565550235652	4	76008	76000	TEC_Heartbeat
4565550235666	18	76009	76000	TEC_Heartbeat_missed
4565550235650	2	76010	76000	TEC_Maintenance
4565550235663	15	76011	76000	TEC_Cleanup_event
4565550235661	13	76012	76000	Escalate_event
4565550235649	1	76013	76000	DB_Cleanup_event
4565550235660	12	76014	76000	TEC_Generic
4565550235653	5	76015	76000	TEC_Tick
4565550235675	27	76016	76000	TEC_LOGGING_BASE
4565550235654	6	76017	76000	TEC_GwR_Event
4565550235656	8	76018	76017	TEC_GwR_Start
4565550235673	25	76019	76017	TEC_GwR_Stop
4565550235674	26	76020	76017	TEC_GwR_ReStart
4565550235662	14	76021	76017	TEC_GwR_Notice
4565550235658	10	76022	76017	TEC_GwR_Error
4565550235671	23	76023	76000	UPS
4565550235664	16	76024	76023	UPS_Fan_Down
4565550235668	20	76025	76023	UPS_Temp_Degraded
4565550235669	21	76026	76023	UPS_Fan_Up

Figure 6-94 Content of OMNibus table master.class\_membership



RowID	RowSerial	KeyField	Colname	Value	Conversion
4501125727068	860	Class69000	Class	69000	PsyQ
4501125727105	897	Class76000	Class	76000	EVENT
4501125727107	899	Class76001	Class	76001	TASK
4501125727097	889	Class76002	Class	76002	TASK_COMPLETE
4501125727110	902	Class76003	Class	76003	TEC_Error
4501125727112	904	Class76004	Class	76004	TEC_Notice
4501125727095	887	Class76005	Class	76005	TEC_Start
4501125727091	883	Class76006	Class	76006	TEC_Stop
4501125727099	891	Class76007	Class	76007	TEC_DB
4501125727092	884	Class76008	Class	76008	TEC_Heartbeat
4501125727106	898	Class76009	Class	76009	TEC_Heartbeat_missed
4501125727090	882	Class76010	Class	76010	TEC_Maintenance
4501125727103	895	Class76011	Class	76011	TEC_Cleanup_event
4501125727101	893	Class76012	Class	76012	Escalate_event
4501125727089	881	Class76013	Class	76013	DB_Cleanup_event
4501125727100	892	Class76014	Class	76014	TEC_Generic
4501125727093	885	Class76015	Class	76015	TEC_Tick
4501125727115	907	Class76016	Class	76016	TEC_LOGGING_BASE
4501125727094	886	Class76017	Class	76017	TEC_GWR_Event
4501125727096	888	Class76018	Class	76018	TEC_GWR_Start
4501125727113	905	Class76019	Class	76019	TEC_GWR_Stop
4501125727114	906	Class76020	Class	76020	TEC_GWR_ReStart
4501125727102	894	Class76021	Class	76021	TEC_GWR_Notice
4501125727098	890	Class76022	Class	76022	TEC_GWR_Error
4501125727111	903	Class76023	Class	76023	UPS
4501125727104	896	Class76024	Class	76024	UPS_Fan_Down
4501125727108	900	Class76025	Class	76025	UPS_Temp_Degraded
4501125727109	901	Class76026	Class	76026	UPS_Fan_Up

Figure 6-95 Content of OMNibus table alerts.conversions

Based on the master.class\_membership table, we will create a lookup table to be used in the tivoli\_eif.rules file to fill the ObjectServer Class field, depending on the content of the TEC class name. First we create an ObjectServer file that will be filled with lookup table information by using an SQL procedure. To create this file we execute the following command:

```
create or replace file lookup '/tmp/tec_lookup' maxfiles 1 maxsize 1M;
go
```

Now we create the SQL procedure to fill:

```
create or replace procedure class_lookup_create()
begin
  for each row tempclass in master.class_membership
  where tempclass.Class >= 76000 and tempclass.Class <= 86000
```

```

begin
  write into lookup
    values (tempclass.ClassName ,',to_char (tempclass.Class));
end;
go

```

Next, we execute this procedure with the following SQL statements:

```

execute procedure class_lookup_create;
go

```

Now the lookup file looks like Example 6-2.

*Example 6-2 New lookup file*

---

TASK	76001
EVENT	76000
TEC_Error	76003
TASK_COMPLETE	76002
TEC_Notice	76004
TEC_Start	76005
TEC_DB	76007
TEC_Stop	76006
TEC_Heartbeat_missed	76009
TEC_Heartbeat	76008
TEC_Cleanup_event	76011
TEC_Maintenance	76010
Escalate_event	76012
DB_Cleanup_event	76013
TEC_Generic	76014
TEC_Tick	76015
TEC_LOGGING_BASE	76016
TEC_GWR_Event	76017
UPS	76023
TEC_GWR_Stop	76019
TEC_GWR_Start	76018
TEC_GWR_ReStart	76020
TEC_GWR_Notice	76021
TEC_GWR_Error	76022
UPS_Temp_Degraded	76025
UPS_Fan_Down	76024
UPS_Fan_Up	76026

---

We can now use this file in our `tivolil_eif.rules` file at the top of the file by inserting the next two lines:

```
table tec_class = "/tmp/tec_lookup1"
default = "Unknown"
```

To use this in the rules file we provide the next statement as follows:

```
$Class = lookup($ClassName,tec_class)
```

Here we fill the `ObjectServer Class` field with a number that is dependent on the TEC class name.

The goal of this section is to implement a similar automation function regarding the TEC hierarchy as the above implemented TEC rule. Therefore, we developed the next SQL procedure. See Figure 6-96.

```
begin
  for each row tmp in alerts.status
  begin
    if ( instance_of (tmp.Class, 'UPS') = TRUE ) THEN
      set tmp.Summary = 'hierarchy-test-ok';
    end if;
  end;
end
```

Figure 6-96 OMNibus SQL procedure for hierarchical TEC events

Here all rows of the `alerts.status` are inspected as to whether the content of the `AlertGroup` field is a child class of the TEC parent class 'UPS'. In this case the `Summary` field is set to 'hierarchy-test-ok'. Figure 6-97 and Figure 6-98 show our test before and after the execution of the above procedure. You must pay attention to the `Summary` field to see the differences.

Node	Alert Group	Summary	Last Occurrence	Count	Type	Agent
9.3.5.201	UPS_Temp_Degraded	test-hierarchy-02	10/8/2007 3:26:39 PM	1	Problem	TEC
9.3.5.201	UPS_Fan_Down	test-hierarchy-01	10/8/2007 3:26:23 PM	1	Problem	TEC

Figure 6-97 OMNibus events before procedure execution

Node	Alert Group	Summary	Last Occurrence	Count	Type	Agent
9.3.5.201	UPS_Temp_Degraded	hierarchy-test-ok	10/8/2007 3:26:39 PM	1	Problem	TEC
9.3.5.201	UPS_Fan_Down	hierarchy-test-ok	10/8/2007 3:26:23 PM	1	Problem	TEC

Figure 6-98 OMNibus events after procedure execution

We demonstrated that we can execute OMNibus automation in the context of a TEC class hierarchy.

## 6.5 TEC information/URL information for events

Many customers use the TEC information button on the TEC console to display adapted help text. Figure 6-99 shows this button in the right bottom corner.

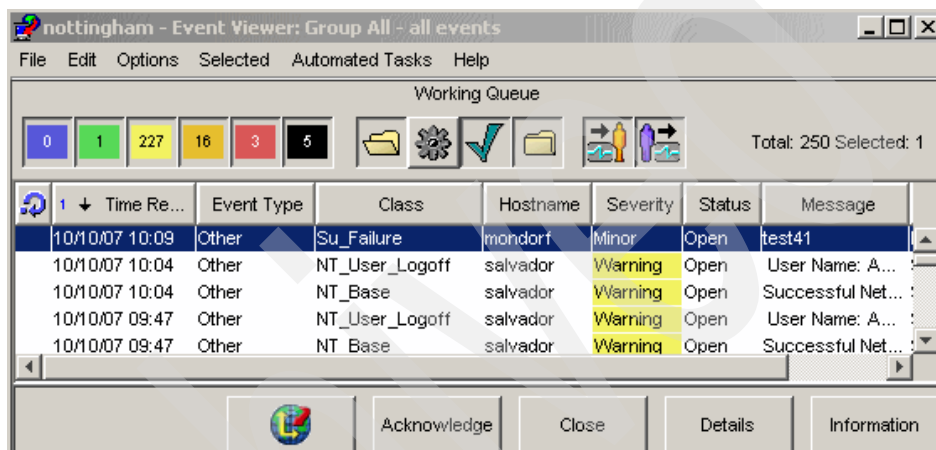


Figure 6-99 TEC console with information button

When you click this button the browser starts and the next URL is shown. Here we selected the Su\_Failure event to demonstrate this.

## Sample event information file for class Su\_Failure

1. [How do I handle the Su\\_Failure event?](#)
2. [Where can I find additional information?](#)
3. [Whom should I talk to if I can't resolve this problem?](#)
4. [Here are the details for the event.](#)

---

### How do I handle the Su\_Failure event?

Here, you might place a description of how this event is handled at your site.

[Back to Top](#)

---

### Where can I find additional information?

Here, you might place a description of where additional information can be found.

[Back to Top](#)

---

### Whom should I talk to if I can't resolve this problem?

Here, you might give the name and/or number of a person to contact for additional assistance.

[Back to Top](#)

---

### Your script can key off of any of the attributes of the event:

Attribute	
acl	[ admin]
adapter_host	N/A
cause_date_reception	0
cause_event_handle	0
class	Su_Failure

Figure 6-100 TEC sample event information for class Su\_Failure

What happens behind the scene? The TEC console starts the perl script `tec_help.pl` from the directory `/Tivoli/bin/aix4-r1/TAS/HTTPd/cgi-bin`. Let us look at parts of this script (Figure 6-101).

```
...part 1...
$class = $input{'class'};
$classD0Thostname .= $class . "." . $input{'hostname'};
$classD0ThostnameD0Tsource .= $classD0Thostname . "." .
$input{'source'};
$classD0ThostnameD0TsourceD0Tseverity .= $classD0ThostnameD0Tsource
. "." . $input{'severity'};
...part 2...
$prefilename = "/Tivoli/bin/aix4-r1/./generic/HTTPd/Tec/";
$locfilename = $prefilename . $locale . '/';
$locfilename0 = $locfilename .
$classD0ThostnameD0TsourceD0Tseverity;
$locfilename1 = $locfilename . $classD0ThostnameD0Tsource;
$locfilename2 = $locfilename . $classD0Thostname;
$locfilename3 = $locfilename . $class;
...partr 3...
$testfilename0 = $prefilename .
$classD0ThostnameD0TsourceD0Tseverity;
$testfilename1 = $prefilename . $classD0ThostnameD0Tsource;
$testfilename2 = $prefilename . $classD0Thostname;
$testfilename3 = $prefilename . $class;
...part 4...
if ( -e $testfilename0) {
    &PRINT_OUT_FILE_CONTENTS($testfilename0);
}
elsif ( -e $testfilename1) {
    &PRINT_OUT_FILE_CONTENTS($testfilename1);
}
elsif ( -e $testfilename2) {
    &PRINT_OUT_FILE_CONTENTS($testfilename2);
}
}
```

Figure 6-101 Extract of the file `tec_help.pl`

In the first part the TEC attribute contents are assigned to local variables. In the second and the third part the specific file name, dependent on the event attributes, is constructed. In part four we test whether the specific file name (starting from the specific file name to the common file name) exists and, if so, whether it will be printed to standard out, which is piped to the browser here. The HTML input files are located in the directory `/Tivoli/bin/generic/HTTPd/Tec`.

Su\_Failure is one of the file names in this directory and contains the content shown in Figure 6-102.

```
<!-- TRANSLATORS: Do NOT change any occurrences of Su_Failure.
      This file is a template. During the automated build of the
product, that
      text is replaced.
-->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=ISO-8859-1">
<TITLE>Su_Failure</TITLE>

</HEAD>

<BODY>

<H2>Sample event information file for class
      <A NAME="top">Su_Failure</A>
</H2>
```

Figure 6-102 First lines of file Su\_Failure

It will be very helpful for customers if the content of this directory can be used in OMNIbus without modification. We show two solutions that cover this topic.

### Solution one

In OMNIbus there is a standard attribute named URL. If this field is filled at the probe level the operator can use the standard tool of the OMNIbus event console: **Alert** → **Tool** → **URL**. After this the browser opens with this specified URL. We test this functionality by inserting an event via nco\_sql with the statement shown in Figure 6-103

```
insert into alerts.status (Identifier, Summary, Node, Agent, Type,
Severity, AlertKey, AlertGroup, Manager, LastOccurrence, URL) values
('itso-help-01', 'Su_Failure event', 'mondorf', 'Agent', 1, 3,
'UPS', 'SNMP', 'nco_sql', getdate(),
'/tec-http/HTTPd/Tec/Su_Failure');
go
```

Figure 6-103 nco\_sql input for an event with a filled URL field

After sending this event with nco\_sql it shows up at the event list, as shown in Figure 6-104.

Node	Alert Group	Summary	Last Occurrence	URL	Count	Type
mondorf	SNMP	Su_Failure event	/10/2007 01:13:15	/tec-http/HTTpd/Tec/Su_Failure	1	Problem

Figure 6-104 OMNibus event list for the specific URL event



When we now select the event and use the above described tool, we get the browser appearance shown in Figure 6-105.

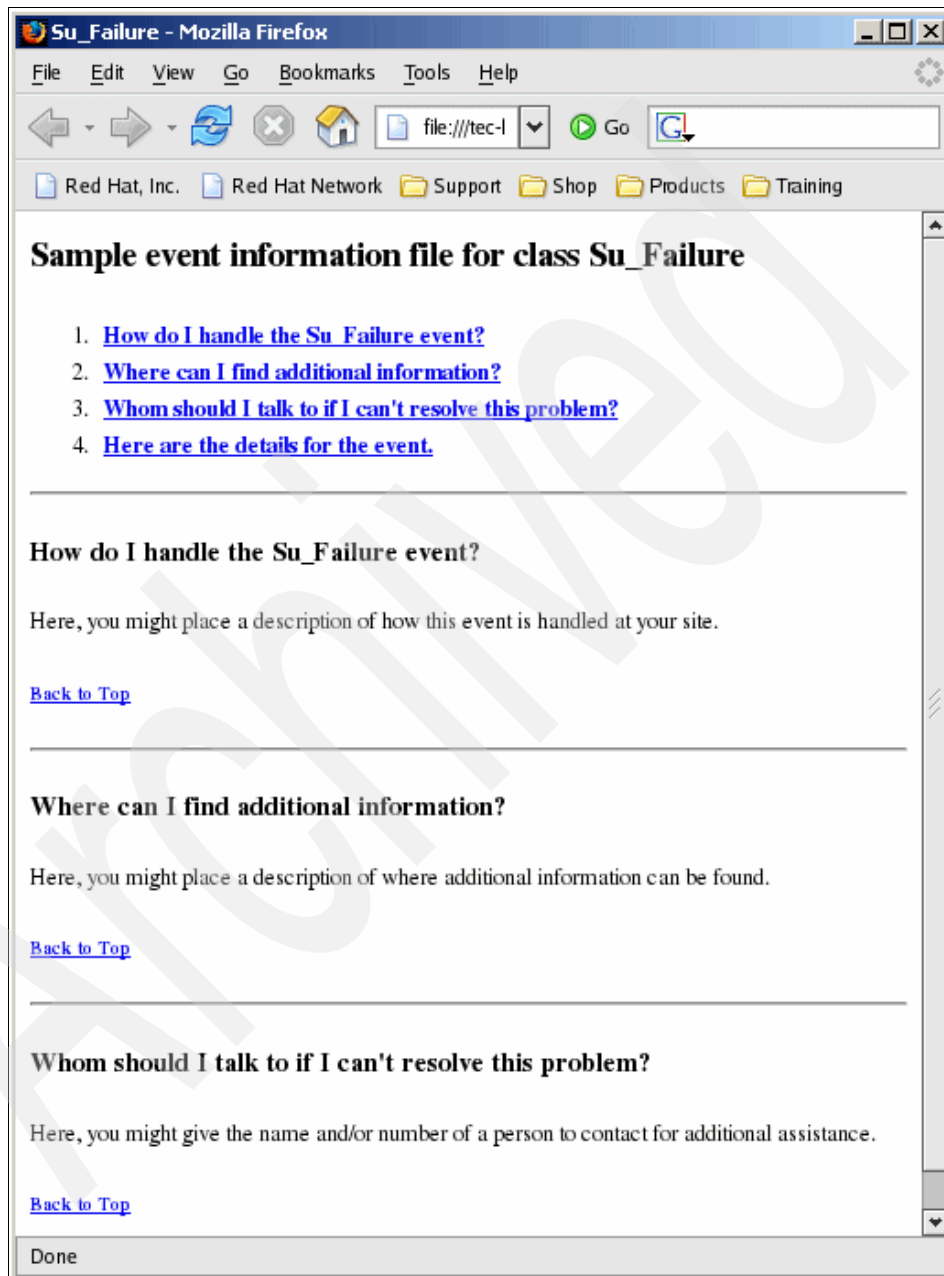


Figure 6-105 Browser output for the specified URL

To get this output we copied the directory /Tivoli/bin/generic/HTTPd/Tec and its content files from the TEC machine nottingham to the directory /tec-http/HTTPd/Tec onto the OMNIBus machine weimar.

In this solution you must define the help URL at the probe level for every event type.

## Solution two

This solution depends on a script that you use centrally at the OMNIBus event list similar to the TEC information script. It would be nice to use the same script as it is delivered with TEC. This will not work because the attribute names and the attribute contents are passed from the TEC console as environment variables to this script and OMNIBus does this differently. In OMNIBus you can execute a script with parameters, and these parameters contain the content of event attributes.

Let us now look at how to start this tool. We changed the tool “Open URL” from \$OMNIBROWSER @URL to:

```
$OMNIBROWSER
"http://weimar/cgi-bin/tec_help.pl?class=@AlertGroup;sub_origin=@Node;prio=@Severity"
```

Let us look at the sample script `tec_help.pl` in Example 6-3, which we used in our lab environment. We put the variables in **bold**. To get this script working we installed on the machine weimar the Web server from the Red Hat Linux CD, copied this script to /var/www/cgi-bin, and changed the attributes so that they can be executed. This script reads several files from flat files that are located under the /tecinfo directory. We will reference them here.

*Example 6-3 Perl script to generate dynamic HTML pages*

---

```
#!/usr/bin/perl
#-----#
# filename: tec_help.pl                                #
#-----#
#
# Description:                                          #
# This CGI script creates an information html page    #
#
#-----#

### modules ###
use CGI qw/:standard/;

### variables and structures ###
```

```

my %Parameter;
my
@month=('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Nov','Dec');
my $InfoPath="/tecinfo";
my @WhatText=();
my @HowText=();

### Main ###
foreach $key (param()) {
#  system("mkdir -p $InfoPath/$key") if(! -d "$InfoPath/$key" );
  $Parameter{$key}=param($key);
}
my @ClassText=`cat $InfoPath/class/$Parameter{'class'}`;
my @WhoText=`cat $InfoPath/sub_origin/$Parameter{'sub_origin'}`;

foreach $_ (@ClassText) {
  chomp;
  if( /\[Description/ ) {
    $readflg=1;
  } elsif( /\[Action/ ) {
    $readflg=2;
  } else {
    push(@WhatText,"$_\n")if $readflg == 1 ;
    push(@HowText,"$_\n")if $readflg == 2 ;
  }
}

html_header();
info_menu();
get_prio();
event_what();
event_how();
event_who();
event_detail();
html_end();

sub html_header {
  print "Content-type: text/html;Charset=iso-8859-1\n";
  print "\n";
  print "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 3.2 Final//EN\">\n";
  print "<HTML>\n";
  print "<HEAD>\n";
  print "<META HTTP-EQUIV=\"Content-Type\" CONTENT=\"text/html; charset=ISO-8859-1\">\n";
}

```

```

        print "<TITLE>".$Parameter{'class'}."</TITLE>\n";
        print "</HEAD>\n";
        print "<BODY>\n";
    }
    sub html_end {
        print "</BODY>\n";
        print "</HTML>\n";
    }
    sub info_menu {
        print "<H2>Event information for class\n";
        print "<FONT COLOR=#FF0000><A
NAME=\"top\">".$Parameter{'class'}."</A></FONT>\n";
        print "</H2>\n";
        print "<OL>\n";
        print "<LI><A HREF=\"#what\"><STRONG>What does this message
mean?</STRONG></A></LI>\n";
        print "<LI><A HREF=\"#how\"><STRONG>What should I do with this
message?</STRONG></A></LI>\n";
        print "<LI><A HREF=\"#who\"><STRONG>Who should be
informed?</STRONG></A></LI>\n";
        print "<LI><A
HREF=\"#vars\"><STRONG>Event-Details</STRONG></A></LI>\n";
        print "</OL>\n";
        print "<HR>\n";
    }

    sub event_what {
        print "<H3><A NAME=\"what\">What does this message
mean?</a></h3>\n";
        print "<FONT SIZE=+1>\n";
        foreach $_ (@WhatText) {
            print "$_<br>\n";
        }
        print "</FONT>\n";
        print "<H5><A HREF=\"#top\">Back to top</A></H5>\n";
        print "<HR>\n";
    }

    sub event_how {
        print "<H3><A NAME=\"how\">What should I do with this
message?</a></h3>\n";
        print "<FONT SIZE=+1>\n";
        foreach $_ (@HowText) {
            print "$_<br>\n";
        }
    }

```

```

    print "</FONT>\n";
    print "<H5><A HREF=\"#top\">Back to top</A></H5>\n";
    print "<HR>\n";
}

sub event_who {
    print "<H3><A NAME=\"who\">Who should I inform?</a></h3><p>\n";
    print "<FONT SIZE=+1>\n";
    foreach $_ (@WhoText) {
        print "$_<br>\n";
    }
    print "</FONT>\n";
    print "<H5><A HREF=\"#top\">Back to top</A></H5>\n";
    print "<HR>\n";
}

sub event_detail {
    print "<h3><a name=\"vars\">Passed parameters:</a></h3><p>\n";
    print "<TABLE BORDER=1 CELLPADDING=4 WIDTH=90%>\n";
    print "<TR><TD><B>Parameter</B></TD><TD><B>Wert</B></TD></TR>\n";
    foreach $_ (sort keys %Parameter) {
        if(/last_modified_time/ || /date_reception/ ) {
            ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst)=localtime($Parameter{$_});
            $value=sprintf("%3s %02d %02d:%02d:%02d
%04d\n",$month[$mon],$mday,$hour,$min,$sec,$year+1900);
        } else {
            if($Parameter{$_}) {
                $value=$Parameter{$_};
            } else {
                $value="&nbsp;";
            }
        }

        print "<TR><TD>$_</TD><TD><FONT
COLOR=\"#0000F5\">$value</FONT></TD></TR>\n";
    }
    print "</TABLE>\n";
    print "</p><H5><A HREF=\"#top\">Back to top</A></H5>\n";
}

sub get_prio {
    @PrioText=`cat $InfoPath/prio/$Parameter{'prio'}~`;

```

```

if( $Parameter{'prio'} == 1) {
    print "<FONT COLOR=#FF0000>\n";
} elseif ( $Parameter{'prio'} == 2) {
    print "<FONT COLOR=#D46A02>\n";
} elseif ( $Parameter{'prio'} == 3) {
    print "<FONT COLOR=#0000FF>\n";
}
foreach $_ (@PrioText) {
    print "<center>$_<br></center>\n";
}
print "</FONT>\n";
}

```

---

This script uses the passed parameters and generates, with the help of text files, dynamic HTML code, which is presented in the browser that is referenced with the variable OMNIBROWSER.

The ClassText variable will contain the content of the file /tecinfo/classes/<class-name>. In our example this is the content of the attribute AlertGroup. The WhoText variable contains the content of the file /tecinfo/sub\_origin/<who-name>. Here the variable who-name is substituted with the content of the attribute node. Finally, we have the third and last variable PrioText, which will contain the priority depending on the attribute Severity field.

We will send two events with nco\_sql to OMNIBus and will look at the URLs by using the URL tool depending on the selection of an event. The first event is:

```

insert into alerts.status (Identifier, Summary, Node, Agent, Type,
Severity, AlertKey, AlertGroup, Manager, LastOccurrence ) values
('itso-help-02', 'Su_Failure event', 'mondorf', 'Agent', 1, 2, 'UPS',
'Su_Failure', 'nco_sql', getdate());
go

```

The execution of this SQL statement results in the event list shown in Figure 6-106.

Node	Alert Group	Summary	Last Occurrence	Count	Type	ExpireTime
mondorf	Su_Failure	Su_Failure event	0/11/2007 11:53:01 A	1	Problem	Not Set

Figure 6-106 OMNIBus event list for the first help event

When we use now the modified URL tool with **Alerts** → **Tools** → **URL**, we get the browser content shown in Figure 6-107 and Figure 6-108.

**Event information for class Su\_Failure**

- [1. What does this message mean?](#)
- [2. What should I do with this message?](#)
- [3. Who should be informed?](#)
- [4. Event-Details](#)

---

\*\*\*\*\*

Priority 2 (Medium)

\*\*\*\*\*

**What does this message mean?**

Someone tried to switch the UID  
Refer to the attribut user\_to

[Back to top](#)

---

**What should I do with this message?**

If the taget user is root, call the college which is referenced in the attribute from\_user

[Back to top](#)

Figure 6-107 Browser help text for Su\_Failure event - part one

Let us look at the content of the file /tecinfo/classes/Su\_Failure (Figure 6-108).

[Description]  
Someone tried to switch the UID  
Refer to the attribut user\_to

[Action]  
If the taget user is root, call the college which is referenced in  
the attribute from\_user

Figure 6-108 Content of file /tecinfo/classes/Su\_Failure

Here you can see that the content of the text file is shown in the browser. Another part of the browser content shows the file /tecinfo/prio/2, which contains the text shown in Figure 6-109.

\*\*\*\*\*  
Priority 2 (Medium)  
\*\*\*\*\*

Figure 6-109 Content of file /tecinfo/prio/2

In the second part of the browser’s content you see who should be informed, and you can see the passed parameters (Figure 6-110).

Who should I inform?

\*\*\*\*\*

Unix Support

\*\*\*\*\*

Monday thru Sunday

00:00 - 24:00

Please send a message to

unix-group

\*\*\*\*\*

[Back to top](#)

Passed parameters:

Parameter	Wert
class	<a href="#">Su_Failure</a>
prio	2
sub_origin	<a href="#">mondorf</a>

[Back to top](#)

Figure 6-110 Browser help text for Su\_Failure event - part two



Now it is time to look at the file /tecinfo/sub\_origin/mondorf (Figure 6-111).

```
*****
Unix Support
*****

Monday thru Sunday
00:00 - 24:00

Please send a message to
unix-group

*****
```

Figure 6-111 Content of file /tecinfo/sub\_origin/mondorf

You can see that this is the part that is shown under the topic “Who should I inform” in the browser.

Now we send the second event:

```
insert into alerts.status (Identifier, Summary, Node, Agent, Type,
Severity, AlertKey, AlertGroup, Manager, LastOccurrence ) values
('itso-help-03', 'UPS_Fan_down event', 'nottingham', 'Agent', 1, 3,
'UPS', 'UPS_Fan_Down', 'nco_sql', getdate());
go
```

For this event we use the same tool as the OMNibus event list. Let us look at this event (Figure 6-112).

Node	Alert Group	Summary	Last Occurrence	Count	Type	ExpireTime
nottingham	UPS_Fan_Down	UPS_Fan_Down event	0/12/2007 03:24:45 P	1	Problem	Not Set

Figure 6-112 OMNibus event list for the second help event

Now we can inspect the corresponding browser and file contents (Figure 6-113).

**Event information for class UPS\_Fan\_Down**

- [What does this message mean?](#)
- [What should I do with this message?](#)
- [Who should be informed?](#)
- [Event-Details](#)

\*\*\*\*\*  
Priority 3 (low)  
\*\*\*\*\*

**What does this message mean?**

The fan of the UPS is down.

[Back to top](#)

**What should I do with this message?**

Call the service 999 and inform them about this topic

[Back to top](#)

Figure 6-113 Browser help text for TEC\_Start event - part one

The content of the file /tecinfo/classes/UPS\_Fan\_Down looks like Figure 6-114.

[Description]  
The fan of the UPS is down.

[Action]  
Call the service 999 and inform them about this topic

Figure 6-114 Content of the file /tecinfo/classes/UPS\_Fan\_Down

Another assignment shows the file `/tecinfo/prio/3`, which contains the text shown in Figure 6-115.

```
*****  
Priority 3 (low)  
*****
```

*Figure 6-115 Content of file `/tecinfo/prio/3`*

The content of the file `/tecinfo/sub_origin/nottingham` looks like Figure 6-116.

```
*****  
UPS Support  
*****  
  
Monday thru Sunday  
00:00 - 24:00  
  
Please send a message to  
UPS-group  
  
*****
```

*Figure 6-116 Content of file `/tecinfo/sub_origin/nottingham`*

The second part of the browser content shows the text shown in Figure 6-117.

Who should I inform?

\*\*\*\*\*

UPS Support

\*\*\*\*\*

Monday thru Sunday

00:00 - 24:00

Please send a message to

UPS-group

\*\*\*\*\*

[Back to top](#)

Passed parameters:

Parameter	Wert
class	<a href="#">UPS_Fan_Down</a>
prio	3
sub_origin	<a href="#">nottingham</a>

[Back to top](#)

Figure 6-117 Browser help text for TEC\_Start event - part two

In the second solution we provided, at a central location, a script to generate dynamic HTML pages. Combined with the first solution we cover the most used scenarios of the TEC world regarding this topic.

The advantage of the second solution is that the messages that are presented in the browser are plain text. Therefore, they can easily be created from the appropriate responsible department.

## 6.5.1 Rule best practices for performance

There are several features of OMNIbus and the OMNIbus probe language that can be utilized to handle high event rates in a large production environment. These techniques can be used in high traffic environments, as well as simply to process event suppression and other event control situations, for example:

- ▶ Using direct advantage of de-duplication to clear events
- ▶ Using the probe to send to tables other than `alerts.status`
- ▶ Keeping alarms that must be consolidated, out of `alerts.status`
- ▶ Using the load functions within the probe rules

Typically, the Generic Clear automation is used to match a resolution event to a problem event and clear the event in the Event List. With more control over de-duplication in v7.x, you can use common identifiers for problem and resolution events, and properly update the problem event to indicate the circumstances of the event resolution. The benefits in using de-duplication for clearing are:

- ▶ Reduce Generic Clear overhead - The Generic Clear automation requires a good deal of processing because of two stages of gathering potentially large tables in memory.
- ▶ Instant clearing - Clearing by deduplication happens instantly. Generic Clear runs on a timed cycle.
- ▶ Reduce overall number of events - The number of events is reduced by half for all cases implemented.

## 6.5.2 Debugging using Netcool IDE

Given a customer-supplied rules and raw capture file, you can set up the Netcool IDE on Windows to allow raw capture files to be played back without processing the events through an ObjectServer. It provides an offline development and dry run replaying method for developing probe rules file logic. It provides replayed alarms, seeing how the developed rules file affect the event in real time.

Unless you are running the probes on a Windows machine, you will most likely need to modify the paths in the rules files that refer to external files, such as include files and lookup files. To correct for these discrepancies the IDE has a tool that will either make the changes in its memory resident copy of the rules file or optionally on disk.

The IDE needs to have an ObjectServer schema defined that it can validate the rules file. You need either an ObjectServer with the correct schema or a .sql file with the correct schema. In OMNIbus Version 7 this would be the `application.sql` file from `$OMNIHOME/etc`. If you have a sql file with your fields, use that. If not, you can use the enclosed V7 file (`application.sql`).

Remember that you are not using an actual ObjectServer, so certain features that we take for granted are not available (like deduplication). Each event in the capture file will be represented as a new event in the list rather than deduplicating. Since this application is memory resident you will quickly use several hundred megabytes of RAM, so take advantage of the pause button located to the left of the event list.

### 6.5.3 Netcool Knowledge Library

The Netcool/OMNIBus Netcool Knowledge Library (NcKL) is an important component of the correlation capabilities. The Knowledge Library improves the capability of the Netcool/OMNIBus by providing more valuable information. The Knowledge Library is written to a common standard, and provides improved correlation and causal analysis for the IBM Tivoli Netcool suite. A set of formally tested *ready to run* probe rules for specific devices identify which alarms indicate actual failures. This allows repair efforts to focus on the issues (or root causes) that truly affect the operation of the infrastructure, without the distraction of the symptomatic or informational events. The device-specific rules dictate how events should be correlated by providing greater detail on the specific containment of events for a particular device. These rules complement the current out-of-the-box event correlation capabilities of Netcool/OMNIBus and Tivoli Network Manager, enabling enhanced root cause analysis.

Root cause analysis has become an issue of paramount importance for the management of communications and information systems infrastructures. Loosely defined, root cause analysis is the process of making sense of large numbers of alert, status, and informational messages (events) that might be generated by such infrastructures.

While some events indicate actual failures that require correction, many others are simply symptoms of the actual failures, or informational messages about normal operations of the infrastructure. Netcool/OMNIBus Knowledge Library aims to identify which alarms indicate actual failures, allowing repair efforts to focus on the issues (or root causes) that truly affect the operation of the infrastructure, without the distraction of the symptomatic or informational events. The end result is a reduction of *mean time to repair* and increased availability of the systems.

The Knowledge Library additionally increases the ability of the Tivoli Netcool/OMNIBus ObjectServer automations to correlate alarms and identify root causes by employing the following techniques:

- Event pre-classification: This process identifies and flags events within the probe rules files to indicate the causal relevance of events, where this can be determined without the need for correlation.

- ▶ Intra-device correlation: This process enhances probe rules files and adds automations to the ObjectServer to perform correlation beyond deduplication and problem/resolution correlation, identifying intra-device root causes and symptoms.
- ▶ AMOS extended event recognition (for IBM Tivoli Network Manager IP Edition integration): This process provides IBM Tivoli Network Manager IP Edition with a larger data set upon which to perform topology-based event correlation, by identifying a larger set of events for analysis.

To effectively install IBM Tivoli Netcool/OMNIBus Knowledge Library and fully realize the benefits delivered, you must be familiar with the underlying principles of IBM Tivoli Netcool/OMNIBus v7.x, including the following:

- ▶ The IBM Tivoli Netcool/OMNIBus components
  - The ObjectServer (including the database tables and columns)
  - Probes (including editing probe properties and stopping/restarting probes)
  - Desktop tools (including event lists, filters, and views)
  - Administration tools (including the IBM Tivoli Netcool/OMNIBus Administrator, the SQL interactive interface, and process control)
- ▶ The IBM Tivoli Netcool/OMNIBus v7.x directory structure and necessary configuration files
- ▶ Basic rules file syntax including the use of lookup tables (both inline and separate files)
- ▶ Permissions, conversions, and automations

Archived



## Configuring the event sources

In this chapter we describe how to perform the final tasks of event sources migration and configuration in order to enable OMNIbus to receive events from the entire infrastructure.

## 7.1 Adding a rule to forward raw events to OMNibus

The first phase of the upgrade process, once the TEC and the OMNibus environments are installed, is to forward on events from TEC, *before they have gone through TEC rules*, to the EIF probe and then on to OMNibus. Doing this when they are in the *raw state* allows a better comparison of the event processing, and is particularly useful while developing the OMNibus event automations.

To ensure that the events have not been processed by TEC before being passed on, it is important to load the rule as the first rule in the rulebase on TEC.

We also recommend using the TEC rules predicate 're\_send\_event\_conf' rather than 'forward\_event' so that multiple destinations can be specified in the configuration files, and the corresponding cache files configured, too. This is important if your TEC already forwards to another system. We created a rule called forward.rls.

The configuration steps are:

1. Create the rule shown in Figure 7-1.

```
/*
*****/
/*
  This ruleset is used for forwarding events
*/

/***** Reception Rules
*****/
rule: forward:
(
  event: _event of_class EVENT,
  reception_action: forward:
  (
    re_send_event_conf('eif-weimar',_event),
    re_send_event_conf('eif-bari',_event)
  )
).

```

Figure 7-1 TEC forwarding rules

2. The configuration files, in our example, eif-weimar and eif-bari, must reside in the active rulebase's TEC\_RULES directory. Figure 7-2 is an example of one of them.

```
ServerLocation=bari
ServerPort=9999

EventMaxSize=4096
NO_UTF8_CONVERSION=YES

BufEvtMaxSize=1000
BufEvtPath=$TIVOLIHOME/tec/tecad_logfile_bari.cache
```

*Figure 7-2 Forwarding configuration file for one host*

Note that we increased the default cache (BufEvtMaxSize) from 64 k to 1000 k or 1 Mb to give more resilience should the EIF probe not be available.

The command to put them in the EventServer target directory is:

```
wrb -imptgtdata eif-bari.conf EventServer residency_rb
```

3. Import the rule with:

```
wrb -imprbrule forward.rls residency_rb
```

4. Ensure that the forwarding rule is first in the list of rule\_sets\_Eventserver:

```
wrb -imptgtrule forward -before cleanup Eventserver residency_rb
```

```
rule_set: forward
rule_set: cleanup
rule_set: lab
rule_set: netview
rule_set: omegamon
data: eif-bari.conf
data: eif-weimar.conf
```

*Figure 7-3 rule\_sets\_EventServer file*

5. Compile the rulebase with:

```
wrb -comprules residency_rb
```

6. Load the rulebase with:

```
wrb -loadrb residency_rb
```

7. Restart the eventserver:  
`wstop evsr; wstart evsr`
8. Check that events are now received both in the TEC database and in the ObjectServer database.

**Note:** You will now have the same events in *both* TEC and OMNibus. The main intention for this method is to assist the development and testing process. Before you go into production be sure to either turn this off or handle the parallel events accordingly. Particular attention should be paid to any automations or rules that send off automatic actions or alerts and e-mails, as you do not want duplicates of these.

## 7.2 Integration between Netcool/OMNibus and Tivoli NetView

Tivoli NetView can be configured to send events directly to Netcool/OMNibus without using Tivoli Enterprise Console as an intermediate event collector. This section describes how to enable this configuration.

## 7.2.1 Netcool/OMNIBus 7.2 and Tivoli NetView integration overview

Figure 7-4 shows an integration scenario between Netcool/OMNIBus and Tivoli NetView.

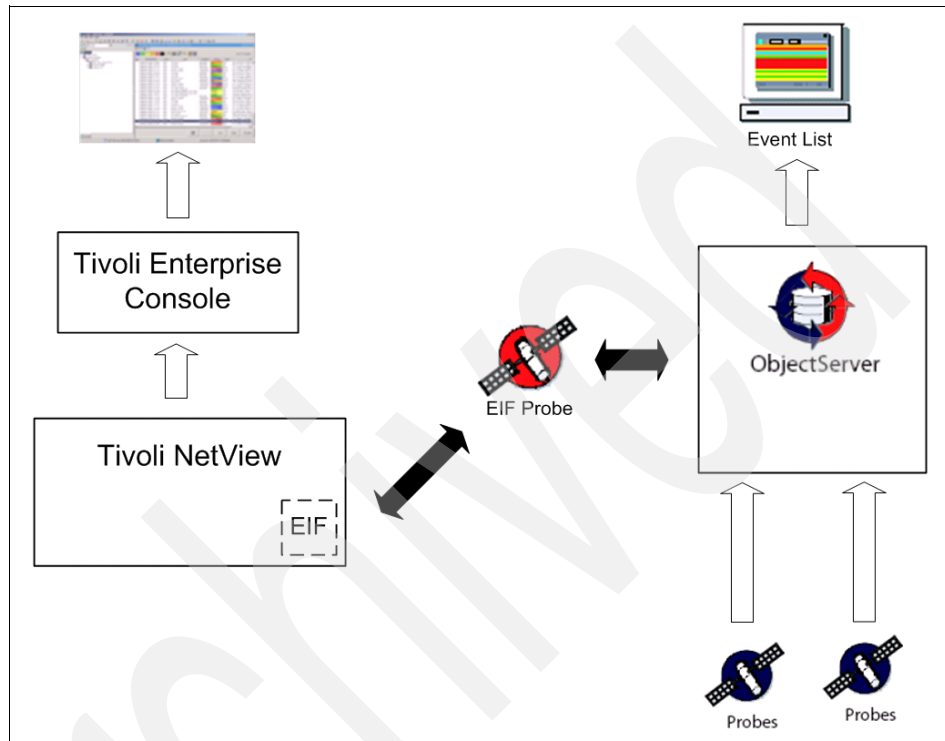


Figure 7-4 Netcool/OMNIBus and NetView integration

In this architecture, Netcool/OMNIBus is the event management focal point for the entire infrastructure, collecting alerts from sources such as Tivoli Network Manager IP Edition, Syslog Probe, Windows NT Event Log Probe, and so on. The standard interfaces for event synchronization are delivered out of the box, with predefined automation rules for handling several update and synchronization scenarios.

The integration of the two products can be achieved through the following steps:

1. Tivoli EIF Probe installation
2. Configuring OMNIBus ObjectServer
3. NetView Tivoli Enterprise Console adapter configuration
4. Configuration of the correlation behavior

**Note:** The Netcool suite also includes the NetView probe for the integration between OMNIBus and Tivoli NetView, but we suggest adopting the strategy described below as the best practice.

## 7.2.2 Installing Netcool/OMNIBus probe for Tivoli EIF

The Netcool/OMNIBus probe for Tivoli EIF is a prerequisite for the integration between Tivoli Enterprise Console and Netcool/OMNIBus, as it is to the integration between Netcool/OMNIBus and IBM Tivoli Monitoring.

## 7.2.3 Configuring OMNIBus ObjectServer

In this step you configure the OMNIBus ObjectServer to receive the event information forwarded by NetView and display them in the OMNIBus console.

In this section we describe the UNIX procedure to customize the integration.

### Updating OMNIBus ObjectServer schema

Additions to the Netcool/OMNIBus database schema are needed to reflect the information being sent to Netcool/OMNIBus from NetView. These modifications are read from SQL. The command to configure OMNIBus pipes the SQL command set into the SQL command-line tool and performs the updates to the ObjectServer.

Update the ObjectServer database with the new NetView attributes with the following commands:

```
$OMNIHOME/bin/nco_sql -user <username> -password <password> -server  
<server_name> < <path_to_file>/tec_db_update.sql
```

Where:

<b>\$OMNIHOME</b>	The system-defined variable defining the install location of OMNIBus
<b>&lt;username&gt;</b>	OMNIBus ObjectServer user name
<b>&lt;password&gt;</b>	OMNIBus ObjectServer password
<b>&lt;path_to_file&gt;</b>	Fully qualified path to specified SQL file

**Note:** If you have already executed this command before, this process might results in some duplicate column error messages. Ignore these messages.

## 7.2.4 Configuring the Tivoli EIF probe

This step configures the probe with the rules for mapping situation events to OMNIbus events. Configuring the mapping involves updating the `tivoli_eif.rules` file installed with the probe. In fact, we have to enable the probe to correctly tokenize the TEC and NetView events information into ObjectServer attributes in order to generate understandable events. You must restart the probe after you update the file.

**Attention:** We recommend creating a backup copy of the current `tivoli_eif.rules` file before going through the next steps.

To update the rules file, update the following file:

`$OMNIHOME/probes/<os_dir>/tivoli_eif.rules`

Where:

**\$OMNIHOME**      System-defined variable defining the install location of OMNIbus

**<os\_dir>**        Operating system specific, for example, `aix5`

Delete the content of your existing `tivoli_eif.rules` file and replace it with the content displayed in “ObjectServer WEIMAR probe `nco_p_tivoli_eif` rules (TEC, NetView)” on page 382.

This rule file has been developed and tested in our lab environment, and it ensures that events coming from both TEC and NetView are correctly managed in OMNIbus.

## 7.2.5 Configuring the NetView TEC adapter to send to the EIF probe

On the NetView server run the following to change the `/usr/OV/conf/tecint.conf` (UNIX) or `\usr\ov\conf\tecad_nv6k.conf` (Windows) adapter configuration file. This step is required to enable the TEC adapter to send events to the Tivoli EIF Probe instead of TEC.

- ▶ On UNIX:

```
/usr/OV/bin/tecits_upgrade -s <servername> -p <port>
```

- ▶ On AIX via:

```
smit nv6000>Configure>Configure Event Forwarding to TEC>
```

- ▶ On Windows:

```
\usr\OV\bin\tec_config.bat
```

Where:

<b>&lt;servername&gt;</b>	Host name of system hosting the EIF probe
<b>&lt;port&gt;</b>	Port the EIF probe listens on (default 9999)

On NetView 7.1.4 and earlier versions it may be necessary to restart the adapter with the following commands:

```
/usr/OV/bin/nvtecia -stop  
/usr/OV/bin/nvtecia -reload
```

Send some test events to confirm the connectivity, for example:

```
/usr/OV/bin/event -e NDWN_EV -h keyworth  
/usr/OV/bin/event -e IDWN_EV -h 9.3.15.210
```

## 7.2.6 Automatic event management customization

As soon as the NetView and OMNIBus integration procedure is complete, the first thing to do is replicate some event management tasks that are performed in NetView through the netview.rls TEC rule. For example, the node\_correlate\_interface TEC rule runs upon receipt of a TEC\_ITS\_NODE\_STATUS event with nodestatus equal to DOWN, MARGINAL or UNREACHABLE. When this event is received, the event cache is searched for any TEC\_ITS\_INTERFACE\_STATUS events for the same host with status equal to DOWN, ADMIN\_DOWN, or UNREACHABLE. If any such effect events are found, they are correlated using the link\_effect\_to\_cause predicate, downgraded to HARMLESS, and closed.

This rule is useful because it helps reduce the noise of *critical interface down* events that are due to the fact that a node in the network is down. Its default behavior can be also defined in Netcool/OMNIBus ObjectServer. To simplify this example, we limited the scope to *interface down* and *node down* events. It is plain to see that there are just little modifications to do to manage the MARGINAL and UNREACHABLE cases.



In OMNIbus terminology, the algorithm performed by the rule is like Figure 7-5.

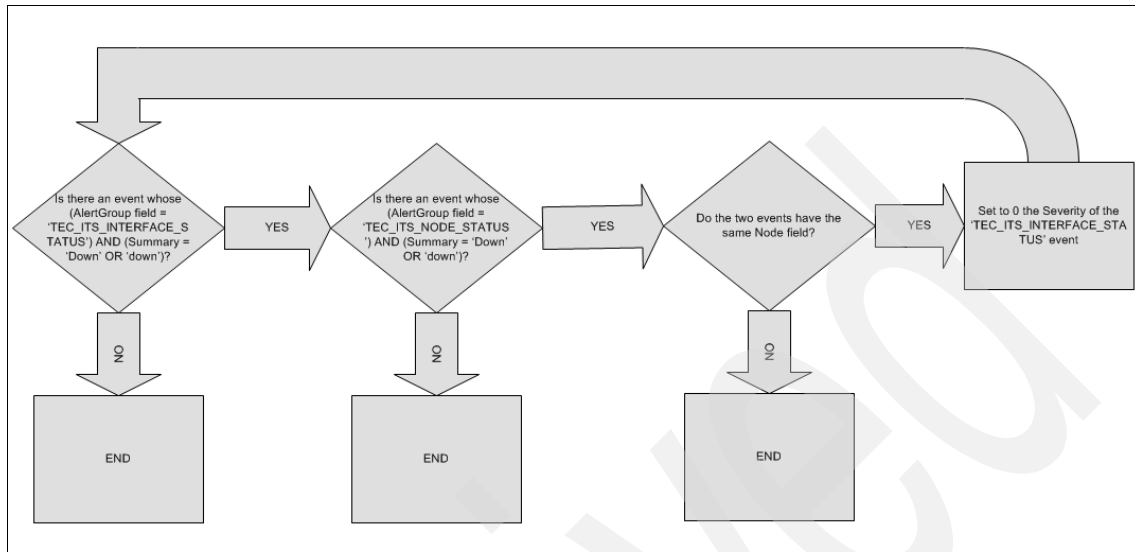


Figure 7-5 Algorithm for `node_correlate_interface` rule

Given the algorithm, it has to be translated in SQL syntax.

In our example, we define `node_corr_new` temporal triggers that fire every 5 seconds, correlating interface down and node down events coming from the same host.

Figure 7-6 shows the Action tab of this trigger.

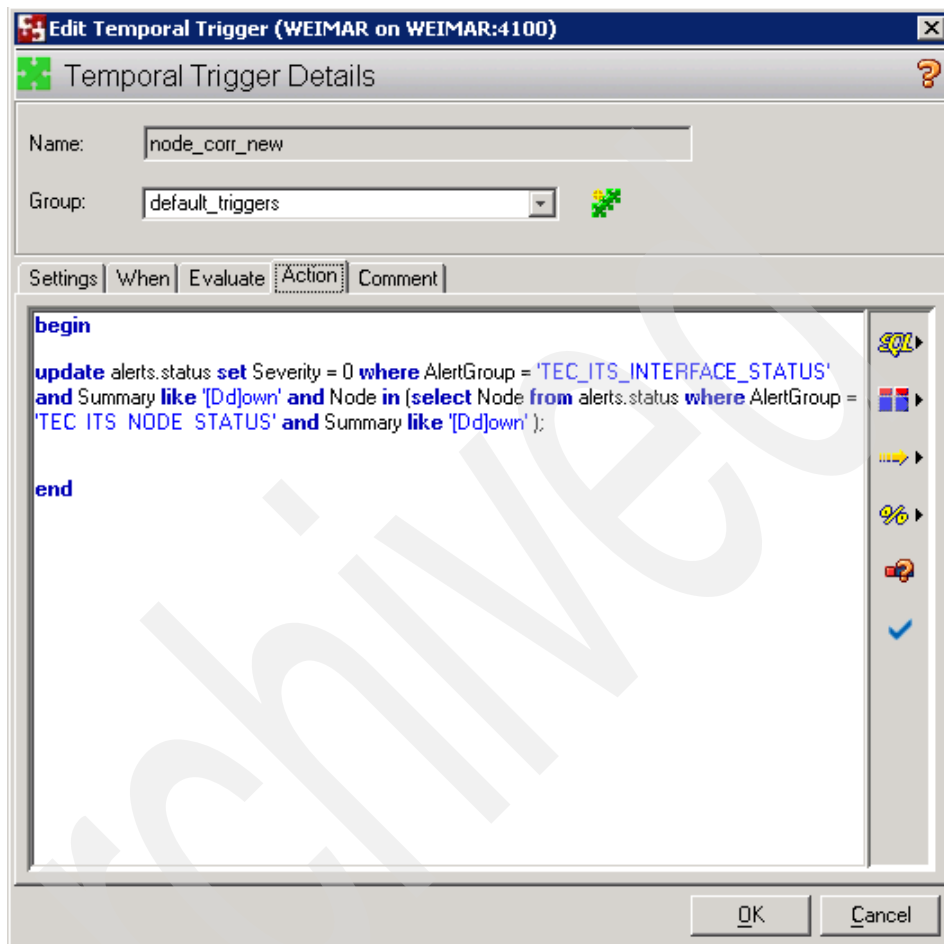


Figure 7-6 *node\_corr\_new* trigger

At this point we have translated the TEC rule into an OMNIbus trigger with just a little effort.

The result of this is that the interface down alarms are successfully cleared if the ObjectServer contains a node down event caused by the same network node. See Figure 7-7.

The screenshot shows the Netcool/OMNIBus Event List window with the filter set to "Node Down". The table displays two events for node 9.3.5.208: a "Node Down" event and an "Interface Down" event, both occurring at 10/8/2007 7:24:20. The status bar at the bottom shows 0 rows selected, the current time is 10/8/2007 8:28:41 PM, and the user is root.

Node	Alert Group	Summary	Last Occurrence	Count	Type
9.3.5.208	TEC ITS_NODE_STATUS	"Node Down"	10/8/2007 7:24:20	1	Problem
9.3.5.208	TEC ITS_INTERFACE_S...	"Interface Down"	10/8/2007 7:23:20 ...	1	Problem

0 rows selected      10/8/2007 8:28:41 PM      root      WEIMAR [PRI]

Figure 7-7 Successful correlation of node down and interface down events

The marginal and unreachable cases can be managed in almost the same way, simply by changing the filtering clause in the trigger Action tab.

Tivoli NetView event correlation behavior can be further refined. In fact, we may want to clear the interface down event also in the case when the malfunctioning node or router comes up again.

Figure 7-8 shows the sequence of actions that we would like to perform.

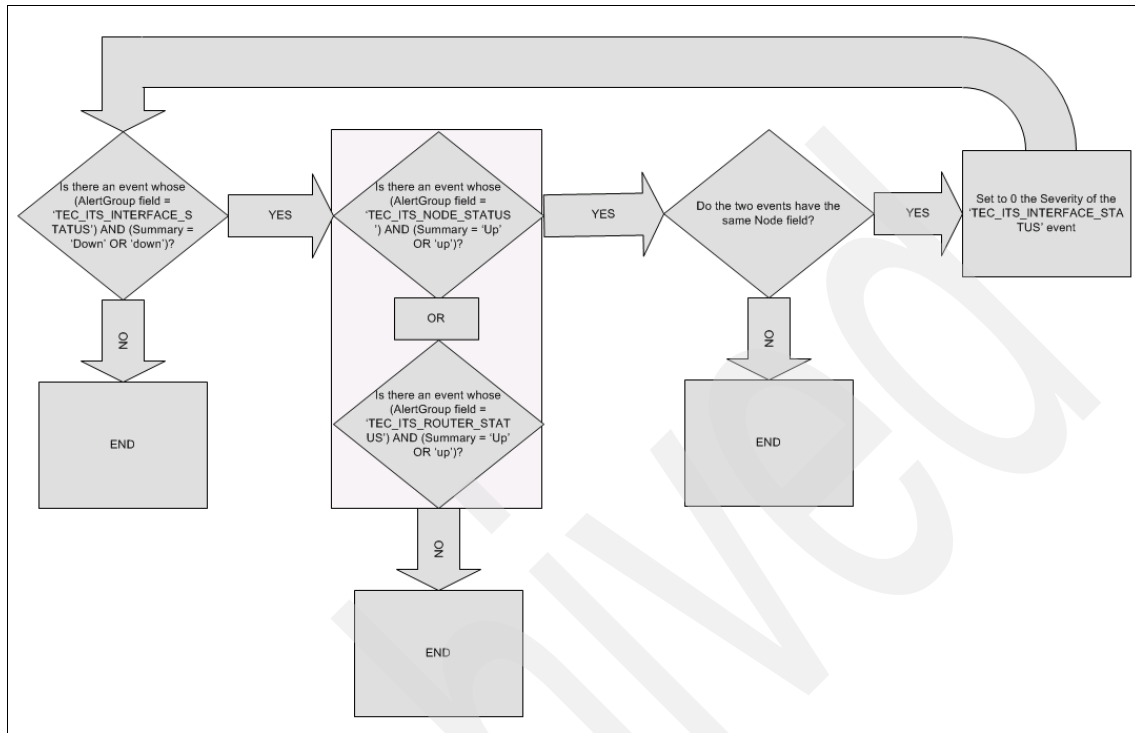


Figure 7-8 Algorithm for advanced interface down - node up and router up correlation

The temporal trigger that corresponds to this algorithm is shown in Figure 7-9.

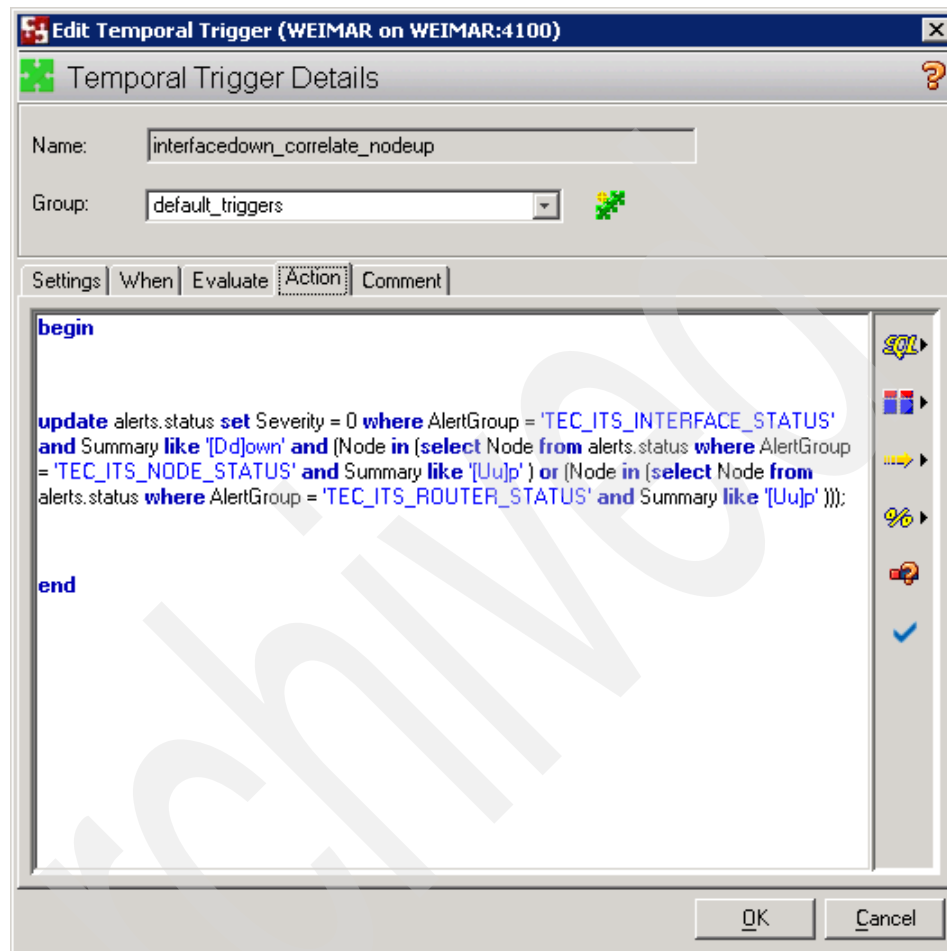


Figure 7-9 *interface\_correlate\_nodeup* trigger

The result of this is that the interface down alarms are successfully cleared if the ObjectServer contains a node up or a router up event originated from the same node in the network. See Figure 7-10.

The screenshot shows the Netcool/OMNIBus Event List window. The title bar reads "Netcool/OMNIBus Event List : Filter='All Events', View='Default'". The menu bar includes File, Edit, View, Alerts, Tools, and Help. Below the menu bar is a toolbar with icons for various functions. The main area displays a table of events:

Node	Alert Group	Summary	Last Occurrence	Count	Type
9.3.5.201	TEC_ITS_NODE_STATUS	"Node Up"	10/9/2007 6:26:02	2	Resolution
9.3.5.201	TEC_ITS_INTERFACE_S...	"Interface Down"	10/9/2007 6:25:52	1	Problem

Below the table is a horizontal bar with colored segments representing event counts: 2 (green), 38 (purple), 18 (blue), 4 (yellow), 3 (orange), and 12 (red). To the right of this bar is a button labeled "All Events". At the bottom of the window, a status bar displays the message "Window is frozen - results will not be displayed until you re...", the date and time "10/9/2007 7:30:46 PM", the user "root", and the location "WEIMAR [PRI]".

Figure 7-10 Successful correlation of node up and interface down events

In a similar way as proposed by our examples, it is possible to translate every NetView rule into OMNIBus automations.

Furthermore, this is a good example that can be applied to other event correlation scenarios, and could also be applied to other occasions where multiple events come from the same host from a number of sources. For example, we could include ITM events also.

## 7.3 Integration between Netcool/OMNIBus and IBM Tivoli Monitoring

In this section we describe step by step how to implement the integration between IBM Tivoli Monitoring 6.2 and Netcool/OMNIBus 7.2 in order to be able to send ITM events directly to OMNIBus without using Tivoli Enterprise Console as an intermediate event collector.

### 7.3.1 Netcool/OMNibus 7.2 and IBM Tivoli Monitoring 6.2 integration

Figure 7-11 shows an integration scenario between Netcool/OMNibus and IBM Tivoli Monitoring.

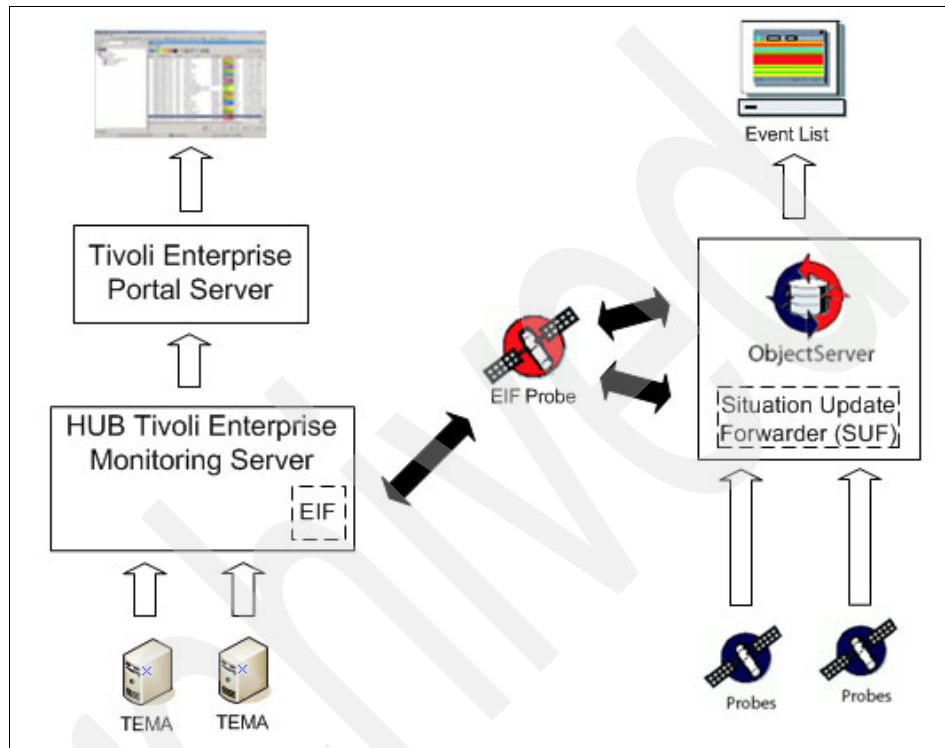


Figure 7-11 Netcool/OMNibus and ITM integration

In this architecture, Netcool/OMNibus is the event management focal point for the entire infrastructure, collecting alerts from sources such as Tivoli Network Manager IP Edition, syslog probe, and Windows NT Event Log Probe. The standard interfaces for event synchronization are delivered with the media for IBM Tivoli Monitoring Version 6.2, with predefined automation rules for handling several update and synchronization scenarios.

The integration of the two products can be achieved through the following steps:

1. Netcool/OMNibus probe for Tivoli EIF (EIF Probe) installation
2. Event synchronization installation
3. Configuring OMNibus server
4. Configuring Tivoli Event Integration Facility (EIF) Interface

### 7.3.2 Installing Netcool/OMNIBus probe for Tivoli EIF

The Netcool/OMNIBus Probe for Tivoli EIF is a prerequisite for the integration between Tivoli Enterprise Console and Netcool/OMNIBus, as it is to the integration between Netcool/OMNIBus and IBM Tivoli Monitoring. This was covered in Chapter 4, “Upgrade strategies” on page 147.

### 7.3.3 Installing event synchronization

In our test environment, the EIF probe and Netcool/OMNIBus are installed on a machine running the Red Hat 4.0 operating system. The event synchronization component will be installed on the same machine.

The installation program for the event synchronization component can be found in the TEC folder of IBM Tivoli Monitoring 6.2 installation media.

The installation steps are:

1. Run ESsync2000Linux.bin to initialize the installation wizard.

**Note:** ESsync2000<xxx>.bin is the executable name, where xxx indicates the appropriate operating system.

**Note:** There are two other ways to install event synchronization: from the command line and the command line using a silent install mode.



The first window is shown in Figure 7-12:

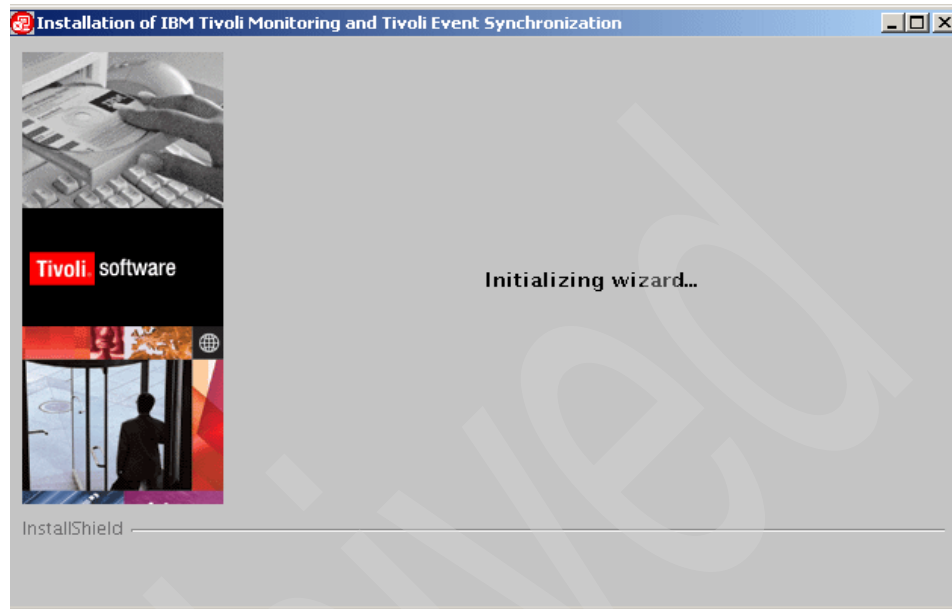


Figure 7-12 Initializing Event Synchronization installation wizard

2. Click **Next** to continue the installation process. See Figure 7-13.

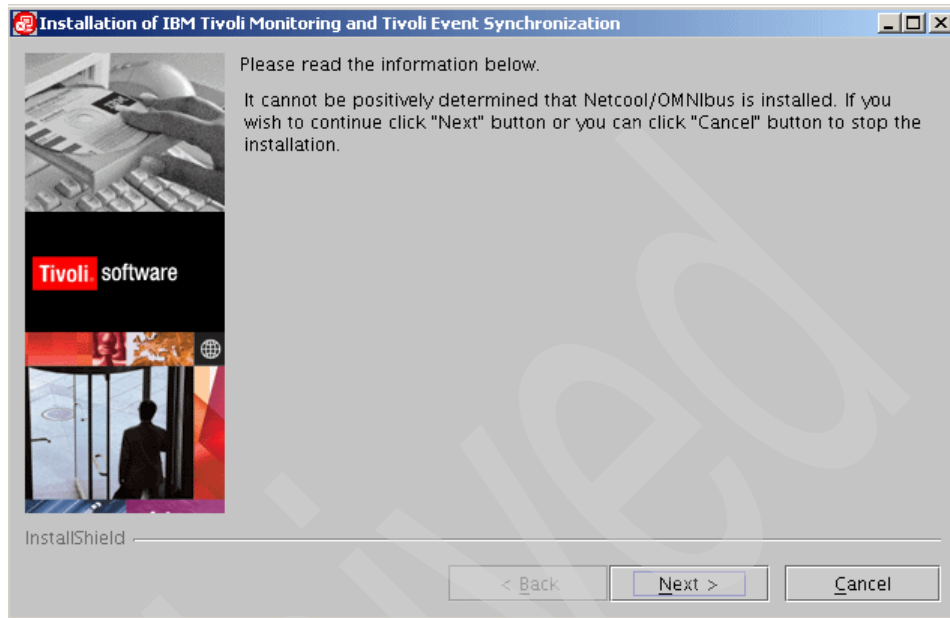


Figure 7-13 Installation of Netcool/OMNibus not determined

3. Click **Next** to continue the installation process. Figure 7-14 shows the Welcome window.

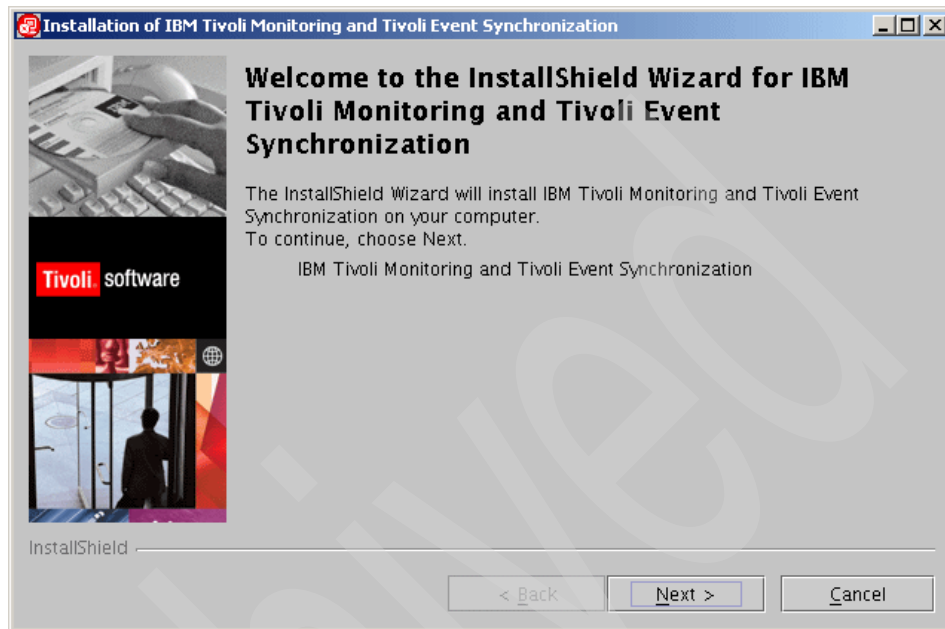


Figure 7-14 Welcome to the InstallShield wizard

4. Accept the software license agreements. Click the option that is shown in Figure 7-15.

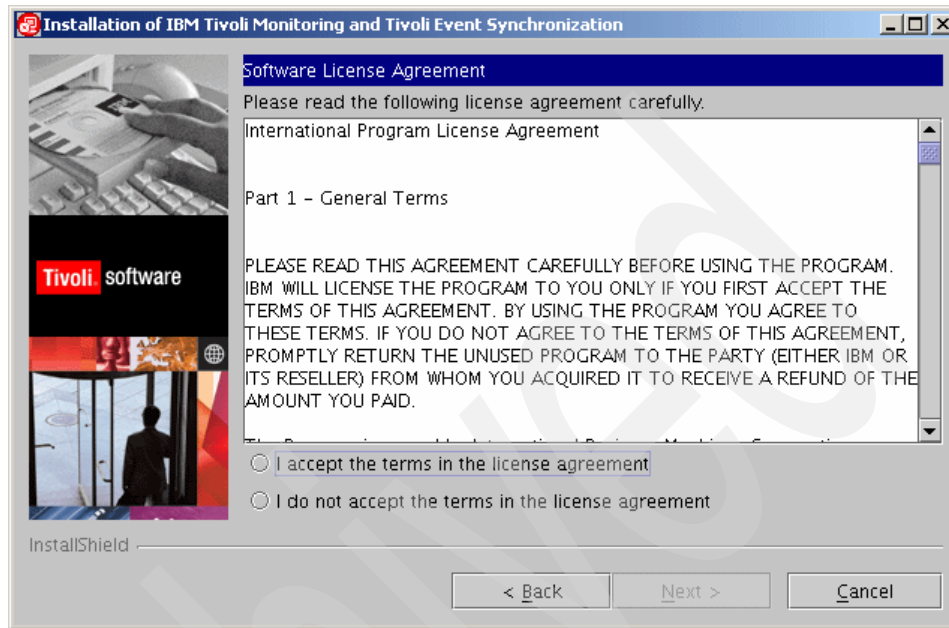


Figure 7-15 Software license agreement acceptance

5. Enter the installation directory path. See the Figure 7-16 with the default value.

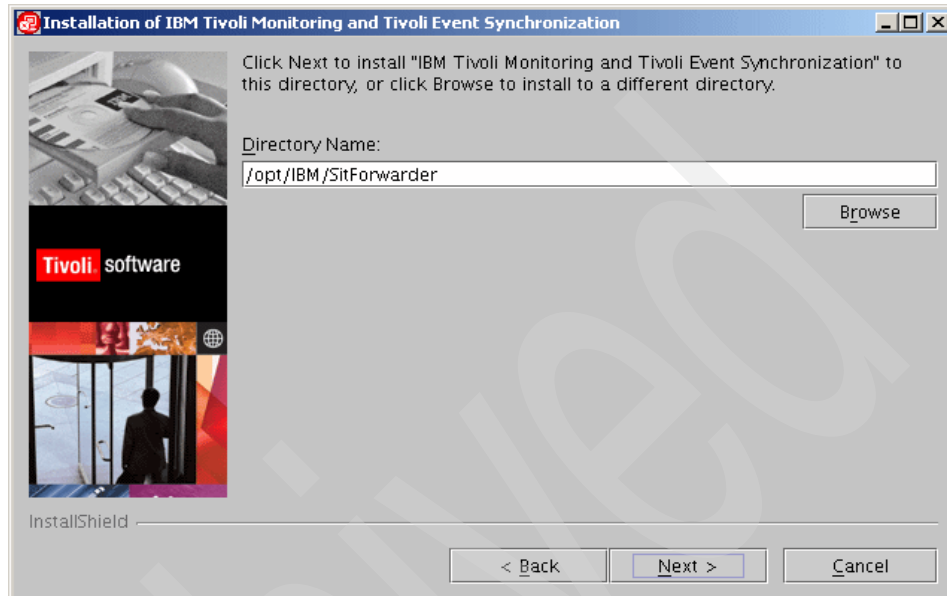


Figure 7-16 Installation directory prompt

6. Customize the required parameters. Configuration can be changed at a later time after installation. See Figure 7-17 for the default values.

Installation of IBM Tivoli Monitoring and Tivoli Event Synchronization

Name of configuration file  
situpdate.conf

Number of seconds to sleep when no new situation updates  
3

Number of bytes to use to save last event  
50

URL of the TEMS SOAP server  
cms/soap

Rate for sending SOAP requests to TEMS from Event Sync via Web Services  
10

Level of debug detail for log  
low

InstallShield

< Back Next > Cancel

Figure 7-17 Configuration parameters - part 1

7. Continue configuring parameters. See Figure 7-18 for the other parameters.

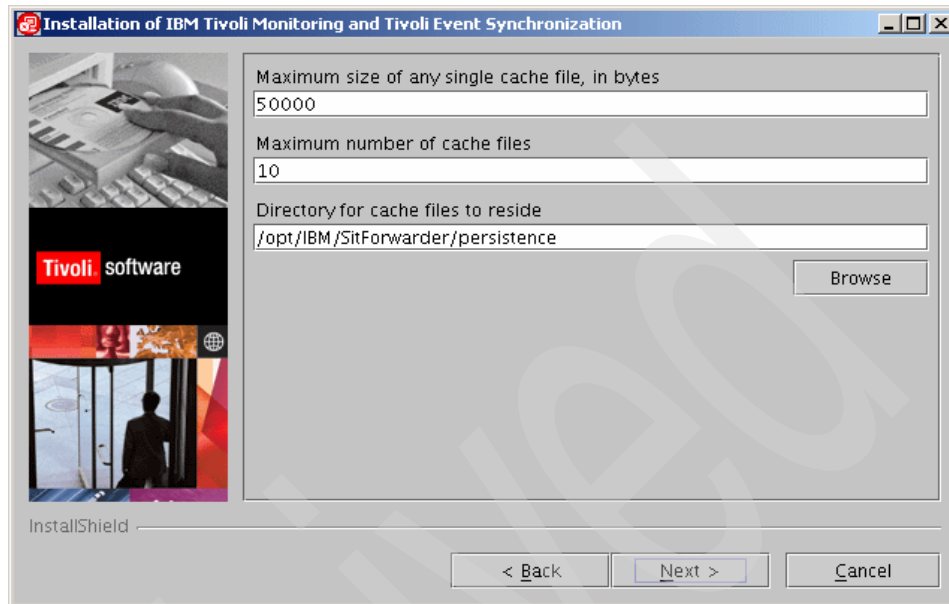


Figure 7-18 Configuration parameters - part 2

8. You are prompted for Tivoli Enterprise Monitoring Server (TEMS) connection parameters. See Figure 7-19.

Installation of IBM Tivoli Monitoring and Tivoli Event Synchronization

Tivoli Enterprise Monitoring server Information

Host name	User ID

Host name User ID Password Confirmation

server2.itsc.austin. sysadmin \*\*\*\*\* \*\*\*\*\*

Add

< Back Next > Cancel

InstallShield

Figure 7-19 Tivoli Enterprise Monitoring Server information input



9. Enter the appropriate host name, user ID, and password. Confirm the information in the configuration screen. Figure 7-20 shows sample contents.

Installation of IBM Tivoli Monitoring and Tivoli Event Synchronization

Tivoli Enterprise Monitoring server Information

Host name	User ID
server2.itsc.austin.ibm.com	sysadmin

Edit  
Remove

Host name User ID Password Confirmation

Add

InstallShield

< Back Next > Cancel

Figure 7-20 Tivoli Enterprise Monitoring Server information confirmation

10. Figure 7-21 shows two options:

- Automatically install rulebases and classes.
- Manually install rules and classes.

Choose **Automatically install rule bases and classes**. This screen is related to the Tivoli Enterprise Console configuration only, so you can skip this section.

**Note:** As this screen comes with a beta installation, we have installed a beta code of IBM Tivoli Monitoring 6.2 and will therefore be removed in the GA version.

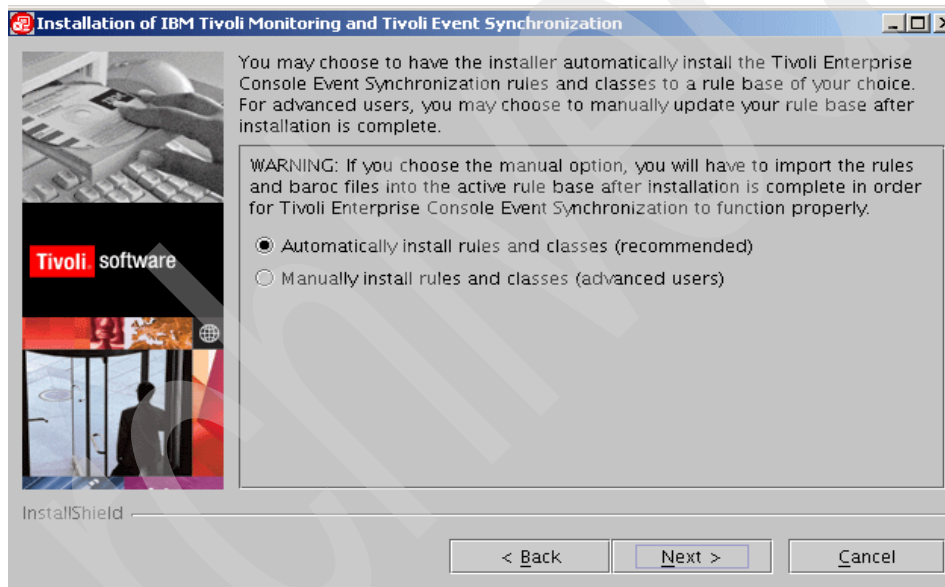


Figure 7-21 Rules and classes for Tivoli Enterprise Console configuration

11. The next window shows the installation summary information. Figure 7-22 shows the directory and disk space that was set up.

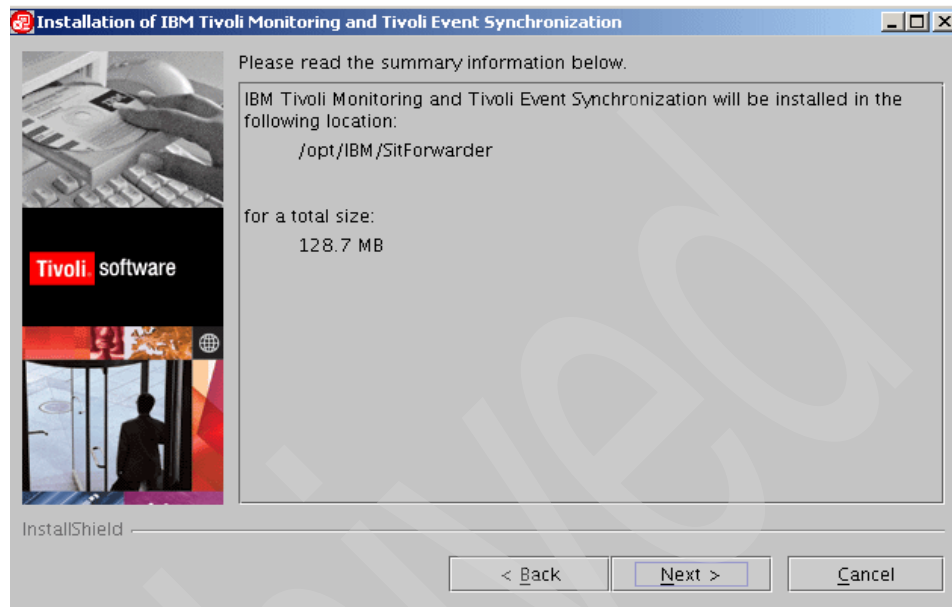


Figure 7-22 Installation directory and disk space required

12. Installation process starts and completion percentage is displayed. See Figure 7-23.



Figure 7-23 Installation completion percentage

13. Wait for the installation process to finish, as shown in Figure 7-24.

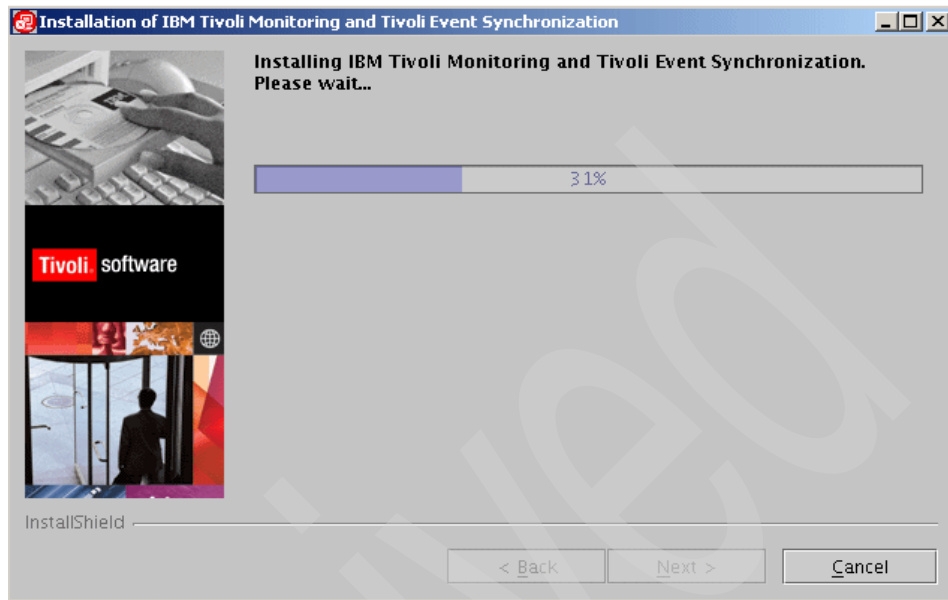


Figure 7-24 Installation progress

14. The installation process ends, as shown in Figure 7-25.

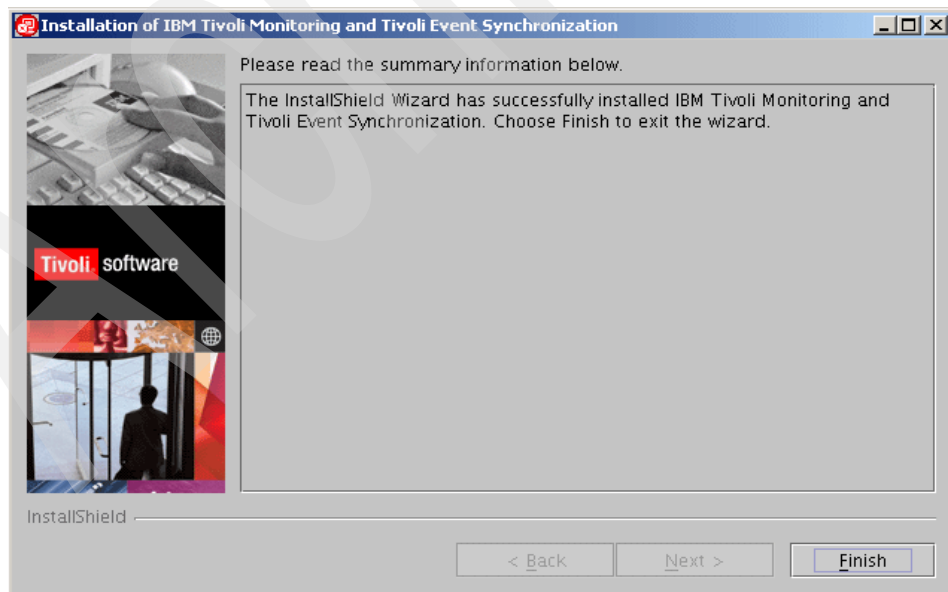


Figure 7-25 Installation process has finished successfully

## 7.3.4 Configuring the OMNIBus server

In this step you configure the OMNIBus ObjectServer to receive and map the situation event information forwarded by a monitoring server and to reflect the events to the OMNIBus console. You also configure the OMNIBus server to send event synchronization information back to the originating monitoring server and configure the EIF probe to map the situation event attributes to OMNIBus event attributes.

In this section we describe the UNIX procedure to customize the integration. For the Windows configuration, see the procedure in the *IBM Tivoli Monitoring Installation Guide*.

### Configuring the OMNIBus server for program execution from scripts

To run the event synchronization program from SQL automation scripts for sending synchronization events to ITM, the OMNIBus server must be running under the process agent and the properties *PA.Username* and *PA.Password* must be set in the `$OMNIHOME/etc/NCOMS.props` file, where `$OMNIHOME` is the system-defined variable defining the installation location of OMNIBus.

By default, the process agent grants access to the members of the default group `ncoadmin`. For default configuration, create a `ncoadmin` group and add `root` as a user to this group. The *PA.Username* property must be set to the username for connecting to the process agent agent. On UNIX, the default value is `root`. The *PA.Password* property must be set to the password for the user connecting to the process agent agent. For the default setting, specify the password of the root user.

### Updating the OMNIBus db schema

The command to configure OMNIBus pipes the SQL command set into the SQL command-line tool and performs the updates to the ObjectServer.

Update the ObjectServer database fields in order to support ITM attributes and ITM event management. Issue the following commands:

```
$OMNIHOME/bin/nco_sql -user <username> -password <password> -server  
<server_name> < <path_to_file>/itm_db_update.sql  
$OMNIHOME/bin/nco_sql -user <username> -password <password> -server  
<server_name> < <path_to_file>/itm_proc.sql  
$OMNIHOME/bin/nco_sql -user <username> -password <password> -server  
<server_name> < <path_to_file>/itm_sync.sql
```

Where:

<b>\$OMNIHOME</b>	Is the system-defined variable defining the install location of OMNIBus
<b>&lt;username&gt;</b>	Is the OMNIBus ObjectServer user name
<b>&lt;password&gt;</b>	Is the OMNIBus ObjectServer password
<b>&lt;path_to_file&gt;</b>	Is the fully qualified path to specified SQL file

Note that the original procedure described in a beta *IBM Tivoli Monitoring Installation Guide* instructs us to run the itm\_proc.sql before itm\_db\_update.sql, but as some tables are required to run the procedure tasks, the correct order is update the database, run procedure tasks, and then do the synchronization task.

## Configuring the Tivoli EIF probe

This step configures the probe with the rules for mapping situation events to OMNIBus events. Configuring the mapping involves updating the tivoli\_eif.rules file installed with the probe. In fact, we have to enable the probe to correctly tokenize the TEC, NetView, and ITM events information into ObjectServer attributes in order to generate understandable events. You must restart the probe after you update the file.

**Attention:** We recommend creating a backup copy of the current tivoli\_eif.rules file before going through the next steps.

To update the rules file:

1. Update the following file:

\$OMNIHOME/probes/<os\_dir>/tivoli\_eif.rules

Where:

<b>\$OMNIHOME</b>	System-defined variable defining the install location of OMNIBus
<b>&lt;os_dir&gt;</b>	Operating system, such as Windows or AIX

Delete the content of your existing tivoli\_eif.rules file and replace it with the content displayed in “ObjectServer WEIMAR probe nco\_p\_tivoli\_eif rules (TEC, NetView, ITM)” on page 388.

This rule file has been developed and tested in our lab environment, and it ensures that all events coming from TEC, NetView, and ITM are correctly managed in OMNIBus.

2. Force the probe to read the new rules file.

On UNIX, issue the following command from command line:

```
kill -HUP <pid>
```

Where <pid> is the probe process ID.

### Configuring error event flow to OMNibus (optional)

To send error events to the OMNibus system when errors are detected in the event synchronization process, update the values for the following parameters in the <eventsync\_install>/omnibus/errorevent.conf file:

ServerName

ServerPort

Where:

**<eventsync\_install>** The location where the event synchronization program is installed (On Windows the default install directory is C:\Program Files\IBM\SitForwarder, and on Linux and UNIX operating systems the default is /opt/IBM/SitForwarder.)

**ServerName** The name of the machine where the EIF probe is running.

**ServerPort** The listening port for the EIF probe. The default value is 9999.

### Customizing the OMNibus configuration

The procedure `get_config_parms` in the <event\_sync\_install\_dir>/omnibus/itm\_proc.sql file defines three configuration parameters:

```
set sit_ack_expired_def_action = 'REJECT'  
set sit_resurface_def_action = 'ACCEPT'  
set situpdate_conf_file = 'situpdate.conf'
```

The variable `sit_ack_expired_def_action` defines the action to be taken for an event by the OMNibus server when acknowledgement expiration information is received for an event from a monitoring server. The default action is to reject the request. OMNibus sends information to change the state of the event to acknowledge back to the monitoring server. If you would like to change the action taken by the OMNibus server to accept the acknowledgement expiration, modify the statement to set `sit_ack_expired_def_action = 'ACCEPT'`.

The variable `sit_resurface_def_action` defines the action to be taken by the OMNibus server when a situation event has resurfaced. The default action of the OMNibus server is to accept this request and deacknowledge the event. If you would like to change the action taken by the OMNibus server to reject the



resurface of the event, modify the statement to set `sit_resurface_def_action = 'REJECT'`. OMNIBus then sends information back to the monitoring server to change the state of the event back to acknowledge.

The variable `situpdate_conf_file` specifies the name of the configuration file to be used by the SitUpdate forwarder. If you would like to change the name of the configuration file, modify the statement to set `situpdate_conf_file = 'newname.conf'`.

After modifying `itm_proc.sql`, issue the following command (for UNIX):

```
$OMNIHOME/bin/nco_sql -user <username> -password <password> -server  
<server_name> < <path_to_file>/itm_proc.sql
```

## Situation Update Forwarder

The Situation Update Forwarder is the Web service based Java process that is used to communicate with the TEMS from Netcool/OMNIBus.

The configuration files for the Situation Update Forwarder are located in `/opt/IBM/SitForwarder/` by default on a Linux configuration and has the content shown in Example 7-1.

*Example 7-1 /opt/IBM/SitForwarder/etc directory*

---

```
# pwd  
/opt/IBM/SitForwarder/etc  
# ls -la  
total 32  
drwxrwxrwx  2 root    root      4096 Sep 28 18:47 .  
drwxrwxrwx 12 netcool ncoadmin 4096 Sep 28 18:02 ..  
-rw-r--r--  1 root    root       652 Sep 28 18:14  
server2.itsc.austin.ibm.com.pwd  
-rwxrwxrwx  1 root    root        27 Sep  5 23:29 sit_timeouts.conf  
-rwxrwxrwx  1 root    root       166 Sep 28 18:19 situpdate.conf  
-rwxrwxrwx  1 root    root       25 Sep 28 15:54 situpdate.properties  
-rwxrwxrwx  1 root    root       121 Sep 28 18:33 situser.conf  
-rw-r--r--  1 root    root       125 Sep 28 18:14 situser.conf.1
```

---

The `situpdate.conf` file contains information about the configuration of the event synchronization, as shown in Example 7-2.

*Example 7-2 situpdate.conf file*

---

```
fileSize=50000  
fileNumber=10  
fileLocation=/opt/IBM/SitForwarder/persistence
```

```
pollingInterval=3
crcBytecount=50
cmsSoapUrl=cms/soap
bufferFlushRate=10
logLevel=verbose
```

---

The situser.conf file contains information about the user ID and password for interaction with SOAP Web services, as shown in Example 7-3.

*Example 7-3 situser.conf file*

---

```
serverid=server2.itsc.austin.ibm.com
userid=root
passwordfile=/opt/IBM/SitForwarder/etc/server2.itsc.austin.ibm.com.pwd
```

---

These files must only be modified with the scripts provided with the Situation Update Forwarder, and any changes require the Situation Update Forwarder to be restarted.

Example 7-4 shows the content of the /opt/IBM/SitForwarder/bin directory.

*Example 7-4 /opt/IBM/SitForwarder/bin*

---

```
#ls -la
drwxrwxrwx  2 root    root      4096 Sep 28 18:49 .
drwxrwxrwx 12 netcool ncoadmin 4096 Sep 28 18:02 ..
-rwxrwxrwx  1 root    root       732 Sep  5 23:29 launch.sh
-rw-----  1 netcool ncoadmin    0 Sep 28 18:02 nohup.out
-rwxrwxrwx  1 root    root       699 Sep 28 15:51 query_state.sh
-rw-r--r--  1 netcool ncoadmin 2218 Sep 28 18:14 sitconfigsvruser.log
-rw-r--r--  1 netcool ncoadmin    0 Sep 28 18:14
sitconfigsvruser.log.lck
-rwxrwxrwx  1 root    root       531 Sep  5 23:29 sitconf.sh
-rwxrwxrwx  1 root    root       538 Sep  5 23:29 sitconfuser.sh
-rwxrwxrwx  1 root    root     21119 Sep 28 18:49 situpdate.sh
-rwxrwxrwx  1 root    root       713 Sep 28 15:51 startSUF.sh
-rwxrwxrwx  1 root    root       649 Sep 28 15:51 stopSUF.sh
-rwxrwxrwx  1 root    root       697 Sep 28 15:51 test.000
-rwxrwxrwx  1 root    root       699 Sep 28 15:51 test.sh
```

---

### 7.3.5 Configuring the monitoring server

Before the monitoring server forwards any situation events to Netcool/OMNIBus, you must enable event forwarding and define a default event destination.

You can configure the EIF probe and port information for the monitoring server during installation. If you decide to configure after installation, use the following steps to configure the hub monitoring server.

**Note:** If you will be configuring user security, you should use the root login ID to configure.

1. At the command line, change to the /opt/IBM/ITM/bin directory (or the directory where you installed IBM Tivoli Monitoring).

2. Run the following command:

```
./itmcmd config -S -t tems_name
```

Where *tems\_name* is the name of your monitoring server (for example, HUB\_itmdev17).

3. Follow the configuration steps until you reach this question:

```
Tivoli Event Integration Facility? [1=YES, 2=NO]
```

Complete the following additional steps:

- ▶ For the EIF server, type the host name of the TEC event server or the host name of the Netcool/OMNIBus EIF probe and press Enter.
- ▶ For the EIF port, type the EIF reception port number for the TEC event server or the Netcool/OMNIBus EIF probe and press Enter.

Example 7-5 shows a sample EIF configuration.

#### *Example 7-5 EIF configuration*

---

```
[root@server2] [/opt/IBM/ITM/bin]-> ./itmcmd config -S -t HUB_TEMS  
Configuring TEMS...
```

```
Hub or Remote [1=*LOCAL, 2=*REMOTE] (Default is: 1):
```

```
TEMS Host Name (Default is: server2):
```

```
Network Protocol 1 [ip, sna, ip.pipe or ip.spipe] (Default is: ip.pipe):
```

```
Now choose the next protocol number from one of these:
```

- ip
- sna
- ip.spipe

```

- 0 for none
Network Protocol 2 (Default is: ip):

Now choose the next protocol number from one of these:
- sna
- ip.spice
- 0 for none
Network Protocol 3(Default is: ip.spice):
IP Port Number (Default is: 1918):
IP.PIPE Port Number (Default is: 1918):
Enter name of KDC_PARTITION (Default is: null):
Enter path and name of KDC_PARTITIONFILE (Default is:
/opt/IBM/ITM/tables/HUB_TEMS/partition.txt):
IP.SPIPE Port Number (Default is: 3660):

Configuration Auditing? [1=YES, 2=NO] (Default is: 1):
Hot Standby TEMS Host Name or type 0 for "none" (Default is: 0):
Enter Optional Primary Network Name or type 0 for "none" :(Default is: 0):
Security: Validate User ? [1=YES, 2=NO] (Default is: 1):
LDAP Security: Validate User with LDAP ? [1=YES, 2=NO] (Default is: 2):

Tivoli Event Integration Facility? [1=YES, 2=NO] (Default is: 2): 1
EIF Server?(Default is: none): weimar
EIF Port? (Default is: 5529): 9999
Disable Workflow Policy/Tivoli Emitter Agent Event Forwarding? [1=YES, 2=NO]
(Default is: 2):
... Writing to database file for ms.

Hubs
##      CMS_Name
1      ip.pipe:HUB_TEMS[1918]

1)Add, 2)Remove ##, 3)Modify Hub ##, 4)UserAccess ##, 5)Cancel, 6)Save/exit: 6
... creating config file "/opt/IBM/ITM/config/server2_ms_HUB_TEMS.config"
... creating file "/opt/IBM/ITM/tables/HUB_TEMS/glb_site.txt."
... updating "/opt/IBM/ITM/config/kbbenv"
... verifying Hot Standby.
Info - Checking TEMS User Authentication requirements.
Info - OK, TEMS User Authentication requirements complete.
TEMS configuration completed...

```

---

## Integrating the Netcool/OMNIBus events to Common Event Console

In ITM 6.2, the Common Event Console can concentrate events from ITM itself, TEC, and Netcool/OMNIBus in the same workspace. To achieve this target, some customization has been made.

TEMS has a default ITM connector configuration, as you can see in Figure 7-26.

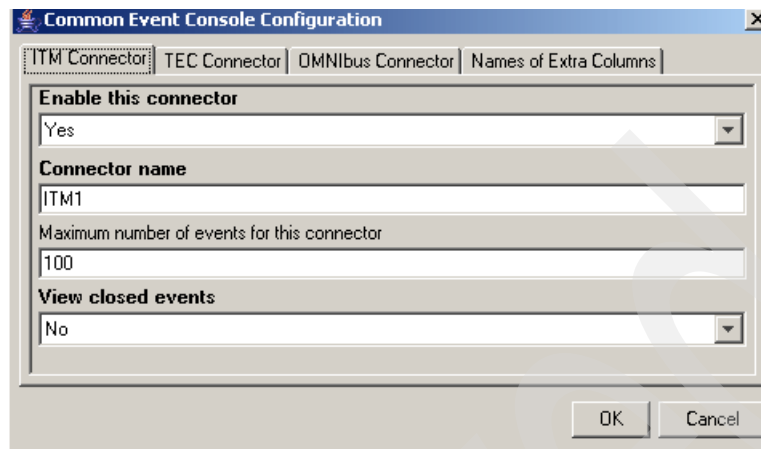


Figure 7-26 ITM connector configuration

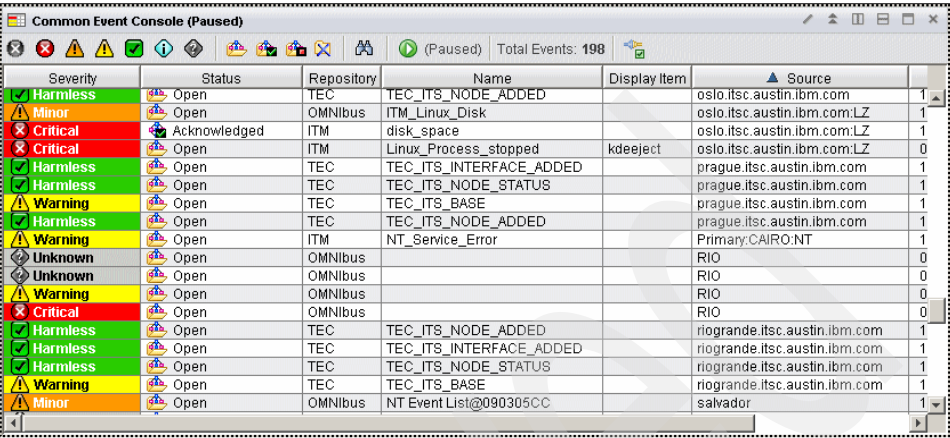
To configure the OMNibus connector, select the respective folder, click **New**, and fill in the parameters as shown in Figure 7-27.

The image shows a Windows-style dialog box titled "Common Event Console Configuration". It has four tabs: "ITM Connector", "TEC Connector", "OMNibus Connector" (which is selected), and "Names of Extra Columns". In the top right corner of the dialog is a "New..." button. Below the tabs, the "OMNibus Connector" section is active. It contains a "Delete" button in the top right. The configuration fields are as follows: "Connector name" with the value "weimar"; "Maximum number of events for this connector" with the value "100"; "Computer name of event system" with the value "weimar"; "Port number of event system" with the value "4100"; "User name for accessing event system" with the value "root"; "Password" and "Confirm Password" fields, both empty; "SQL WHERE clause that restricts events for common event console" with an empty text box; "View cleared events" with a dropdown menu set to "Yes"; "Time interval (in minutes) for checking for new or changed events" with the value "1"; "Time interval (in seconds) between reconnection attempts" with the value "20"; "Maximum number of consecutive reconnection attempts" with the value "10"; and "Field type for extra column 1" with a dropdown menu. At the bottom right are "OK" and "Cancel" buttons.

Figure 7-27 OMNibus connector configuration

The other parameters that are not shown above were left with the default values.

Figure 7-28 shows a sample Common Event Console with three connectors: ITM, TEC, and Netcool/OMNibus.



Common Event Console (Paused)						
(Paused) Total Events: 198						
Severity	Status	Repository	Name	Display Item	Source	
Harmless	Open	TEC	TEC_ITS_NODE_ADDED		oslo.itsc.austin.ibm.com	1
Minor	Open	OMNibus	ITM_Linux_Disk		oslo.itsc.austin.ibm.com:LZ	1
Critical	Acknowledged	ITM	disk_space		oslo.itsc.austin.ibm.com:LZ	1
Critical	Open	ITM	Linux_Process_stopped	kdeeject	oslo.itsc.austin.ibm.com:LZ	0
Harmless	Open	TEC	TEC_ITS_INTERFACE_ADDED		prague.itsc.austin.ibm.com	1
Harmless	Open	TEC	TEC_ITS_NODE_STATUS		prague.itsc.austin.ibm.com	1
Warning	Open	TEC	TEC_ITS_BASE		prague.itsc.austin.ibm.com	1
Harmless	Open	TEC	TEC_ITS_NODE_ADDED		prague.itsc.austin.ibm.com	1
Warning	Open	ITM	NT_Service_Error		Primary:CAIRO:NT	1
Unknown	Open	OMNibus			RIO	0
Unknown	Open	OMNibus			RIO	0
Warning	Open	OMNibus			RIO	0
Critical	Open	OMNibus			RIO	0
Harmless	Open	TEC	TEC_ITS_NODE_ADDED		riogrande.itsc.austin.ibm.com	1
Harmless	Open	TEC	TEC_ITS_INTERFACE_ADDED		riogrande.itsc.austin.ibm.com	1
Harmless	Open	TEC	TEC_ITS_NODE_STATUS		riogrande.itsc.austin.ibm.com	1
Warning	Open	TEC	TEC_ITS_BASE		riogrande.itsc.austin.ibm.com	1
Minor	Open	OMNibus	NT Event List@090305CC		salvador	1

Figure 7-28 Common Event Console

One situation was create as an example to fire a disk space with the space available percent less than 90. See Figure 7-29 for the basic configuration.

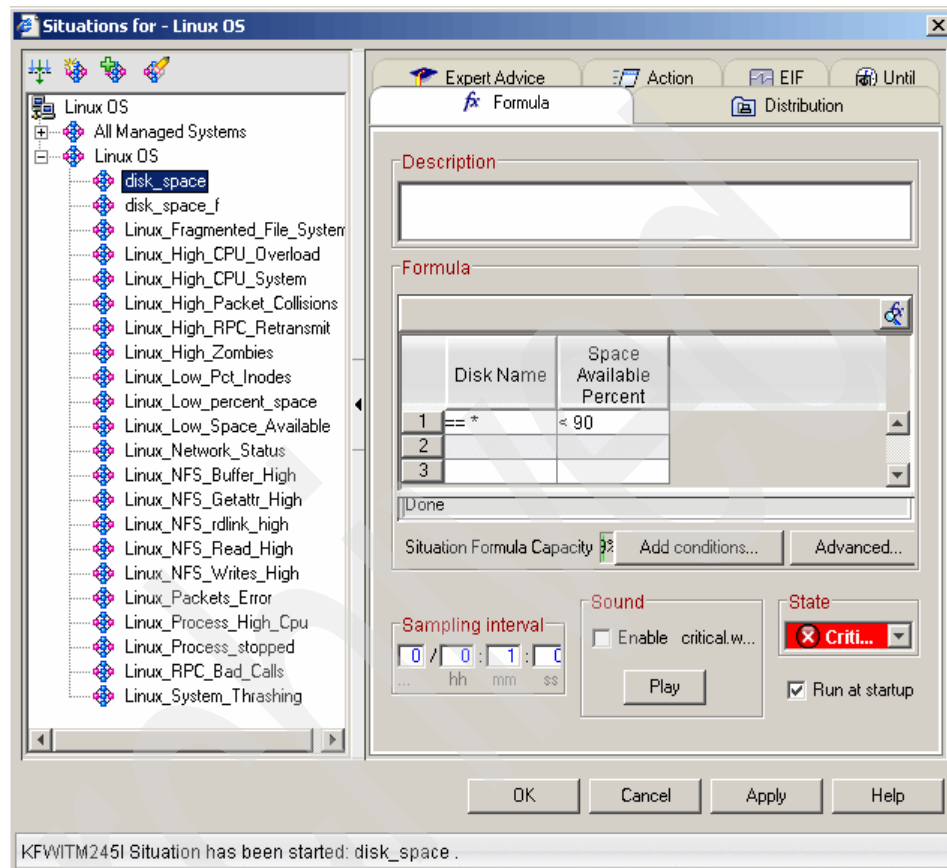


Figure 7-29 Situation configuration



In the situation configuration, click the **EIF** folder to select the EIF receiver for the situation alert. Figure 7-30 shows a Windows example.

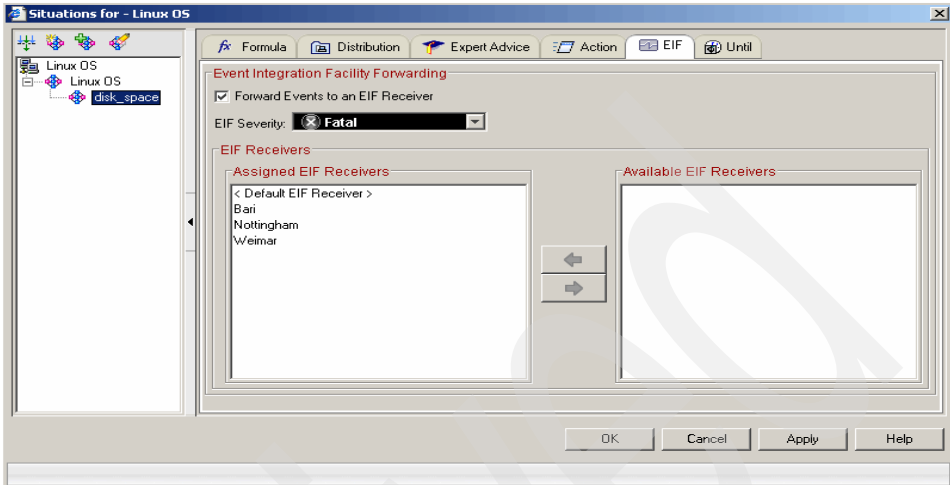


Figure 7-30 EIF receivers

This example situation was fired to generate an event to Netcool/OMNIBus—an alert where Alert Group with the ITM\_Linux\_Disk value was received on Netcool/OMNIBus. Figure 7-31 shows the native Netcool/OMNIBus console.

Node (+)	Alert Group	Summary (+)	Last Occurrence	Count
mondorf	SNMP	node down	03/2007 02:10:01 P	1
oslo.itsc.austin.ibm	ITM_Linux_Disk	disk_space(Disk_NameLIKE"" AND Space_Available_Percent<90 ) ON oslo.itsc	03/2007 02:34:08 P	12
salvador	Administrator	A Administrator process running on salvador has connected as username root	03/2007 02:15:34 P	1
salvador	Windows Event List	A NT Event List@090305CC process running on salvador has connected as user	03/2007 12:00:20 P	1
salvador	Windows Event List	A NT Event List@090305CC process running on salvador has connected as user	03/2007 12:00:21 P	1
salvador	Windows Event List	A NT Event List@090305CC process running on salvador has connected as user	03/2007 11:43:42 A	1
salvador	Windows Event List	A NT Event List@090305CC process running on salvador has connected as user	03/2007 11:43:44 A	1
salvador	Probe	A PROBE process running on salvador has disconnected as username probe	03/2007 11:43:30 A	1
salvador	NT Event List@090305C	Attempt to login as netcool from host salvador failed	03/2007 10:22:45 A	1
salvador		mhntlog probe on salvador:Host - salvador : Log - Application -> Attempting to c	03/2007 11:23:30 A	1
salvador		mhntlog probe on salvador:Host - salvador : Log - Application -> Successfully co	03/2007 11:23:30 A	1
salvador		mhntlog probe on salvador:Host - salvador : Log - Security -> Attempting to con	03/2007 11:23:30 A	2
salvador		mhntlog probe on salvador:Host - salvador : Log - Security -> Successfully conn	03/2007 11:23:30 A	2
salvador		mhntlog probe on salvador:Host - salvador : Log - System -> Attempting to conn	03/2007 11:23:30 A	2
salvador		mhntlog probe on salvador:Host - salvador : Log - System -> Successfully conn	03/2007 11:23:30 A	2
waco.itsc.austin.ibm	ITM_Linux_Disk	disk_space(Disk_NameLIKE"" AND Space_Available_Percent<90 ) ON waco.itsc	03/2007 02:34:06 P	12
weimar	Administrator	A Administrator process running on weimar has connected as username root	03/2007 02:07:09 P	1
weimar	WEBCON	A WEBCON process running on weimar has connected as username root	03/2007 11:35:44 A	1

Figure 7-31 Netcool/OMNIBus event list

The user can acknowledge the event on the TEP workspace, as shown in Figure 7-32.

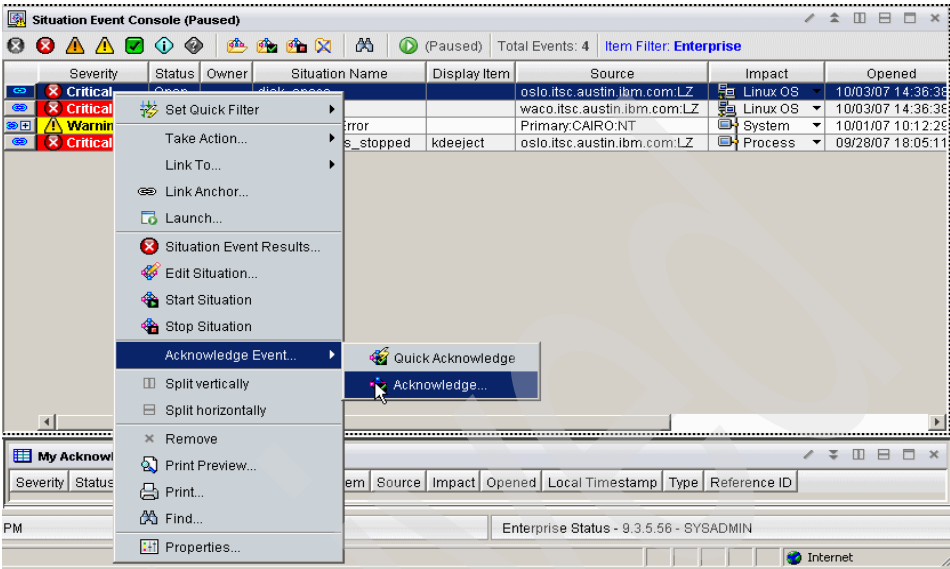


Figure 7-32 Acknowledging event

Then the event is acknowledged in the TEP workspace, as shown in Figure 7-33.

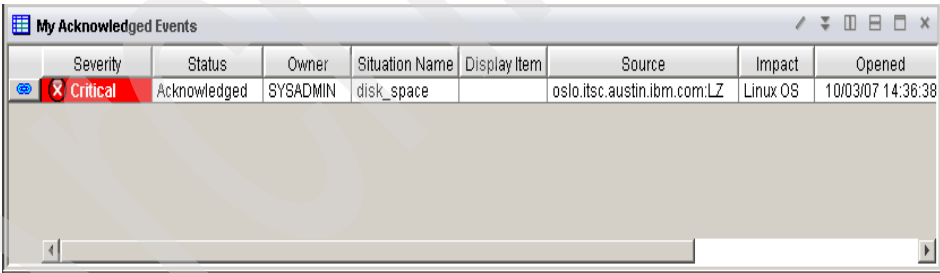


Figure 7-33 Event acknowledged

In the Netcool/OMNibus event list, we can see that event has been acknowledged. See Figure 7-34.

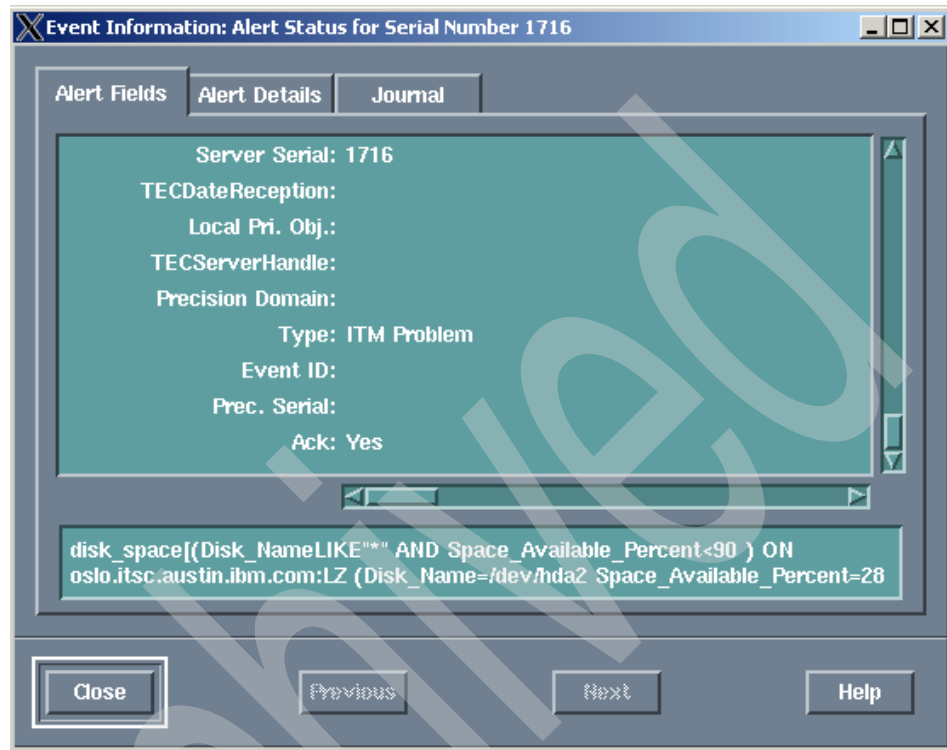
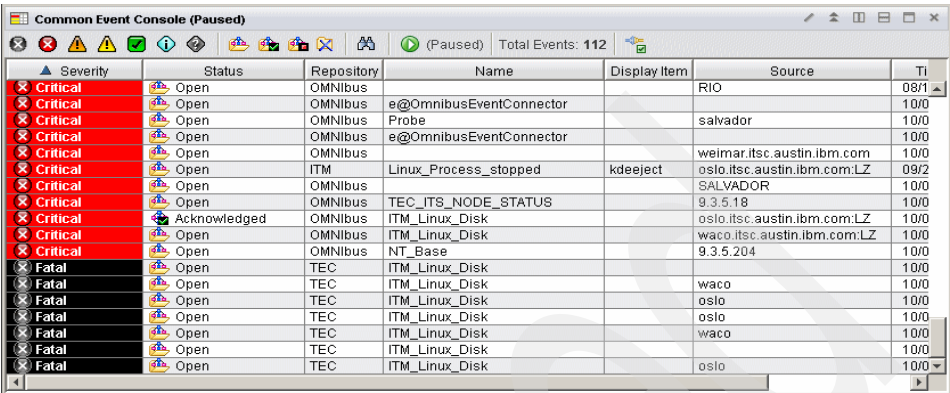


Figure 7-34 Alert fields updated

Figure 7-35 shows the acknowledged status from the Netcool/OMNIBus connector interface.



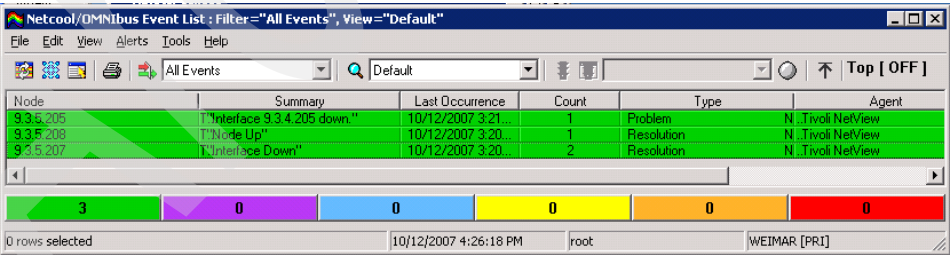
Severity	Status	Repository	Name	Display Item	Source	TI
Critical	Open	OMNIBus			RIO	08/1
Critical	Open	OMNIBus	e@OmnibusEventConnector			10/0
Critical	Open	OMNIBus	Probe		salvador	10/0
Critical	Open	OMNIBus	e@OmnibusEventConnector			10/0
Critical	Open	OMNIBus			weimar.itsc.austin.ibm.com	10/0
Critical	Open	ITM	Linux_Process_stopped	kdeeject	oslo.itsc.austin.ibm.com:LZ	09/2
Critical	Open	OMNIBus			SALVADOR	10/0
Critical	Open	OMNIBus	TEC_ITS_NODE_STATUS		9.3.5.18	10/0
Critical	Acknowledged	OMNIBus	ITM_Linux_Disk		oslo.itsc.austin.ibm.com:LZ	10/0
Critical	Open	OMNIBus	ITM_Linux_Disk		waco.itsc.austin.ibm.com:LZ	10/0
Critical	Open	OMNIBus	NT_Base		9.3.5.204	10/0
Fatal	Open	TEC	ITM_Linux_Disk			10/0
Fatal	Open	TEC	ITM_Linux_Disk		waco	10/0
Fatal	Open	TEC	ITM_Linux_Disk		oslo	10/0
Fatal	Open	TEC	ITM_Linux_Disk		oslo	10/0
Fatal	Open	TEC	ITM_Linux_Disk		waco	10/0
Fatal	Open	TEC	ITM_Linux_Disk		oslo	10/0
Fatal	Open	TEC	ITM_Linux_Disk		oslo	10/0

Figure 7-35 Common Event Console

## 7.4 Deduplication configuration

During our testing we observed that after the installation of IBM Tivoli Monitoring 6.2 integration with Netcool/OMNIBus, the deduplication trigger automation works in an unexpected way. We have requested that this behavior be changed. However, we illustrate it below, as it serves as a good example. If, during your automation development, you also observe this behavior, then you can make the same changes.

Figure 7-36 shows the behavior of deduplication.



Node	Summary	Last Occurrence	Count	Type	Agent
9.3.5.205	Interface 9.3.4.205 down"	10/12/2007 3:21	1	Problem	N_Tivoli NetView
9.3.5.208	"Node Up"	10/12/2007 3:20	1	Resolution	N_Tivoli NetView
9.3.5.207	"Interface Down"	10/12/2007 3:20	2	Resolution	N_Tivoli NetView

3 0 0 0 0 0

0 rows selected 10/12/2007 4:26:18 PM root WEIMAR [PRI]

Figure 7-36 Deduplication automation unexpected behavior

As you can see, a node down event severity is correctly cleared whenever a *node up* event comes in the ObjectServer, but its summary is not. Therefore, we see a cleared event along with a node down summary.

This unexpected behavior can be easily fixed in the following way:

1. Access the Trigger menu in the Administrator panel.
2. Edit the deduplication trigger, adding the following lines in the section related to ITM events:

```
        if ((old.Severity = 0) and (new.Severity > 0)) then set  
old.Severity = new.Severity;  
end if;
```

The deduplication trigger should now look like Figure 7-37.

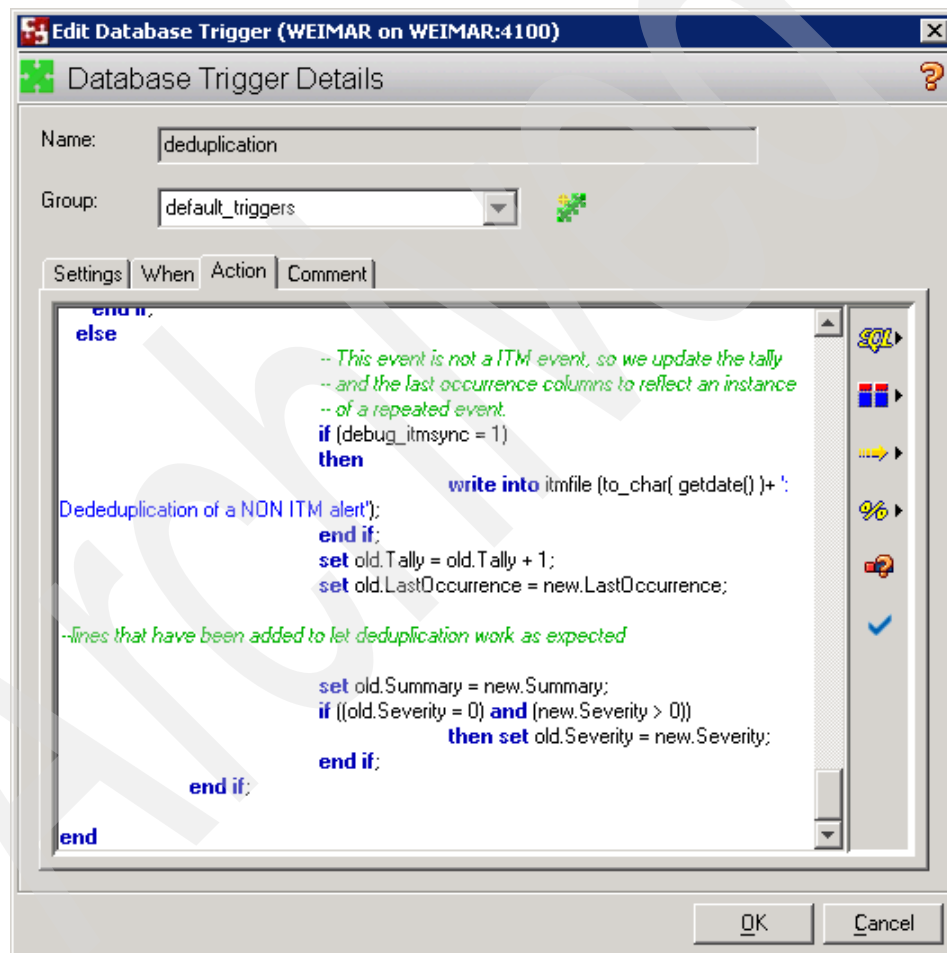


Figure 7-37 Edit Database Trigger

3. Click **OK** to save the updates to the trigger.

The deduplication trigger will work correctly after this easy fix. The summary is correctly updated and the count is increased by 1.

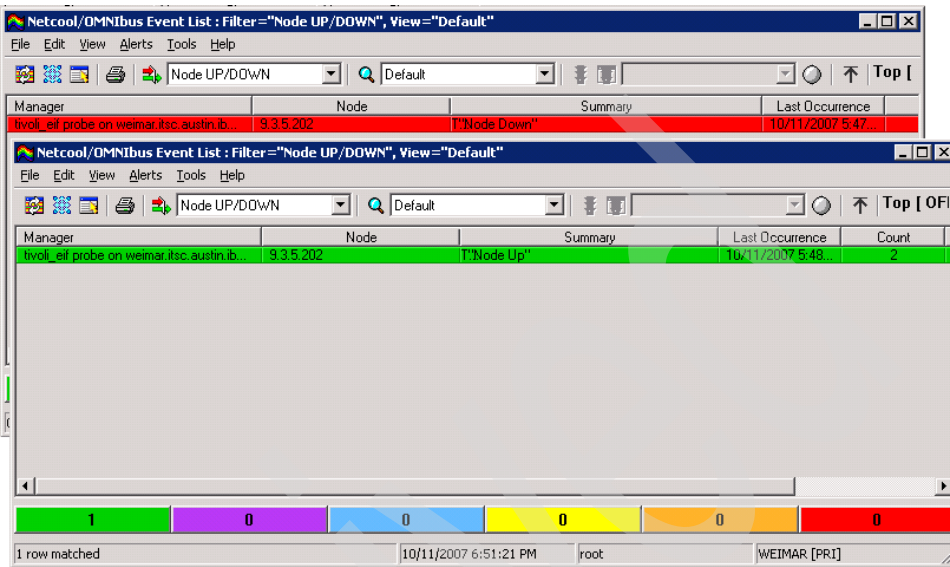


Figure 7-38 Deduplication now works correctly

**Note:** As with all database trigger edits, the change is dynamic, and the impact can be monitored immediately.

## 7.5 Migrating the TEC Windows event log adapter

One of the most common customer scenarios involves collecting events from log files related to applications, operating system, security, and so on. Most of the time, when particular events are received in TEC, some actions are initiated on the target machine to execute scripts that perform housekeeping tasks.

We replicated a sample of this scenario, installing a TEC Windows Event Log Adapter on the Windows machine SALVADOR. The TEC adapter is successfully up and running and sending events to Tivoli Enterprise Console.

In this section we explain how to achieve the same results using Netcool/OMNIBus and the Multi-Headed NT Windows Event Log file probe (MH NT), which has been identified as the best solution to collect events from a Windows operating system.

At the end of the following tasks, we expect the Windows Event Log File installed on SALVADOR to send events to the OMNibus ObjectServer WEIMAR on the WEIMAR machine. We also expect to be able to execute scripts and external procedures on the Windows box.

After verification that events are correctly collected and managed with OMNibus, it will be possible to disable the TEC Windows Event Log Adapter from sending events to TEC.

### 7.5.1 Installing and configuring the Windows NT Event Log probe

To be able to install the probe, a Netcool/OMNibus installation is necessary on the target machine. Execute the following steps to install OMNibus and then the MH NT Windows Event Log probe:

1. Unzip the OMNibus installer software archive in a directory of your choice. See Figure 7-39.

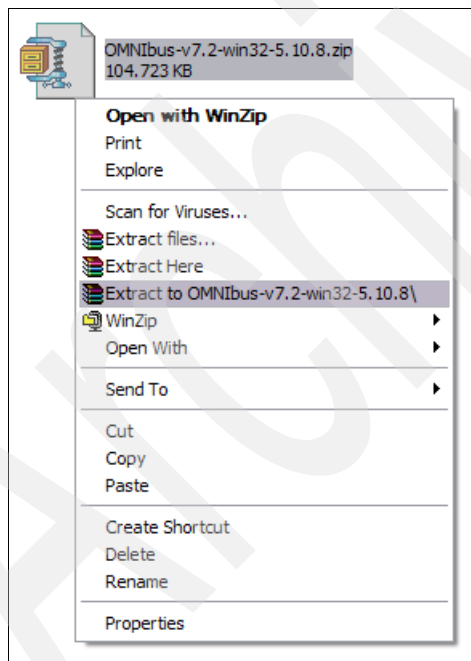


Figure 7-39 Unzip OMNibus installer software archive

2. Double-click **setup.exe** (Figure 7-40).

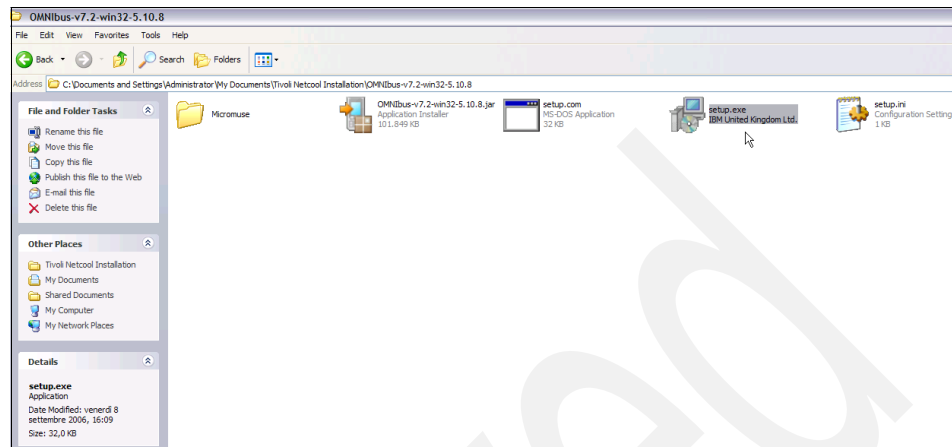


Figure 7-40 Double-click **setup.exe**

3. The Netcool installer welcome screen displays (Figure 7-41). Click **Next** to continue.



Figure 7-41 Welcome screen for the Netcool installer



4. Accept the license agreement and click **Next** (Figure 7-42).



Figure 7-42 The license agreement

5. Enter the installation directory and click **Next** (Figure 7-43).

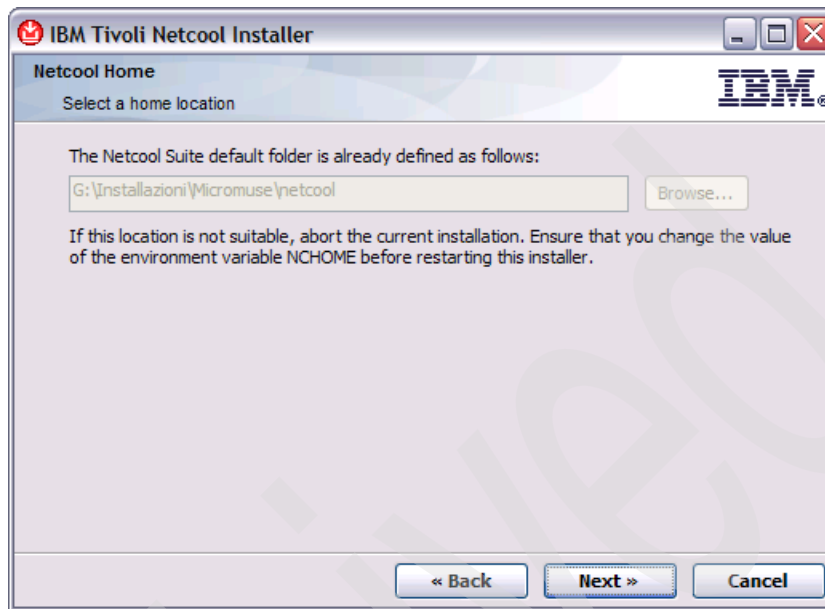


Figure 7-43 Enter the installation directory

6. Choose the components that you want to install. If you need to install only the Netcool/OMNibus MT NT Event Log Probe, select the following features:

- Process agent
- (Optionally) Desktop component to be able to configure the ObjectServer to be used by the probe in GUI mode

If you also need to be able to execute external procedures and scripts on the Windows machine, select the following features:

- Process agent
- Desktop
- Servers, to be able to configure an ObjectServer on SALVADOR
- Administrator, to manage the ObjectServer and the process agent

This difference is due to the fact that the ObjectServer WEIMAR installed on the WEIMAR Red Hat box cannot be used to execute an external procedure on SALVADOR (that is, a Windows machine).

7. After you choose the components to install, click **Next** (Figure 7-44).

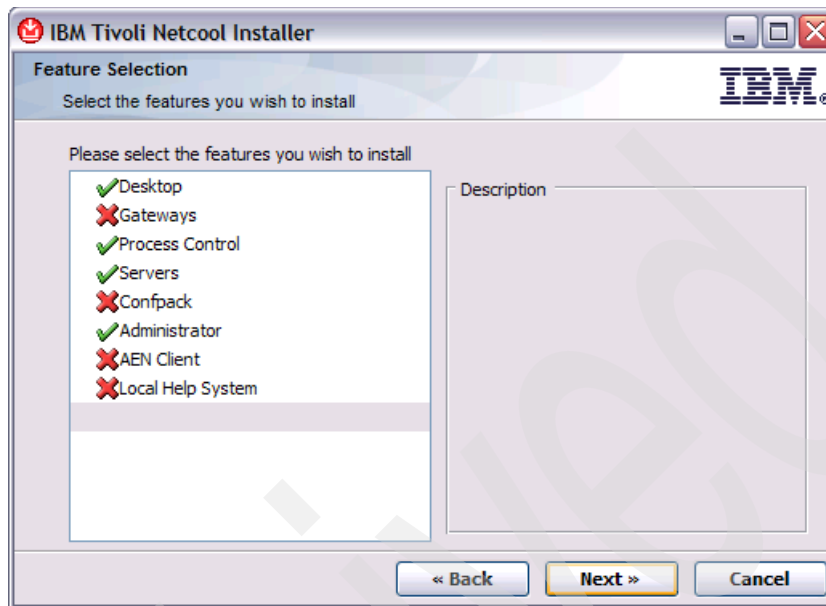


Figure 7-44 Selection of the features to install

8. Review your installation settings and click **Next** (Figure 7-45).

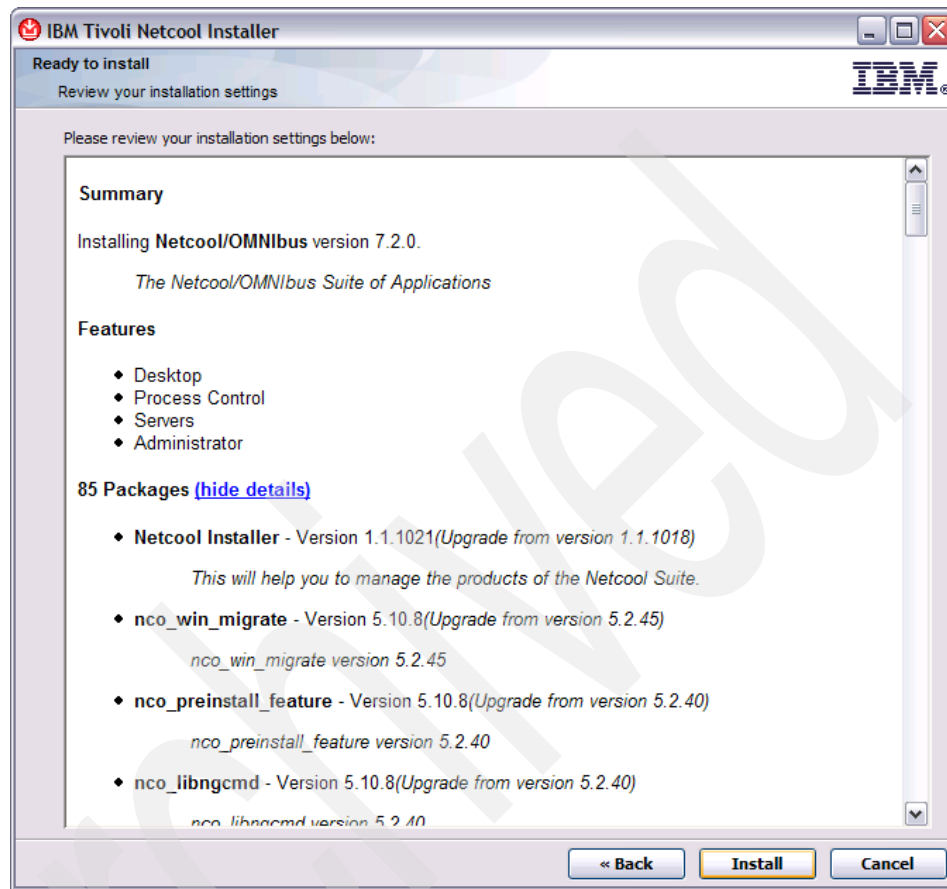


Figure 7-45 Installation settings review

9. Wait for the installation process to finish (Figure 7-46).

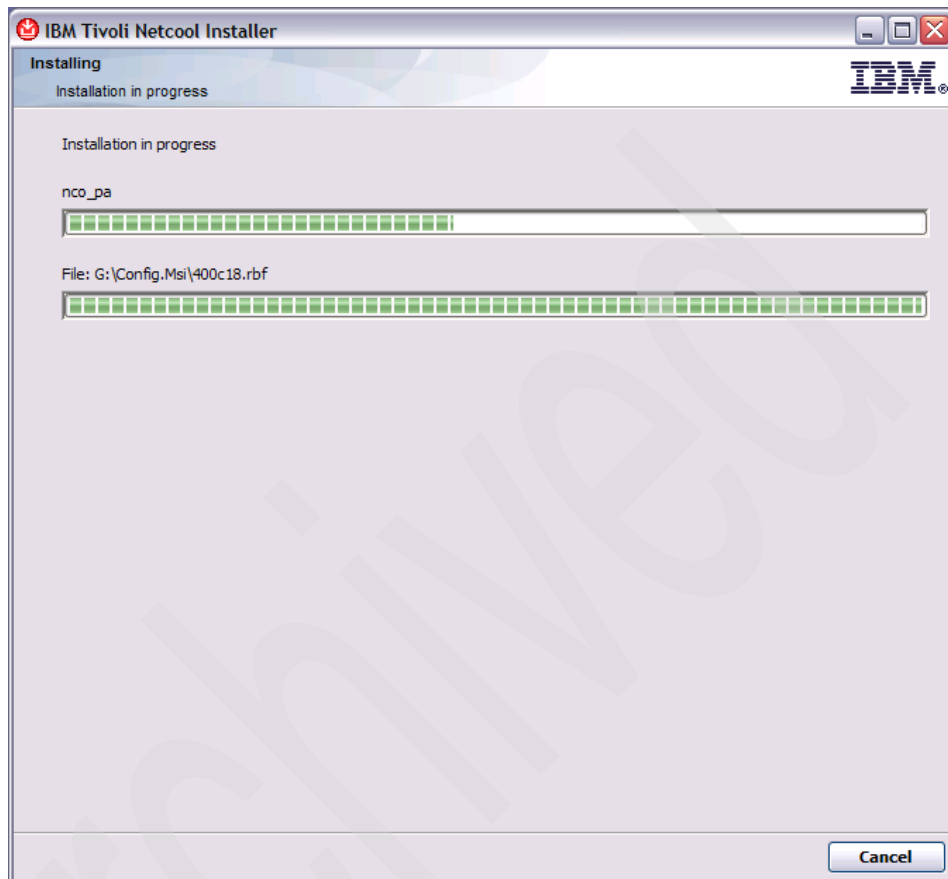


Figure 7-46 Installation in progress

10. The last panel informs you about the successful installation. Reboot the system after clicking **Finish** (Figure 7-47).

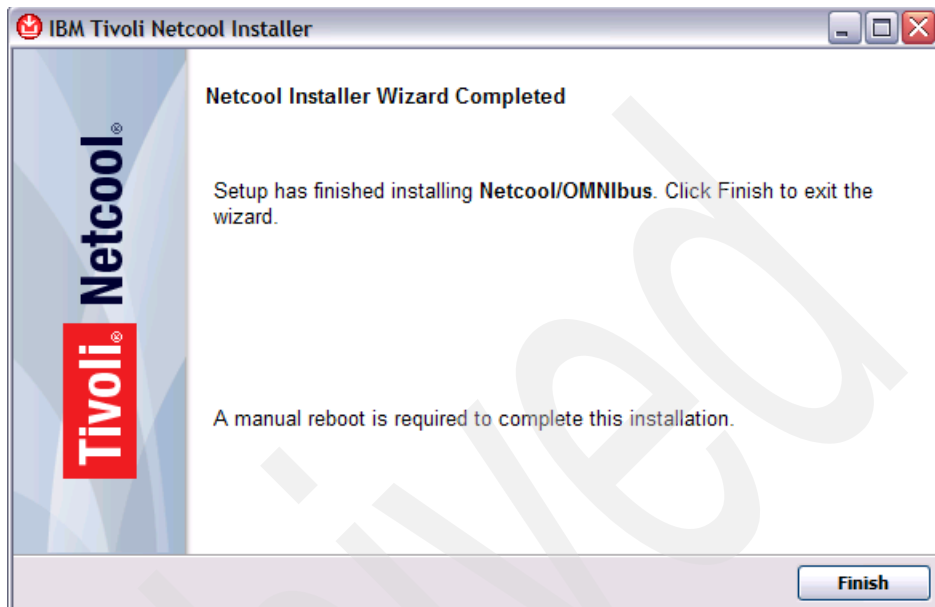


Figure 7-47 Installation completed

11. To start the installation of the probe, open the zipped archive containing the probe files (Figure 7-48).

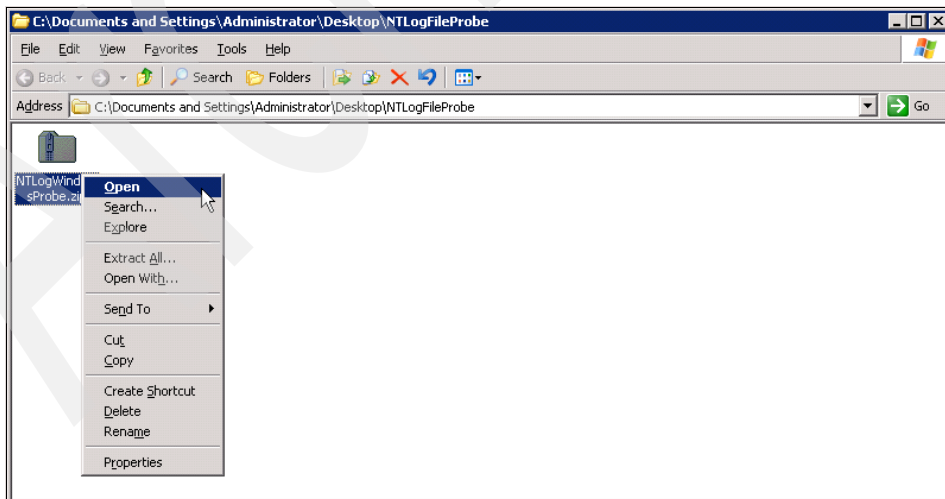


Figure 7-48 Open the zipped archive that contains the probe files

12. Double-click the directory to open it (Figure 7-48 on page 330).

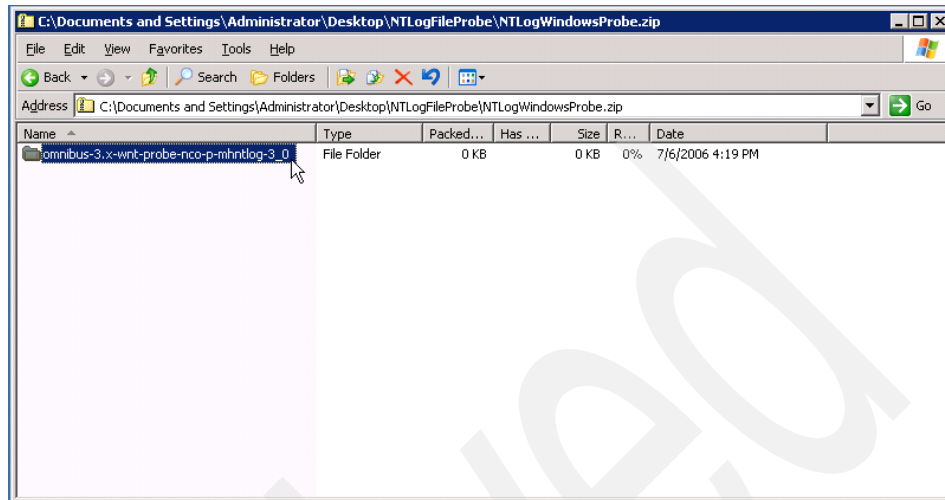


Figure 7-49 Open the unzipped directory that contains the probe files

13. Since we are installing the probe on OMNibus 7.2, we need to use the files contained in the post36 directory (Figure 7-50).

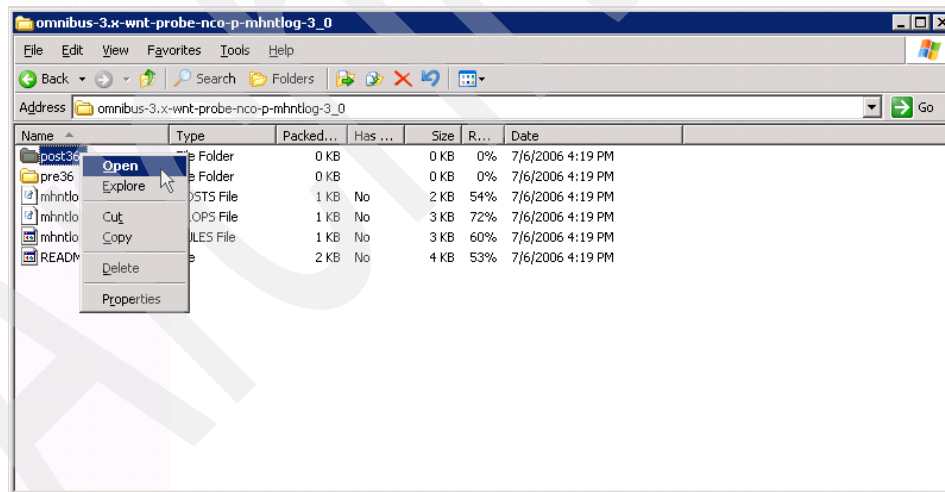


Figure 7-50 Open the post36 directory for an OMNibus 7.2 installation

14. Copy the two files nco\_p\_mhntlog.exe and nco\_p\_mhntlog.dll contained in the post36 directory (Figure 7-51).

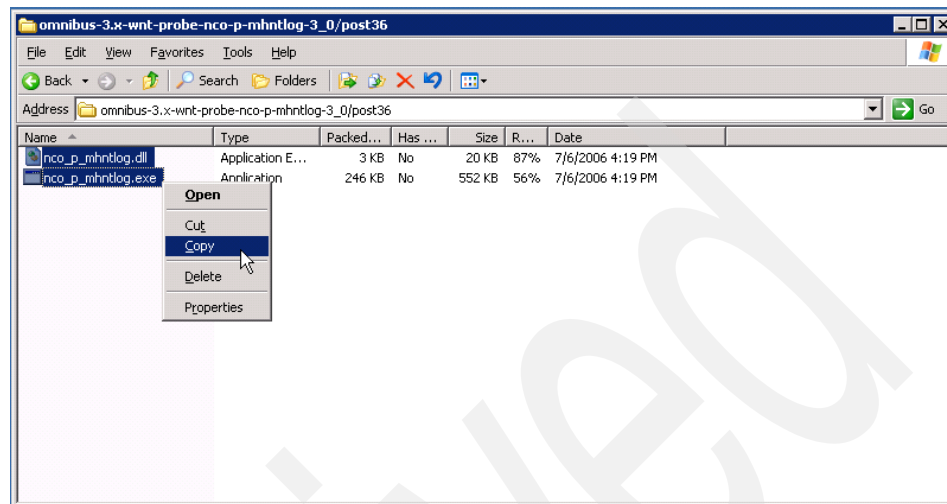


Figure 7-51 Copy the two files contained in the post36 directory



15. Paste them in the %OMNIHOME%\probes\win32 directory, where %OMNIHOME% is the installation directory for Netcool/OMNIBus (Figure 7-52).

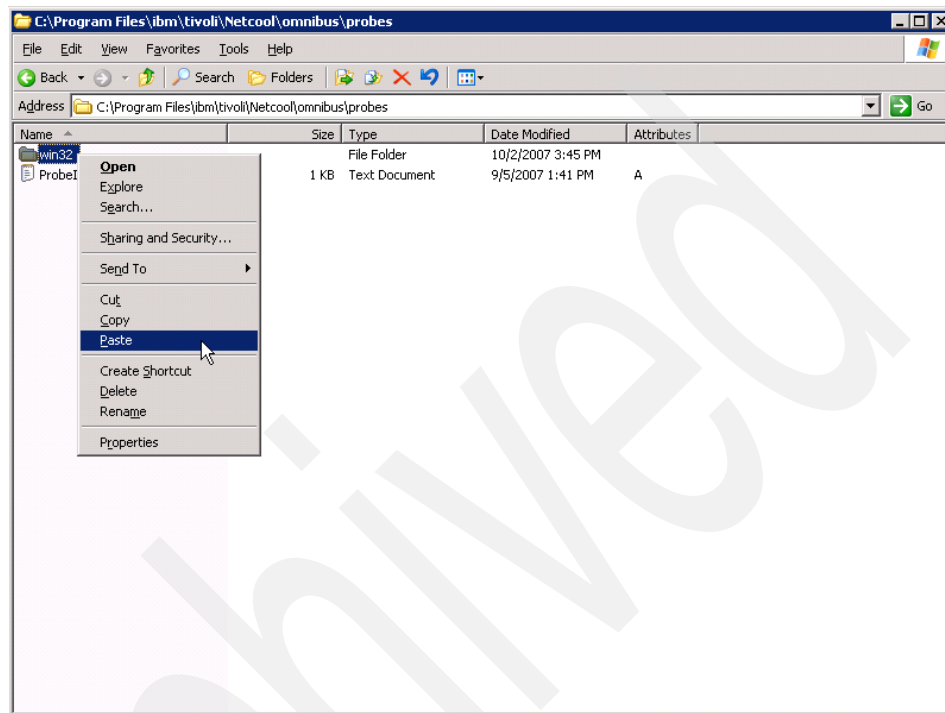


Figure 7-52 Paste the files in the win32 directory

16. Now return to the directory containing the configuration files for the probe: `mhntlog.hosts`, `mhntlog.props`, and `mhntlog.rules`. See Figure 7-53.

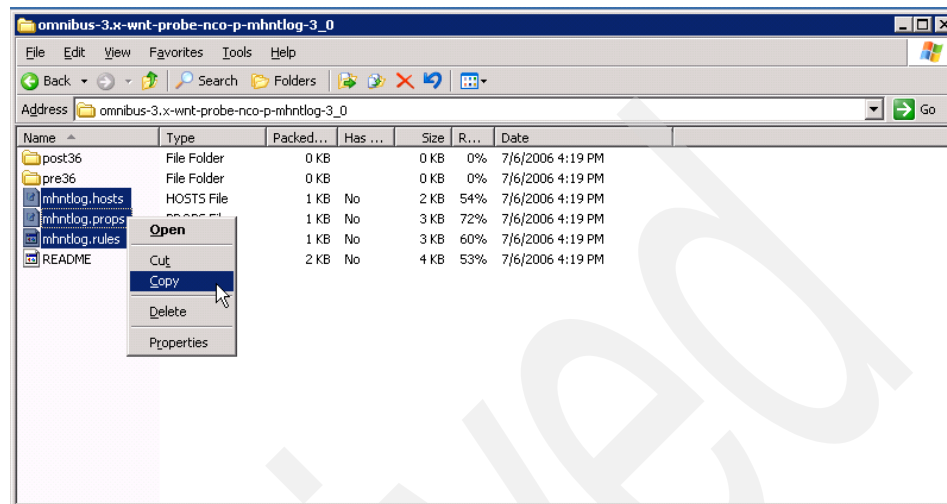


Figure 7-53 Copy the configuration files for the probe

17. Finally, paste these files into the %OMNIHOME%\probes\win32 directory, where %OMNIHOME% is the installation directory for Netcool/OMNIBus (Figure 7-54).

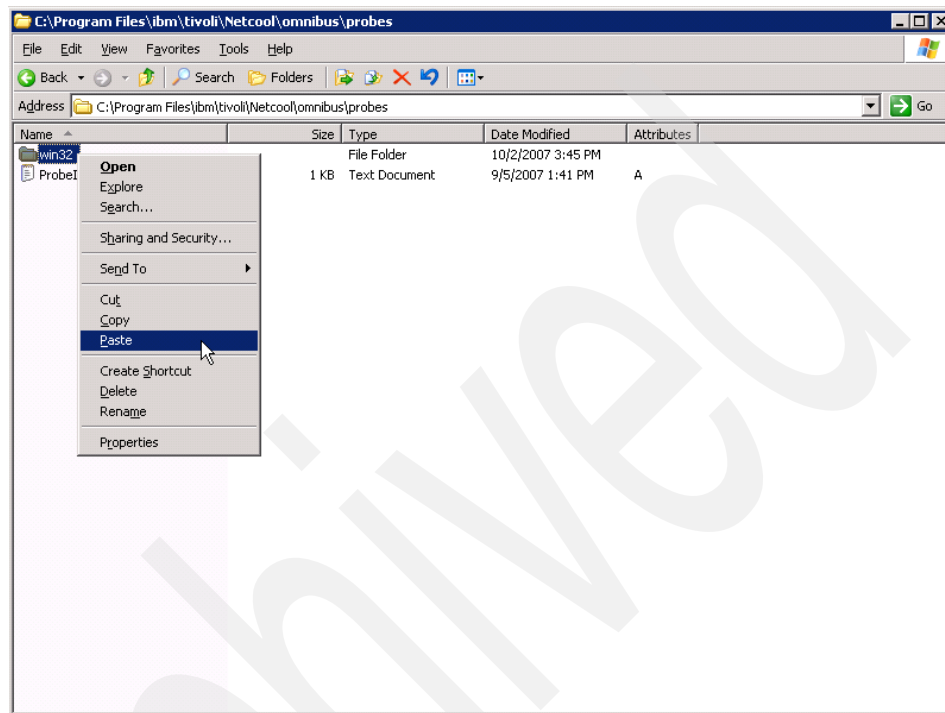


Figure 7-54 Paste the files in the %OMNIHOME%\probes\win32 directory

18. The %OMNIHOME%\probes\win32 directory should now look like Figure 7-55.

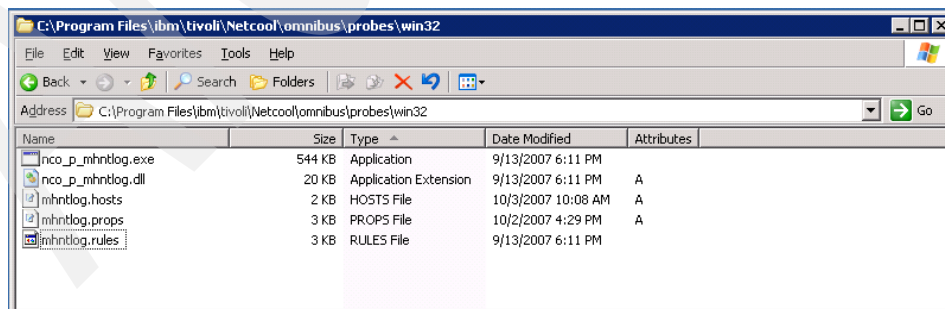


Figure 7-55 %OMNIHOME%\probes\win32 directory after the installation of the probe

19.Now start the server editor (Figure 7-56).

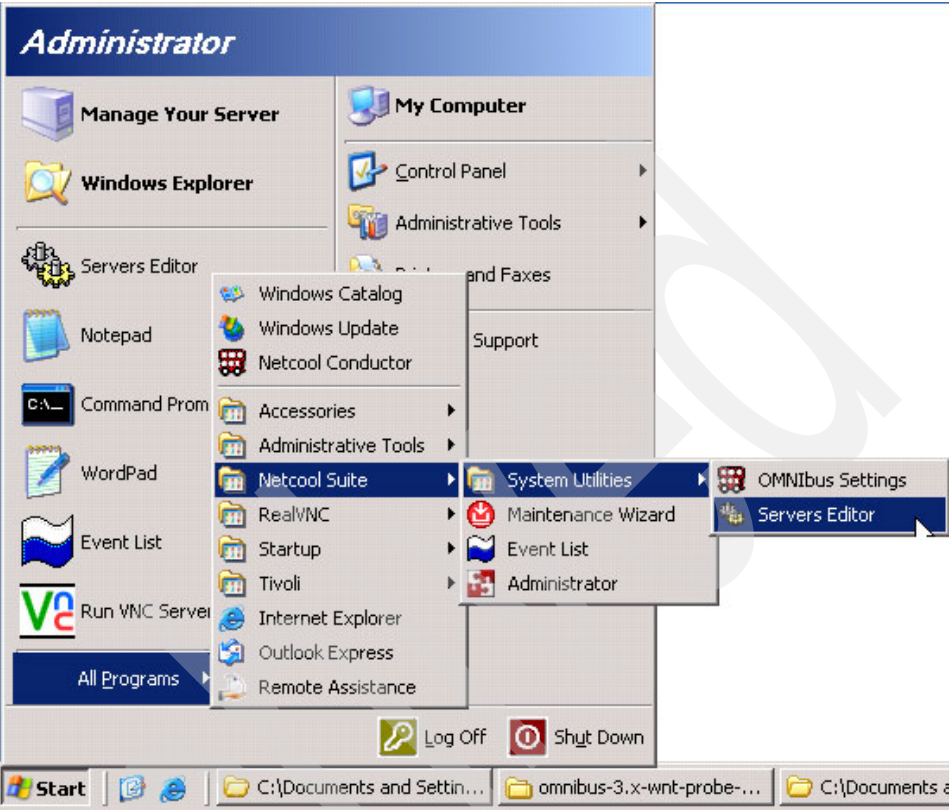


Figure 7-56 Start the servers editor

20.In the Servers Editor window, replace the default NCOMS entries with the ObjectServer that you want to send Windows log files events to, in this case WEIMAR on host WEIMAR on port 4100 (Figure 7-57).

WEIMAR	WEIMAR	4100	Winsock TCP/IP
Listeners: WEIMAR		4100	Winsock TCP/IP

Figure 7-57 Servers editor entry for WEIMAR ObjectServer

21. Configure the properties for the probe. In particular, you want the probe to be aware of the ObjectServer it has to send events to (in our case, WEIMAR). Edit the mhntlog.props file in the %OMNIHOME%\probes\win32 directory in the way shown in Example 7-6.

*Example 7-6 mhntlog.rules file after configuration*

```
#####
#
#      Copyright (C) 2005 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.
#
#####

#####
#
# Property Name                                Default
#
# Generic Properties
#
# AuthPassword                                :      ""
# AuthUserName                                :      ""
# AutoSAF                                     :      0
# BufferSize                                  :      10
# Buffering                                    :      0
# Help                                        :      0
# LookupTableMode                            :      3
# Manager                                     :      "mhntlog"
# MaxLogFileSize                             :      1048576
# MaxRawFileSize                             :      -1
# MaxSAFFileSize                             :      1048576
# MessageLevel                               :      "warn"
# MessageLog                                 :      "%OMNIHOME%\log\mhntlog.log"
# MsgDailyLog                                :      0
# MsgTimeLog                                 :      "0000"
# Name                                        :      "mhntlog"
```

```

# NetworkTimeout          :      0
# PollServer              :      0
# PropsFile                :
"%OMNIHOME%\probes\nt351\mhntlog.props"
# RawCapture               :      0
# RawCaptureFile           :      "%OMNIHOME%\var\mhntlog.cap"
# RawCaptureFileAppend     :      0
# RawCaptureFileBackup     :      0
# RetryConnectionCount     :     15
# RetryConnectionTimeOut   :     30
# RulesFile                :
"%OMNIHOME%\probes\nt351\mhntlog.rules"
# SAFFilename              :      ""
# Server                   :      "NCOMS"
# ServerBackup              :      ""
# StoreAndForward           :      1
# Version                   :      0
#
# Specific Properties
#
# Cleanstart                :      0
# HostFile                  :      "$OMNIHOME/probes/<arch>/mhntlog.hosts"
# NoNameResolution          :      0
# PollTime: 1
# ReadFileInterval          :      3
# RecoveryFile              :      "$OMNIHOME/probes/<arch>/mhntlog.reco"
# Replay                    :      0
#
#####

#####

#
# Add your settings here
#
#####
Server      :      "WEIMAR"

```

---

22. Finally, specify in the mhntlog.hosts file the host that you want to monitor and the kind of monitoring that you want to apply. The format for this configuration file is:

```

# HOST_INFO=<hostname>:MAX_RETRIES=<max
retries>;RETRY_INTERVAL=<retry interval>;LOG_FILES=<log file
name>,<log file name> etc.:PROBE_STATUS=<status>

```

We want to monitor the localhost (SALVADOR) security, application, and system log files. The maximum number of retries can be conveniently set to 3, and the time interval between each try can be set to 20 seconds. Edit the file as shown in Example 7-7.

*Example 7-7 mhnlog.hosts file after configuration*

---

```
#
# This is a sample host configuration file for use with the Multi-headed NT Event Log
# probe.
# In order to use it, uncomment the relevant lines and change the appropriate
# parameters to conform to the specific configuration required.
#
# The host configuration file (the list of hosts and logs to monitor) must be written
# in a specific format, described below.
#
# Format for host configuration file
#
# HOST_INFO=<hostname>:MAX_RETRIES=<max retries>:RETRY_INTERVAL=<retry
# interval>:LOG_FILES=<log file name>,<log file name> etc.:PROBE_STATUS=<status>
#
#
# Key:
#     <hostname>           : host name (or IP address) of machine on which log
# to be monitored resides
#     <max retries>        : number of times to attempt to connect to host
#     <retry interval>     : time (in seconds) to wait in between each connect
# attempt
#     <log file name>      : name of the log file to be monitored (eg.
# Application)
#     <status>             : ON or OFF - if set to OFF, the probe will not
# monitor this host at all
#
# Separators:
#   Colon (':') is the terminator for each individual entry.
#   Each log file to be monitored should be separated by a comma (,).
#   There should be NO spaces between any individual element and each option should
# be separated from its value by an equals (=).
#
# Comment lines are preceded by '#', blank lines are ignored.
# Following is a sample 2 lines of an example list of hosts/logs to monitor...

#
HOST_INFO=examplehost:MAX_RETRIES=3:RETRY_INTERVAL=20:LOG_FILES=Application,System:PR
OBE_STATUS=ON
```

```
#
HOST_INFO=192.168.34.60:MAX_RETRIES=3:RETRY_INTERVAL=20:LOG_FILES=Application,Dummy:PROBE_STATUS=ON

HOST_INFO=SALVADOR:MAX_RETRIES=3:RETRY_INTERVAL=20:LOG_FILES=Application,System,Security:PROBE_STATUS=ON
```

---

23. You can now start the probe from the command line, as suggested in Example 7-8.

*Example 7-8 Start the probe from the command line*

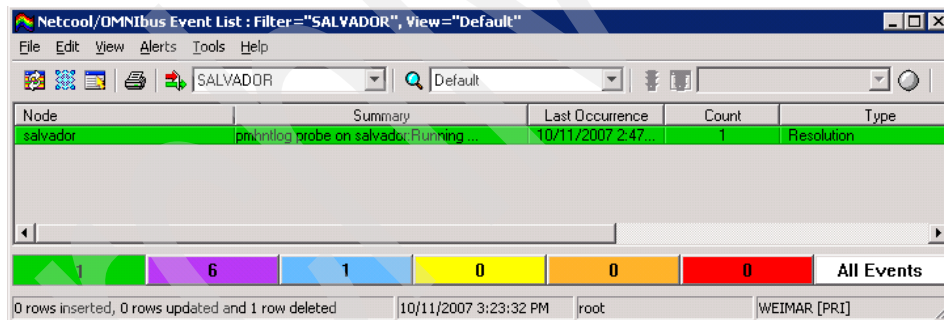
---

```
C:\Program Files\ibm\tivoli\Netcool\omnibus\probes\win32>nco_p_mhntlog.exe
Netcool/OMNIBus MT NT Event Log Probe - Version 7.2
(C) Copyright IBM Corp. 1994, 2007
```

Information: Requested to execute in CONSOLE mode

---

If everything worked, you should see a green probe running an event in the WEIMAR event list (Figure 7-58).



The screenshot shows the 'Netcool/OMNIBus Event List' window with the filter set to 'SALVADOR'. The table below represents the data shown in the window:

Node	Summary	Last Occurrence	Count	Type
salvador	phntlog probe on salvador Running	10/11/2007 2:47	1	Resolution

Below the table, there is a summary bar with colored segments: 6 (green), 1 (blue), 0 (yellow), 0 (orange), and 0 (red). The status bar at the bottom indicates '0 rows inserted, 0 rows updated and 1 row deleted' and shows the time '10/11/2007 3:23:32 PM' and user 'root'.

*Figure 7-58 The event list shows that the probe has started successfully*

## 7.5.2 Installing and configuring the process agent on Windows

As mentioned above, to enable external procedure execution on a target machine it is necessary to have a process agent up and running on it. In the case of a Windows machine, process agent cannot connect to an ObjectServer running on a UNIX or Linux box.

In this scenario, therefore, the only solution consists of the installation of an additional ObjectServer on a Windows machine.



After the installation of an ObjectServer on a Windows machine and the configuration of process agent, we will be able to execute external procedures and scripts on it.

The steps we went through to achieve this in our lab environment are the following:

1. Install an ObjectServer called SALVADOR on your Windows machine, executing the following from the DOS command line:  

```
C:\Program Files\ibm\tivoli\Netcool\omnibus\bin>nco_dbinit -server SALVADOR
```
2. Configure the properties file for the SALVADOR ObjectServer. It has to connect to the process agent named SALVADOR\_PA that will be started later. Edit the salvador.props file in the way shown in Example 7-9.

*Example 7-9 salvador.props file after configuration*

```
#####  
#  
# Licensed Materials - Property of IBM  
#  
# 572404800  
#  
# (C) Copyright IBM Corp. 1994, 2007. All Rights Reserved  
#  
# US Government Users Restricted Rights - Use, duplication  
# or disclosure restricted by GSA ADP Schedule Contract  
# with IBM Corp.  
#  
#  
# Ident: $Id: NCOMS.props 1.4 2003/06/17 09:38:14 stephenc Development $  
#  
#####  
#  
# AlertSecurityModel: 0 # INTEGER (Desktop security model)  
# AllowConnections: TRUE # BOOLEAN (Specifies whether or not non-root users can  
connect)  
# AllowISQL: TRUE # BOOLEAN (Specifies whether or not isql connections are allowed)  
# AllowISQLWrite: TRUE # BOOLEAN (Specifies whether or not modifications by isql  
connections are allowed)  
# AllowTimedRefresh: FALSE # BOOLEAN (Allow desktops to apply timed refresh)  
# Auto.Debug: FALSE # BOOLEAN (Automation debug )  
# Auto.Enabled: TRUE # BOOLEAN (Automation enable), "Automation  
enable (default: %s)",  
# Auto.StatsInterval: 60 # INTEGER (Automation statistics interval)
```

```

# BackupObjectServer: FALSE # BOOLEAN (Backup ObjectServer)
# Connections: 30 # INTEGER (Number of connections permitted)
# DTMaxTopRows: 100 # INTEGER (Desktop maximum top rows)
# DeleteLogFile: '$OMNIHOME/log/NCOMS_deletes_file.log' # STRING (Log file to record
all delete commands)
# DeleteLogging: FALSE # BOOLEAN (Turn on the delete logging)
# DeleteLogLevel: 0 # INTEGER (The level of detail sent to the log file)
# DeleteLogSize: 1024 # INTEGER (The maximum size of the log file)
# GWDeduplication: 0 # INTEGER (Gateway deduplication mode)
# Help: FALSE # BOOLEAN (Display help information.)
# Granularity: 60 # INTEGER (Iduc update granularity)
# Iduc.ListeningHostname: 'localhost' # STRING (Hostname to listen for Iduc
connections)
# Iduc.ListeningPort: 0 # INTEGER (Iduc port to listen on)
# Ipc.NumberConnections: 30 # INTEGER (Number of connections permitted)
# Ipc.QueueSize: 1024 # INTEGER (Size of middleware internal server queues)
# Ipc.ServerOverrideSybase: FALSE # BOOLEAN (Override the automatic SYBASE setting
(DEBUG only))
# Ipc.SingleThreaded: FALSE # BOOLEAN (Single Threaded IPC)
# Ipc.SSLCertificate: '$OMNIHOME/etc/NCOMS.crt' # STRING (SSL certificate)
# Ipc.SSLEnable: FALSE # BOOLEAN (Enable SSL)
# Ipc.SSLPrivateKeyPassword: '' # STRING (Private key password)
# Ipc.StackSize: 67584 # INTEGER (Size of middleware internal server thread stacks)
# Ipc.TruncateVendorLogFile: TRUE # BOOLEAN (Truncate vendor log file on start up)
# Ipc.VendorLogFileSize: 1024 # INTEGER (Maximum size of middleware vendor's log file
(kB))
# MaxLogFileSize: 1024 # INTEGER (Maximum log file size in kbytes.)
# Memstore.DataDirectory: '$OMNIHOME/db' # STRING (Memory storage directory)
# MessageLevel: 'warn' # STRING (Message reporting level)
# MessageLog: '$OMNIHOME/log/NCOMS.log' # STRING (Path to the message log file.)
# Name: 'NCOMS' # STRING (Server name)
# NoProbeParser: FALSE # BOOLEAN (Disable the ObjectServer's fast probe parser)
# PA.Name: 'NCO_PA' # STRING (Name of process agent)
# PA.Password: '' # STRING (Password to use when connecting to the process agent)
# PA.Username: 'root' # STRING (Username to use when connecting to the process agent)
# PAAware: 0 # INTEGER ()
# PaAwareName: '' # STRING ()
# PasswordEncryption: 'DES' # STRING (The encryption scheme to use for users
passwords. DES or AES)
# PasswordFormat: '8:1:1:0' # STRING (The password restriction
format.[<min_len>:<alpha>:<digit>:<punct>])
# Profile: FALSE # BOOLEAN (Enable profiling)
# ProfileStatsInterval: 120 # INTEGER (Profiler statistics report interval)
# Props.CheckNames: TRUE # BOOLEAN (Cause program to abort if any property is not
understood at the time it is read)

```

```

# PropsFile: '$OMNIHOME/etc/NCOMS.props' # STRING (Path to gateways property file.)
# Region.ProtectAll: FALSE # BOOLEAN (Make all regions write-protected)
# RestrictionUpdateCheck: TRUE # BOOLEAN (Specifies whether or not users can update
events that will not appear in their view
# RestrictProxySQL: FALSE # BOOLEAN (Restrict the set of SQL commands proxies can
submit)
# Sec.AuditLevel: 'warn' # STRING (Security audit level)
# Sec.AuditLog: 'stdout:' # STRING (Security audit trail)
# Sec.UsePam: TRUE # BOOLEAN (Use Pluggable Authentication Modules)
# SecureMode: FALSE # BOOLEAN (Secure authentication)
# UniqueLog: FALSE # BOOLEAN (Make log file name unique (adds PID))
# Version: FALSE # BOOLEAN (Display version information.)

```

```

AlertSecurityModel: 0
AllowConnections: TRUE
AllowISQL: TRUE
AllowISQLWrite: TRUE
AllowTimedRefresh: FALSE
Auto.Debug: FALSE
Auto.Enabled: TRUE
Auto.StatsInterval: 60
BackupObjectServer: FALSE
Connections: 30
DTMaxTopRows: 100
DeleteLogging: FALSE
DeleteLogLevel: 0
DeleteLogSize: 1024
GWDeduplication: 0
Granularity: 60
Iduc.ListeningPort: 0
Ipc.SSLCertificate: ''
Ipc.SSLEnable: FALSE
Ipc.SSLPrivateKeyPassword: ''
MaxLogFileSize: 1024
Memstore.DataDirectory: '$OMNIHOME/db'
MessageLevel: 'warn'
PA.Name: 'SALVADOR_PA'
PA.Password: ''
Profile: FALSE
ProfileStatsInterval: 60
RestrictPasswords: FALSE
RestrictProxySQL: FALSE
RestrictionUpdateCheck: TRUE
Sec.AuditLevel: 'warn'
UniqueLog: FALSE

```

# End of File

---

3. Open the Servers Editor utility as explained above.
4. Add the following entries in the Servers Editor window:
  - ObjectServer SALVADOR on host SALVADOR and port 4600
  - Process agent SALVADOR\_PA on host SALVADOR and port 4500

The result of this is shown in Figure 7-59.

SALVADOR	SALVADOR	4600	Winsock TCP/IP
Listeners:	SALVADOR	4600	Winsock TCP/IP
SALVADOR_PA	SALVADOR	4500	Winsock TCP/IP
Listeners:	SALVADOR	4500	Winsock TCP/IP

*Figure 7-59 Servers editor entries for SALVADOR ObjectServer and process agent*

5. Start the SALVADOR ObjectServer issuing the following from the command line:

```
C:\Program Files\ibm\tivoli\Netcool\omnibus\bin>nco_objserv -name
SALVADOR
Netcool/OMNIBus Object Server - Version 7.2
(C) Copyright IBM Corp. 1994, 2007
```

```
Information: Requested to execute in CONSOLE mode
```

```
Server 'SALVADOR' initialised - entering RUN state.
```

6. Ensure that the ObjectServer is successfully reached by selecting it and clicking **Test** in the servers editor (Figure 7-60).

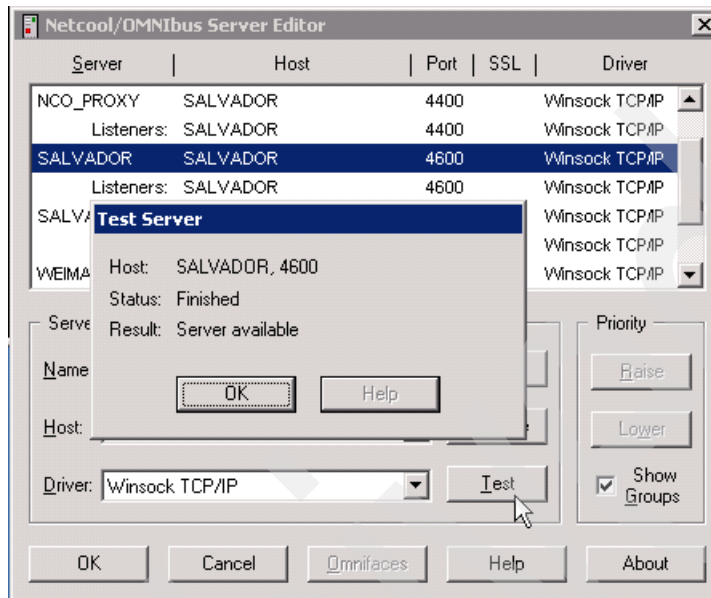


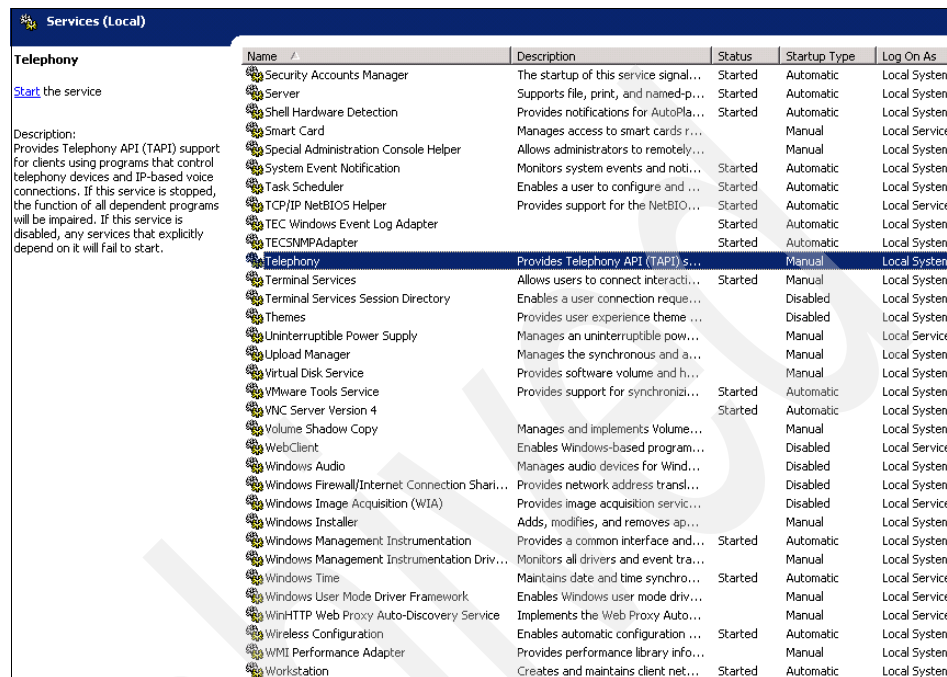
Figure 7-60 Ensure that the SALVADOR ObjectServer is reached successfully

7. Start the SALVADOR\_PA process agent in the following way:

```
C:\Program Files\ibm\tivoli\Netcool\omnibus\bin>nco_pa -name  
SALVADOR_PA  
Netcool/OMNIBus Process Agent - Version 7.2  
(C) Copyright IBM Corp. 1994, 2007
```

Information: Requested to execute in CONSOLE mode

8. To make sure that process agent is working correctly, we try to execute an external procedure that starts the sample Windows Telephony service. Before starting this test, Telephony service is not started.



The screenshot shows the Windows Services (Local) console. The 'Telephony' service is highlighted in blue. Its status is 'Stopped', and its startup type is 'Manual'. The description for the Telephony service is 'Provides Telephony API (TAPI) support for clients using programs that control telephony devices and IP-based voice connections. If this service is stopped, the function of all dependent programs will be impaired. If this service is disabled, any services that explicitly depend on it will fail to start.'

Name	Description	Status	Startup Type	Log On As
Security Accounts Manager	The startup of this service signal...	Started	Automatic	Local System
Server	Supports file, print, and named-p...	Started	Automatic	Local System
Shell Hardware Detection	Provides notifications for AutoPla...	Started	Automatic	Local System
Smart Card	Manages access to smart cards r...	Manual	Manual	Local Service
Special Administration Console Helper	Allows administrators to remotely...	Manual	Manual	Local System
System Event Notification	Monitors system events and noti...	Started	Automatic	Local System
Task Scheduler	Enables a user to configure and ...	Started	Automatic	Local System
TCP/IP NetBIOS Helper	Provides support for the NetBIO...	Started	Automatic	Local Service
TEC Windows Event Log Adapter		Started	Automatic	Local System
TECSNMPAdapter		Started	Automatic	Local System
Telephony	Provides Telephony API (TAPI)s...	Stopped	Manual	Local System
Terminal Services	Allows users to connect interacti...	Started	Manual	Local System
Terminal Services Session Directory	Enables a user connection reque...	Disabled	Disabled	Local System
Themes	Provides user experience theme ...	Disabled	Disabled	Local System
Uninterruptible Power Supply	Manages an uninterruptible pow...	Manual	Manual	Local Service
Upload Manager	Manages the synchronous and a...	Manual	Manual	Local System
Virtual Disk Service	Provides software volume and h...	Manual	Manual	Local System
VMware Tools Service	Provides support for synchronizi...	Started	Automatic	Local System
VNC Server Version 4		Started	Automatic	Local System
Volume Shadow Copy	Manages and implements Volume...	Manual	Manual	Local System
WebClient	Enables Windows-based program...	Disabled	Disabled	Local Service
Windows Audio	Manages audio devices for Wind...	Disabled	Disabled	Local System
Windows Firewall/Internet Connection Shari...	Provides network address transl...	Disabled	Disabled	Local System
Windows Image Acquisition (WIA)	Provides image acquisition servic...	Disabled	Disabled	Local Service
Windows Installer	Adds, modifies, and removes ap...	Manual	Manual	Local System
Windows Management Instrumentation	Provides a common interface and...	Started	Automatic	Local System
Windows Management Instrumentation Driv...	Monitors all drivers and event tra...	Manual	Manual	Local System
Windows Time	Maintains date and time synchro...	Started	Automatic	Local Service
Windows User Mode Driver Framework	Enables Windows user mode driv...	Manual	Manual	Local Service
WinHTTP Web Proxy Auto-Discovery Service	Implements the Web Proxy Auto...	Manual	Manual	Local Service
Wireless Configuration	Enables automatic configuration ...	Started	Automatic	Local System
WMI Performance Adapter	Provides performance library info...	Manual	Manual	Local System
Workstation	Creates and maintains client net...	Started	Automatic	Local System

Figure 7-61 Telephony Windows service is not started

9. Create a TelephonyServiceStart.bat file containing the instruction shown in Figure 7-62.

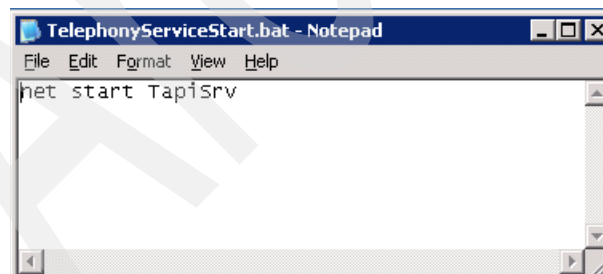


Figure 7-62

The **net start Tapisrv** instruction is responsible for the start of the Telephony Windows Service.

10. Open the **Administrator** panel.



Figure 7-63 Open Administrator window

11. The Import Connection Wizard is displayed. It will enable us to import the connection details for ObjectServers and process agents in a guided way. Click **Next**.



Figure 7-64 Connection Import Wizard



12.Import the ObjectServer details as shown in Figure 7-65. Click **Next**.

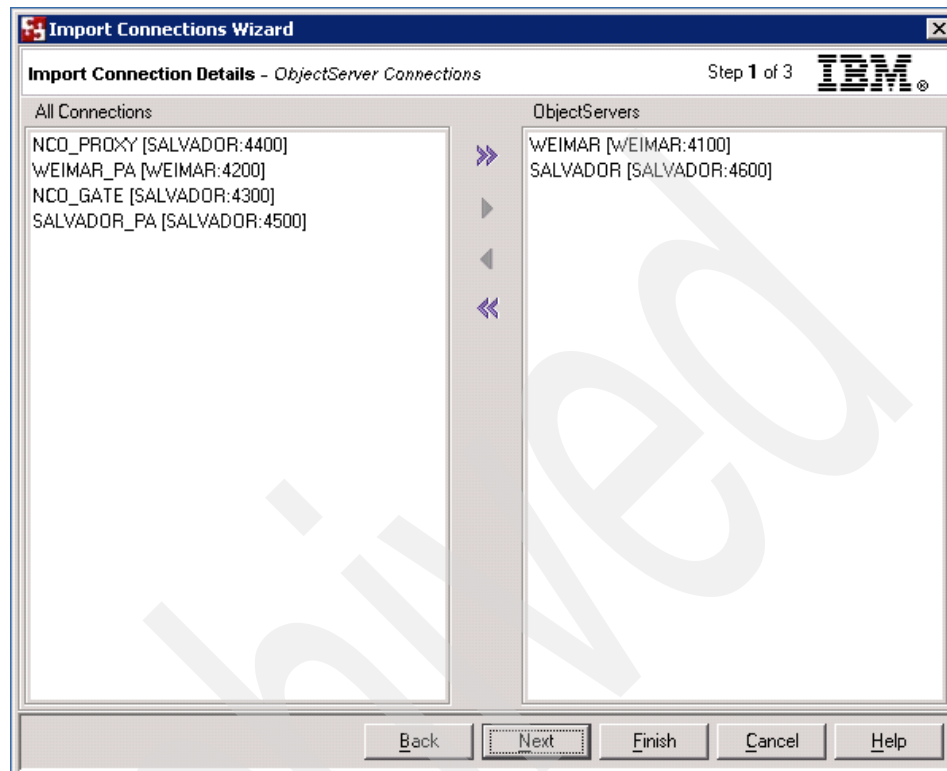


Figure 7-65 Import ObjectServer connections

13.Import process agent details as shown in Figure 7-66. Click **Next**.

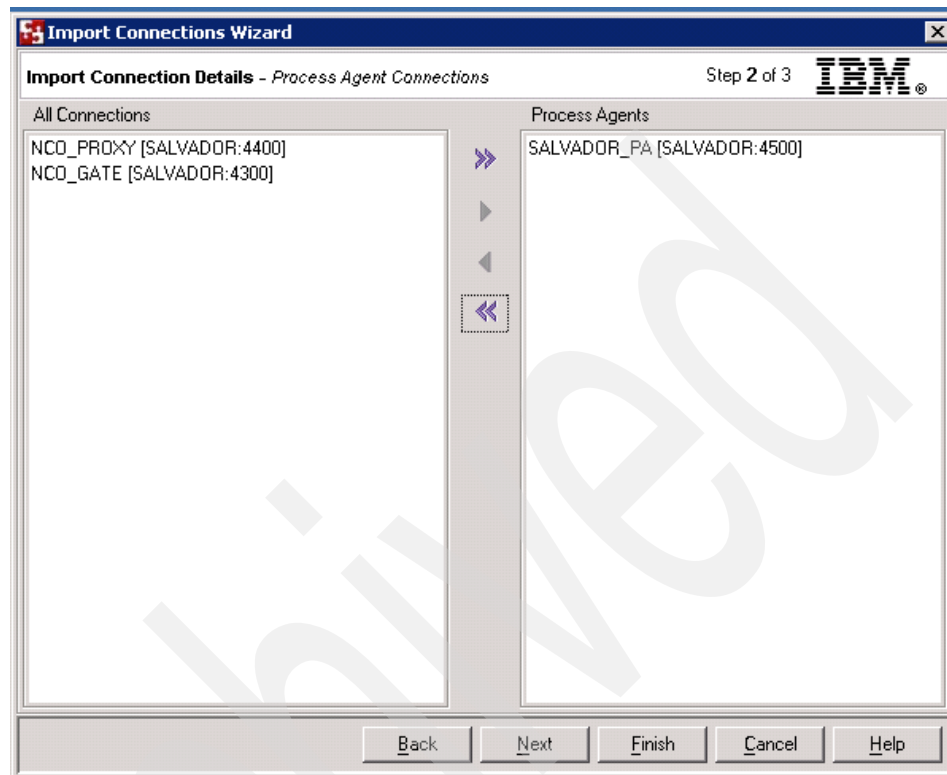


Figure 7-66 Import process agent connections

14. The connection import process ends (Figure 7-67). Click **Finish**.

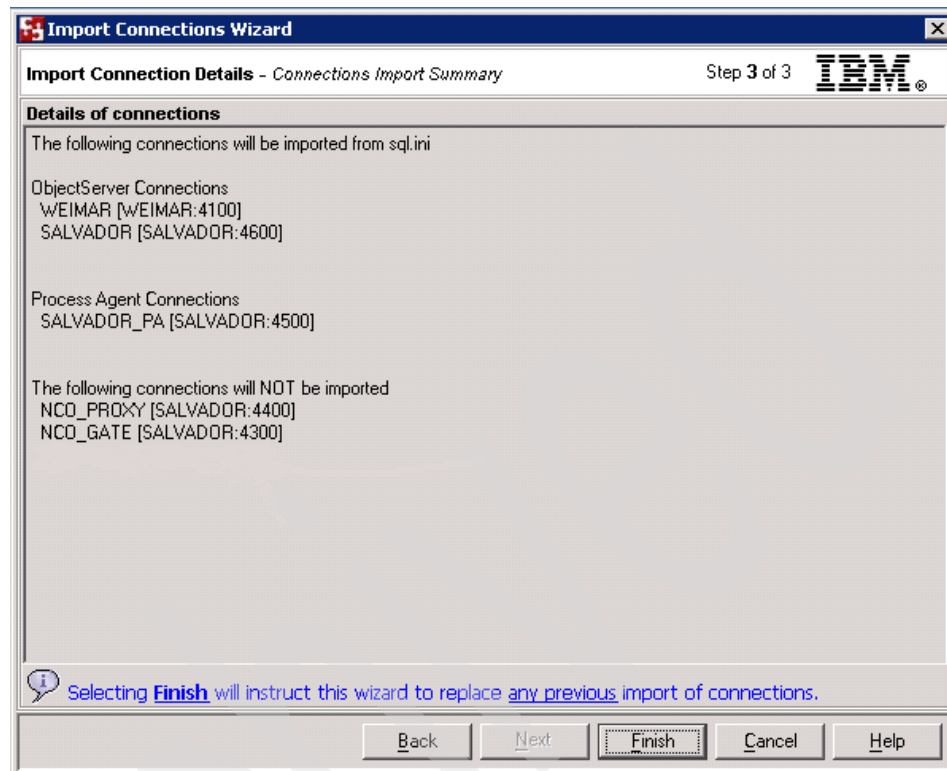


Figure 7-67 Connection import summary

15. Click **Hosts** → **SALVADOR** → **SALVADOR, SALVADOR:4600**. Insert the default username and password combination (root/”) to log in to the SALVADOR ObjectServer on the SALVADOR Windows box (Figure 7-68).

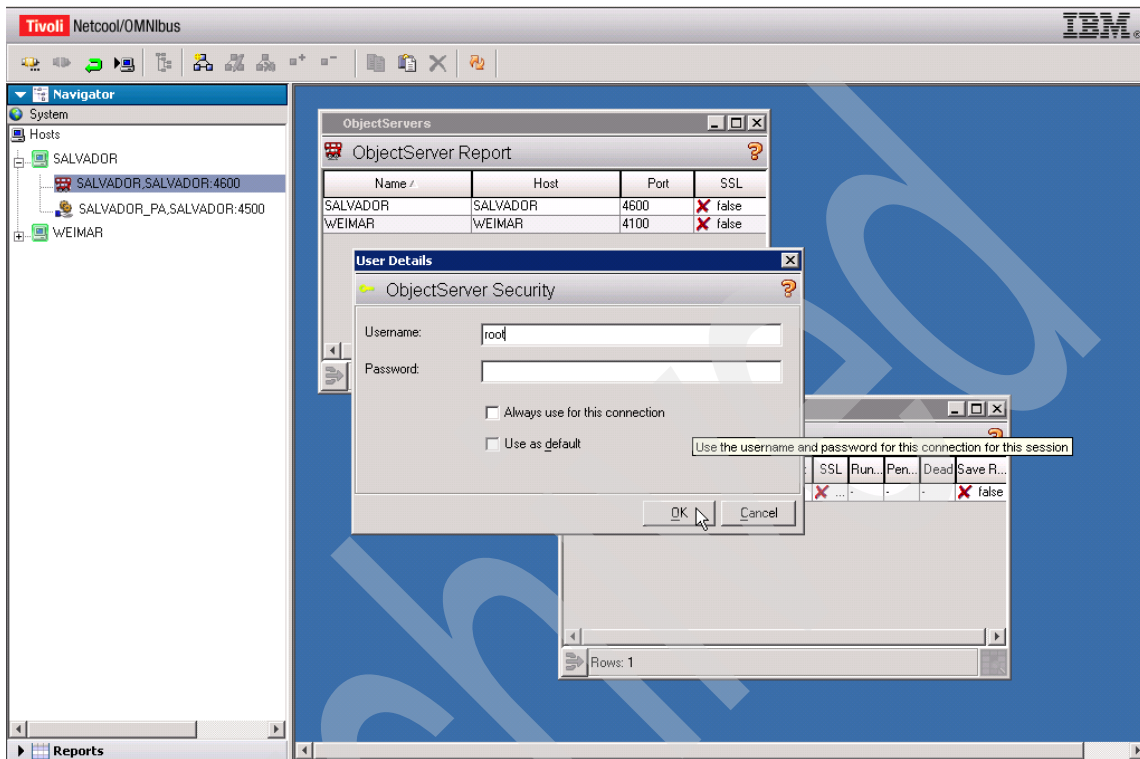


Figure 7-68 Log in to SALVADOR ObjectServer

16. Navigate to **Automation** → **Procedures**. Right-click in the empty grey space on the right and select **Add external procedure** (Figure 7-69).

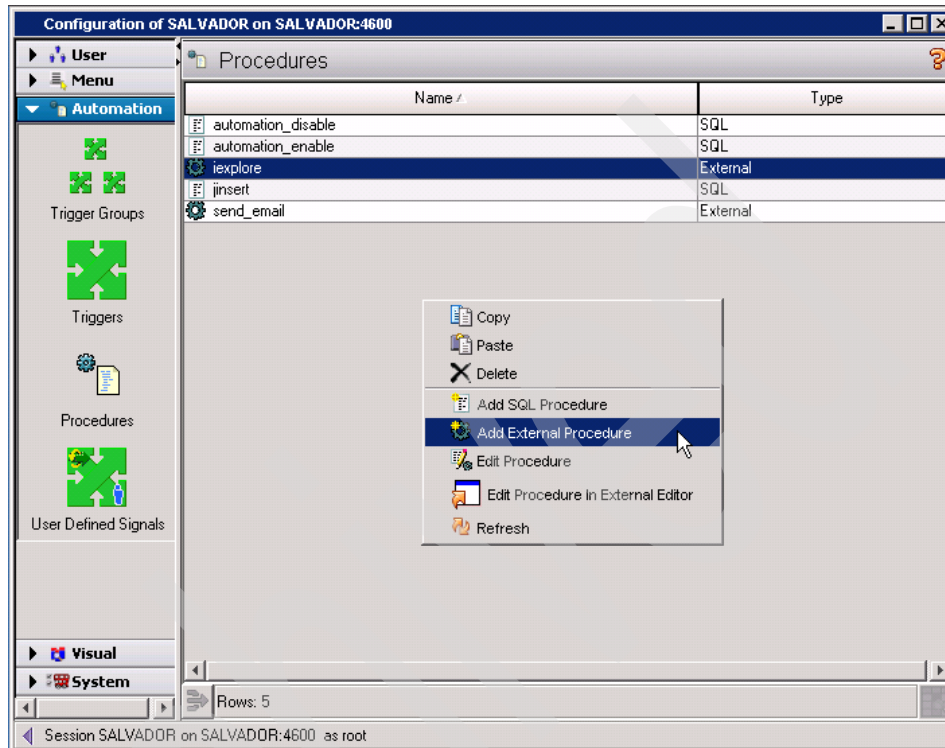


Figure 7-69 Select Add External Procedure

17. Give the telephony\_service name to the external procedure, and in the Executable field enter the path to the TelephonyServiceStart.bat file created in step 8:

C:\\Documents and  
Settings\\Administrator\\Desktop\\TelephonyServiceStart.bat.

You can also use the **Browse** button to choose directly the file (Figure 7-70).

The screenshot shows a Windows-style dialog box titled "Edit Procedure (SALVADOR on SALVADOR:4600)". Inside, there's a tab labeled "External Procedure Details". The "Name:" field contains "telephony\_service". Below it is a "Parameters:" section with an empty list box and navigation arrows. Further down, there are fields for "In/Out:" (set to "in"), "Name:" (empty), "Data Type:" (set to "Integer"), and "Array:" (unchecked). The "Executable:" field contains a file path "C:\\Documents and Settings\\Administrator\\Desktop\\TelephonyServiceStart.bat" and a "Browse..." button. Below that is an empty "Arguments:" list box. The "Host:" field contains "SALVADOR". At the bottom, there are "User ID:" and "Group ID:" fields, both set to "0". The dialog ends with "OK" and "Cancel" buttons.

Figure 7-70 Create procedure *telephony\_service*

18. Navigate to **System** → **SQL** and insert the appropriate SQL instruction that calls the *telephony\_service* procedure:

```
execute procedure telephony_service
```

19. Click the **Execute** button to execute the SQL statement (Figure 7-71).

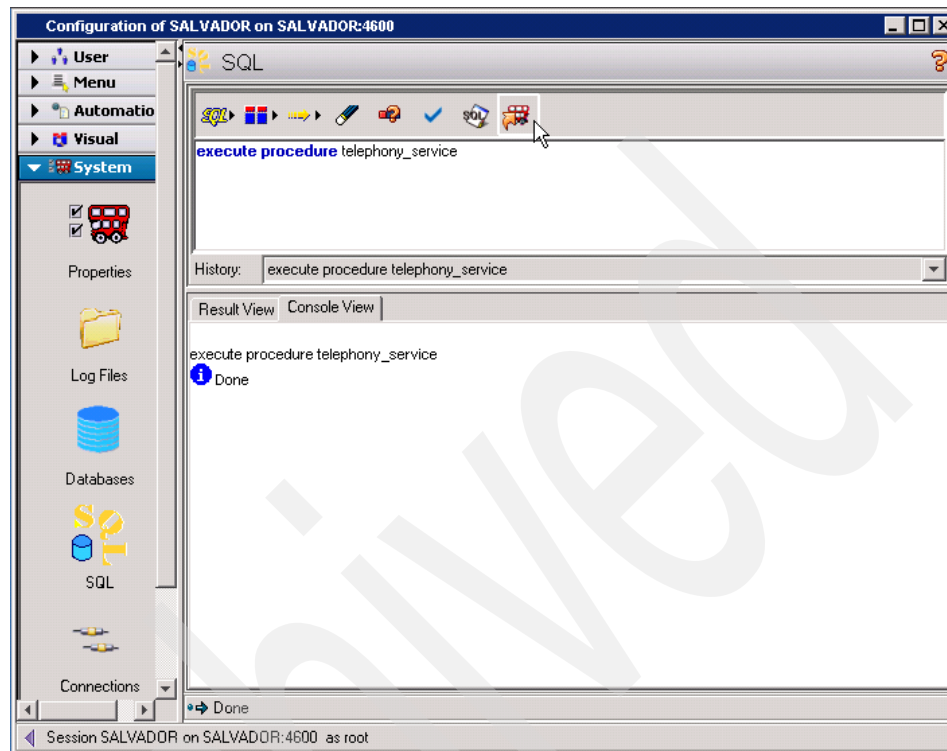


Figure 7-71 Write and execute the SQL statement that calls the procedure

20. Check the Windows service and make sure that the Telephony service has started (Figure 7-72).

Services (Local)						
Telephony						
<a href="#">Stop</a> the service <a href="#">Pause</a> the service <a href="#">Restart</a> the service	Name	Description	Status	Startup Type	Log On As	
	Portable Media Serial Num...	Retrieves t...		Manual	Local System	
	Print Spooler	Manages al...	Started	Automatic	Local System	
Description: Provides Telephony API (TAPI) support for clients using programs that control telephony devices and IP-based voice connections. If this service is stopped, the function of all dependent programs will be impaired. If this service is disabled, any services that explicitly depend on it will fail to start.	Protected Storage	Protects st...	Started	Automatic	Local System	
	Remote Access Auto Conn...	Detects un...		Manual	Local System	
	Remote Access Connection...	Manages d...		Manual	Local System	
	Remote Desktop Help Sessi...	Manages a...		Manual	Local System	
	Remote Procedure Call (RPC)	Serves as t...	Started	Automatic	Network S...	
	Remote Procedure Call (RP...	Enables re...		Manual	Network S...	
	Remote Registry	Enables re...	Started	Automatic	Local Service	
	Removable Storage	Manages a...		Manual	Local System	
	Resultant Set of Policy Pro...	Enables a...		Manual	Local System	
	Routing and Remote Access	Enables mu...		Disabled	Local System	
	Secondary Logon	Enables st...	Started	Automatic	Local System	
	Security Accounts Manager	The startu...	Started	Automatic	Local System	
	Server	Supports fil...	Started	Automatic	Local System	
	Shell Hardware Detection	Provides n...	Started	Automatic	Local System	
	Smart Card	Manages a...		Manual	Local Service	
	Special Administration Cons...	Allows adm...		Manual	Local System	
	System Event Notification	Monitors s...	Started	Automatic	Local System	
	Task Scheduler	Enables a...	Started	Automatic	Local System	
	TCP/IP NetBIOS Helper	Provides s...	Started	Automatic	Local Service	
	TEC Windows Event Log A...		Started	Automatic	Local System	
	TECSNMPAdapter		Started	Automatic	Local System	
	Telephony	Provides T...	Started	Manual	Local System	
	Terminal Services	Allows user...	Started	Manual	Local System	
	Terminal Services Session ...	Enables a...		Disabled	Local System	
	Themes	Provides u...		Disabled	Local System	
	Uninterruptible Power Supply	Manages a...		Manual	Local Service	
	Upload Manager	Manages t...		Manual	Local System	
	Virtual Disk Service	Provides s...		Manual	Local System	
	VMware Tools Service	Provides s...	Started	Automatic	Local System	
	VNC Server Version 4		Started	Automatic	Local System	
	Volume Shadow Copy	Manages a...		Manual	Local System	
	WebClient	Enables Wi...		Disabled	Local Service	

Figure 7-72 Telephony Windows service is started



21. Check the event list for WEIMAR and you can see a purple event coming from the MH NT event log probe, showing that the Telephony service has started (Figure 7-73).

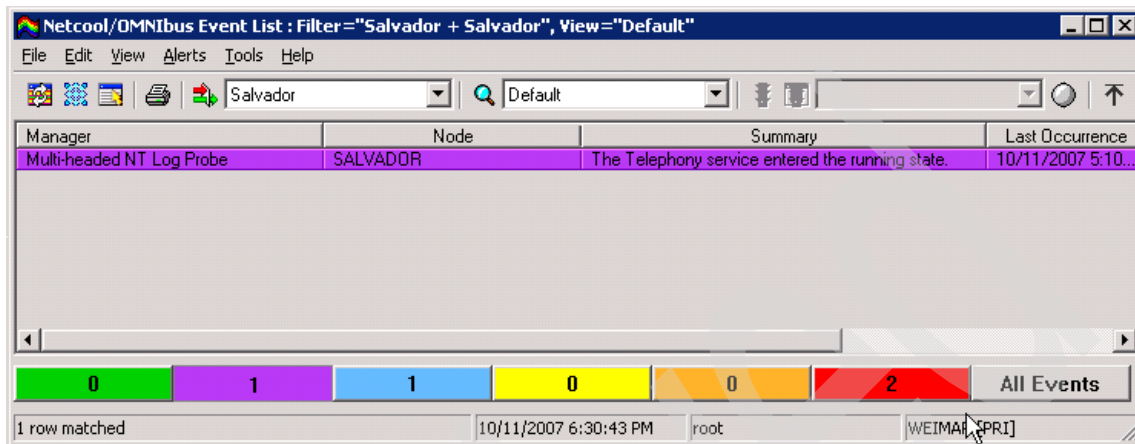


Figure 7-73 Event sent to WEIMAR by the MH NT event log probe

In the same way it is possible to execute any other external procedure or script with OMNIBus and process agent on Windows.

## 7.6 Syslog probe event configuration

Here we show that it is possible to have TEC adapters and OMNIBus probes concurrently reading from same syslog. We use named pipes, but the example will work equally for flat logfile reading, too.

This could be useful for testing or during a phased implementation.

1. Install the syslog probe.

2. Examine the existing entry in `/etc/syslog.conf` (the configuration file that controls the output of the syslog daemon). Copy the line added by the Tivoli TEC adapter and edit the entry to send messages to the named pipe that you are going to create (in this example, `/var/log/ncopipe`). We chose to send fewer messages, and so removed the warnings, notices, and information. The configuration will look something like Figure 7-74.

```
# Log anything (except mail) of level info or higher.
*.info;mail.none;authpriv.none;cron.none /var/log/messages
# The authpriv file has restricted access.
authpriv.* /var/log/secure
# Log all the mail messages in one place.
mail.* /var/log/maillog
# Log cron stuff
cron.* /var/log/cron
# Everybody gets emergency messages
*.emerg *
# Save news errors of level crit and higher in a special file.
uucp,news.crit /var/log/spooler
# Save boot messages also to boot.log
local7.* /var/log/boot.log
# Tivoli logfile adapter entry,
*.emerg;*.alert;*.crit;*.err;*.warning;*.notice;*.info
|/tmp/.tivoli/.tecad_logfile.fifo.bonn.itsc.austin.ibm.com.
# End of logfile adapter entry,
# Netcool syslog probe entry
*.emerg;*.alert;*.crit;*.err|/var/log/ncopipe
# End of Netcool syslog entry
```

Figure 7-74 `/etc/syslog.conf`

3. Create the named pipe with the command:

```
mknod /var/log/ncopipe p
```

4. Edit the probe file's properties file `syslog.props`, and add `/var/log/ncopipe` to the `FifoName` field (this is empty by default), as in Figure 7-75.

**Note:** There should be only one of `FiFoName` or `Logfile` name uncommented. When the probe starts up, if it finds a `FifoName` entry, it starts in FIFO mode.

```
#####  
#  
# Add your settings here  
#  
#####  
MessageLevel      :      "warn"  
Server            :      "WEIMAR"  
PeerHost          :      'weimar'  
Peerport          :      4100  
InactivityAlarmTime :      0  
FifoName :  "/var/log/ncopipe"  
#Logfile :  "/var/lof/ncolog"  
#####
```

Figure 7-75 `/opt/netcool/omnibus/probes/linux2x86/syslog.props`

5. Restart `syslogd`, or refresh the configuration without interrupting the service, with:  
`kill -HUP pid of syslogd`
6. Start or restart the probe to reread the properties file:  
`kill -9 PID (of nco_p_syslog)`  
`/opt/netcool/omnibus/probes/nco_p_syslog &`
7. Confirm that both TEC and OMNibus receive the messages.
8. When the TEC adapter is no longer required, the Tivoli adapter entry can be removed from the `/etc/syslog.conf`, the `syslog` daemon refreshed, and the TEC adapter retired.

## 7.7 Completed upgrade

Figure 7-76 is the OMNIBus event list after we migrated all the event sources and turned off the TEC forwarding rule. Remember that you can decide to keep the TEC adapters in place (and just change their configuration) when they cannot be replaced by OMNIBus probes.

Netcool/OMNIBus Event List : Filter="All Events", View="Default"

File Edit View Alerts Tools Help

All Events Default Top [ OFF ] 0 ?

Node	Summary	Last Occurrence	Count	Type	Agent	Manager
9.3.5.207	T*Interface Down"	10/12/2007 3:18	1	Problem	N_Tivoli NetView	tivoli_elf probe on weimar.itsc.aust...
9.3.5.208	T*Interface Down"	10/12/2007 3:18	1	Problem	N_Tivoli NetView	tivoli_elf probe on weimar.itsc.aust...
9.3.5.206	T*Node Down"	10/12/2007 3:17	1	Problem	N_Tivoli NetView	tivoli_elf probe on weimar.itsc.aust...
oslo.itsc.austin.ibm.com/LZ	[disk_space]([Disk_NameLIKE"...	10/12/2007 3:13	5	ITM Problem	N_ITM	tivoli_elf probe on weimar.itsc.aust...
waco.itsc.austin.ibm.com/LZ	[disk_space]([Disk_NameLIKE"...	10/12/2007 3:13	5	ITM Problem	N_ITM	tivoli_elf probe on weimar.itsc.aust...
	[disk_space	10/12/2007 3:09	1	ITM Resolution	N_ITM	tivoli_elf probe on weimar.itsc.aust...
oslo.itsc.austin.ibm.com/LZ	[disk_space]([Disk_NameLIKE"...	10/12/2007 3:09	2	ITM Problem	N_ITM	tivoli_elf probe on weimar.itsc.aust...
waco.itsc.austin.ibm.com/LZ	[disk_space]([Disk_NameLIKE"...	10/12/2007 3:09	2	ITM Problem	N_ITM	tivoli_elf probe on weimar.itsc.aust...
	[disk_space	10/12/2007 3:09	1	ITM Resolution	N_ITM	tivoli_elf probe on weimar.itsc.aust...
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 3:16	1	Problem	N_	ConnectionWatch
	TTEC Event Server shut down	10/12/2007 3:04	1	Problem	N_TEC	tivoli_elf probe on weimar.itsc.aust...
nottingham	UIPS_Fan_Down event	10/12/2007 2:24	1	Problem	N_Agent	nco_sq
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 2:24	1	Problem	N_	ConnectionWatch
	TTEC Event Server initialised	10/12/2007 3:05	1	Problem	N_TEC	tivoli_elf probe on weimar.itsc.aust...
bonn	[root] CMD [run-parts /etc/cron...	10/12/2007 3:01	5	Type Not Set	N_cron	Syslog
bonn	Accepted password for root from...	10/12/2007 2:36	1	Type Not Set	N_sshd	Syslog
bonn	session opened for user root by r...	10/12/2007 2:36	1	Type Not Set	N_sshd	Syslog
salvador	VA.NT Event List@090305CC pro...	10/12/2007 3:15	2	Resolution	N_	ConnectionWatch
salvador	AA.Administrator process running...	10/12/2007 3:15	1	Resolution	N_	ConnectionWatch
salvador	AA.Administrator process running...	10/12/2007 3:14	1	Problem	N_	ConnectionWatch
salvador	VA.NT Event List@090305CC pro...	10/12/2007 3:12	2	Problem	N_	ConnectionWatch
salvador	VA.NT Event List@090305CC pro...	10/12/2007 3:12	2	Resolution	N_	ConnectionWatch
salvador	VA.NT Event List@090305CC pro...	10/12/2007 3:07	2	Problem	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 2:24	1	Resolution	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 2:23	1	Problem	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 2:23	1	Resolution	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 10:5	1	Problem	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 10:5	1	Resolution	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 10:4	1	Problem	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 10:4	1	Resolution	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 10:4	1	Problem	N_	ConnectionWatch
weimar.itsc.austin.ibm.com	[A]sql process running on weima...	10/12/2007 10:4	1	Resolution	N_	ConnectionWatch
Mondorf	L.2007-10-05 15:08:28 Mondorf a...	10/12/2007 9:55	1	Problem	N_Linux	LINUX

16 4 0 4 0 9

0 rows selected 10/12/2007 4:21:34 PM root WEIMAR [PRI]

Figure 7-76 The final picture!

## 7.8 Troubleshooting the event flow

The *Netcool Administration Guide* covers many troubleshooting hints and tips about the health of the Netcool components. However, Table 7-1 provides a useful comparison for troubleshooting the event flow and comparing it to the equivalent TEC method where appropriate.

Table 7-1 Event flow troubleshooting

Question	TEC	OMNibus
Has the event left the event source?	Is the adapter or agent running?	Is the probe or monitor process running? <code>ps -ef   grep nco_p</code>
	Is the event formatted okay or filtered out? <code>tec_logfile.fmt</code> <code>tec_logfile.conf</code>	Is the matched event stream shown in the probe logfile? <code>\$OMNIHOME/log/&lt;Probe&gt;.log</code>
	Check logfile. Turn on debug on the adapter.	Turn on probe debug logging. Either start the probe with the properties parameter <code>MessageLevel : debug</code> or change the debug level of a running probe by sending a USR2 signal to the probe process ID as follows: <pre>&gt; kill -USR2 &lt;pid&gt;</pre> Each time that you issue the command above the message level is decreased to increase information.
	Check whether the cache file on the source host has entries (for example, <code>tec_logfile.cache</code> ).	Check whether the cache file on the probe source has entries, for example, <code>\$OMNIHOME/var/tivoli_eif.cache</code> .
	Check that endpoint framework communications are okay, or whether non-time to the next level are okay (that is, gateway or server).	Check the Netcool/OMNibus components with <code>\$OMNIHOME/nco_ping</code> .

Question	TEC	OMNibus
	Check the route and port in the config file (for example, tecad_logfile.conf /usr/OV/conf/tecint.conf /TEMS/tec/om_tec.conf)	Check the probe config file, for example, \$OMNIHOME/probes/aix5/tivoli_eif.props.
Has the event arrived at the gateway?	Is the gateway running or the SCE gateway active?	Is the Netcool southbound component, for example, NCO_Proxy, or an event centralizing ObjectServer up and running and receiving events?
	Check whether the cache on the gateway has entries (for example, tec_gateway_sce.cache tec_gateway.cache)	Check NCO_Proxy or ObjectServer logfiles.
	Check communications to the server. Look for firewall blocked ports or framework issues.	Check the Netcool/OMNibus components with \$OMNIHOME/nco_ping. Look for firewall blocked ports.
	Check the route and port in the config file on the gateway tec_gateway.conf.	Check the NCO_Proxy and ObjectServer configuration parameters.
		<p>Is the probe connected to the ObjectServer? To make sure it is, do one of the following options:</p> <ul style="list-style-type: none"> <li>▶ Look at the Connections view in the Administrator panel (nco_config from the command line).</li> <li>▶ Use ObjectServer SQL (nco_sql from the command line) and issue the query “select * from catalog.connections”.</li> </ul>

Question	TEC	OMNibus
Has the event arrived at the server?	Is the TEC server running correctly? /tmp/tec_* logfiles	Is the ObjectServer up and running? Check with \$OMNIHOME/nco_ping. Try to log in with \$OMNIHOME/bin/nco_sql. Check NCO_Proxy's and ObjectServer's logfiles.
	Is the database RIM okay (for example, wrimtest)?	N/A
	Is the server listening port correct? .tec_config Check whether events arrived in the reception log (for example, wtdumpri).	Is the ObjectServer running on the correct defined port? Check \$OMNIBUS/bin/nco_xigen .
Has the event been processed?	Check the TEC server configuration parameters. wlsesvrcfg	Check the ObjectServer configuration parameters \$OMNIBUS/etc/<name>.p rops. Check \$OMNIBUS/bin/nco_xigen .
	Check that events have been parsed correctly. Load the correct baroc file.	Check the RAW capture file of the probe turned on, especially in the probe configurations file. Check the probe logfile turned on, especially in debug mode.
	Check the rules cache. wtdumper	Check whether the cache file on the probe source has entries, for example, \$OMNIHOME/var/tivoli_eif .cache.

Question	TEC	OMNibus
	Trace the rules to see what is firing	Check the RAW capture file of the Probe turned on in the Probe configurations file. Check the Probe logfile turned on in debug mode. Turn on ObjectServer debug logfile mode and check the ObjectServer Trigger logfiles in \$OMNIHOME/log.
Is the event hidden?	Check the console event group views. Check the console filters.	Check the active event lists in the ObjectServer native GUI and their configured views and filters. If the event can be seen with the ObjectServer SQL (nco_sql from the command line) but not in the event list, check filters and restriction filters that have been applied to the user/group.



## Appendixes

In Appendix A, “Lab configuration” on page 367, we provide some lab configuration examples that may be helpful as references to follow in actual upgrade scenarios.

In Appendix B, “Additional material” on page 399, we provide information about how to access additional online materials that are available as part of this IBM Redbooks publication.

Archived

# Lab configuration

In this appendix we provide example content to better demonstrate various installation and configuration steps. The appendix topics include:

- ▶ “TEC installation steps” on page 368
- ▶ “TEC event source generation commands and scripts” on page 372
- ▶ “Netcool/OMNIbus directory structure reference” on page 374
- ▶ “IBM Tivoli Netcool default port usage” on page 375
- ▶ “User profile” on page 377
- ▶ “Netcool Process Automation startup script” on page 377
- ▶ “ObjectServer WEIMAR\_PA Process Automation configuration” on page 379
- ▶ “ObjectServer WEIMAR probe nco\_p\_tivoli\_eif rules (TEC, NetView)” on page 382
- ▶ “ObjectServer WEIMAR probe nco\_p\_tivoli\_eif rules (TEC, NetView, ITM)” on page 388

## TEC installation steps

To install:

1. Host name resolution tables were set up in DNS with fully qualified host names configured. On the TMR server we added a single entry to `/etc/hosts` with the fully qualified and the short host name.
2. Framework 4.1.1 was installed and patched as the root user.
3. DB2 8.1 FP07 was installed with DB2 user `db2inst1`, group `db2admin`.
4. The framework RIM was configured as shown in Figure A-1.

RIM Host:	nottingham
RDBMS User:	db2inst1
RDBMS Vendor:	DB2
Database ID:	tec
Database Home:	/usr/opt/db2_08_01
Server ID:	tcpip
Instance Home:	/home/db2inst1
Instance Name:	db2inst1

*Figure A-1 RIM configuration*

5. TEC 3.9 was installed and patched to FP06 (Figure A-2).

```
Product List
*-----*
Tivoli Management Framework 4.1.1
IBM Tivoli Enterprise Console Adapter Configuration Facility 3.9
IBM Tivoli Enterprise Console JRE 3.9
IBM Tivoli Enterprise Console Sample Event Information 3.9
IBM Tivoli Enterprise Console Console 3.9
IBM Tivoli Enterprise Console Server 3.9
IBM Tivoli Enterprise Console User Interface Server 3.9
*-----*

Patch List
*-----*
3.9.0 Tivoli Enterprise Console ACF Fix Pack 6
3.9.0 Tivoli Enterprise Console Console Fix Pack 6
3.9.0 Tivoli Enterprise Console Sample Event Information Fix Pack 6)
3.9.0 Tivoli Enterprise Console JRE Fix Pack 6
3.9.0 Tivoli Enterprise Console Server Fix Pack 6
3.9.0 Tivoli Enterprise Console User Interface Server Fix Pack 6
Tivoli Framework Patch 4.1.1-LCF-0049 (build 04/12)
Tivoli Framework Patch 4.1.1-TMF-0090 (build 05/17)
```

Figure A-2 TMR server wlsinst output

6. TEC and framework operators created ops\_uk, ops\_br, ops\_it, and ops\_de.

7. TEC rules used initially are shown in Figure A-3.

```
rule_set('forward', 'forward.rls', active).
rule_set('netview', 'netview.rls', active).
rule_set('maintenance_mode', 'maintenance_mode.rls', active).
rule_set('dependency', 'dependency.rls', active).
rule_set('ebusiness', 'ebusiness.rls', active).
rule_set('heartbeat', 'heartbeat.rls', active).
rule_set('cleanup', 'cleanup.rls', active).
rule_set('event_activity', 'event_activity.rls', inactive).
rule_set('forwarding', 'forwarding.rls', inactive).
rule_set('correlation', 'correlation.rls', inactive).
rule_set('db_cleanup', 'db_cleanup.rls', inactive).
rule_set('escalate', 'escalate.rls', inactive).
rule_set('notify', 'notify.rls', inactive).
rule_set('troubleshoot', 'troubleshoot.rls', inactive).
```

Figure A-3 TEC rule sets

8. TEC barocs, mainly the addition of ITM 5.1.1, ITM 6.1, and ITM 6.2 baroc files. See Figure A-4.

```
root.baroc
tec.baroc
omegamon.baroc
Tmw2k.baroc
hb_events.baroc
ebusiness.baroc
netview.baroc
dependency.baroc
tecad_logfile.baroc
tecad_nt.baroc
tecad_snmp.baroc
ups.baroc
kib.baroc
klz.baroc
knt.baroc
kul.baroc
kux.baroc
Sentry.baroc
ups.baroc
universal.baroc
DMXFileSystem.baroc
DMXMemory.baroc
DMXCpu.baroc
TMW_LogicalDisk.baroc
TMW_MemoryModel.baroc
TMW_Processor.baroc
```

*Figure A-4 TEC BAROC files*

9. The managed node and gateway were created on host mondorf.itso.austin.ibm.com.
10. Endpoints were created on the TMR, gateway, and host bonn.itso.austin.ibm.com.
11. The non-TME TEC adapter tecad\_win was installed on Windows2003 server salvador.itso.austin.ibm.com.
12. ACP adapter profiles were created as the default to send all events (filtermode=out).

13. The Gateway\_SCE profile was modified with additional lines, as shown in Figure A-5.

```
ServerLocation=@EventServer
GatewaySendInterval=5
GatewayQueueSize=40000
BufEvtPath=/etc/Tivoli/tec/tec_gateway_sce.cache
Pre37Server=no
UseStateCorrelation=YES
gwr_Enable=YES
SendEventPort=5561
ReceiveAckPort=5562
ReceiveEventPort=5563|
SendAckPort=5564
StateCorrelationConfigURL=file:/etc/Tivoli/tec/tecroot.xml
```

Figure A-5 SCE configuration

14. ITM 6.1 and ITM 6.2 forwarding to TEC was set up on the systems cairo.itso.austin.ibm.com and server2.itso.austin.ibm.com (the two hub ITM TEMS systems). On both systems the file /opt/IBM/ITM/tables/HUB\_TEMS/TECLIB/om\_tec.config was configured to forward to the TEC server, as in Figure A-6.

```
#this variable points to the T/EC host where we forward events
ServerLocation=nottingham
ServerPort=0
#EventMaxSize=4096
RetryInterval=5
getport_total_timeout_usec=50500
NO_UTF8_CONVERSION=YES
ConnectionMode=co
BufferEvents=YES
BufEvtMaxSize=4096
BufEvtPath=./TECLIB/om_tec.cache
FilterMode=OUT
Filter:Class=ITM_Generic;master_reset_flag='';
```

Figure A-6 om\_tec.config

Additionally, with ITM 6.2 there is a new feature to allow the forwarding of events to more than one EIF receiver simultaneously, so that could be any combination of TECs or Netcool EIF probes. It also allows specific ITM situations to send to one EIF and others to another EIF. These destinations are created with the ITM command:

```
tacmd createeventdest
```

For a full description of this command see the *ITM6.2 Installation and Deployment guide* and *ITM 6.2 Administration guides*.

This produces a configuration file per EIF receiver, as in Figure A-7

```
ServerLocation=nottingham
ServerPort=0
RetryInterval=5
getport_total_timeout_usec=50500
NO_UTF8_CONVERSION=YES
ConnectionMode=co
BufEvtMaxSize=4096
BufferEvents=YES
BufEvtPath=./TECLIB/evtdst001.cache
```

Figure A-7 *evtdst001.config*

15. NetView to TEC forwarding was set up via `/usr/OV/conf/tecint.conf`.

```
ServerLocation=9.3.5.205
TecRuleName=forwardall.rs
ServerPort=0
DefaultEventClass=TEC_ITS_BASE
BufferEvents=YES
BufEvtPath=/usr/OV/log/tec_cache
UseStateCorrelation=YES
StateCorrelationConfigURL=file:///usr/OV/conf/nvsbcrule.xml
```

Figure A-8 *NetView-TEC forwarding configuration*

16. Finally, all event sources were tested.

## TEC event source generation commands and scripts

After setting up the TEC environment and the different event sources, a random sample of genuine events and test scripts were allowed to flow into to TEC. We then took a snapshot of this via the `wtdumpr1` command. This resulted in 565 events being collected, which were then replayed again to create 1130 events. The following commands were then used to replay the events in a controlled manner:

```
cd $BINDIR/TME/TEC/contrib
wtdumpr1 | ./ParseEvents.pl -d /Tivoli/parse1909
wtdbc clear -left 0
./SendEvents.pl -d /Tivoli/parse1909 -t0
```



NB: -t0 sends all events immediately, -t1 gives a 1-second pause, and no -t flag replays them at the original time interval.

Additionally, a script to run multiple copies of postzmsg was used. This particular example sends 100 messages to the gateway receiver on the managed node. It is worth noting that postzmsg is required, rather than postmsg, to be able to select a port number, without the need for a configuration file.

```
#!/bin/ksh
COUNT=1
while [ $COUNT -lt 101 ]
do
/Tivoli/lcf/bin/aix4-r1/bin/postzmsg -S mondorf -p 5539 -m
"sendevent $COUNT `date +%T`" -r CRITICAL hostname=bonn TEC_Error LOGFILE
COUNT=`expr $COUNT + 1`
done
```

*Figure A-9 Script to send 100 postzmsgs to TEC*

This script was used to direct messages to an OMNibus EIF probe as well as a TEC gateway or server component by changing the -S and -p flags, for example, to send to an EIF probe (where bari is the ObjectServer and 9999 the listening port).

```
#!/bin/ksh
COUNT=1
while [ $COUNT -lt 101 ]
do
/Tivoli/lcf/bin/aix4-r1/bin/postzmsg -S bari -p 9999 -m "sendevent
$COUNT `date +%T`" -r CRITICAL hostname=bonn TEC_Error LOGFILE
COUNT=`expr $COUNT + 1`
done
```

*Figure A-10 Script to send 100 postzmsgs to EIF probe*

We also performed some limited, controlled stress tests, sending significant event flows between the TEC and OMNibus servers and direct from event sources to OMNibus. The events used were a mixture of genuine and simulated events from the installed adapters, **postzmsg** commands, ITM, and NetView. The results were discussed in Chapter 7, “Configuring the event sources” on page 277.

# Netcool/OMNIBus directory structure reference

Figure A-11 shows the directory structure of the deployed Netcool/OMNIBus environment in the default configuration.

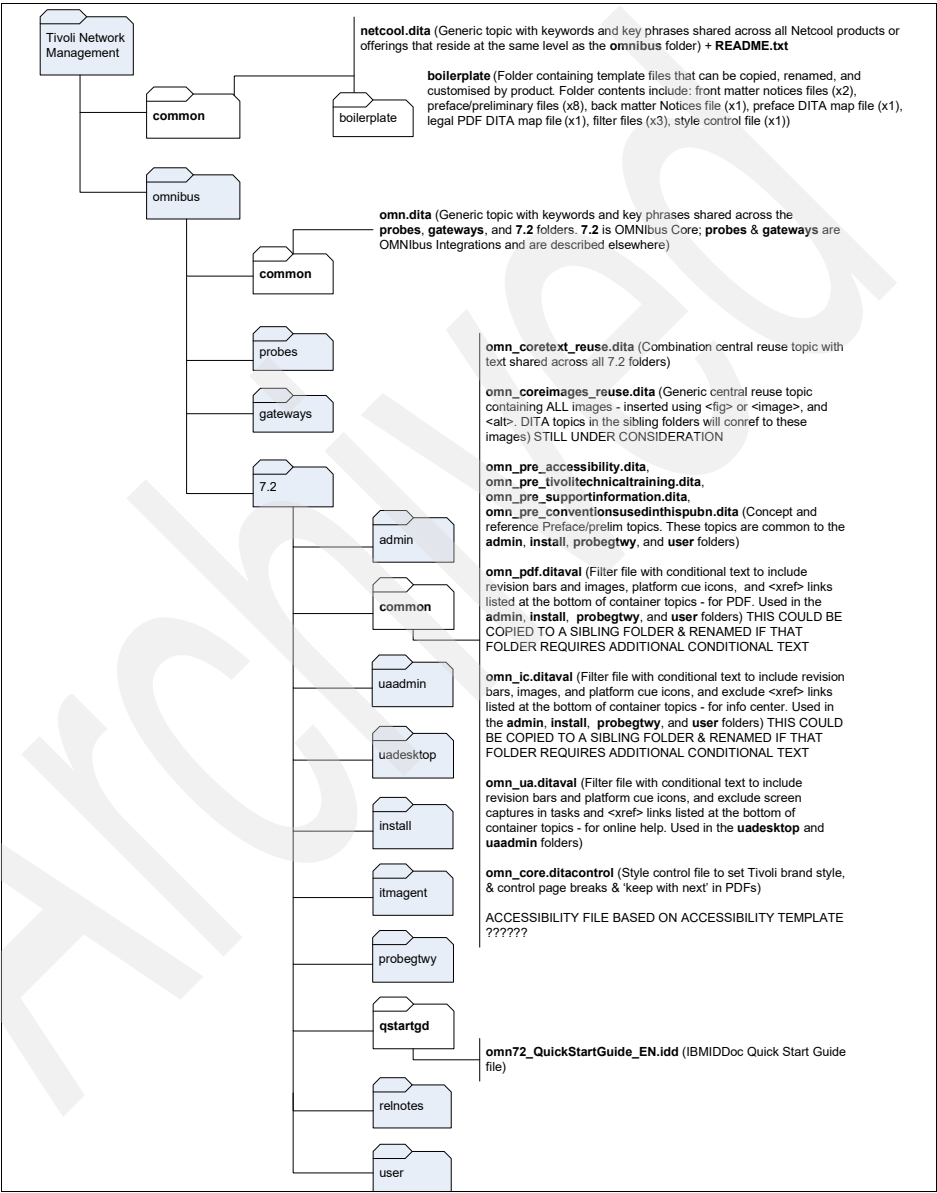


Figure A-11 Netcool/OMNIBus directory structure

# IBM Tivoli Netcool default port usage

This list has been created to simplify a Netcool installation and to assist during the deployment phase, help in order to avoid communication port conflicts, and to get an overview of used ports by the current and commonly used Netcool products.

## IBM Tivoli Netcool/OMNibus

Table A-1 shows default port usage of IBM Tivoli Netcool/OMNibus used in our lab environment.

Table A-1 Default port usage: OMNibus

Component	Default port	Additional information/port configuration
ObjectServer (NCOMS)	4100	These default port numbers are defined in the server editor, but are configurable and rarely used with the default values. Amend the port numbers as necessary, and then save your changes, as described in Customizing Server Definitions in the server editor.
Process control agent (NCO_PA)	4200	
Gateway server	4300	
Proxy server	4400	On UNIX systems that do not have a graphical interface, you can amend the port numbers by editing the \$NCHOME/etc/omni.dat file.
IDUC	Variable value (6969 - UDP)	The operating system supplies the port number for the Insert Delete Update Control (IDUC) communication channel.

## IBM Tivoli Netcool Security Manager

Table A-2 shows default port usage of IBM Tivoli Netcool Security Manager used in our lab environment.

Table A-2 Default port usage: NSM

Component	Default port	Additional information/port configuration
Server port	1275	Security Manager Server port
Server DB port	5600	Security Manager database port
Server HTTP port	8077	Security Manager Web port

## IBM Tivoli Netcool/Webtop

Table A-3 shows default port usage of IBM Tivoli Netcool/Webtop used in our lab environment.

*Table A-3 Default port usage: Netcool/Webtop*

Component	Default port	Additional information/port configuration
Netcool WebTop WAAPISecure	4433	The port number of the secure Netcool/Webtop server. Webtop Administration Application Program Interface (WAAPIS).
Netcool/Webtop	8080	The port number of the Netcool/Webtop server.

## IBM Tivoli Netcool probes

Table A-4 shows default port usage of IBM Tivoli Netcool probes used in our lab environment.

*Table A-4 Default port usage: probes*

Component	Default port	Additional information/port configuration
nco_p_glf	5555	Communication port between master and subordinate
	1122	Listening port
	9999	Peer port
nco_p_syslog	5555	Communication port between master and subordinate
	1122	Listening port
	9999	Peer port
nco_p_nonnative	No port	No port
nco_p_tivoli_eif	5555	Communication between master and subordinate
	9999	Listening port
	9999	Peer port

## User profile

Example A-1 shows the profile of the user Netcool as a reference of the lab environment.

*Example: A-1 AIX /home/netcool/.profile*

---

```
# more .profile

export
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/opt/netcool/platform/aix5/jre_
1.5.4/jre/bin:/opt/netcool/bin:/opt/netcool/omnibus/bin:/usr/bin/X11:/sbin:/opt
/netcool
:/opt/netcool/omnibus:/opt/netcool/omnibus/probes:.
export NCHOME=/opt/netcool
export OMNIHOME=/opt/netcool/omnibus

export DISPLAY=~echo $SSH_CLIENT | awk '{print $1}':0.0
export PS1="\u@\H:\w$ "

if [ -s "$MAIL" ]          # This is at Shell startup. In normal
then echo "$MAILMSG"      # operation, the Shell checks
fi                        # periodically.

cd /opt/netcool

bash
```

---

## Netcool Process Automation startup script

The rc.nco\_pa startup script starts the Process Automation daemon automatically after a system startup. Then the Process Automation daemon starts all configured processes and services and controls the availability of them.

Example A-2 shows the “rc.nco\_pa” startup script in the “/etc/inittab” file that we used in our lab environment.

*Example: A-2 AIX /etc/rc.nco\_pa*

---

```
#!/bin/bsh
# Start the Netcool/OMNibus Process Agent
#
# Add Netcool related environment variables here
NCHOME=/opt/netcool
OMNIHOME=/opt/netcool/omnibus
```

```

# End of Netcool environment variables
#

NCO_PA="BARI_PA"
START_NCO="Y"
SECURE=n
NETCOOL_LICENSE_FILE=270000@localhost
PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:/home/netcool/bin:/opt/netcool/platform/a
ix5/jre_1.5.4/jre/bin:/opt/netcool/bin:/opt/netcool/omnibus/bin:/sbin:/opt/netc
ool/omnibus/pro
bes:/usr/bin/X11:.

export NCHOME OMNIHOME NETCOOL_LICENSE_FILE PATH
case "$START_NCO" in
    'Y')
        if [ -x $OMNIHOME/bin/nco_pad ]; then
            grep ${NCO_PA} $NCHOME/etc/omni.dat >/dev/null
2>/dev/null
            if [ $? -eq 0 ]; then
                echo "Netcool/OMNibus startup : Starting
Process Control..."
                if [ $SECURE = "Y" ]; then
                    $OMNIHOME/bin/nco_pad -name ${NCO_PA}
                    -secure -configfile $OMNIHOME/etc/BARI_PA.conf &
                else
                    $OMNIHOME/bin/nco_pad -name ${NCO_PA}
                    -configfile $OMNIHOME/etc/BARI_PA.conf &
                fi
            else
                echo "Netcool/OMNibus startup : Process
Control not configured"
            fi
        else
            echo "Netcool/OMNibus startup : nco_pad not
executable"
        fi
    ;;
    'N')
        echo "Netcool/OMNibus startup : Process Control not starting"
    ;;
    *)
        echo "Netcool/OMNibus startup : START_NCO incorrectly set"
    ;;
esac

# END OF Netcool/OMNibus ADDITIONS

```

---

# ObjectServer WEIMAR\_PA Process Automation configuration

Example A-3 shows the properties file of the Process Automation called “WEIMAR\_PA” on the AIX lab system with host name “weimar”.

**Note:** The contents of the file shown in Example A-3 are included in the additional materials downloadable file for this book. See Appendix B, “Additional material” on page 399.

*Example: A-3 \$OMNIHOME/etc/WEIMAR\_PA.conf*

```
#NCO_PA3
#
# Process Agent Daemon Configuration File 1.1
#
# Ident: $Id: nco_pa.conf 1.3 2002/05/21 15:28:10 renate Development $
#

#
# List of processes
#
nco_process 'MasterObjectServer'
{
    Command '$OMNIHOME/bin/nco_objserv -name WEIMAR -messagelevel debug
-pa WEIMAR_PA -propsfile $OMNIHOME/etc/WEIMAR.props' run as 501
    Host = 'weimar'
    Managed = True
    RestartMsg = '${NAME} running as ${EUID} has been restored
on ${HOST}.'
    AlertMsg = '${NAME} running as ${EUID} has died on
${HOST}.'
    RetryCount = 0
    ProcessType = PaPA_AWARE
}

nco_process 'SecurityManager'
{
    Command '$NCHOME/security/bin/ncsm_server' run as 501
    Host = 'weimar'
    Managed = True
    RestartMsg = '${NAME} running as ${EUID} has been restored
on ${HOST}.'
    AlertMsg = '${NAME} running as ${EUID} has died on
${HOST}.'
    RetryCount = 0
```

```

        ProcessType      =      PaPA_AWARE
    }

    nco_process 'nco_p_tivoli_eif'
    {
        Command '$OMNIHOME/probes/nco_p_tivoli_eif -propsfile
$OMNIHOME/probes/linux2x86/tivoli_eif.props' run as 501

        Host              =      'weimar'
        Managed            =      True
        RestartMsg         =      '${NAME} running as ${EUID} has been restored
on ${HOST}.'
        AlertMsg          =      '${NAME} running as ${EUID} has died on
${HOST}.'
        RetryCount        =      0
        ProcessType       =      PaPA_AWARE
    }

    nco_process 'nco_p_syslog'
    {
        Command '$OMNIHOME/probes/nco_p_syslog -propsfile
$OMNIHOME/probes/linux2x86/syslog.props' run as 501

        Host              =      'weimar'
        Managed            =      True
        RestartMsg         =      '${NAME} running as ${EUID} has been restored
on ${HOST}.'
        AlertMsg          =      '${NAME} running as ${EUID} has died on
${HOST}.'
        RetryCount        =      0
        ProcessType       =      PaPA_AWARE
    }

    nco_process 'nco_p_glf'
    {
        Command '$OMNIHOME/probes/nco_p_glf -propsfile
$OMNIHOME/probes/linux2x86/glef.props' run as 501

        Host              =      'weimar'
        Managed            =      True
        RestartMsg         =      '${NAME} running as ${EUID} has been restored
on ${HOST}.'
        AlertMsg          =      '${NAME} running as ${EUID} has died on
${HOST}.'
        RetryCount        =      0
        ProcessType       =      PaPA_AWARE
    }

```



```

#
# List of Services
#
nco_service 'Core'
{
    ServiceType      =      Master
    ServiceStart     =      Auto
    process 'MasterObjectServer' NONE
    process 'SecurityManager' 5
    process 'nco_p_tivoli_eif' 5
    process 'nco_p_syslog' 5
    process 'nco_p_glf' 5
}

#
# This service should be used to store processes that you want to temporarily
# disable. Do not change the ServiceType or ServiceStart settings of this
# process.
#
nco_service 'InactiveProcesses'
{
    ServiceType      =      Non-Master
    ServiceStart     =      Non-Auto
}

#
# ROUTING TABLE
#
# 'user'      - (optional) only required for secure mode PAD on target host
#              'user' must be member of UNIX group 'ncoadmin'
# 'password'  - (optional) only required for secure mode PAD on target host
#              use nco_pa_crypt to encrypt.
nco_routing
{
    host 'weimar' 'WEIMAR_PA'
    host 'bari' 'BARI_PA'
    host 'salvador' 'SALVADOR_PA'
}

```

---

## ObjectServer WEIMAR probe nco\_p\_tivoli\_eif rules (TEC, NetView)

Example A-4 shows the rules file of the ObjectServer WEIMAR probe “nco\_p\_tivoli\_eif” on the Red Hat lab system considering NetView and TEC.

**Note:** The contents of the file shown in Example A-4 are included in the additional materials downloadable file for this book. See Appendix B, “Additional material” on page 399.

*Example: A-4 \$OMNIHOME/probes/linux2x86/tivoli\_eif.rules*

```
# -----
# --
# -- Copyright (C) 1994 - 2006, Micromuse, Ltd. (an IBM Company)
# -- 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: $Id$
# --
# -----
#####
# This rulesfile has been developed in accordance to the IBM Netcool
# Rules Files Best Practices to perform the following functionality
#
# 1. De-duplicate events
# 2. Generic-Clear to correlate Problems/Resolutions
# 3. Readable alarm summaries
#####
# Available elements:
# $ClassName - Class name of the TEC event
# $EventSeqNo - Event sequence number of this event
# All other elements are dynamically created, based on the name/value
# pairs in the event.
#####

# Lookup table "status, severity, type"
```

```

# Lookup table valid for snmpstatus,nodestatus,servicestatus,
# and routerstatus
table Status1 = {
    { "1", "1", "2" },
    { "2", "5", "1" },
    { "3", "3", "1" },
    { "4", "2", "1" },
    { "THRESHOLD_EXCEEDED", "1", "2" },
    { "REARMED", "5", "1" },
    { "UP", "1", "2" },
    { "DOWN", "5", "1" },
    { "MARGINAL", "3", "1" },
    { "UNREACHABLE", "2", "1" },
}

# Lookup table "status, severity, type"
# Lookup table valid for reachability and isdnstatus
table Status2 = {
    { "1", "3", "1" },
    { "2", "1", "2" },
    { "UNREACHABLE", "3", "1" },
    { "REACHABLE_AGAIN", "1", "2" },
    { "ACTIVE", "3", "1" },
    { "DORMANT", "1", "2" },
}

# Lookup table "status, severity, type"
# Lookup table valid for ifstatus
table Status3 = {
    { "1", "1", "2" },
    { "2", "5", "1" },
    { "3", "1", "2" },
    { "4", "2", "1" },
    { "UP", "1", "2" },
    { "DOWN", "5", "1" },
    { "ADMIN_DOWN", "1", "2" },
    { "UNREACHABLE", "2", "1" },
}

# Lookup table "status, severity, type"
# Lookup table valid for pixfailstatus
table Status4 = {
    { "FAILOVER", "3", "1" },
    { "RECOVERED", "1", "2" },
}

array servers;

```

```

array server_info;
array server_detail;

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
{
    @Manager = "tivoli_eif probe on " + hostname()
    @AlertGroup = $ClassName
    @Class = 6601
    @Agent = $source
    @Type = 1
    @Grade = "1"

    if(exists($msg))
    {
        if(regmatch($msg, "^'.*'$"))
        {
            $msg = extract($msg, "^'(.*)'$")
        }
    }

    if(exists($date))
    {
        if(regmatch($date, "^'.*'$"))
        {
            $date = extract($date, "^'(.*)'$")
        }
    }

    if(exists($hostname))
    {
        @Identifier = $hostname
    }
}

```

```

}

if(exists($source))
{
    @AlertKey = $source
    @Identifier = @Identifier + ":" + $source
}

if(exists($sub_source))
{
    @AlertKey = @AlertKey + ":" + $sub_source
    @Identifier = @Identifier + ":" + $sub_source
}

if(exists($sub_origin))
{
    @AlertKey = @AlertKey + ":" + $sub_origin
    @Identifier = @Identifier + ":" + $sub_origin
}

if( exists( $severity ) )
{
    switch ( $severity )
    {
        case "FATAL" :
            @Severity = 5
        case "60":
            @Severity = 5
        case "CRITICAL":
            @Severity = 5
        case "50":
            @Severity = 5
        case "MINOR":
            @Severity = 3
        case "40":
            @Severity = 3
        case "WARNING":
            @Severity = 2
        case "30":
            @Severity = 2
        default:
            @Severity = 1
    }
}

if(exists($origin))
{
    @Node = $origin
    @NodeAlias = $origin
}

```

```

    }

    @Identifier = @Identifier + ":" + $ClassName

    if(exists ($server_path))
    {
        $num_servers = split($server_path, servers, ",")
        $num_detail = split(servers[$num_servers], server_detail, "")
        $num_info = split(server_detail[int($num_detail)-1],
server_info, " ")
        @TECServerHandle=server_info[2]
        @TECDateReception = server_info[3]
        @TECEventHandle=server_info[4]
    }

    @Summary = $msg
    @TECHostname = $hostname
    @TECFQHostname = $fqhostname
    @TECDate = $date
    @TECRepeatCount = $repeat_count
    @LastOccurrence = getdate
    @FirstOccurrence = getdate
    @TECStatus = $status

    #
    # Handle TEC event status
    #

    switch ($status)
    {
        CASE "CLOSED":
            @Type = 2
            @Severity = 0
        CASE "30":
            @Type = 2
            @Severity = 0
        CASE "ACK":
            @Acknowledged = 1
        CASE "20":
            @Acknowledged = 1
        default:
    }

    ###
    ### Tivoli NetView Events
    ###

```

```

if ( match($source, "NV6K" ) OR match($source,"nvserverd"))
{
    @Agent = "Tivoli NetView"
    switch ($ClassName)
    {
    case "TEC_ITS_SNMPCOLLECT_THRESHOLD":
        [@Severity, @Type] = lookup($snmpstatus, Status1)
        update(@Severity)
        update(@Type)
    case "TEC_ITS_NODE_STATUS":
        [@Severity, @Type] = lookup($nodestatus, Status1)
        update(@Severity)
        update(@Type)
    case "TEC_ITS_SERVICE_STATUS":
        [@Severity, @Type] = lookup($servicestatus, Status1)
        update(@Severity)
        update(@Type)
    case "TEC_ITS_ROUTER_STATUS":
        [@Severity, @Type] = lookup($routerstatus, Status1)
        update(@Severity)
        update(@Type)
    case "TEC_ITS_INTERFACE_STATUS":
        [@Severity, @Type] = lookup($ifstatus, Status3)
        update(@Severity)
        update(@Type)
    case "TEC_ITS_SUBNET_CONNECTIVITY":
        [@Severity, @Type] = lookup($reachability, Status2)
        update(@Severity)
        update(@Type)
    case "TEC_ITS_ISDN_STATUS":
        [@Severity, @Type] = lookup($isdnstatus, Status2)
        update(@Severity)
        update(@Type)
    case "TEC_ITS_PIXFAIL_STATUS":
        [@Severity, @Type] = lookup($pix_state, Status4)
        update(@Severity)
        update(@Type)
    default:
        discard
    }
}
}

```

---

## ObjectServer WEIMAR probe nco\_p\_tivoli\_eif rules (TEC, NetView, ITM)

Example A-5 shows the rules file of the ObjectServer WEIMAR probe “nco\_p\_tivoli\_eif” on the Linux-based lab system.

**Note:** The contents of the file shown in Example A-5 are included in the additional materials downloadable file for this book. See Appendix B, “Additional material” on page 399.

*Example: A-5 \$OMINHOME/probes/linux2x86/tivoli\_eif.rules*

```
# -----
# --
# -- Copyright (C) 1994 - 2006, Micromuse, Ltd. (an IBM Company)
# -- 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: $Id$
# --
# -----
#####
# This rulesfile has been developed in accordance to the IBM Netcool
# Rules Files Best Practices to perform the following functionality
#
# 1. De-duplicate events
# 2. Generic-Clear to correlate Problems/Resolutions
# 3. Readable alarm summaries
#####
# Available elements:
# $ClassName - Class name of the TEC event
# $EventSeqNo - Event sequence number of this event
# All other elements are dynamically created, based on the name/value
# pairs in the event.
#####

# Lookup table "status, severity, type"
# Lookup table valid for snmpstatus,nodestatus,servicestatus,
```



```

# and routerstatus
table Status1 = {
    { "1", "1", "2" },
    { "2", "5", "1" },
    { "3", "3", "1" },
    { "4", "2", "1" },
    { "THRESHOLD_EXCEEDED", "1", "2" },
    { "REARMED", "5", "1" },
    { "UP", "1", "2" },
    { "DOWN", "5", "1" },
    { "MARGINAL", "3", "1" },
    { "UNREACHABLE", "2", "1" },
}

# Lookup table "status, severity, type"
# Lookup table valid for reachability and isdnstatus
table Status2 = {
    { "1", "3", "1" },
    { "2", "1", "2" },
    { "UNREACHABLE", "3", "1" },
    { "REACHABLE_AGAIN", "1", "2" },
    { "ACTIVE", "3", "1" },
    { "DORMANT", "1", "2" },
}

# Lookup table "status, severity, type"
# Lookup table valid for ifstatus
table Status3 = {
    { "1", "1", "2" },
    { "2", "5", "1" },
    { "3", "1", "2" },
    { "4", "2", "1" },
    { "UP", "1", "2" },
    { "DOWN", "5", "1" },
    { "ADMIN_DOWN", "1", "2" },
    { "UNREACHABLE", "2", "1" },
}

# Lookup table "status, severity, type"
# Lookup table valid for pixfailstatus
table Status4 = {
    { "FAILOVER", "3", "1" },
    { "RECOVERED", "1", "2" },
}

array servers;
array server_info;

```

```

array server_detail;

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
{
    @Manager = "tivoli_eif probe on " + hostname()
    @AlertGroup = $ClassName
    @Class = 6601
    @Agent = $source
    @Type = 1
    @Grade = "1"

    if(exists($msg))
    {
        if(regmatch($msg, "^'.*'$"))
        {
            $msg = extract($msg, "^'(.*)'$")
        }
    }

    if(exists($date))
    {
        if(regmatch($date, "^'.*'$"))
        {
            $date = extract($date, "^'(.*)'$")
        }
    }

    #Part taken from ITM rules file

    if(exists($situation_name))
    {

```

```

        if(regmatch($situation_name, "^'.*'$"))
        {
            $situation_name = extract($situation_name, "^'(.*?)'$")
        }
    }

    if(exists($situation_origin))
    {
        if(regmatch($situation_origin, "^'.*'$"))
        {
            $situation_origin = extract($situation_origin,
            "^'(.*?)'$")
        }
    }

    if(exists($situation_displayitem))
    {
        if(regmatch($situation_displayitem, "^'.*'$"))
        {
            $situation_displayitem =
            extract($situation_displayitem, "^'(.*?)'$")
        }
    }

    if(exists($source))
    {
        if(regmatch($source, "^'.*'$"))
        {
            $source = extract($source, "^'(.*?)'$")
        }
    }

    if(exists($situation_status))
    {
        if(regmatch($situation_status, "^'.*'$"))
        {
            $situation_status = extract($situation_status,
            "^'(.*?)'$")
        }
    }

    if(exists($integration_type))
    {
        if(regmatch($integration_type, "^'.*'$"))
        {
            $integration_type = extract($integration_type,
            "^'(.*?)'$")
        }
    }
}

```

```

if(exists($situation_time))
{
    if(regmatch($situation_time, "^'.*'$"))
    {
        $situation_time = extract($situation_time, "^'(.*)'$")
    }
}

if(exists($situation_eventdata))
{
    if(regmatch($situation_eventdata, "^'.*'$"))
    {
        $situation_eventdata = extract($situation_eventdata,
"'^'(.*)'$")
    }
}

if(exists($situation_type))
{
    if(regmatch($situation_type, "^'.*'$"))
    {
        $situation_type = extract($situation_type, "^'(.*)'$")
    }
}

if(exists($situation_thrunode))
{
    if(regmatch($situation_thrunode, "^'.*'$"))
    {
        $situation_thrunode = extract($situation_thrunode,
"'^'(.*)'$")
    }
}

if(exists($cms_hostname))
{
    if(regmatch($cms_hostname, "^'.*'$"))
    {
        $cms_hostname = extract($cms_hostname, "^'(.*)'$")
    }
}

if(exists($cms_port))
{
    if(regmatch($cms_port, "^'.*'$"))
    {
        $cms_port = extract($cms_port, "^'(.*)'$")
    }
}

```

```

    }

    if(exists($master_reset_flag))
    {
        if(regmatch($master_reset_flag, "^'.*'$"))
        {
            $master_reset_flag = extract($master_reset_flag,
"^(.*)'$")
        }
    }

    if(exists($hostname))
    {
        if(regmatch($hostname, "^'.*'$"))
        {
            $hostname = extract($hostname, "^'(.*)'$")
        }
    }

    if(exists($fqhostname))
    {
        if(regmatch($fqhostname, "^'.*'$"))
        {
            $fqhostname = extract($fqhostname, "^'(.*)'$")
        }
    }

    if(exists($repeat_count))
    {
        if(regmatch($repeat_count, "^'.*'$"))
        {
            $repeat_count = extract($repeat_count, "^'(.*)'$")
        }
    }

    if(exists($severity))
    {
        if(regmatch($severity, "^'.*'$"))
        {
            $severity = extract($severity, "^'(.*)'$")
        }
    }

    if(exists($hostname))
    {
        @Identifier = $hostname
    }

    if(exists($source))

```

```

{
    @AlertKey = $source
    @Identifier = @Identifier + ":" + $source
}

if(exists($sub_source))
{
    @AlertKey = @AlertKey + ":" + $sub_source
    @Identifier = @Identifier + ":" + $sub_source
}

if(exists($sub_origin))
{
    @AlertKey = @AlertKey + ":" + $sub_origin
    @Identifier = @Identifier + ":" + $sub_origin
}

if( exists( $severity ) )
{
    switch ( $severity )
    {
        case "FATAL" :
            @Severity = 5
        case "60":
            @Severity = 5
        case "CRITICAL":
            @Severity = 5
        case "50":
            @Severity = 5
        case "MINOR":
            @Severity = 3
        case "40":
            @Severity = 3
        case "WARNING":
            @Severity = 2
        case "30":
            @Severity = 2
        default:
            @Severity = 1
    }
}

if(exists($origin))
{
    @Node = $origin
    @NodeAlias = $origin
}

@Identifier = @Identifier + ":" + $ClassName

```

```

if(exists ($server_path))
{
    $num_servers = split($server_path, servers, ",")
    $num_detail = split(servers[$num_servers], server_detail, "")
    $num_info = split(server_detail[int($num_detail)-1],
server_info, " ")
    @TECServerHandle=server_info[2]
    @TECDateReception = server_info[3]
    @TECEventHandle=server_info[4]
}

@Summary = $msg
@TECHostname = $hostname
@TECFQHostname = $fqhostname
@TECDate = $date
@TECRepeatCount = $repeat_count
@LastOccurrence = getdate
@FirstOccurrence = getdate
@TECStatus = $status

#
# Handle TEC event status
#

switch ($status)
{
    CASE "CLOSED":
        @Type = 2
        @Severity = 0
    CASE "30":
        @Type = 2
        @Severity = 0
    CASE "ACK":
        @Acknowledged = 1
    CASE "20":
        @Acknowledged = 1
    default:
}

###
### Tivoli NetView Events
###
if ( match($source, "NV6K" ) OR match($source,"nvserverd"))
{
    @Agent = "Tivoli NetView"
}

```

```

        switch ($ClassName)
        {
        case "TEC_ITS_SNMPCOLLECT_THRESHOLD":
            [@Severity, @Type] = lookup($snmpstatus, Status1)
            update(@Severity)
            update(@Type)
        case "TEC_ITS_NODE_STATUS":
            [@Severity, @Type] = lookup($nodestatus, Status1)
            update(@Severity)
            update(@Type)
        case "TEC_ITS_SERVICE_STATUS":
            [@Severity, @Type] = lookup($servicestatus, Status1)
            update(@Severity)
            update(@Type)
        case "TEC_ITS_ROUTER_STATUS":
            [@Severity, @Type] = lookup($routerstatus, Status1)
            update(@Severity)
            update(@Type)
        case "TEC_ITS_INTERFACE_STATUS":
            [@Severity, @Type] = lookup($ifstatus, Status3)
            update(@Severity)
            update(@Type)
        case "TEC_ITS_SUBNET_CONNECTIVITY":
            [@Severity, @Type] = lookup($reachability, Status2)
            update(@Severity)
            update(@Type)
        case "TEC_ITS_ISDN_STATUS":
            [@Severity, @Type] = lookup($isdnstatus, Status2)
            update(@Severity)
            update(@Type)
        case "TEC_ITS_PIXFAIL_STATUS":
            [@Severity, @Type] = lookup($pix_state, Status4)
            update(@Severity)
            update(@Type)
        default:
            discard
        }
    }

    if ( exists( $situation_name ) )
    {
        @Identifier = $situation_name + ":" + $situation_origin + ":" +
        $situation_displayitem + ":" + $ClassName
        @AlertKey = $situation_name
        @Node = $situation_origin
        @NodeAlias = $situation_origin
        @Agent = $source
        @Type = 20
    }

```



```

@ITMDisplayItem = $situation_displayitem
@ITMStatus = $situation_status
@ITMTime = $situation_time
@ITMEventData = $situation_eventdata
@ITMHostname = $cms_hostname
@ITMPort = $cms_port
@ITMIntType = $integration_type
@ITMSitType = $situation_type
@ITMThruNode = $situation_thrunode

# if (( match($situation_status, "E") OR
#       match($situation_status, "F")) AND
#       match($integration_type, "U") )
#{
#   @ITMStatus= "Y"
#}
#update(@ITMStatus)
#update(@ITMEventData)
#update(@ITMHostname)
#update(@ITMTime)

if( exists($master_reset_flag) )
{
    @ITMResetFlag = $master_reset_flag
    update(@ITMResetFlag)
}

#
# Handle additional sync with ITM Server.
#
if(    match($situation_status, "Y") AND
      match($integration_type, "U") )
{
    update(@ITMStatus)
    update(@ITMEventData)
    update(@ITMHostname)
    update(@ITMTime)
}
if(    match($situation_status, "P") AND
      match($integration_type, "U") )
{
    # Situation Stop - Resolution.
    @Type = 21
    # avoid de-dup with any event
    @Identifier = @Identifier + ':' + @Type
}
else if( match($situation_status, "D") AND
        match($integration_type, "U") )
{

```



## Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

### Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG247557>

Alternatively, you can go to the IBM Redbooks Web site at:

[ibm.com/redbooks](http://ibm.com/redbooks)

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247557.

## Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
<b>SG247557_addmat.zip</b>	Example system configuration files

### How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder. Check the README file contained within zip file for further instructions.

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 407. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *Migrating to Netcool/Precision for IP Networks --Best Practices for Migrating from IBM Tivoli NetView*, SG24-7375
- ▶ *Creating EIF Events with Tivoli Directory Integrator for Tivoli Netcool/OMNIBus and Tivoli Enterprise Console*, REDP-4352

## Online resources

These Web sites are also relevant as further information sources:

- ▶ Netcool/OMNIBus Primer for IBM Tivoli Enterprise Console Administrators  
[ftp://ftp.software.ibm.com/software/tivoli/education/WhitePapers/OMNIBus\\_TEC\\_Admin.pdf](ftp://ftp.software.ibm.com/software/tivoli/education/WhitePapers/OMNIBus_TEC_Admin.pdf)
- ▶ Datasheet Netcool/OMNIBus  
<ftp://ftp.software.ibm.com/software/tivoli/datasheets/ds-netcool-omnibus.pdf>
- ▶ IBM Tivoli Open Process Automation Library (OPAL): IBM Tivoli Netcool/OMNIBus Integration Best Practices  
<http://catalog.lotus.com/topal?NavCode=1TW10NC10>
- ▶ Tivoli and Netcool Event Flow Integration  
<http://catalog.lotus.com/ncomnibus?NavCode=1TW10EC01>
- ▶ Universal Agent Solution for Netcool/OMNIBus ObjectServer Monitoring and UNIX  
<http://catalog.lotus.com/ncomnibus?NavCode=1TW10TM34>

- ▶ IBM Business Partner support and resources  
<http://www.ibm.com/partnerworld/>
- ▶ Sybase technical documents - technical documents about Sybase using the search in the global header or the Tech Docs Focused search below, which allows you to search by title, doc id, and so on  
<http://www.sybase.com/support/techdocs>
  - Sybase Open ClientConnect  
[http://www.sybase.com/detail\\_list?id=8072&multi=true](http://www.sybase.com/detail_list?id=8072&multi=true)
  - Sybase Open Server  
<http://www.sybase.com/products/databasemanagement/openserver/technicalsupport>
  - Sybase Open ServerConnect  
[http://www.sybase.com/detail\\_list?id=8102&multi=true](http://www.sybase.com/detail_list?id=8102&multi=true)
- ▶ Subversion - an open-source version control system  
<http://subversion.tigris.org/>
- ▶ IBM Software Support Toolbar  
<http://www.ibm.com/software/support/toolbar/>
- ▶ DeveloperWorks Tivoli - technical resources for Tivoli software and security products  
<http://www.ibm.com/developerworks/tivoli>
- ▶ DeveloperWorks Netcool/OMNIBus Forum  
[http://www.ibm.com/developerworks/forums/dw\\_forum.jsp?forum=1144&cat=15](http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=1144&cat=15)
- ▶ Open Process Automation Library - Ready for IBM Tivoli software  
[http://www.ibm.com/developerworks/forums/dw\\_forum.jsp?forum=1100&cat=15](http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=1100&cat=15)
- ▶ IBM Tivoli Network Manager - IP Edition and Transmission Edition  
[http://www.ibm.com/developerworks/forums/dw\\_forum.jsp?forum=1136&cat=15](http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=1136&cat=15)
- ▶ IBM Tivoli Monitoring (ITM) 6.1  
[http://www.ibm.com/developerworks/forums/dw\\_forum.jsp?forum=796&cat=15](http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=796&cat=15)
- ▶ Security Management  
[http://www.ibm.com/developerworks/forums/dw\\_forum.jsp?forum=259&cat=15](http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=259&cat=15)

- ▶ Tivoli User Groups - Through this URL you will find resources on news and events, hot links, newsletters, acquisitions, and user community groups  
[http://www-306.ibm.com/software/tivoli/tivoli\\_user\\_groups/index.html](http://www-306.ibm.com/software/tivoli/tivoli_user_groups/index.html)  
<http://netcoolusers.org/>
- ▶ Tivoli TME10 Mailing List  
[http://www.ibm.com/developerworks/forums/dw\\_forum.jsp?forum=374&cat=15](http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=374&cat=15)
- ▶ Tivoli Netcool Mailing List  
<http://netcoolusers.org/MailingLists>
- ▶ Tivoli Training for Netcool  
<http://www-306.ibm.com/software/tivoli/welcome/micromuse/education.html>

## **Tivoli Netcool/OMNIBus technical information**

### **Release notes**

This document describes the associated publications, new features, prerequisites, resolved issues, and known issues for Netcool/OMNIBus v7.1:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/rn/om71rn1.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/rn/om71rn1.pdf)

### **Installation and deployment guide**

This book provides instructions for installing and deploying IBM TivoliNetcool/OMNIBus, and includes details of the supported platforms and requirements:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/ig/om71ig.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/ig/om71ig.pdf)

### **Administration guide**

This book describes how to perform administrative tasks using IBM Tivoli Netcool/OMNIBus Administrator GUI, command-line tools, and process control:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/ag/om71ag.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/ag/om71ag.pdf)

## User guide

This guide describes how to use the Netcool/OMNIBus desktop to manage events. It provides an overview of Netcool/OMNIBus components, as well as a description of the operator tasks related to using the desktop tools.

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/ug/om71ug.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/ug/om71ug.pdf)

## Probe and gateway guide

This book provides general introductory and reference information about probes and gateways. Documentation on the specific probes discussed within these release notes can be found at the following Web site:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/pg/om71pg.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/pg/om71pg.pdf)

### Generic Log File Probe guide

The Generic Log File Probe is a multi-platform probe that reads a flat log file and parses the values using specified value and line separators. The probe creates dynamic, numbered elements according to the resulting parsed data.

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/probes/genlf/genlf-pdf.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/probes/genlf/genlf-pdf.pdf)

### Syslog probe guide

Syslog is a logging mechanism implemented on UNIX platforms and does not require any special hardware. The probe logs messages in an appropriate system log and writes it to the system console, forwards it to a list of users, or forwards it to another UNIX host over the network. There are three probes that acquire data from syslogd: syslog probe, syslogd probe, and juniper syslog probe.

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/probes/syslog/syslog-pdf.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/probes/syslog/syslog-pdf.pdf)

### Probe for Tivoli EIF guide

A range of Tivoli products generates Event Integration Facility (EIF) messages. The Netcool/OMNIBus probe for Tivoli EIF can receive EIF events sent from any of these Tivoli devices and sends them to the ObjectServer.

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/probes/Tivoli{EIF/tveif-pdf.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/probes/Tivoli{EIF/tveif-pdf.pdf)



## IBM Tivoli Netcool/Security Manager

### Release notes

This document contains supplemental information about the Netcool/Security Manager:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_sm.doc/sm13rn.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_sm.doc/sm13rn.pdf)

### Installation guide

This document contains installation information about the Netcool/Security Manager:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_sm.doc/sm13inst.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_sm.doc/sm13inst.pdf)

## IBM Tivoli Netcool/Webtop

### Release notes

These release notes describe the installation prerequisites and known issues for Netcool/Webtop 2.1:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_wt.doc/rn/GI11-8191-00.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_wt.doc/rn/GI11-8191-00.pdf)

### Administration guide

This guide describes how to install, administer, and use IBM Tivoli Netcool/Webtop. The chapters and appendixes describe each functional area, and task-oriented examples are provided to assist users and administrators in configuring and using the application:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_wt.doc/ag/Administration\\_Guide.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_wt.doc/ag/Administration_Guide.pdf)

## IBM Tivoli Netcool GUI Foundation

### Release notes

These release notes introduce and describe the known issues of the Netcool GUI Foundation:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_gui.doc/rn/NGF\\_1.1.359\\_RelNotes.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_gui.doc/rn/NGF_1.1.359_RelNotes.pdf)

## **Administration guide**

This guide describes how to administer the Netcool GUI Foundation. The chapters and appendixes describe each functional area, and provide task-oriented examples to assist administrators in configuring and using the application:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_gui.doc/admin/NGF\\_1.1.359\\_Admin.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_gui.doc/admin/NGF_1.1.359_Admin.pdf)

## **IBM Tivoli Netcool tools and utilities**

### **Using IBM Tivoli Include Library**

This document provides further reference information about IBM Tivoli Netcool/OMNIBus Knowledge Library 1.3 and is located at:

[http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool\\_OMNIBus.doc/probes/nck1/nck1-pdf.pdf](http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIBus.doc/probes/nck1/nck1-pdf.pdf)

### **Global Advanced Technology team tools and utilities**

IBM Tivoli customers with approved login accounts can find useful tools and utilities (for example, the Netcool IDE and the mib2rules). The GAT tools and utilities Web site is:

<http://www-03.ibm.com/software/tivoli/gat/toolsutils>

#### **Netcool IDE for Windows**

[http://www-03.ibm.com/software/tivoli/gat/toolsutils?Page=show\\_list&objectId=85](http://www-03.ibm.com/software/tivoli/gat/toolsutils?Page=show_list&objectId=85)

#### **Netcool IDE for UNIX**

[http://www-03.ibm.com/software/tivoli/gat/toolsutils?Page=show\\_list&objectId=61](http://www-03.ibm.com/software/tivoli/gat/toolsutils?Page=show_list&objectId=61)

#### **mib2rules for Windows**

This is a fully featured MIB browser. Export to csv, html, Netcool lookup tables, or Netcool rulesfiles. Generate test traps and issue SNMPGETs to SNMP-enabled devices.

[http://www-03.ibm.com/software/tivoli/gat/toolsutils?Page=show\\_list&objectId=221](http://www-03.ibm.com/software/tivoli/gat/toolsutils?Page=show_list&objectId=221)

## The Netcool MIB database

The Netcool MIB database houses the largest repository of SNMP MIBs on the Internet. It has been compiled and made available to any IBM customer with a support contract at no additional cost. Users can search for objects by name or OID, or can search for MIBs by MIB name. They can then add any MIBs to the cart for later download. With mib2rules technology inside, MIBs in a user's cart can be downloaded in a number of different formats, including:

- ▶ Standalone rules for any of the Netcool trapd probes
- ▶ NCiL format rules suitable of inclusion in an existing Netcool Include Library (NCiL) deployment
- ▶ NCKL format rules suitable of inclusion in an existing Netcool Knowledge Library (NCKL) deployment
- ▶ Lookup table suitable for including in any Netcool rulesfile
- ▶ HTML with frames suitable for making available on a Web page or simply browsing from the desktop
- ▶ HTML without frames suitable for using in a Netcool Eventlist tool
- ▶ CSV files for importing into a spreadsheet
- ▶ Individual plain text files suitable for popping up in a help window (that is, nco\_message)
- ▶ mib2rules data files suitable for reading with a standalone version of mib2rules (v5.0 and later)

<http://www-03.ibm.com/software/tivoli/gat/mibdb/mibdb>

## How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

# Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

## RSS feed list

RSS feeds allow you to stay up to date with the latest content created for specific IBM Software products. For more information about RSS and how to get started, visit this RSS help page:

<http://www-306.ibm.com/software/support/rsshelp.html>

Example B-1 shows a collection of RSS feeds defined in Outline Processor Markup Language (OPML<sup>1</sup>) format. The feed definitions in Example B-1 focus on IBM Tivoli and Netcool channels only. This OPML file should be ready for import into any compatible RSS feed reader.

*Example: B-1 RSSOwl feeds.opml*

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!--XML generated by RSSOwl (http://www.rssowl.org) on 11.10.07 11:17-->
<opml version="1.1">
  <head>
    <title>OPML generated by RSSOwl (http://www.rssowl.org)</title>
    <dateCreated>11.10.07 11:17</dateCreated>
    <ownerName>DietgerBahn</ownerName>
  </head>
  <body>
    <outline text="+ IBM Feeds +">
      <outline text="IBM Tivoli Enterprise Console">
        <outline text="OPAL (IBM Tivoli Open Process Automation Library)">
          <outline text="IBM Tivoli Enterprise Console - new entries"
            title="IBM Tivoli Enterprise Console - new entries" type="rss"
            xmlUrl="http://catalog.lotus.com/feeds/IBM_Tivoli_TEC_new_rss.xml#1"
            rssOwlUpdateInterval="60" htmlUrl="http://catalog.lotus.com/rss"
            language="en-us" description="IBM Tivoli Enterprise Console - new entries" />
        </outline>
      </outline>
      <outline text="IBM Tivoli Monitoring">
```

---

<sup>1</sup> <http://www.opml.org>

```

        <outline text="Tivoli - IBM Tivoli Monitoring Version 6" title="Tivoli
- IBM Tivoli Monitoring Version 6" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/2366.xml?rss=s2366&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliMonitori
ngV6.html" language="en-us" description="Tivoli support provides technical
self-help information to help troubleshoot technical problems with Tivoli" />
    </outline>
    <outline text="IBM Tivoli Netcool">
        <outline text="IBM Netcool Service Quality Manager">
            <outline text="Tivoli - Tivoli Netcool Service Quality Manager"
title="Tivoli - Tivoli Netcool Service Quality Manager" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3257.xml?rss=s3257&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/TivoliNetcoolServ
iceQualityManager.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli" />
        </outline>
        <outline text="IBM Tivoli Netcool/Impact">
            <outline text="Tivoli - IBM Tivoli Netcool/Impact" title="Tivoli -
IBM Tivoli Netcool/Impact" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3124.xml?rss=s3124&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliNetcoolI
mpact.html" language="en-us" description="Tivoli support provides technical
self-help information to help troubleshoot technical problems with Tivoli
products." />
            </outline>
            <outline text="IBM Tivoli Netcool/Reporter">
                <outline text="Tivoli - IBM Tivoli Netcool/Reporter" title="Tivoli -
IBM Tivoli Netcool/Reporter" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3123.xml?rss=s3123&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliNetcoolR
eporter.html" language="en-us" description="Tivoli support provides technical
self-help information to help troubleshoot technical problems with Tivoli
products." />
                </outline>
                <outline text="IBM Tivoli Netcool Performance Manager for Wireless">
                    <outline text="Tivoli - Tivoli Netcool Performance Manager for
Wireless" title="Tivoli - Tivoli Netcool Performance Manager for Wireless"
type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3256.xml?rss=s3256&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/TivoliNetcoolPerf
ormanceManagerforWireless.html" language="en-us" description="Tivoli support
provides technical self-help information to help troubleshoot technical
problems with Tivoli" />
                    </outline>
                </outline>
            </outline>
        </outline>
    </outline>

```

```

</outline>
<outline text="Netcool/OMNIBus">
  <outline text="OPAL (IBM Tivoli Open Process Automation Library)">
    <outline text="IBM Tivoli Monitoring - new entries" title="IBM
Tivoli Monitoring - new entries" type="rss"
xmlUrl="http://catalog.lotus.com/feeds/IBM_Tivoli_TM_new_rss.xml"
rssOwlUpdateInterval="60" htmlUrl="http://catalog.lotus.com/rss"
language="en-us" description="IBM Tivoli Monitoring - new entries" />
    <outline text="IBM Tivoli Netcool OMNIBus - new entries" title="IBM
Tivoli Netcool OMNIBus - new entries" type="rss"
xmlUrl="http://catalog.lotus.com/feeds/IBM_Tivoli_TNO_new_rss.xml"
rssOwlUpdateInterval="60" htmlUrl="http://catalog.lotus.com/rss"
language="en-us" description="IBM Tivoli Netcool OMNIBus - new entries" />
  </outline>
  <outline text="Support - OMNIBus Virtual Operator" title="Support -
OMNIBus Virtual Operator" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3128.xml?rss=s3128&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolOMNIBusVir
tualOperator.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli products." />
  <outline text="DW - Netcool OMNIBus" title="DW - Netcool OMNIBus"
type="rss"
xmlUrl="http://www.ibm.com/developerworks/forums/dw_forum_rss.jsp?forum=1144&a
mp;full=true" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/developerworks/forums//dw_forum.jsp?forum=1144&am
p;cat=15&amp;ca=drs-fo" language="en" description="RSS of Netcool OMNIBus topics"
/>
  <outline text="Tivoli - IBM Tivoli Netcool/OMNIBus" title="Tivoli -
IBM Tivoli Netcool/OMNIBus" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3120.xml?rss=s3120&a
mp;ca=rsstivoli#1" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliNetcoolO
MNIBus.html" language="en-us" description="Tivoli support provides technical
self-help information to help troubleshoot technical problems with Tivoli
products." />
  <outline text="International Netcool Users Group"
title="International Netcool Users Group" type="rss"
xmlUrl="http://lists.netcoolusers.org/archives/users/recent.rss"
rssOwlUpdateInterval="60" htmlUrl="http://netcoolusers.org" language="en-us"
description="The Netcool Users mailinglist provides you with direct access to
Netcool users, operators and administrators around the World. The purpose of
this list is to help people who are tasked with using, operating and supporting
the Netcool Suite of products (aka Cisco InfoCenter)." />
</outline>
<outline text="Netcool/Webtop">
  <outline text="Tivoli - IBM Tivoli Netcool/Webtop" title="Tivoli -
IBM Tivoli Netcool/Webtop" type="rss"

```

```

xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3121.xml?rss=s3121&a
mp;ca=rsstivoli#1" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliNetcoolW
ebtop.html" language="en-us" description="Tivoli support provides technical
self-help information to help troubleshoot technical problems with Tivoli
products." />
  <outline text="Support - Netcool GUI" title="Support - Netcool GUI"
type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3144.xml?rss=s3144&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolGUI.html"
language="en-us" description="Tivoli support provides technical self-help
information to help troubleshoot technical problems with Tivoli products." />
  </outline>
  <outline text="Netcool Carrier VoIP Manager">
    <outline text="Tivoli - Tivoli Netcool Carrier VoIP Manager"
title="Tivoli - Tivoli Netcool Carrier VoIP Manager" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3322.xml?rss=s3322&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliNetcoolC
arrierVoIPManager.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli" />
    </outline>
    <outline text="Network Manager (Netcool/Precision)">
      <outline text="Support - IBM TNM TN Edition" title="Support - IBM TNM
TN Edition" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3117.xml?rss=s3117&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliNetworkM
anagerTransmissionEdition.html" language="en-us" description="Tivoli support
provides technical self-help information to help troubleshoot technical
problems with Tivoli products." />
      <outline text="OPAL - NetView" title="OPAL - NetView" type="rss"
xmlUrl="http://catalog.lotus.com/feeds/IBM_Tivoli_TN_new_rss.xml"
rssOwlUpdateInterval="60" htmlUrl="http://catalog.lotus.com/rss"
language="en-us" description="IBM Tivoli NetView - new entries" />
      <outline text="DW - IBM TNM IP & TN" title="DW - IBM TNM IP &
TN" type="rss"
xmlUrl="http://www.ibm.com/developerworks/forums/dw_forum_rss.jsp?forum=1136&a
mp;full=true" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/developerworks/forums//dw_forum.jsp?forum=1136&a
mp;cat=15&ca=drs-fo" language="en" description="RSS of IBM Tivoli Network
Manager - IP Edition and Transmission Edition topics" />
      <outline text="Support - IBM TNM IP Edition" title="Support - IBM TNM
IP Edition" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3118.xml?rss=s3118&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliNetworkM

```

```

anagerIPEdition.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli products." />
</outline>
<outline text="TBSM - (Netcool/RAD)">
  <outline text="Support - Realtime Active Dashboards" title="Support -
Realtime Active Dashboards" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3126.xml?rss=s3126&a
mp;ca=rssstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolRealtimeAc
tiveDashboards.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli products." />
  <outline text="Support - IBM TBSM" title="Support - IBM TBSM"
type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3189.xml?rss=s3189&a
mp;ca=rssstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliBusiness
ServiceManager.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli products." />
</outline>
<outline text="Service Monitor Reporter" title="Service Monitor
Reporter" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3160.xml?rss=s3160&a
mp;ca=rssstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolServiceMon
itorReporter.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli products." />
<outline text="Portal" title="Portal" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3122.xml?rss=s3122&a
mp;ca=rssstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolPortal.htm
l" language="en-us" description="Tivoli support provides technical self-help
information to help troubleshoot technical problems with Tivoli products." />
  <outline text="System Service Monitor" title="System Service Monitor"
type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3127.xml?rss=s3127&a
mp;ca=rssstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolSystemServ
iceMonitor.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli products." />
  <outline text="Service Monitor Global Perspective ISM" title="Service
Monitor Global Perspective ISM" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3162.xml?rss=s3162&a
mp;ca=rssstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolServiceMon

```



```

itorGlobalPerspectiveISM.html" language="en-us" description="Tivoli support
provides technical self-help information to help troubleshoot technical
problems with Tivoli products." />
    <outline text="Netcool for VoIP" title="Netcool for VoIP" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3119.xml?rss=s3119&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolforVoIP.ht
ml" language="en-us" description="Tivoli support provides technical self-help
information to help troubleshoot technical problems with Tivoli products." />
    <outline text="Visionary" title="Visionary" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3116.xml?rss=s3116&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolVisionary.
html" language="en-us" description="Tivoli support provides technical self-help
information to help troubleshoot technical problems with Tivoli products." />
    <outline text="Data Center Monitors" title="Data Center Monitors"
type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3129.xml?rss=s3129&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolDataCenter
Monitors.html" language="en-us" description="Tivoli support provides technical
self-help information to help troubleshoot technical problems with Tivoli
products." />
    <outline text="Service Monitor Client Diagnostic" title="Service
Monitor Client Diagnostic" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3161.xml?rss=s3161&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolServiceMon
itorClientDiagnostic.html" language="en-us" description="Tivoli support
provides technical self-help information to help troubleshoot technical
problems with Tivoli products." />
    <outline text="Service Monitor for Network Usage" title="Service
Monitor for Network Usage" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3159.xml?rss=s3159&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolServiceMon
itorforNetworkUsage.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli products." />
    <outline text="Netcool for Asset Management" title="Netcool for Asset
Management" type="rss"
xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3146.xml?rss=s3146&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolforAssetMa
nagement.html" language="en-us" description="Tivoli support provides technical
self-help information to help troubleshoot technical problems with Tivoli
products." />
    <outline text="Netcool for Security Management" title="Netcool for
Security Management" type="rss"

```

```

xmlUrl="http://www-306.ibm.com/software/support/rss/tivoli/3130.xml?rss=s3130&a
mp;ca=rsstivoli" rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/sysmgmt/products/support/NetcoolforSecurit
yManagement.html" language="en-us" description="Tivoli support provides
technical self-help information to help troubleshoot technical problems with
Tivoli products." />
</outline>
<outline text="IBM Tivoli Open Process Automation Library - updated
entries" title="IBM Tivoli Open Process Automation Library - updated entries"
type="rss"
xmlUrl="http://catalog.lotus.com/feeds/IBM_Tivoli_TOPAL_updated_rss.xml"
rssOwlUpdateInterval="60" htmlUrl="http://catalog.lotus.com/rss"
language="en-us" description="IBM Tivoli Open Process Automation Library -
updated entries" />
<outline text="Education - Tivoli" title="Education - Tivoli" type="rss"
xmlUrl="http://www-306.ibm.com/software/tivoli/education/rss/ibm_ed.rss"
rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/software/tivoli/education/rss/" language="en-us"
description="This is the Tivoli Education RSS Feed" />
<outline text="IBM Redbooks - Tivoli" title="IBM Redbooks - Tivoli"
type="rss" xmlUrl="http://w3.itso.ibm.com/rss/tivoli.xml"
rssOwlUpdateInterval="60" htmlUrl="http://w3.itso.ibm.com" language="en-us"
description="Latest publications from IBM Redbooks. Most are made available
online for free in PDF and HTML formats. This feed is updated in real-time." />
<outline text="developerWorks : Tivoli : Technical library"
title="developerWorks : Tivoli : Technical library" type="rss"
xmlUrl="http://www.ibm.com/developerworks/views/tivoli/rss/libraryview.jsp"
rssOwlUpdateInterval="60"
htmlUrl="http://www.ibm.com/developerworks/index.html" language="en-us"
description="The latest content from IBM developerWorks" />
<outline text="IBM Podcasts" title="IBM Podcasts" type="rss"
xmlUrl="http://www.ibm.com/investor/ibm_ir_podcast.xml"
rssOwlUpdateInterval="60" htmlUrl="http://www.ibm.com/investor/"
language="en-us" description="Audio interviews, point of views and news
pertinent to IBM investors" />
</outline>
</body>
</opml>

```

---

# Index

## Symbols

\$OMNIHOME 282–283, 307  
.load\_classes 126, 249

## A

ACF 167  
Acknowledged 28  
ACP profiles 123  
action buttons 138  
actions 104  
Active Event List (AEL) 92  
adapter 9  
adapters 104  
Additional materials 399  
Administration Interface 74  
AES 34  
AFTER IDUC 199  
Agent 196  
agent sources  
    Omegamon 110  
agents 175  
Agile ATM Switch Management 80  
AlertGroup 196  
AlertKey 196  
alerts.conversions 251, 253  
alerts.details 65  
alerts.journal 65  
alerts.problem\_events 224  
alerts.status 64, 196  
alerts.status table 224  
AMOS 275  
Analyze and Automate Layer 61  
Apache 91  
API Probes 81  
Application Dependency Discovery Manager 107  
architecture 37  
ARS 135  
as400 adapter 108  
AS400 messages 114  
assess.pl 107  
Assigned to users 28  
attributes  
    Agent 221

AlertGroup 222  
AlertKey 222  
LastOccurrence 222  
Manager 222  
Node 222  
Severity 222  
audit 105  
audit2d1 206  
auditing database 136  
Authentication 41  
Authorization 41  
automated alerting 104  
automations 221  
    delete\_clears 224  
    generic\_clear 221

## B

BARI.props 174  
BARI\_PA.conf 176  
BAROC  
    file definitions 126  
BAROC class 194  
BAROC class information 138  
baroc file 108  
barocs 370  
Benefits 23  
best practices 100  
bi-directional gateway 94, 135  
Bidirectional gateways 89  
Borland 82  
bottlenecks 135  
buttons 138

## C

Cause 133  
cause event 218  
CEC 141  
Central Event processing 127  
Certificate Authority (CA) 99  
Change rule 128  
Checklist 143  
class\_lookup\_create() 253  
classes 302

- cleanup.rls 49
- Clearing 133
- clearing event 220
- coexistence 147
- Collection Layer 61
- collection layer 96
- collector rule 130
- Color patterns 141
- Command line sources 115
- Command-line framework commands and tasks 48
- commands
  - assess.pl 107
  - epreport.pl 107
  - ESync2000Linux.bin 292
  - event 284
  - itmcmd 311
  - lcf\_env.sh 196
  - nco\_baroc2sql 127, 249
  - nco\_dbinit 171, 341
  - nco\_objserv 179, 344
  - nco\_p\_glf 194
  - nco\_p\_syslog 359
  - nco\_pa 345
  - nco\_ping 361
  - nco\_sql 194, 282, 306, 309
  - nco\_xigen 174
  - nvp\_add 214, 245
  - nvtecia 284
  - tec\_agent\_demo 220
  - tecits\_upgrade 283
  - wgetsub 106
  - wlookup 106
  - wls 106
  - wlsac 106
  - wlscurrb 126
  - wlsinst 369
  - wpostmsg 191, 195, 207
  - wrb 126, 279
  - wrimtest 363
  - wtdbclear 372
  - wtdumpri 372
- Common Event Console 141, 315
- connection\_less mode 121
- connection\_oriented mode 121
- Consolidate Layer 61
- Consolidation 25
- Conversions 73
- CORBA 80
- CORBA Probes 81

- correlated events 225
- correlation 8, 288
- Correlation rule 128
- correlation statement 228
- correlation.rls 49
- correlations 133
- custom buttons 138
- Custom EIF applications 115

## D

- daemon
  - Process Automation 179
- Data Manipulation Language (DML) 188
- database trigger 133, 202
- db\_cleanup.rls 49
- DB2 156, 167
- db2inst1 229
- Deacknowledged 28
- deduplicated events 96
- de-duplication 100, 111
- Deduplication configuration 320
- Default Port Usage 375
- default rulesets 8
- delete\_clears automation 224
- Deleted 28
- dependency.rls 49
- Deployment considerations 116
- DES 34
- Desktop upgrade 136
- Device Probes 80
- DHCP 10
- Dispatch engine process 44
- Distributed Event Processing 123
- Distributed Monitoring 109
- distribution 49
- DML 188
- DNS 10
- dup\_detect attributes 191
- dup\_detect modifier 130, 194
- duplicate events 130, 194
- duplication of consoles 151
- Dynamic configuration and administration 24
- dynamic HTML Lightweight Event List (LEL) 92

## E

- ebusiness.rls 49
- Effect 133
- effect event 218

- EIF
  - Probe considerations 122
- EIF Configuration example 311
- EIF Gateway 150
- EIF integration 167
- EIF interface 149
- EIF Java API 83
- EIF mechanism 150
- EIF Probe 82, 110, 283
  - configuring 283
- EIF receiver 372
- encryption 34
- Endpoint 39
- Enterprise Monitoring Server 300
- Enterprise OID information 112
- Environmental Assessment 103
- epreport.pl 107
- errorevent.conf 308
- escalate.rls 49
- ESync2000Linux.bin 292
- event 284
  - UPS\_Fan\_Down 215
  - UPS\_Temp\_Degraded 216
- event class 228
- event enrichment 32
- Event Flow
  - end-to-end 104
- Event flow 154
- Event Flow Integration 147
- Event flow integration
  - TEC based 149
- Event Forwarder 56
- Event Integration Facility (EIF) 82
- Event List 14
- event list
  - correlated events 225
- event log Adapter 322
- Event Log probe 323
- Event management 8
- event management
  - customization 284
- event management system audit 105
- Event Processing 187
  - distributed 123
  - inputs and outputs 104
- event processing 145
- event processing examples 129
- Event Server 39
- event severities 75
- Event Sources 48
- event sources
  - configuring 277
  - IBM Tivoli Monitoring 109
  - NetView 110
- event storms 121
- Event Synchronization 292
  - installation wizard 293
- event synchronization 151
- Event Viewer 12
- Event visualization 11
- event visualization and management 28
- event\_activity.rls 49
- event\_filtering.rls 50
- event\_thresholds.rls 50
- EventList 73
- events 278
  - attributes 306
  - cause 218
  - clearing 220
  - effect 218
  - Escalation 235
  - handling duplicate events 194
  - Interface Down 287
  - manipulation 104
  - Node Down 287
  - OMNIBus values 221
  - Pre-Classification 275
  - Processing 188
  - Processing Migration 189
  - resolution 188
  - severities 75
  - severity 235
  - Source Hosts 106
  - sources 104
  - storms 121
  - SU\_Failure 245
  - throughput 120
  - volume 120
- EventServer 123
- exec\_program 128
- exec\_task 128
- expire automation 204
- extended attributes 213
  - content 214
  - database trigger 215
  - EIF rules file 244
  - nvp\_add 213
  - nvp\_exists 213

- nvp\_get 213
- nvp\_remove 213
- nvp\_set 213
- Extended Event Recognition 275
- extended glf.rules file 206
- ExtendedAttr 190, 213, 245
- External Event Database 47
- external Omnibus procedure 234

## F

- fact files 239
- Failover 27
- failover 146
- failover configuration 26
- files
  - .load\_classes 126, 249
  - audit2d1 206
  - BARI.props 174
  - BARI\_PA.conf 176
  - errorevent.conf 308
  - glf.props 194
  - glf.rules 194
  - hosts 212
  - inittab 176, 377
  - itm\_proc.sql 308
  - mhntlog.hosts 339
  - mhntlog.props 337
  - mhntlog.rules 337
  - nco\_igen 172
  - nco\_p\_mhntlog.dll 332
  - nco\_p\_mhntlog.exe 332
  - nco\_xigen 363
  - NCOMS.props 306
  - om\_tec.conf 362
  - omni.dat 172
  - probe-name.log 243
  - rule\_sets\_EventServer 127, 279
  - situpdate.conf 308–309
  - situser.conf 310
  - syslog.conf 358
  - syslog.props 359
  - tec\_config.bat 283
  - tec\_gateway.cache 362
  - tec\_gateway.conf 123, 362
  - tec\_gateway\_sce.cache 362
  - tec\_logfile.cache 361
  - tec\_logfile.conf 361
  - tec\_logfile.fmt 361

- tecad\_logfile.baroc 244
- tecint.conf 372
- tecroot.xml 126
- tivoli\_eif.cache 361
- tivoli\_eif.rules 244, 283, 307
- Filter Builder 29
- filter2b 199
- Filtering 125
- filtering 104
- Filtering Events 140
- filter-out 198
- final configuration 161
- Firewall considerations 98
- firewall considerations 121
- firewalls 146
- flash\_not\_ack 134
- Flex 64
- formats 8
- Formatting 123
- forward\_event 278
- Forwarder
  - Situation Update 310
- forwarding events 278
- forwarding.rls 50
- Framework 368
- Framework and TEC Tasks 48
- Framework Authentication 41
- FTP 10
- Fully interconnected connections 52
- functions
  - geteventcount 242
  - getload 242
  - updateload 242

## G

- Gateway 39, 167
- Gateway Receiver 167
- Gateway Receiver component 42
- Gateways 87
- gateways 22
- General suggestions 130
- Generic Clear automation 100
- Generic Clear overhead 100
- generic\_clear automation 133, 221
- generic\_clears automation 191
- geteventcount 132, 209, 242
- getload 242
- glf.props 194

glf.rules 194, 201  
global TMR Roles 41  
GUI 92

## H

heartbeat 122  
heartbeat.rls 50  
Heroix 80  
hierarchical TEC events 255  
hierarchy of TEC servers 135  
hierarchy-test-ok 255  
hostname 131  
hosts 212  
html 138  
HTTP 10  
HTTPS 10  
HUB 58  
Hub and spoke connections 51

## I

IBM Maximo 135  
IBM Service Management 4  
IBM Tivoli Monitoring 54, 290  
IBM Tivoli Monitoring (ITM) V5.1.x 21  
IBM Tivoli Monitoring (ITM) V6.1 21  
IBM Tivoli Monitoring (ITM) V6.2 21  
Identifier 196  
IDUC  
    AFTER IDUC command 199  
IDUC connection 98  
Iduc.ListeningPort property 98  
IMAP 10  
Inform Layer 61  
inittab 176, 377  
insert.sql 250  
installation in phases 147  
Instant Clearing 100  
integration  
    Netcool/OMNIBus and NetView 281  
interface\_correlate\_nodeup 289  
Interfaces file  
    generation 173  
Intra-Device Correlation 275  
ITM 156, 370  
    attributes 306  
    event management 306  
    integration 291  
ITM 5.1.x 109

ITM 6.1 109  
ITM 6.2 110  
ITM Agents 54  
ITM\_Linux\_Disk 317  
itm\_proc.sql 308  
itmcmd 311

## J

Java Console 139  
Java Event Console 46  
java utility 126  
Java-based Active Event List (AEL) 92

## L

Lab configuration 367  
LastOccurrence 196  
LastOccurrence field 237  
lcf\_env.sh 196  
libOpl 82  
Lightweight Event List (LEL) 92  
link\_effect\_to\_cause predicate 217  
Load balancing 24  
Localization support 94  
Log File Probes 80  
logfile adapter 49, 108, 198  
Logfile adapter for linux 167  
Logfile messages 114  
logic 105  
lookup file  
    example 254

## M

mail\_on\_critical trigger 135  
maintenance\_mode.rls 50  
Managed Node 39  
Manager 196  
Manager of Managers 135  
manager of managers 147  
Map definition file 90  
Marconi 80  
MASTER 240  
Master process 44  
master.class\_membership 250, 252  
Master-remote connections 52  
Maximo 135, 145  
MH NT Event Log probe 357  
mhntlog.hosts 339

- mhntlog.props 337
- mhntlog.rules 337
- Microsoft Windows 49
- Miscellaneous Probes 81
- monitoring server
  - configuration 311
- Monitors 87, 140
- multi-byte character sets 83
- Multiple HUB TEMS to HUB/Spoke TEC 58
- Multiple region architectures 51
- multiple Tivoli regions 50

## N

- NCKL 86
- NcKL 274
- nco\_baroc2sql 126–127, 249
- nco\_baroc2sql script 248
- nco\_config 179
- nco\_dbinit 171, 341
- nco\_event 179
- nco\_igen 172
- nco\_objserv 179, 344
- nco\_p\_glf 194
- nco\_p\_mhntlog.dll 332
- nco\_p\_mhntlog.exe 332
- nco\_p\_syslog 359
- nco\_p\_tivoli\_eif 185, 382, 388
- nco\_pa 345
- nco\_pa\_start 176
- nco\_pa\_stop 176
- nco\_patch 181, 184
- nco\_ping 361
- nco\_routing 179
- nco\_sql 127, 194, 224, 251, 282, 306, 309
- nco\_sql statements 137
- nco\_xigen 174, 363
- NCOMS 98, 185
- NCOMS.props 306
- Netcool GUI Foundation 92
- Netcool java utility 126
- Netcool Knowledge Library 86, 274
- Netcool Probe
  - configuration 185
- Netcool/Impact 22, 153
- Netcool/OMNIBus
  - Directory Structure Reference 374
- Netcool/OMNIBus rules 100
- Netcool/Precision IP/TN 95

- Netcool/Provisio 136
- Netcool/Reporter 136
- Netcool/Security Manager 137
- Netcool/Webtop 91, 136
- NetView 156, 167
- NetView 7.1.4/5 108
- NetView event sources 110
- netview.rls 50, 111, 225
- new\_row trigger 131
- NGF 92
- Node 196
- node\_corr\_new 285
- node\_correlate\_interface rule 285
- NodeDown 111
- NodeUp 111
- non-TME transport 123
- Non-TME windows adapter 167
- notify.rls 50
- nvp\_add 213–214, 245
- nvp\_exists 213
- nvp\_get 213
- nvp\_remove 213
- nvp\_set 213
- nvtecia 284

## O

- Object Query Language (OQL) 22
- ObjectServer 62
  - configuring 282
  - database Initialization 171
  - desktop 136
  - event source routes
    - as400 adapter 108
    - DM 3.7 108
    - ITM 5.1.x 108
    - ITM 6.1 108
    - ITM 6.2 108
    - logile adapter 108
    - NetView 7.1.4/5 108
    - postmsg 109
    - postzmsg 109
    - snmp adapter 108
    - unix syslogd 108
    - windows adapter 108
    - wpostmsg 108
    - wpostzmsg 109
  - interfaces 172
  - logfile definition 205



- Properties Configuration 174
- schema 282
- shutdown 180
- startup 179
- ObjectServer attributes
  - Agent 221
  - AlertGroup 222
  - AlertKey 222
  - LastOccurence 222
  - Manager 222
  - Node 222
  - Severity 222
- ObjectServer automations 221
- ObjectServer SQL 18
- ODBC 42
- ODBC Gateways 158
- OID 112
- om\_tec.conf 362
- Omegamon agent sources 110
- omni.dat 172
- OMNIBus
  - as Manager of Manager 151
  - db schema 306
  - desktop 158
  - Directory Structure Reference 374
  - typical event flow 151
- Omnibus
  - external procedure 234
- OMNIBus server
  - configuring 306
- one-way integration 111
- One-way region connections 51
- Open URL tool 138
- OQL 22
- Oracle 79
- outputs 104
- ov\_default.rls 50
- Owned 28

**P**

- PA.NAME 179
- PA.PASSWORD 179
- PA.Username 306
- parallel installation 147
- parsed input fields 201
- parseEvents.pl 144
- Parsing 243
- PARSING FAILED 243
- parsing failed message 127
- PasswordEncryption 34
- Peer-to-peer failover 28
- performance 99
- Performance considerations 83
- phased migration 158
- Plain rule 128
- Planning Guidelines 103
- policy based distribution and subscription 49
- POP 10
- Popup\_Message 135
- Port Usage 98
- ports 375
  - Netcool Probes 376
  - Netcool Security Manager 375
  - Netcool/OMNIBus 375
  - Netcool/Webtop 376
- postmsg 48, 109
- postzmsg 48, 109, 115, 373
- predicate
  - link\_effect\_to\_cause 217
- pre-filtering 113
- pre-insert trigger 131
- Prioritized 28
- Probe
  - Netcool configuration 185
- Probe for Heroix RoboMon Element Manager 80
- Probe for Marconi ServiceOn EMOS 80
- Probe rule
  - lookup tables 211
- probe rule
  - glf.rules 201
- probe watch 122
- probe-name.log 243
- probe-nco-p-nonnative probe 82
- procedure execution 255
- Process Agent 306
  - installing on Windows 340
- Process Automation
  - configuration 175
  - configuring 176
  - daemon 377
  - startup script 377
- Process Automation daemon 179
- Process Control
  - toggle feature 181
- Process Control agents 175
- Process control agents on Windows machines 129
- processes

- nco native event GUI 179
- nco\_config 179
- nco\_event 179
- nco\_objserv 179
- profile 377
- Profiling 24
- Propagating status change 225
- Properties file 90
- Provisio 136
- proxies 146
- Proxy Server 75

## R

- rc.nco\_pa 176
- rc.nco\_pa startup 176
- RDBMS Server 39
- re\_send\_event.conf predicate 145
- re\_send\_event\_conf 135, 278
- reception action 131
- Reception engine process 44
- Red Hat environment 170
- Redbooks Web site 407
  - Contact us xvi
- region architectures 51
- region connections 51
- regular expressions 212
- Reliability 26
- Remedy 135, 145
- Remote procedure execution 128
- repeat\_count 206–207
- Reporter 136
- resilience 146
- Resolved 28
- Restriction Filters 73
- RIM 368
- RIM Host 39
- RIM Object 42
- Root cause analysis 274
- RouterDown 111
- routing gateway 96
- rule bases 302
- Rule engine process 44
- rule types 128
- rule\_sets\_EventServer 127
- rules
  - forwarding raw events 278
  - frequently used 128
  - nco\_p\_tivoli\_eif 388

- node\_correlate\_interface 285
- rules file
  - example 382
- rules processing 104
- Rulesets 49

## S

- safety net mechanism 108
- Scalability 138
- scalability 24
- SCE 123, 125
  - configuration 371
- Scenarios 50
- schema 211, 306
- scripts
  - nco\_baroc2sql 248
  - rc.nco\_pa 176
  - rc.nco\_pa startup 176
  - security.sql 136
  - tec\_help.pl 258
- second stage 159
- Secure Sockets Layer 98
- Secure Sockets Layer (SSL) 34
- Security 33
- Security Manager 137
- security.sql 136
- Self monitoring 243
- send\_email 135
- sendEvents.pl 122
- Service level reporting 136
- Service Management 4
- Severity 196
- SG247557\_addmat.zip 400
- Simple Network Management Protocol 48
- Simple rules 128
- Single HUB TEMS 60
- sit\_ack\_expired\_def\_action 308
- sit\_resurface\_def\_action 308
- SitForwarder 309
- situation update forwarder (SUF) 145
- situpdate.conf 308–309
- situser.conf 310
- slot definitions 8
- slots 130
- SMS 158
- SNMP 10, 48, 156
- SNMP Adapter 167
- snmp adapter 108

- SNMP probe 161
- SNMP traps 114
- Sorting Events 140
- Sources 48
- Spokes 58
- SQL 188
  - correlation statement 228
- SSL tunnel 98
- Startup command file 90
- State Correlation Engine 125
- State Correlation Engine processing 125
- StateCorrelationConfigURL 194
- store and forward 79
- strategy 103
- SU\_Failure event 245
- sub\_source attribute 241
- subscription 49
- SUF 145
- Summary 196
- SWITCH statement 245
- Sybase 47, 64
- synchronization 94, 111, 151
- synchronization tools 89
- syncronization 145
- syslog 114
- syslog daemon 114
- syslog probe 114
- syslog.conf 358
- syslog.props 359
- syslogd probe 114
- system audit 105

## T

- Table replication file 90
- Tables 64
- tables
  - alerts.status 224
- TADDM 107
- TapiSrv 346
- Task engine process 44
- TBSM 116
- TCP/IP Port usage 121
- TEC
  - adapter pre-filtering 113
  - Adapters 154
  - analyzing the TEC rule base 127
  - barocs 370
  - Class UPS\_Temp\_Degraded 199, 208

- command line utilities 48
- Components 41
- components 166
- Console 41
- custom buttons 138
- Dispatch engine process 44
- Environmental Assessment 103
- event flow 43
- event flows 166
- event log Adapter 322
- Event Server 43
- Event Source generation commands and scripts 372
- Event Synchronization 56
- Event Viewer 56
- extracting console information 137
- fact files 239
- forwarding rules 278
- installation 368
- intermediate event collector 280
- Master process 44
- multi-region environment 53
- NetView Adapter configuration 283
- Operator actions 139
- Outputs 134
- Reception engine process 44
- replacement strategy 152
- Rule engine process 44
- rule sets 369
- sample format statements 124
- Task engine process 44
- Tasks 135
- TEC - ITM integration Architecture 56
- TEC ACF Gateway 40
- TEC Adapter 40
- TEC Adapter (non-TME) 40
- TEC Console 40
- TEC Gateway 40
- TEC SCE Gateway 40
- tec\_agent\_demo tool 220
- tec\_config.bat 283
- tec\_dispatch 43
- tec\_forward 135
- tec\_gateway 123
- tec\_gateway.cache 362
- tec\_gateway.conf 123, 362
- tec\_gateway\_sce.cache 362
- tec\_help.pl 258
- TEC\_ITS\_INTERFACE\_STATUS 284

- TEC\_ITS\_NODE\_STATUS 284
- tec\_logfile.cache 361
- tec\_logfile.conf 361
- tec\_logfile.fmt 361
- TEC\_OMNIBus.tar 158
- tec\_reception 43
- tec\_rule) 43
- tec\_server 43
- tec\_task 43
- tecad\_logfile.baroc 244
- tecad\_nv390fwd.rls 50
- tecad\_nv390msg.rls 50
- tecad\_snaevent.rls 50
- tecad\_win 113
- tecint.conf 372
- TEC-ITM integration event flow 57
- tecits\_upgrade 283
- tecroot.xml 126
- telephony\_service 354
- temporal 237
- TEMS 300
  - ITM connector configuration 313
- TEMS Hub 54
- TEMS HUB infrastructure ( 58
- tems\_name 311
- TEP 40
- TEP client 54
- TEP workspace 318
- TEPS 54, 141
- testing plan 144
- textual-conventions 85
- third stage 160
- thread pool 83
- three-tier architecture 97
- threshold rule 130, 208
- timed array window 132
- time-out 122
- Timer rule 128
- Tivoli & Netcool Integration package 158
- Tivoli Application Dependency Discovery Manager 107
- Tivoli Business Service Manager 4.1 23, 152
- Tivoli Business Systems Manager 116
- Tivoli Configuration Manager 107
- Tivoli Data Warehouse gateway integration 136
- Tivoli Desktop 40
- Tivoli EIF 282
- Tivoli EIF Probe
  - configuring 283, 307
- Tivoli Enterprise Console 7
- Tivoli Enterprise Portal 40
- Tivoli Framework 166
- Tivoli framework commands 106
- Tivoli Monitoring 167
- Tivoli NetView 20
- Tivoli Network Manager Entry Edition 113
- Tivoli Network Manager IP Edition 22, 152, 154
- Tivoli regions 50
- Tivoli secure logon 45
- tivoli\_eif.cache 361
- tivoli\_eif.props 84
- tivoli\_eif.rules 244, 307
- tivoli\_eif.rules file 283
- TME UNIX 49
- TMR 368
- TMR Region 39
- TMR Resource Roles 41
- TMR Roles 41
- TMR Server 39, 42
- TNM 152
- tokenize 85
- Tomcat 91
- Tools 73
- trapd daemon 112
- trigger
  - interface\_correlate\_nodeup 289
  - node\_corr\_new 286
- trigger flash\_not\_ack 237
- triggers 25
  - flash\_not\_ack 134
  - mail\_on\_critical 135
  - new\_row 131
  - pre-insert 131
- trouble ticket systems 104
- Trouble Ticketing 158
- trouble ticketing system. 134
- Troubleshooting 361
- troubleticket.rls 50
- tunnel 98
- Two-tiered architecture 95
- Two-way region connections 51
- Type 196
- typical event flow 151

**U**

- uni-directional gateway 135
- unidirectional ObjectServer gateway 89

- unix syslogd 108
- updateload 132, 209, 242
- Upgrade
  - strategies 147
- upgrade 154
  - TEC to OMNIBus 154
- upgrading 165
- UPS\_Fan\_Down 215
- UPS\_Temp\_Degraded 199, 208, 216
- User Interface Server 44
- Users, Roles and Groups 72
- UseStateCorrelation 194

## V

- variable \_rc 207
- View Builder 29
- View builder 30
- Virtual ObjectServer configuration 76
- VisiBroker Object Request Broker (ORB) 82
- VMware 122

## W

- WAAPI 74
- Web Administration Application Programming Inter-  
face (WAAPI) 74
- Web Console 140
- Web material 399
- WebSphere 92
- WebSphere Console Server 44
- Webtop 136, 185
- wgetsub 106
- Windows 49
- windows adapter 108
- Windows NT Event Log 161
- Windows NT event logs
  - probe installation 183
- wlookup 106
- wls 106
- wlsac 106
- wlscurrb 126–127
- wlsvrcfg 363
- wlsinst 369
- Working Queue 139
- wpostmsg 48, 108, 191, 195, 207, 210, 236, 244
- wpostzmsg 48, 109
- wrb 126, 279
- wrimtest 363
- wtdbclear 48, 372

- wtddumper 48, 363
- wtddumprl 48, 108, 144, 363, 372





Redbooks

## Best Practices for IBM Tivoli Enterprise Console to Netcool/OMNIbus Upgrade









# Best Practices for IBM Tivoli Enterprise Console to Netcool/OMNIBus

**Integration and upgrade strategies for TEC-based environments**

**Provides detailed guidelines for planning an upgrade**

**Includes upgrade scenarios and best practice recommendations**

The acquisition of Netcool Inc. brings new opportunities for all involved in IBM Systems Management and the development of a new and exciting strategy. All existing customers who have Tivoli and Netcool products will be looking for IBM direction and guidance on the methods to join these two Systems Management product portfolios together in ways that maximize value.

This IBM Redbooks publication should be used in planning and implementing an integration and upgrade strategy from TEC to OMNIBus. In this book we provide recommended best practices and describe strategies for upgrading existing installations in a way that should best suit the needs of existing TEC-based environments.

The audience for this book is anyone involved in the Systems Management discipline, but it applies primarily to both those with a Tivoli or Netcool background, and is aimed at customers with an existing Tivoli Enterprise Console investment who are looking to evaluate the comparative characteristics of TEC and Netcool/OMNIBus so that they can perform a system upgrade at some point in the future.

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

### BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)