Tivoli® software

IBM

IBM Tivoli Monitoring: Implementation and Performance Optimization for Large Scale Environments

Best practices - large scale deployment of IBM Tivoli Monitoring V6.1 and V6.2

Performance, scalability and high availability considerations

Includes CCMDB integration

Vasfi Gucer
Naeem Altaf
Erik D Anderson
Dirk-Gary Boldt
Murtuza Choilawala
Isabel Escobar
Scott A Godfrey
Mauricio Morilha Sokal
Christopher Walker

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM Tivoli Monitoring: Implementation and Performance Optimization for Large Scale Environments

February 2008

SG24-7443-00

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

First Edition (February 2008)

This edition applies to IBM Tivoli Monitoring Version 6.1.0 and Version 6.2.0, IBM Tivoli OMEGAMON XE Version 4.1.0.

Contents

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 on 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

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

Access360®
AIX®
AS/400®
Candle®
Collation®
Cyanea®
CICS®
CUA®
DataPower®
Domino®
DB2 Universal Database™
DB2®
DFS™
eBA®
Enterprise Asset Management®
ETEWatch®
HACMP™
i5/OS®
Informix®
iSeries®
Isogon®
IBM®
IMS™
Lotus Notes®
Lotus®
Maestro™
Micromuse®
MVS™
Netcool/OMNIbus™
Netcool®
NetView®
Notes®
OMEGACENTER®
OMEGAMON II®
OMEGAMON®
OMEGAVIEW II®
OMEGAVIEW®
OS/2®
OS/390®
OS/400®
Parallel Sysplex®
PQEdit®
Rational®
Redbooks®
Redbooks (logo) ®
RACF®
System p™
Tivoli Enterprise Console®
Tivoli®
TME®
WebSphere®
z/OS®
z/VM®
zSeries®

The following terms are trademarks of other companies:

mySAP, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

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

Snapshot, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

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.

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.

Java, JavaScript, JDBC, JMX, JNI, JRE, JVM, J2EE, Solaris, Sun, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Expression, Internet Explorer, Microsoft, SQL Server, Windows NT, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries 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.

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

Preface

The IBM® Tivoli® Monitoring solution is the next generation of the IBM Tivoli family of products that help monitor and manage critical hardware and software in distributed environments. This IBM Redbooks® publication provides a practical guide to implementing, using and optimizing IBM Tivoli Monitoring, including best practices for performance tuning, sizing, high availability, scalability, reporting, IBM Change and Configuration Management Database integration and firewall considerations.

You will find a great deal of information about IBM Tivoli Universal Agents, including versioning, remote deployment and management, and meta servers.

We also delve into details of IBM Tivoli Monitoring of the components, such as how these components interact, what is the underlying technology, details of configuration files, and where to check in case of problems.

This book is a reference for IT professionals who implement and use the IBM Tivoli Monitoring solution in large scale environments.

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.

Vasfi Gucer is an IBM Certified Consultant IT Specialist at the ITSO Austin Center. He started his IBM career as a Network Specialist in 1989. He has over 18 years of experience providing technical support across a variety of IBM products and technologies, including communications, network, and systems management. For the last ten years, he has been working for IBM ITSO, where he has been writing IBM Redbooks and creating and teaching workshops around the world on a variety of topics. In this position, he also has worked on various Tivoli customer projects as a Systems Architect and Consultant. Vasfi holds a Master's degree in Engineering.

Naeem Altaf is a Senior Software Engineer who has worked in IBM Tivoli Monitoring solutions for almost five years. Altaf is currently working as a Solutions Architect, helping Business Partners and clients to integrate their products into IBM Tivoli Monitoring Enterprise Software.

Erik D Anderson is an Advisory Software Engineer with IBM Tivoli. He is certified in multiple Tivoli products, and is currently working with Tivoli Global Response Team (GRT) helping clients resolve critical situations. He provides technical support and expertise on various Tivoli products. He actively presents at various Tivoli Technical User Conferences, User Group meetings and customer briefings. He has been working for IBM Tivoli for the past four years. Erik has a Bachelor's Degree in Computer Technology from Purdue University, where he graduated with Highest Distinction.

Dirk-Gary Boldt is a Certified IT-Architect, Consultant and Instructor working for IT-Services and Solutions GmbH, a German subcompany of IBM. He has a degree in Information Systems and has more than 15 years of experience in Enterprise Systems Management, focusing on the Tivoli monitoring and event management products on multiple platforms. Dirk-Gary works mainly in customer projects and trainings. He is an IBM Tivoli Certified Enterprise Consultant and is certified in ITIL®.

Murtuza Choilawala is a Senior Software Engineer with IBM Tivoli. He is an IBM Certified IT Specialist working with Tivoli Global Response Team (GRT) helping clients resolve critical situations. He provides technical support on various Tivoli products. He actively presents at various Tivoli Technical User Conferences and User Group meetings. He has 14 years of experience in information technology. The last seven years have been with Tivoli. He has a Bachelor's Degree in Computer Engineering and a diploma in Computer Engineering. Prior to joining Tivoli, Murtuza worked with the iSeries® platform. He has also been a Product Manager for AS/400® systems as a Business Partner.

Isabel Escobar has been working for IBM Brazil since 2004. She was a Candle® Software employee and transferred to IBM during the Candle's acquisition. Since then, Isabel has been working as a Tivoli Support Specialist, responsible for supporting OMEGAMON® XE family and IBM Tivoli Monitoring V6.x. Isabel has a Bachelor's Degree in Mathematics with a speciality in Computer Science.

Scott A Godfrey is a Senior IT Specialist with IBM Global Services. He is a Tivoli Technical Team Leader working for the IBM Integrated Technology Delivery Organization helping large Strategic Outsourced customers with their Systems Management design and implementation. He provides technical support on various Tivoli products for both Distributed Systems and Mainframe Systems. He has 12 years of experience in information technology with IBM. Scott has a Bachelor's Degree in Computer Information Systems from Stephen F. Austin State University in Nacogdoches, TX.

Mauricio Morilha Sokal is an IT Specialist working for IGS Strategic Outsourcing/SDC Brazil on the Tivoli Support Team, providing deployment and support in Tivoli systems for outsourcing customers in Brazil. He holds a degree in Systems Analysis, and has nine years of experience working in Information Technology, including four years with Tivoli systems. Prior joining IBM, Mauricio worked for an IBM Business Partner, implementing Tivoli systems for many different customers.

Christopher Walker is a software engineer based at IBM Hursley Labs in the UK. Since joining IBM in 2001 he has worked as a developer within the transaction processing arena before moving to Tivoli, where he has worked both as a system tester and developer for the Tivoli OMEGAMON XE for CICS® and CICS TG products. Christopher has a Masters degree in Software Engineering from University of Sheffield.

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

Budi Darmawan, Arzu Gucer
International Technical Support Organization, Austin Center

Tony Bhe, Edward Bernal, Greg Bowman, Rod Bowser, Ana C Biazetti, Joseph L Carter, Catherine Cook, James A. Edwards, Bing Fan, Dave Fresquez, Matthew Exley, Shayne Grant, Prabhakar Gopalan, Bruce Hartley, Yasutaka Hirasawa, Doug Jenkins, Girish Kulkarni, Flo Kandefer, Timothy Land, John Michael Lake, Yvonne Matute, Greg Modster, Francisco Moraes, Toben Nelson, Heath Newburn, James Paterson, Jerry Saulman, Ben Stern, Laurie B Turner, Divyesh Vaidya, William Van Hornet, Bradford Willard
IBM USA

Axel Hoppe
IBM Germany

Richard Roy
IBM Canada

Fabrizio Salustri, Alessandro Zonin
IBM Italy

Denis C F de Vasconcelos
IBM Brazil

Dirk Wouters
IBM Belgium

Paul Murtagh
IBM Australia

Elise Kushner
ACT IT-Consulting & Services AG

The team would like to express special thanks to Thad Jennings from the Tivoli Performance team, Cathy Cook and Clyde Foster from the IBM Tivoli Monitoring development team and Gary Louw from the Tivoli Global Response Team. Thad worked with the residency team and provided most of the scalability and performance data. Cathy and Clyde authored the component flow diagrams in the Appendix of this document. Gary provided significant input for the "Firewall considerations" chapter, based on his implementation experiences in this area.

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

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

- Send your comments in an e-mail to:

  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

1

Introduction to IBM Tivoli Monitoring V6.1

This chapter introduces the concepts and components behind IBM Tivoli Monitoring Version 6.1. If you are new to Version 6.1, also refer to *IBM Tivoli Monitoring Version 6.1.0. Quick Start Guide*, SC32-1802, which provides concise information about the product.

The following topics are addressed:

- IBM Tivoli at a glance
- IBM Service Management
- Enterprise management challenges
- IBM Tivoli Monitoring solutions
- IBM Tivoli Monitoring V6.1 components
- IBM Tivoli Open Process Automation Library (OPAL)
- What is new in IBM Tivoli Monitoring V6.1 post-GA
- What is new in IBM Tivoli Monitoring V6.2

1.1 IBM Tivoli at a glance

Tivoli’s portfolio spans security, storage, enterprise asset, systems and network management software that enables clients to reduce the complexity of technology through the integration of IT processes as services across an organization’s infrastructure.

By automating the integration of business services, which is known as “Service Management”, companies can speed the flow of processes and free up their technology resources for more strategic projects. They can also ensure compliance with parameters set by Sarbanes-Oxley (Public Company Accounting Reform and Investor Protection Act), HIPAA (Health Insurance Portability and Accountability Act), Basel II (recommendations on banking laws and regulations) and other mandates, which can require clients to account for how resources are used and how sensitive data is accessed and stored.

In June 2006, Gartner ranked IBM as the worldwide market share leader in the IT operations management software marketplace based on total software revenue for 2005. This marks the fifth consecutive year that IBM maintained its leadership in the IT operations management software market.

IBM Tivoli has recently completed 12 strategic acquisitions to boost strengths in key areas in the last four years:

- Access360® (Security Management)
- TrelliSoft (Storage Management)
- MetaMerge (Meta-Directory)
- Think Dynamics (Orchestration and Provisioning)
- Candle (Availability and Performance Management)
- Cyanea® (Application Management)
- Isogon® (Software Asset Management)
- Collation® (Application Discovery and Dependency Mapping)
- CIMS Labs (Virtualization)
- Micromuse® (Network and Business Service Management)
- Rembo Technology (Software installation and configuration)
- MRO Software (Enterprise Asset Management®)

1.2 IBM Service Management

Today, companies focus on providing innovative services. To deliver these services, IT and operations departments must strive to guarantee compliance, security, and continuous uptime. These areas all play a part in helping to ensure these business services are effectively performed to support the organization's business goals.

Yet it is common for companies with organizational silos and traditional implementations to become entrenched in managing things like IT infrastructure technologies, single product revenues and expenses, individual processes and organizational efficiencies, instead of managing integrated solutions and services delivered by the sum of all these components. When this happens, there can be penalties for noncompliance and service level violations.

IBM Service Management offers you a revolutionary way to align your organization and all its related functions with your business. IBM Service Management encompasses the management processes, tactics and best practices needed to deliver business services. IBM Service Management is focused on developing, deploying, and managing services. It helps to reduce IT and operations costs by automating processes. And it helps to more effectively manage compliance. IBM Service Management offers increasing flexibility and getting products, solutions, and services to market more quickly. Ant it can help you to respond to changes more efficiently and effectively than ever before.

IBM Service Management is designed with one thing in mind: to help you manage your business. Because IBM understands that IT and operations are very much a part of your business, it offers powerful tools to help you align the four primary components of your business:

- People
- Processes
- Information
- Technology

IBM Service Management lets you pull these critical components together with an array of tightly integrated solutions that can be viewed as three interconnected layers:

- IBM Process Management
- IBM Operational Management
- IBM Service Management platform

These solutions are based on IBM and industry best practices, such as the IT Infrastructure Library® (ITIL), Control Objectives for Information and related Technology (COBIT) and enhanced Telecom Operations Map (eTOM), helping users to ensure IT and operational processes are consistently designed, automated and executed, and are auditable for compliance adherence.

IBM Service Management helps you anticipate and plan for change by providing timely access to critical information. IBM Service Management can help you react more quickly to shifts in the marketplace and customer demand, and help you stay far ahead of the competition.

1.2.1 IBM Process Management products

IBM Process Management products work with your operational management products to automate repeatable processes, reduce manual tasks, and free your staff to focus on business-critical priorities. Process managers fully integrate with IBM Tivoli Change and Configuration Management Database (CCMDB).

- IBM Tivoli Availability Process Manager automates tasks related to managing incidents and problems across the organization.
- IBM Tivoli Release Process Manager automates process steps and tasks related to managing software and related hardware deployments.
- IBM Tivoli Storage Process Manager automates tasks related to storage provisioning to optimize storage space and protect data integrity.
- IBM Tivoli Change Process Manager (included in Tivoli CCMDB) automates tasks to apply changes to your IT infrastructure.
- IBM Tivoli Configuration Process Manager (included in Tivoli CCMDB) automatically manages the configuration of your IT infrastructure.

1.2.2 IBM Operational Management products

IBM Operational Management products help users to ensure the availability, reliability and performance of business-critical software and hardware, aid in optimizing storage requirements, and help to meet ongoing needs for data security.

- Business Application Management helps maintain availability and optimal performance of business-critical software applications spanning multiple servers, operating systems, and databases.
- Server, Network and Device Management helps users to optimize availability and performance of underlying IT architecture, including networks, operating systems, databases and servers, as illustrated in Figure 1-1 on page 5.
- Storage Management helps users to optimize storage space, protect data integrity, and comply with data retention regulations and policies.
- Security Management automates identity management and security event management.

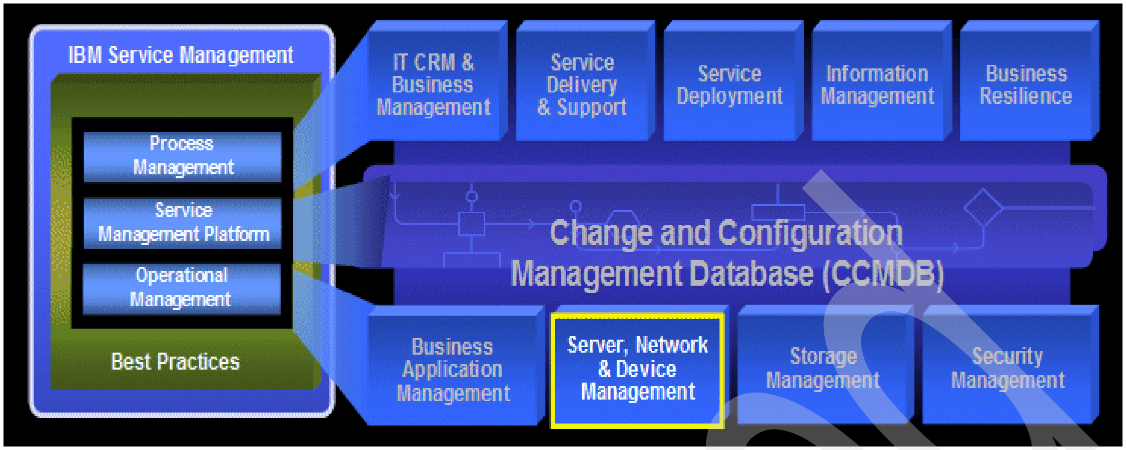


Figure 1-1 ISM Service Management

1.2.3 IBM Service Management platform

One of the key elements of the integration platform in IBM Service Management is Tivoli CCMDB, which collects, stores, and shares dynamic information required for automation. It establishes processes for managing both configuration and change, and it works with IBM Process Management products and IBM Operational Management products to help users ensure that the organization is using current, consistent information by providing:

- Traceable, auditable changes
- Automated application discovery
- Detail on the interactions between systems
- An open platform for data and process integration

1.3 Enterprise management challenges

Many readers will be familiar with the challenges faced by IT departments and IT support teams. The nature of ever-changing business demands and market dynamics often put strains on IT resources, and we are constantly told to “do more, with less.”

The reality we are facing today includes situations where problems occur in systems and the environment well before we are notified, causing IT staff to operate in a reactive, “firefighting” mode. In some environments, the fires seem to get bigger as the pressure, tight deadlines, and high profile of business solutions shadow the need to plan an effective systems management solution.

Traditional enterprise management has to change, as the modus operandi tends to include most if not all of the following:

- It is reactive, not proactive.
- Resources may be healthy while customer service levels are not acceptable.
- Events describe problems, not corrective actions.
- Events flow into the Operations room at an incredibly high rate, and “event storms” have performance impact on systems.
- Fixes are typically manual and inefficient.
- Problems cannot be prioritized because impacts are unknown.
- Most problems cannot be detected; more than 50% of all problems are reported through the help desk.
- Organizational boundaries breed incompatible tools, making end-to-end management and integration very difficult.
- Lack of vision and strategic direction increases costs.

1.3.1 Business driving forces

It can be noted that there are some key business driving forces behind an effective enterprise management. A major force is the need to improve the quality of service delivery and reduce the resources required to implement and use new information technologies. In addition, the following factors must be taken into account when planning and defining the vision of enterprise management.

- The need to increase revenues, reduce costs, and compete more effectively.
- Companies need to deploy informational applications rapidly, and provide business users with easy and fast access to business information that reflects the rapidly changing business environment. Enterprise Management solutions must be transparent to the business solutions being delivered, and they must be proactive to detect, resolve, and escalate potential issues which may impact the business service being delivered.
- The need to manage and model the complexity of today’s business environment.
- Corporate mergers and deregulation means that companies today are providing and supporting a wider range of products and services to a broader and more diverse audience than ever before. Understanding and managing such a complex business environment and maximizing business investment is becoming increasingly more difficult. Enterprise management systems provide more than just basic monitoring mechanisms; they also offer sophisticated issue detection, event correlation, and application and

transaction discovery and performance management tools that are designed to handle and process the complex business information associated with today's business environment.

- The need to reduce IT costs and leverage existing corporate business information.
- The investment in IT systems today is usually a significant percentage of corporate expenses, and there is a need not only to reduce this overhead, but also to gain the maximum business benefits from the information managed by IT systems. New information technologies like corporate intranets, thin-client computing, and subscription-driven information delivery help reduce the cost of deploying business intelligence systems to a wider user audience, especially information consumers like executives and business managers. Maintaining the maximum uptime of these systems is becoming more and more critical.

1.4 IBM Tivoli Monitoring solutions

IBM Tivoli Monitoring solutions provide a means to manage distributed resources through centralized control and configuration, and for many years IBM Tivoli has been a market leader in enterprise monitoring solutions. Figure 1-2 on page 8 illustrates the breadth of IBM Tivoli Monitoring solutions. IBM Tivoli Monitoring has been the backbone for availability monitoring across operating systems and application components.

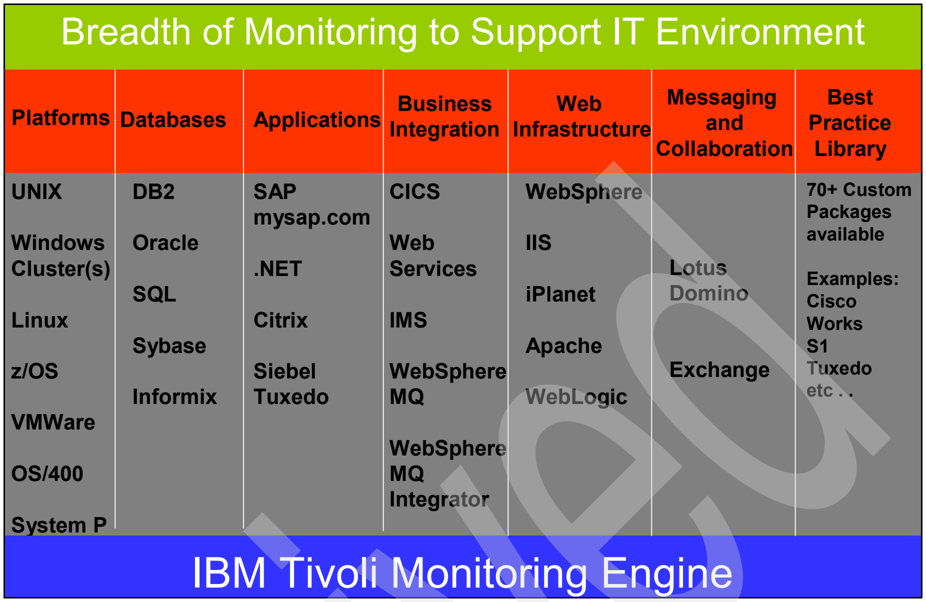


Figure 1-2 IBM Tivoli Monitoring solutions - Integrated end-to-end support for heterogeneous environments

IBM Tivoli Monitoring solutions provide a solid foundation for the development of management solutions addressing the complex needs of today's IT infrastructures. A set of modules built on top of IBM Tivoli Monitoring provide a comprehensive set of solutions for companies facing the challenge of monitoring composite application infrastructures. These modules are delivered through a set of offerings that include:

- IBM Tivoli Monitoring for Applications
- IBM Tivoli Monitoring for Business Integration
- IBM Tivoli Monitoring for Databases
- IBM Tivoli Monitoring for Messaging and Collaboration

IBM Tivoli Monitoring for Composite application infrastructures is illustrated in Figure 1-3 on page 9.

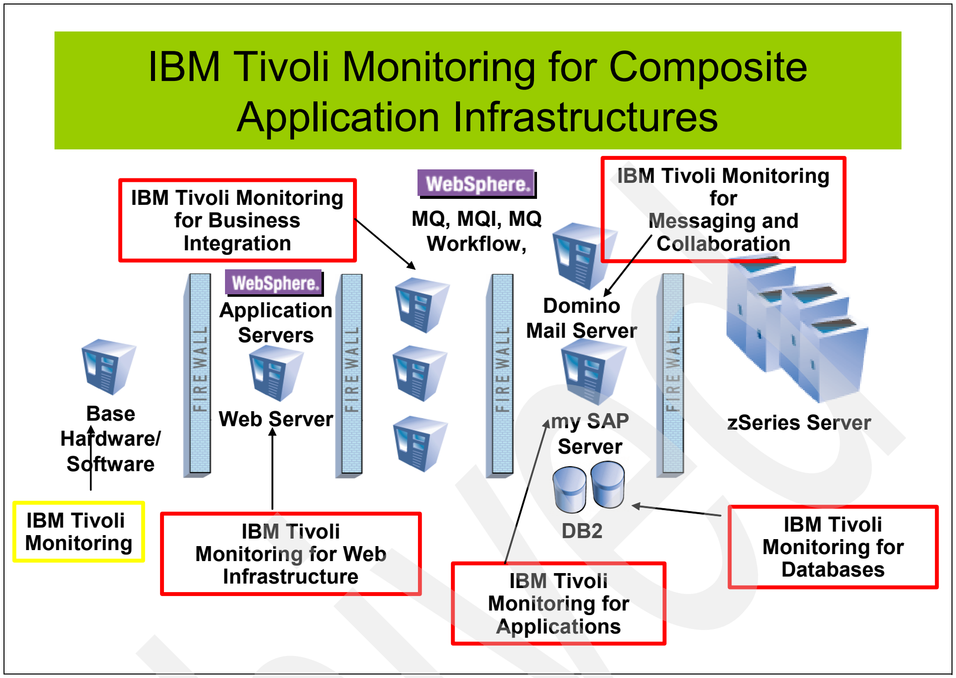


Figure 1-3 Monitoring composite application infrastructures

The latest generation of IBM Tivoli Monitoring solutions have service-oriented themes, and are now focused on:

- Consolidating monitoring platforms.
- Ensuring customers can take advantage of and realize the return on their monitoring investments.
- Improving visualization and analysis of monitoring data.
- Improving management of monitoring environment.
- Simplifying the installation, configuration, and deployment of the solutions with a simplified user interface.
- Improving the integration of Tivoli products.
- Elevating the value of products through process integration.

1.5 IBM Tivoli Monitoring V6.1 components

An IBM Tivoli Monitoring V6.1 installation consists of various components of the IBM Tivoli Monitoring V6.1 infrastructure. This environment is a combination of several vital components. Additionally, optional components can be installed to extend the monitoring functionality.

Figure 1-4 shows the IBM Tivoli Monitoring components.

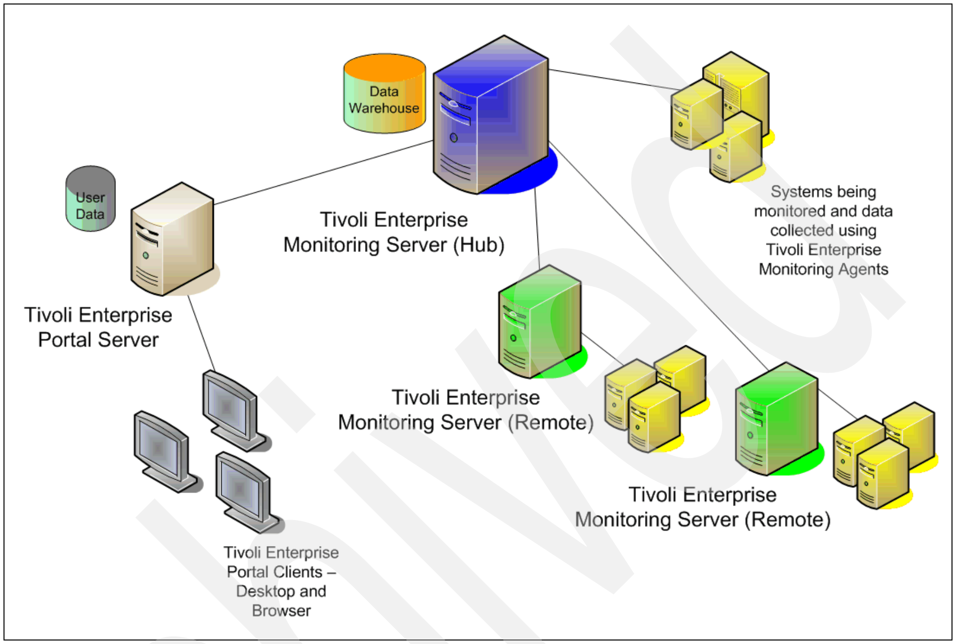


Figure 1-4 IBM Tivoli Monitoring 6.1 components

1.5.1 Tivoli Enterprise Monitoring Server (monitoring server)

The Tivoli Enterprise Monitoring Server (referred to as the *monitoring server*) is the key component on which all other architectural components depend directly. The monitoring server acts as a collection and control point for alerts received from agents, and collects their performance and availability data.

The monitoring server is responsible for tracking the heartbeat request interval for all Tivoli Enterprise Monitoring Agents connected to it. The monitoring server stores, initiates, and tracks all situations and policies, and is the central repository for storing all active conditions on every Tivoli Enterprise Monitoring Agent. Additionally, it is responsible for initiating and tracking all generated actions that invoke a script or program on the Tivoli Enterprise Monitoring Agent.

The monitoring server storage repository is a proprietary database format (referred to as the Enterprise Information Base, or EIB) grouped as a collection of files located on the Tivoli Enterprise Monitoring Server.

The primary monitoring server is configured as a hub (*LOCAL). All IBM Tivoli Monitoring V6.1 installations require at least one monitoring server configured as a hub.

Additional remote (*REMOTE) monitoring servers introduce a scalable hub-spoke configuration into the architecture.This hub/remote interconnection provides a hierarchical design that enables the remote monitoring server to control and collect its individual agent status and propagate the agent status up to the hub monitoring server. This mechanism enables the hub monitoring server to maintain infrastructure-wide visibility of the entire environment.

1.5.2 Tivoli Enterprise Portal Server (portal server)

The Tivoli Enterprise Portal Server (referred to as the *portal server*) is a repository for all graphical presentation of monitoring data. The portal server provides the core presentation layer, which allows for retrieval, manipulation, analysis, and reformatting of data. It manages this access through user workspace consoles.

1.5.3 Tivoli Enterprise Portal (portal or portal client)

The Tivoli Enterprise Portal client (referred to as the *portal* or *portal client*) is a Java™-based user interface that connects to the Tivoli Enterprise Portal Server to view all monitoring data collections. It is the user interaction component of the presentation layer. The portal brings all of these views together in a single window so you can see when any component is not working as expected. The client offers two modes of operation: a Java desktop client and an HTTP browser.

The Tivoli Enterprise Portal can be launched from an Internet Explorer® browser, or can be installed as a client application on a workstation.

IBM Tivoli Monitoring V6.1 Fix Pack 4 introduced Java Web Start capability for administering the desktop client. Java Web Start allows the portal desktop client to be deployed over the network, ensuring the most current version is used. This solution, which can be downloaded from the following official OPAL site, is called “Java Web Start Enablement Kit for the Tivoli Enterprise Portal Desktop Client”:

`http://catalog.lotus.com/wps/portal/topal`

1.5.4 Tivoli Enterprise Monitoring agent (monitoring agent)

The Tivoli Enterprise Monitoring agent (also referred to as *monitoring agents*) is installed on the system or subsystem requiring data collection and monitoring.

The agents are responsible for data gathering and distribution of attributes to the monitoring servers, including initiating the heartbeat status. These agents test attribute values against a threshold and report these results to the monitoring servers. The Tivoli Enterprise Portal displays an alert icon when a threshold is exceeded or a value is matched. The tests are called *situations*.

Tivoli Enterprise Monitoring Agents are grouped into the following types:

- Operating System (OS) Agents

  Operating System Agents retrieve and collect all monitoring attribute groups related to specific operating system management conditions and associated data.

- Application Agents

  Application Agents are specialized agents coded to retrieve and collect unique monitoring attribute groups related to one specific application. The monitoring groups are designed around an individual software application, and they provide in-depth visibility into the status and conditions of that particular application.

- Universal Agent

  The Tivoli Universal Agent is a monitoring agent you can configure to monitor any data you collect. It enables you to integrate data from virtually any platform and any source, such as custom applications, databases, systems, and subsystems.

> **Note:** In addition to these different kinds of agents, IBM Tivoli Monitoring also provides an Agent Builder for creating custom monitoring agents. Using the IBM Tivoli Monitoring Agent Builder, you can quickly create, modify, and test an agent.

1.5.5 Warehouse Proxy agent

The Warehouse Proxy agent (WPA) is a unique agent that performs the task of receiving and consolidating all historical data collections from the individual agents to store in the Tivoli Data Warehouse. You can also multiple Warehouse Proxy agents in your environment.

> **Tip:** If you are planning to use multiple Warehouse Proxy agents support, it is highly recommended that you install Fix Pack 5. Although the support has been available since Fix Pack 2, Fix Pack 5 provides some important fixes and additional functionality in this area.

1.5.6 Warehouse Summarization and Pruning agent

The Summarization and Pruning agent (S&P) is a unique agent that performs the aggregation and pruning functions for the detailed historical data on the Tivoli Data Warehouse. It has advanced configuration options that enable exceptional customization of the historical data storage. Only one S&P agent can be used with the Tivoli Data Warehouse.

In order to minimize network overhead, the S&P agent can be run on the same physical system as the Tivoli Data Warehouse database. If the S&P agent is installed on a separate physical system, the use of a high-speed network connection (100 Mbps or better) between the S&P agent and the warehouse database server is highly recommended for best performance.

> **Note:** To determine the current supported platforms for these components, refer to the latest Fix Pack Readme file. At the time of writing the Fix Pack 5 Readme file contained the most current list.

1.6 IBM Tivoli Open Process Automation Library

The IBM Tivoli Open Process Automation Library (OPAL) Web site is a catalog of solutions provided by IBM and IBM Business Partners; it is accessible at the following site:

`http://www.ibm.com/software/tivoli/opal`

Figure 1-5 on page 14 displays the OPAL site page.

Figure 1-5 OPAL - site page

The OPAL site offers the following options:

- A comprehensive online catalog of more than 300 validated product extensions
- Product extensions to facilitate managing or protecting a specific application or type of application. Examples include:
  - Automation packages
  - Integration adapter, agents,
  - Technical integration papers
  - Trap definitions
  - Plug-in toolkits, or application registration files
- 70+ Universal Agent Solutions
- 209+ ABSM solutions

1.7 What is new in IBM Tivoli Monitoring V6.1 post-GA

Since it became available, IBM Tivoli Monitoring V6.1 has added support, improvements, and a number of features. The highlights of new features added since GA include the following:

- Added support for various new OS, application and database support for all IBM Tivoli Monitoring components. Refer to the current Fix Pack Readme for current supported platforms.
- New **tacmd** commands, as listed in Table 1-1.

Table 1-1 New tacmd commands

Command	Description
tacmd createsystemlist	Creates a new managed system list.
tacmd deletesystemlist	Deletes an existing managed system list.
tacmd editsystemlist	Edits a managed system list.
tacmd exportWorkspaces	Exports a workspace.
tacmd importWorkspaces	Imports a workspace.
tacmd listsystemlist	Lists all existing managed system lists.
tacmd listWorkspaces	Lists workspaces available to export.
tacmd viewsystemlist	Displays a managed system list.

- Multiple Warehouse Proxy support.
- Enhanced firewall functionality through the use of a gateway feature.
- New agents - a number of new agents has been added to IBM Tivoli Monitoring.
- TMS Infrastructure view - the Tivoli Enterprise Monitoring Server has a topology view called the Tivoli Monitoring Services (TMS) Infrastructure view, which visually expresses the relationships and linking of monitoring agents and other components to the hub monitoring servers.
- Ability to create a topology view in the Tivoli Enterprise Portal - for monitoring products that support the topology view, you can add the view to a workspace to graphically show objects and their logical relationships to one another.

  Fix Pack 4 added enhancements to Topology view properties. Configuring the threshold number of objects in the topology view before it switches automatically to a table view is done in the Properties editor. The view also has a Style tab for formatting the view, objects, labels, and connector lines.

- Enhancements to event management through the Tivoli Enterprise Portal - you can now add more detailed note information, as well as attach files, to individual events. A new user permission has been added to enable users to attach files. The way that events are acknowledged has also been improved.
- A Discovery Library adapter (DLA) for use in a configuration management database (CMDB), such as IBM Tivoli Change and Configuration Management Database. The DLA scans the IBM Tivoli Monitoring environment and identifies the managed systems in the environment. You can then feed this information into IBM Tivoli Change and Configuration Management Database or another CMDB. Refer to Chapter 12, "IBM Change and Configuration Management Database integration" on page 421 for more information about this topic.
- Ability to link from a situation event in the embedded Tivoli Enterprise Console® event console to the Event Results workspace in the Tivoli Enterprise Portal.

  To use this new function, right-click a situation event from inside the Tivoli Enterprise Console event console and click **ITM Situations -> Situation Results**.
- A new utility, secureMain, has been added to change the file permissions level for files on Linux® and UNIX® computers. Its usage is as follows:
  - "secureMain lock" tightens file permissions and still allows products to run.
  - "secureMain unlock" loosens file permissions to allow the installation of more products.
- The Linux OS agent has added Linux File and Directory monitoring, file information attributes to refer to file information characteristics.
- The UNIX OS agent has added AIX® Printer Queue Support monitoring, as well as the ability to monitor disk attributes (such as available free space) in megabytes and gigabytes (instead of kilobytes).
- The i5/OS® OS monitoring agent has added support for using the SSL communication protocol for communication between the agent and the monitoring server
- Installing the IBM Tivoli Monitoring Fix Pack 5 on all computers where you are running the Warehouse Proxy agent and the Summarization and Pruning agent provides enhancements to reduce the disk space requirements for the Tivoli Data Warehouse. Specifically, the Warehouse Proxy Agent now trims all trailing whitespace data for character data that it inserts into VARCHAR columns in the Tivoli Data Warehouse.
- Network interface list for z/OS®.
- Take-Action authorization and command execution through NetView®.

Monitoring the AIX/IBM System P environment with IBM Tivoli Monitoring

New entitled offering for AIX/System P customers:

- Light-weight monitoring of System P
- Visualize and manage the health and availability of System p™ environment
- See how virtual resources map to physical ones

Agent Builder

Agent Builder is introduced with Fix Pack 5 and will be enhanced in IBM Tivoli Monitoring V6.2. Figure 1-6 shows the Agent Builder.

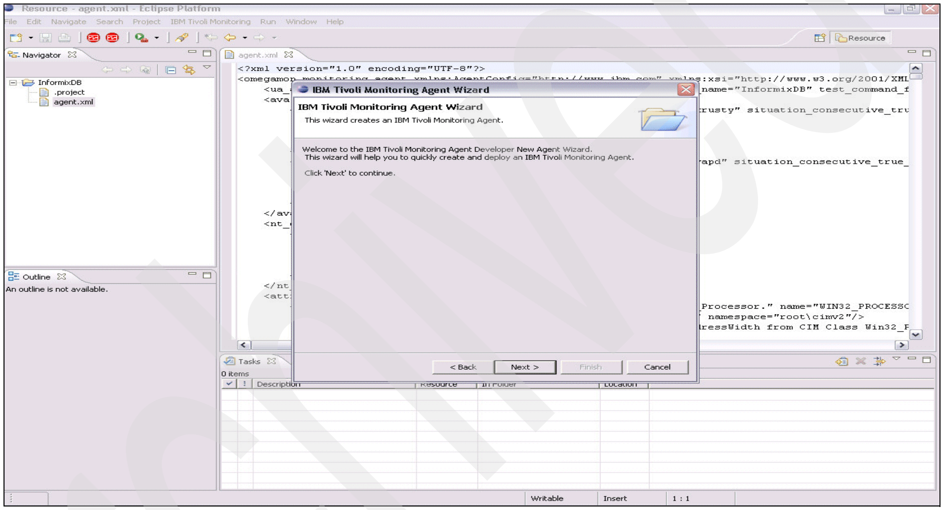


Figure 1-6 Agent Builder

Agent Builder is an Eclipse-based wizard that allows you to quickly and easily build a custom monitoring agent. It can use various data sources, such as WMI (Windows® Management Instrumentation), Windows Performance Monitor (Perfmon), Windows Event Log, SNMP, Script, JMX™ (Java Management Extensions) and more. OPAL Best Practice Library contains many downloadable samples of custom agents created by Agent Builder.

1.8 What is new in IBM Tivoli Monitoring V6.2

This section highlights planned enhancements in IBM Tivoli Monitoring V6.2.

> **Important:** All planned enhancements are subject to change before the general availability time, based on customer requirements and priorities.

IBM Tivoli Monitoring V5.x-to-IBM Tivoli Monitoring V6.2 migration

IBM Tivoli Monitoring V6.2 will provide an "automation-supported upgrade process" for migrating IBM Tivoli Monitoring V5.x resource models to situations.

Security

Product security features will be enhanced in IBM Tivoli Monitoring V6.2, by enabling user authentication via LDAP. You will also be able to manage portal permissions using User Groups. Figure 1-7 illustrates modifying Group Permissions.

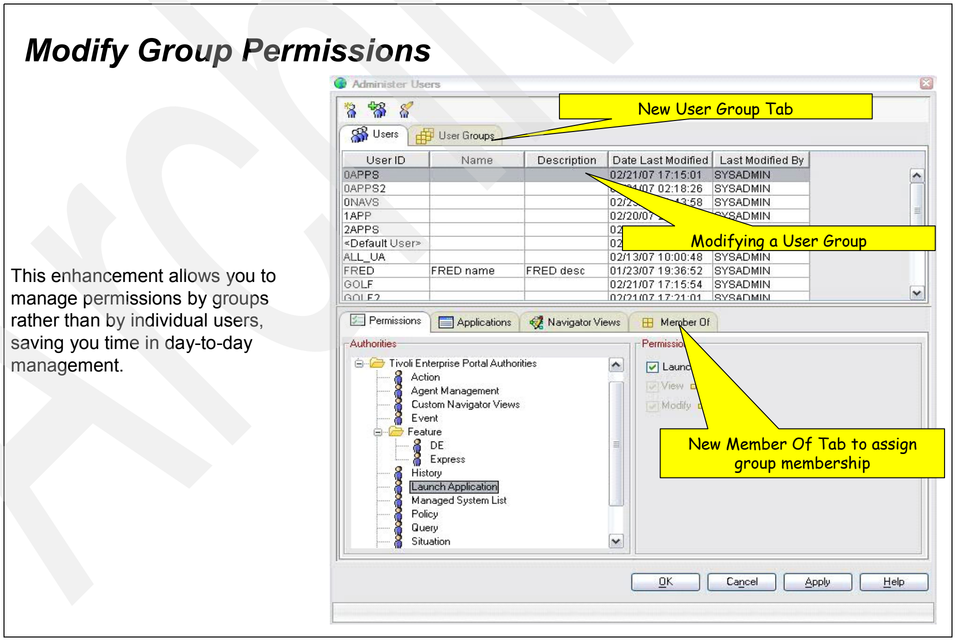


Figure 1-7 Modify Group Permissions

Figure 1-8 illustrates use of the User Groups Tab.

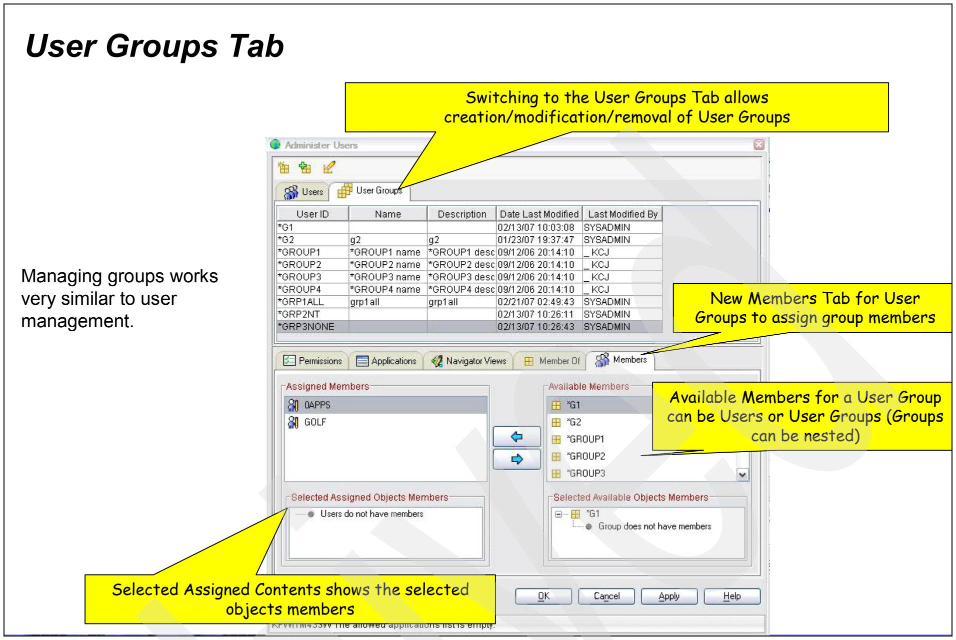


Figure 1-8 User Groups Tab

Infrastructure enhancements

This version will see a number of enhancements in the area of infrastructure:

- Serviceability:
  - Problem Determination data gathering tool
  - Operations Log Enhanced
- Platform updates:
  - Support for Management Clusters
  - Support VMware Management Servers
  - Reduce infrastructure (1500 agents/remote monitoring server)
  - Use Java 1.5 for IBM Tivoli Monitoring Java-based components
  - Support for DB2® V9.1/Include DB2 V9.1 in IBM Tivoli Monitoring BOM
  - Support Tivoli License Manager

Advanced event integration

IBM Tivoli Monitoring V6.2 will enhance the existing Tivoli Enterprise Console integration and allow context based launching. It will allow enable or disable send event, setting the destination IBM Tivoli Enterprise Console server(s) and event severity per situation.

Note: With IBM Tivoli Monitoring V6.2, you will be able to send situation events to more than one event server.

Common Event Viewer will integrate IBM Tivoli Monitoring, IBM Tivoli Enterprise Console and OMNIBUS events in a single console. Refer to 6.4, “IBM Tivoli Monitoring V6.2 enhancements for event integration” on page 229 for a preview of these enhancements.

Broadening integration and improved visualization

There are also enhancements in the visualization area. The embedded HTML Browser will be enhanced to provide better HTML and Active Page support and will be fully functional with Web 2.0, frames and multimedia.

This also means that virtually any product interface that can be rendered via a Web browser can be embedded in the TEP and allow for tighter integration via launch-in-context.

There are also a number of Topology View Integration Chart View improvements, such as multi-source support and multi-line support. See 11.1, “New reporting features in IBM Tivoli Monitoring V6.2” on page 370 for more information about this topic.

Agent enhancements

Agent enhancements include a new monitor for the IBM AIX/System P environment. This represents full performance management of AIX/System P with IBM Tivoli Monitoring V6.2. It provides visualization and performance management of the entire System p environment with historical data collection for improved troubleshooting, capacity planning, and service level reporting.

Other agent improvements include added capability for customer configurable views, situations, and workflows, UNIX Agent Zone Support, and support for more than 64 characters in service names.

2

Planning considerations - large scale deployment of Tivoli Monitoring V6.1 components

This chapter provides considerations and guidelines for deploying IBM Tivoli Monitoring V6.1 into large distributed environments. Although some of the recommendations may also apply to large z/OS-based environments, the focus of this chapter is deployment in large distributed environments.

The following topics are addressed:

- Hardware requirements
  - Hub Tivoli Enterprise Monitoring Server
  - Remote Tivoli Enterprise Monitoring Server
  - Tivoli Enterprise Portal Server
  - Tivoli Enterprise Portal clients
  - Tivoli Data Warehouse
- Scalability considerations
- Additional considerations

2.1 Hardware requirements

IBM Tivoli Monitoring has many different components where each one has a specific functionary. In this section we discuss the hardware and networks requirements for IBM Tivoli Monitoring V6.1 components in a large scale deployment. A large scale environment is considered to be a monitoring environment with 3000 or more monitored agents, with 500 to 1000 monitored agents per remote monitoring server.

2.1.1 Hardware requirements

Table 2-1 lists the minimum recommended hardware requirements for a large scale deployment of IBM Tivoli Monitoring V6.1 in a distributed environment.

Table 2-1 Hardware requirements

Component[a]	CPU Speed	Number of CPUs	Memory[b]	Disk Space[c]
Hub monitoring server	Intel® - 2 GHz RISC - 1 GHz	Single	100 MB	650 MB[d]
Remote monitoring server	Intel - 2 GHz RISC - 1GHz	Single	300 MB	250 MB
Portal Server	Intel - 2 GHz RISC - 1GHz	Single	300 MB	800 MB
Portal Client	Intel - 2 GHz RISC - 1GHz	Single	300 MB	150 MB
Tivoli Data Warehouse (See also Table 2-2 on page 27 for more details)	Intel - 2 GHz RISC - 1GHz	4 CPUs	4 - 8 GB depending on the database configuration parameters	Use the Tivoli Data Warehouse planning worksheets. See 4.3, "Size the Tivoli historical data collection" on page 80
Warehouse Proxy agent	Intel - 2GHz RISC - 1GHz	Single	100 MB	150 MB

Component[a]	CPU Speed	Number of CPUs	Memory[b]	Disk Space[c]
Summarization and Pruning agent	Intel - 2GHz RISC - 1GHz	Single	300 MB	150 MB

a. Add the sizings for individual components to calculate a total for more than one component installed on the same computer.
b. The memory and sizings shown in this table are the amounts required for the individual component beyond the needs of the operating system and any concurrently running applications.
c. The size of log files affects the amount of storage required.
d. The disk requirements for the hub and remote monitoring servers do not include storage for the agent depot, which can require an additional 1 GB or more.

> **Note:** The hardware requirements for z/OS environments are documented in *Program Directory for IBM Tivoli Monitoring Services*, GI11-4105.

For best performance, processor speeds are recommended to be at least 1 GHz for RISC architectures and 2 GHz for Intel architectures. Except for the Tivoli Data Warehouse, single processor systems are suitable when an IBM Tivoli Monitoring infrastructure component is installed on a separate computer from the other components (which is the case in most large scale installations).

Consider using multiprocessor systems in the following scenario:

- You want to run the Tivoli Enterprise Portal client on a computer that is also running one of the server components.
- You have a monitoring environment of 1000 or more monitored agents, and you want to install multiple server components on the same computer; for example:
  - Portal server and hub monitoring server
  - Monitoring server (hub or remote) and Warehouse Proxy agent
  - Warehouse Proxy agent and Summarization and Pruning agent

Except in very small environments, use a multiprocessor system for the Tivoli Data Warehouse database server.

- You can run the Warehouse Proxy agent and the Summarization and Pruning agent on the Warehouse database server to eliminate network transfer from the database processing path:
  - If you install the Warehouse Proxy agent on the warehouse database server, consider using a two-way or four-way processor.

– If you install the Summarization and Pruning agent on the Warehouse database server (with or without the Warehouse Proxy agent), consider using a four-way processor.

For large environments where more CPU resources might be needed, you can run the Summarization and Pruning agent on a computer separate from the Warehouse database server. In this case, ensure that a high-speed network connection exists (100 Mbps or faster) between the Summarization and Pruning agent and the database server.

Hub Tivoli Enterprise Monitoring Server CPU

For a large scale installation, the CPU and memory recommendations for the hub monitoring server in Table 3-1 are sufficient during the periods of low activity. If the enterprise requires faster response to the more intense workloads, a faster CPU may be required. Most of the hub's workloads are executed using a single thread, so a faster CPU will provide greater benefit in executing computationally intense workloads. As mentioned previously, multiple processors can provide additional relief if the hub must execute multiple workloads concurrently. If workloads are rarely executed concurrently, additional CPUs will not provide a performance benefit.

> **Note:** Resource utilization on a hub's host machine reflects the sporadic nature of its interactions. The hub can become CPU bound whenever an associated component is restarted, a large configuration change is initiated, or a huge state change occurs. If the communication or interaction is fairly intense, the hub process can dominate the CPU for several minutes, causing noticeable disruption to other processes or subsystems sharing the hub's host machine. Usually the hub will exhibit low CPU utilization, responding only to update requests generated by the portal server or one of the remote servers.
>
> The following list of activities may cause a period of high CPU usage by the hub monitoring server:
>
> - Restarting the hub
> - Restarting the portal server
> - Restarting a remote monitoring server or installing a new remote monitoring server
> - Creating a large managed system list
> - Distributing a situation or policy to a large managed system list
> - Refreshing the status of historical situations

2.2 Scalability considerations

Before starting the deployment of IBM Tivoli Monitoring V6.1 in a large scale environment, consider the following:

- Number of monitoring agents per server
- Number of situations
- Geographic location of the resources that need to be monitored
- Network bandwidth between the locations
- Firewalls
- Number of portal clients connected to the portal server
- Amount, frequency, and locations of historical data collection
- Types of agents

> **Tip:** This last consideration is due to the fact that non-OS agents generally generate more monitoring data than OS agents.

In a large scale deployment it is best to place the hub monitoring server, remote monitoring server, portal server, and Tivoli Data Warehouse on individual servers or images if you are using LPARs on one physical server.

Refer to “Determining the number of monitoring servers needed” in *IBM Tivoli Monitoring Installation and Setup Guide Version 6.1.0,* GC32-9407, for more information about this topic. That publication was written in Fix Pack 5, and lists some of the factors to consider in determining the number of monitoring servers for a deployment.

> **Note:** At the time of writing, the following IBM Tivoli Monitoring V6.1 planning rules of thumb were based on the largest configurations built in IBM Tivoli Monitoring laboratory test environments.
>
> - One hub monitoring server per 10,000 agents
> - One remote monitoring server per 1000 agents (per 1500 agents for IBM Tivoli Monitoring V6.2)
> - One portal server per 50 concurrent clients

There are many factors that affect the scalability of a hub environment. The primary factors are network latency, number of situations running at the hub, the number of concurrent active users, and historical data collection. Note that agent per remote monitoring server number assumes that agents are configured as the short-term historical data collection location. If short-term historical data is collected on the monitoring servers, you might not be able to achieve the 1000 number.

Tip: Before increasing the number of agents a remote monitoring server supports, track the memory usage and CPU utilization on the monitoring server's host for several hours. The memory usage should be acceptable, and the steady state CPU utilization should be below 10%. Lower average CPU utilization allows the remote server to respond quickly to transient workloads. If both these conditions are satisfied, consider adding more agents.

The restart time of the remote monitoring server can also be affected by the number of associated monitoring agents. Make sure the monitoring server's restart time is acceptable before adding new agents.

As environments grow in size and exceed the 10,000 agents limit per hub server, you must run multiple hub monitoring servers. When using multiple hub monitoring servers, correlation of events is performed in the event management product (Tivoli Enterprise Console or OMNIbus).

An important point that you must keep in mind is that IBM Tivoli Monitoring V6.1 has a different concept compared with the previous versions of the TME®-based products: Agents are *not* the total number of servers that are monitored.

An agent is a component monitored on a server like the UNIX OS Agent, UNIX Log Agent, or DB2 agent. For example, if you have a UNIX server running DB2 and an application that logs its errors to a file, you would install three agents on the one server: KUX, KDB, and KUL. On average for large deployments, a server will have at least two agents per server.

2.2.1 Tivoli Enterprise Monitoring Server deployment recommendations

If agents are geographically located in several medium-to-large sites, then consider the following:

- Placing a remote monitoring server at each site to minimize latency and bandwidth usage for heartbeat, situation, and policy processing.

  Note: Network traffic may be improved by locating a monitoring server closer to the data that it collects. This causes consolidation of data locally before data is transmitted further across the network.

- Network speed between central and remote sites will affect response time of portal requests for agent data.

If agents are geographically dispersed, then consider the following:

- Placing a remote monitoring server in a central location.
- Network bandwidth and potential for sharing network links with other applications.

2.2.2 Portal client deployment recommendations

Portal client processing is the biggest contributor to response time. Consider the following deployment techniques:

- Choose a faster single processor on the client PC for better response time, rather than slower multiple processors.
- Use LAN connectivity for portal clients whenever possible.
  - If dial-up is the only connection speed available, consider making a remote desktop connection (using terminal services) to a machine with LAN connectivity to the portal server.

2.2.3 Tivoli Data Warehouse deployment recommendations

Table 2-2 lists the suggested configurations for setting up the Summarization and Pruning agent and Warehouse Database on the same system (which is the recommended setup), based on the amount of data inserted into the Warehouse Database per day. The server sizing is based on the Warehouse Proxy agent is running on a separate server.

The assumption is that you have new and fast server-class hardware because the Warehouse database server is a high-performing server. Even for smaller environments, we recommend a minimum of three dedicated disks because of the high disk I/O requirements of the Warehouse server.

Table 2-2 Tivoli Data Warehouse deployment recommendations

Data inserted per day	Number of CPUs	Memory	Hard disk storage
0 to 100 K Inserts/day	1 CPU	2 GB	3 or more dedicated SCSI drives.
100 K to 500 K Inserts/day	2 CPUs	2 GB	4 or more dedicated SCSI drives.

Data inserted per day	Number of CPUs	Memory	Hard disk storage
0.5 to 2 Million Inserts/day	2 CPUs	4 GB	RAID Array with 4 or more disks.
2 to 10 Million Inserts/day	4 CPUs	4 GB	Multiple RAID Arrays with 5 or more disks. Or, SAN storage
10 to 20 Million Inserts/day	4 CPUs	8 GB	RAID Arrays with 15 to 20 dedicated disk drives. Or, high- performance SAN storage.
> 20 Million Inserts/day	4 CPUs	8 GB	Multiple RAID arrays with 20 to 25 dedicated disks. Or, high-performance SAN storage.

When deploying the Tivoli Data Warehouse in a large scale environment, consider the following deployment techniques:

- Do not collect, warehouse and summarize data that you do not need.

  Avoid blindly enabling all historical data collection unless it is really needed.

- Configure historical data to keep short-term history files at the *agent,* rather than at the monitoring server.

  Collecting short-term history files at the agent reduces agent-to-monitoring server traffic (because history data will not be sent the monitoring server) and decreases monitoring server workload when doing warehousing.

  The historical data file sizes will be much smaller, because data for only that agent will exist in the file, as opposed to data from all agents connected to a monitoring server, when monitoring servers are selected as the short-term history data location (one binary file is created for each attributed group from which data is collected).

  Some arguments for collection of historical data on the monitoring servers are:

  – Central control of history files.

  – Single point of file management, when custom scripts are used instead of warehousing.

  – When sites require restricted resource usage on agent machines.

  - Firewall considerations (sites using history warehousing with agents outside of firewalls do not require an additional network port to be opened to firewall traffic for the Warehouse Proxy agent).
- To save disk space and processing time for the agent, consider the following:
  - Collect historical data only for attribute groups where there is a planned use for the information.
  - If possible, avoid collecting historical data for attribute groups with a large number of instances (for example, process, threads, and so on).
  - Use the longest data collection interval (5, 15, 30 or 60 minutes) that will provide the desired level of information.
- To save disk space and processing time for the warehouse, consider the following:
  - Enable warehouse collection only for attribute groups where there is a planned use for the information.
  - If possible, avoid warehousing historical data for attribute groups with a large number of instances (for example process, threads, and so on).
- Ensure 100 Mbps or higher network connection between the Warehouse Proxy agent server and Tivoli Data Warehouse database server, otherwise.

  You might consider placing the Warehouse Proxy agent on the warehouse database server or keeping Warehouse Proxy agent on remote monitoring servers (using the multiple Warehouse Proxy agent feature).
- If possible, place the Summarization and Pruning agent on the Tivoli Data Warehouse database server
  - Otherwise, use a multiprocessor machine for the Summarization and Pruning agent with 100 Mbps or higher network connection to the Tivoli Data Warehouse database server.

Important: One scenario that might be suitable for a limited number of environments is to use multiple Tivoli Data Warehouses per hub monitoring server (one Tivoli Data Warehouse per data center). This is typically the case where there are several big data centers or branches, or in a Service Provider-type of environment.

This can be done by configuring one or more Warehouse Proxy agents and one portal server in each data center. Each Warehouse Proxy agent will point to a separate database server. Note that this will only be logical if you do not want the administrators in one data center to be able to view historical data for the other data centers, because each portal server will be configured to point to its local Warehouse and will have visibility to that data, but will not have visibility to the historical data for the other data centers (One portal server can not view data from multiple Warehouses.) In this scenario, multiple Summarization and Pruning agents can handle summarizing and pruning of the multiple warehouses.

Another possible configuration is to share one Tivoli Data Warehouse between multiple hub monitoring servers. However, take care not to exceed the scalability limits of the Tivoli Data Warehouse. With multiple hubs and more than 10,000 agents, you increase the likelihood that you will exceed the capacity of the Warehouse. Be aware of how much data you are collecting to ensure that you do not exceed the capacity of the Warehouse database. There is no hard limit for the capacity of the Warehouse, because this depends on various factors such as the system configuration, database characteristics, and configuration.

In addition to scalability, there are specific deployment requirements when sharing a Warehouse database between hub monitoring servers. First, you can run only one Summarization and Pruning agent in only one of the two monitoring server environments. The single Summarization and Pruning agent is responsible for Summarizing and Pruning the data for all of the data in the Warehouse. Consequently, the Summarization and Pruning configuration is determined by the hub monitoring server where the Summarization and Pruning agent is connected. You cannot set up different Summarization and Pruning schedules for each of the hub monitoring server environments.

You must also ensure that the hub with the Summarization and Pruning agent is patched and maintained so that it is a superset of the two monitoring servers. If you install the database agents in one hub, then you must install the database TEMS/TEPS support in the hub with the Summarization and Pruning agent. If you install a fix pack on one hub, then you must ensure that it is also installed on the hub with the Summarization and Pruning agent, which ensures that the Summarization and Pruning agent is aware of all Attribute Groups and Attributes that can be collected.

2.3 Additional considerations

This section will discuss the additional planning considerations when setting up an IBM Tivoli Monitoring V6.1 environment for large distributed environments.

2.3.1 Where to run the hub Tivoli Enterprise Monitoring Server

> **Note:** Part of this section is based on a whitepaper written by Richard Roy from IBM Canada. The whitepaper entitled "ITM Components Relocation" is published at
> `http://catalog.lotus.com/wps/portal/topal/details?catalog.label=1TW10TM68.`

Consider the following seven factors when deciding where to run the hub monitoring server. Not all factors will have the same significance in every environment; some will be decisive, while others may not matter at all to you.

Portal user authentication

Even though they logon to the portal server, users attempting to use the Tivoli Enterprise Portal application must be authenticated at the hub monitoring server. The hub monitoring server calls the local security authority to confirm that the userid and password are valid (if User Validation is enabled), but the authorization table (what the user can do) is still kept in the portal server database, KFWUSER table.

Where the userid is defined depends on the platform where the hub monitoring server is running, as explained here:

- On z/OS, this will be System Authorization Facility (Resource Access Control Facility (RACF®), Access Control Facility2 (ACF2), or Top Secret Security).
- On Windows, the id could be local to the hub monitoring server machine or defined in the Domain to which the hub monitoring server machine belongs (or a trusted domain).

The id must be granted the “Logon locally” right to the hub monitoring server machine. On UNIX or Linux, this will be local security until support for LDAP is implemented.

> **Note:** IBM Tivoli Monitoring V6.2 will provide LDAP support.

Except for XE for Messaging, the userid is only used to logon to the portal server and will not be used again for the life of the session. With XE for Messaging, the userid of the portal user can be used to control access to view message contents or issue commands (Take Action) against the Queue Manager.

Availability

Availability takes two forms: availability of the platform itself, and availability of the hub monitoring server as an application running on it.

z/OS is generally viewed as highly available, but with higher exploitation of Parallel Sysplex®, availability of any single LPAR is less guaranteed than that of the sysplex. At the limit, any LPAR is expendable because its workload can freely move to any other LPAR. The risk of viruses and attacks and the frequent distribution of patches can make Windows platforms less attractive.

> **Note:** For z/OS hub high availability considerations, you can refer to the "High-Availability HUB TEMS on z/OS" whiteaper at
> http://catalog.lotus.com/wps/poral/topal/details?cataog.label=1TW10TM61.

ASCII/EBCDIC data conversion

Data conversion refers to the need to convert data moving between ASCII and EBCDIC machines. If the hub monitoring server and a component sending data to it are on different platforms, the receiving end of any transmission will need to convert the data stream to the local character format. Depending on the amount of traffic, this could become significant.

Note that the entire presentation layer of IBM Tivoli Monitoring runs on ASCII platforms, so data will eventually have to be converted to ASCII if it is to be viewed by end users. Data conversion is often overlooked in deciding where to place the IBM Tivoli Monitoring hub monitoring server.

Event Forwarding

Event Forwarding is intended to simplify the integration between IBM Tivoli Monitoring and other Tivoli products, specifically the Event Management components of IBM Tivoli Enterprise Console, Netcool/Omnibus and IBM Tivoli Business Systems Manager. It is part of a larger integration called Event Synchronization. This feature is currently not available when running the hub monitoring server on z/OS.

Although it is possible to forward events to IBM Tivoli Enterprise Console, Netcool/Omnibus and IBM Tivoli Business Systems Manager from a z/OS hub monitoring server, the process is more complicated and may require the use of Workflow Automation (Policies) and additional products such as z/NetView or the Universal Agent.

Product mix

Where the majority of agents run may dictate where the hub monitoring server should be. If XE on z/OS or XE for Storage on z/OS is being deployed, there will need to be a monitoring server on every monitored LPAR because those agents can only run inside a monitoring server. If few or no distributed agents exist, it might be logical to simply make one of those monitoring servers the hub monitoring server.

If the XE for Messaging, WebSphere® MQ Configuration product will be used, extensive security rules can be defined to control access to it but only if the hub monitoring server is on z/OS and RACF is used. Access to MQ objects can be controlled in great detail using RACF rules.

If most of the agents will be on Distributed platforms, then it might make more sense to also place the hub monitoring server on a Distributed machine. That is especially true if the Remote Deployment and Update facilities will be needed. Those facilities are not available when the hub monitoring server runs on z/OS.

Availability of hardware and skills

What is available or easily obtained to run the hub monitoring server can weigh heavily in the debate. It should be possible to configure a machine of any of the supported platforms with sufficient capacity to run the hub monitoring server even in the largest environments. *Capacity* refers not only to processor speed, but also to memory and network speed and bandwidth.

z/OS capacity is often viewed as more expensive, but Distributed solutions need to account for duplicate hardware and additional software for clustering, if the solution requires high availability. z/Linux may be attractive if the investment in specialized processors has already been made and the customer is searching for applications to make use of it.

Also of prime importance is the availability of skills to install and maintain the hub monitoring server on the chosen platform.

Politics

If all or the majority of products run on z/OS, the z/OS support staff may not be comfortable with having the center of their monitoring solution reside off the mainframe and under the control of an entirely different group. Distributed platform support groups often complain about z/OS being unresponsive to change because of too much Change Management. z/OS people complain of not enough Change Management on the Distributed platforms.

One customer may have a bias against Windows platforms, while another may have the expanded use of z/Linux as a corporate policy. Many mainframe sites view z/Linux as a closer relative because it runs on the same hardware as z/OS.

There will need to be cooperation and synchronization of certain updates between the z/OS Monitoring layer and the Distributed Presentation layer. Some changes also require that the hub monitoring server be upgraded first. If the hub monitoring server is on a highly available z/OS LPAR, installing maintenance on that system first may not be acceptable to the established Change Management procedures.

2.3.2 Situation formulas and event volumes

The formula of a situation specifies conditions on its dependent attributes before the situation will evaluate to true. The number of conditions is different for each situation. Each condition can consist of one of six possible comparisons between threshold and measured values: equal, not equal, greater than, greater than or equal, less than, less than or equal. The possible threshold values depend on the attribute in question. Situations can also be defined using other situations in either a correlated or embedded configuration.

In large scale environments, situation formulas need to be well thought out. If even a small percentage of the monitoring agents generate a situation event every sampling interval, the generated event volumes can be huge.

Consider the case of 10,000 agents running 30 situations with an average interval of 15 minutes. If only 1% of the situations generate an event every interval, then 12,000 events will be generated per hour. The sheer volume of events would make event interpretation impossible. When creating a situation for a large scale environment, make sure it has the following characteristics:

- Specifies selective threshold values
- Has realistic sampling intervals
- Uses situation persistence conditions

Situation event occurrences should be rare, and should reflect an exceptional or undesirable condition for one or more of the monitoring agents that is both persistent and requires attention. Use historical data to track monitoring agent states.

Tip: As a best practice, after you install IBM Tivoli Monitoring V6.1 turn off all the default situations for each product and unsubscribe them from the default manage system lists. Only turn on and distribute the situations that are needed to monitor the servers and applications in your environment.

2.3.3 Associating agents with the hub monitoring server

Associating monitoring agents directly with the hub is a supported configuration, but is not suggested in large scale environments. Monitoring agents associated with the hub will operate normally, but the additional monitoring workloads can disrupt a hub's abilities to process, correlate, and transmit information to the other IBM Tivoli Monitoring V6.1 components.

2.3.4 Using an event manager

Another consideration when deploying IBM Tivoli Monitoring V6.1 is whether you need to use an event manager such as IBM Tivoli Enterprise Console or Netcool® OMNIBus, in addition to the situation event console.

> **Note:** IBM Tivoli Monitoring V6.2 is shipped with a new common event console. See 6.4.2, "Common event console" on page 231 for a preview of this new function.

It is a best practice to use IBM Tivoli Enterprise Console or Netcool OMNIBus, if any of these conditions are valid:

- You have more than 200 agents and monitor more than 1000 events
- Aggregation of event information is needed from many sources
- Event correlation from many sources is required for an automated action
- Events other than situation events need to be viewed

> **Note:** IBM Tivoli Monitoring V6.2 will introduce a number of enhancements in the event management area. See "IBM Tivoli Monitoring V6.2 enhancements for event integration" on page 217 for more information.

3

Firewall considerations

This chapter describes the IBM Tivoli Monitoring architecture and how it relates to firewalls. We explore different types of firewall designs and explain how to implement IBM Tivoli Monitoring in each.

The following topics are addressed:

- IBM Tivoli Monitoring network components and flows
- How your network firewall design relates to IBM Tivoli Monitoring
  - Simple firewall design
  - Complex firewall design
    - Firewall Gateway Proxy
- Additional considerations

3.1 IBM Tivoli Monitoring network components and flows

In this section we discuss which network IP protocols to use when dealing with firewalls and IBM Tivoli monitoring. We also explain the port requirements and describe how to set up reserved ports for IBM Tivoli Monitoring agents.

3.1.1 Communications protocol selection

If you are installing IBM Tivoli Monitoring components across firewalls, the recommendation is to configure each component with the IP.PIPE or IP.SPIPE (TCP communication) protocol.

The IP (UDP communication) protocol is insufficient for firewall configurations. The connectionless UDP protocol requires opening up multiple ports across firewalls to allow multiple connections from each individual IBM Tivoli Monitoring component. For example, a monitoring agent communicating to the monitoring server using IP (UDP communication) protocol requires multiple ports to operate properly.

IP.PIPE protocol has some notable limitations:

- Only 16 IBM Tivoli Monitoring processes on a single system can share the base listening port (default port 1918) on a single network interface card when using the IP.PIPE protocol. Any processes above 16 will fall back to using the IP protocol (only if configured).

  This mainly is a restriction when running large numbers of Tivoli Enterprise Management Agents on one physical system, for example on a z/OS system. It is not a limitation for the total amount of monitoring agent connecting to one monitoring server.

  This may occur only when a system is required to run more than 16 Universal Agents or has more than 16 Database Agent instances. If firewall restrictions force the use of the IP.PIPE protocol, one solution is to move excess Tivoli Enterprise Management Agents above 16 to another system. Another workaround for this limitation is to use the ephemeral:y flag on KDC_FAMILIES at the monitoring agent, as described in 3.1.4, “Understanding ephemeral connections” on page 42.
- The monitoring server may run out of sockets (listen threads). In this case, the monitoring server log will show evidence of this, as displayed in the following message:

  ```
  message KDSMA010 – Communication did not succeed.
  ```

If this occurs, increase the number of sockets by changing the setting of KDS_NCSLISTEN. The maximum value that can be set is 256. You can change these parameters on the hub monitoring server and the remote monitoring server in the KBBENV and kbbenv.ini files.

Table 4-1 lists the *default* listening ports for the IBM Tivoli Monitoring components. Use this table as a quick reference to understand the standard ports for an installation.

> **Note:** Although modifying these default values is supported, it is *not* recommended and has not been tested by IBM Tivoli Software Support.

Table 3-1 Default port usage for IBM Tivoli Monitoring Components

IBM Tivoli Monitoring Component	Listening Port
Tivoli Enterprise Monitoring Server using IP.PIPE	1918/tcp
Tivoli Enterprise Monitoring Server using IP.SPIPE	3660/tcp[a]
Tivoli Enterprise Portal Server	1920/tcp 15001/tcp[b]
Tivoli Warehouse Proxy agent using IP.PIPE	6014/tcp (see Example 3-1 on page 39)
Tivoli Warehouse Proxy agent using IP.SPIPE	7756/tcp (see Example 3-1 on page 39)
Tivoli Enterprise Console	5529/tcp

a. IP.SPIPE uses SSL - Secure Socket Layer encryption

b. Port 15001 is the default port that the portal server uses for CORBA traffic to its portal clients. Portal clients use HTTP to set up the Java application, which then uses CORBA Interoperable Object Reference (IOR) between client and server over port 15001.

3.1.2 Port requirements

Listening ports can be reserved for specific agents on a server by calculating the listening port with the formula shown in Example 3-1.

Example 3-1 IBM Tivoli Monitoring formula to calculate listening port

```
“well-known port” + (N*4096) = “reserved port”
<where: N=startup sequence>
```

```
1918 + (1*4096) = 6014
3660 + (1*4096) = 7756
```

3.1.3 Understanding the COUNT and SKIP options

The COUNT:N parameter that can be defined on the KDC_FAMILIES environment variable of IBM Tivoli Monitoring Agent component is the mechanism for reserving IP.PIPE and IP.SPIPE listening ports. COUNT:N accommodates multiple monitored servers on a single host, in addition to the Tivoli Enterprise Monitoring Server (monitoring server).

N is the number IP.PIPE or IP.SPIPE ports to reserve on a host (in addition to the monitoring server well-known port). Note the following points:

- Servers with COUNT:N configured are assumed to require one of the N reserved ports.
- Servers with SKIP:N+1 are assumed to *not* require one of the N reserved ports.

> **Important:** When IP.PIPE is referenced in this book, this means that the component uses TCP. If your site is using IP.PIPE, be aware of the following limitations:
>
> - There can be, at most, 16 IP.PIPE processes per host.
> - IP.PIPE uses one, and only one, physical port per process. Port numbers are allocated using a well-known port allocation algorithm. The first process for a host is assigned port 1918, which is the default.

As an example, assume that you have the following scenario:

1. A Windows 2003 system inside a firewall.
2. You expect to run a monitoring server, monitoring agent, and a Warehouse Proxy agent on this Windows 2003 system.
3. The monitoring server and the Warehouse Proxy agent must be accessible from outside the firewall. Accessibility from outside the firewall means that you require IP.PIPE ports that must be permitted at the firewall (thus, these ports must be predictable).

Given this scenario, and to allow for this accessibility, one IP.PIPE port requires reservation (and firewall permission in addition to the monitoring server). The monitoring server always receives the well-known port by default. The Warehouse Proxy agent requires `KDC_FAMILIES=IP.PIPE COUNT:1` and the monitoring agent requires `KDC_FAMILIES=IP.PIPE SKIP:2` to make the necessary reservations for the Warehouse Proxy agent. And if the monitoring server well-known port is 1918, then the Warehouse Proxy agent is assigned port (1918 + (4096 *(1)) = 6014. The monitoring agent attempts to listen on port (1918 + (4096 *(2)) = 10110.

The Warehouse Proxy agent listening port is reserved only on 6014 if keyword KDC_FAMILIES is used in conjunction with the following COUNT option:

- COUNT specifies the number of offsets that the agent can use to retrieve a reserved port number.
- COUNT:N means that all offset ports calculated by the following formula starting from 1 up to N are allowed and the first free one is used:
  - Well-known port 1918 + (X * 4096) for X {1,..,N}
- COUNT:1 means that only 1918+(1*4096)=6014 is used, thus the proxy listening port is fixed. For example:
  - `KDC_FAMILIES=IP.PIPE COUNT:1 PORT:1918 IP use:n SNA use:n` **`IP.SPIPE`** `use:n`

If other IBM Tivoli Monitoring V6.1 components, such as the Universal Agent, portal server, and other components, are running on the same system, then the SKIP option must be used to instruct these components to skip some reserved ports, as described here.

- SKIP:N+1 means that only offset ports starting from multiplier N+1 are allowed:
  - Well-known port 1918 + (X * 4096) for X {N+1,N+2,..,max_port_high}
- SKIP:2 allocates 1918+(2*4096)=10110 as the first port and there is no proxy listening port conflict; for example:
  - `KDC_FAMILIES=IP.PIPE SKIP:2 PORT:1918 IP use:n SNA use:n IP.SPIPE use:n`

To test the connectivity between target machines to make sure the firewall ports are open, you can use the **`telnet`** command as demonstrated in Example 3-2.

Example 3-2 Telnet command to test firewall ports

```
C:>telnet 9.1.2.3 1918
```

> **Note:** If there is no connection, then the **`telnet`** command will timeout and fail.

3.1.4 Understanding ephemeral connections

To accommodate firewalls, we need a way to prevent connections being opened from monitoring server downstream to agents. We can force the monitoring server to use only connections opened from the agent, keep them persistent, and then perform all data transfers across this existing and persistent connection.

This mode of connectivity can be forced from the monitoring server (generally when forcing *all* clients), or from the agent (generally when forcing only a small group of clients).

Example 3-3 shows how to code this parameter at the monitoring *server* level.

Example 3-3 Monitoring server ephemeral implementation

```
KDC_FAMILIES=ip.pipe ephemeral:inbound port:1918
```

Example 3-4 shows how to code this parameter at the monitoring *agent* level.

Example 3-4 Monitoring agent ephemeral implementation

```
KDC_FAMILIES=ip.pipe ephemeral:y port:1918
```

Note: There is a difference in behavior with this parameter, depending on which Fix Pack of IBM Tivoli Monitoring V6.1 you are at.

- With Fix Pack 2 and lower:

  The agent will always open a listener on (1918 + (n*4096)) This means that a total of 15 agents can be installed using ports 6014 (1), 10110 (2), 14206 (3)... 63358 (15).

- With Fix Pack 3 and higher:

  The agent will never open a listener when ephemeral is coded, because in this mode the Tivoli Enterprise Monitoring Server or WPA will never initiate a downcall to the monitoring agent anyway; therefore it is unnecessary to have a listener on the agent.

  With the listener suppressed in this mode, we also remove the 15 agent restriction, because there no 63358 ceiling in the 64 K port range.

IBM Tivoli Monitoring 6.1 communications flow

Figure 3-1 on page 43 illustrates the basic port flows between IBM Tivoli Monitoring V6.1 components using the IP.PIPE protocol.

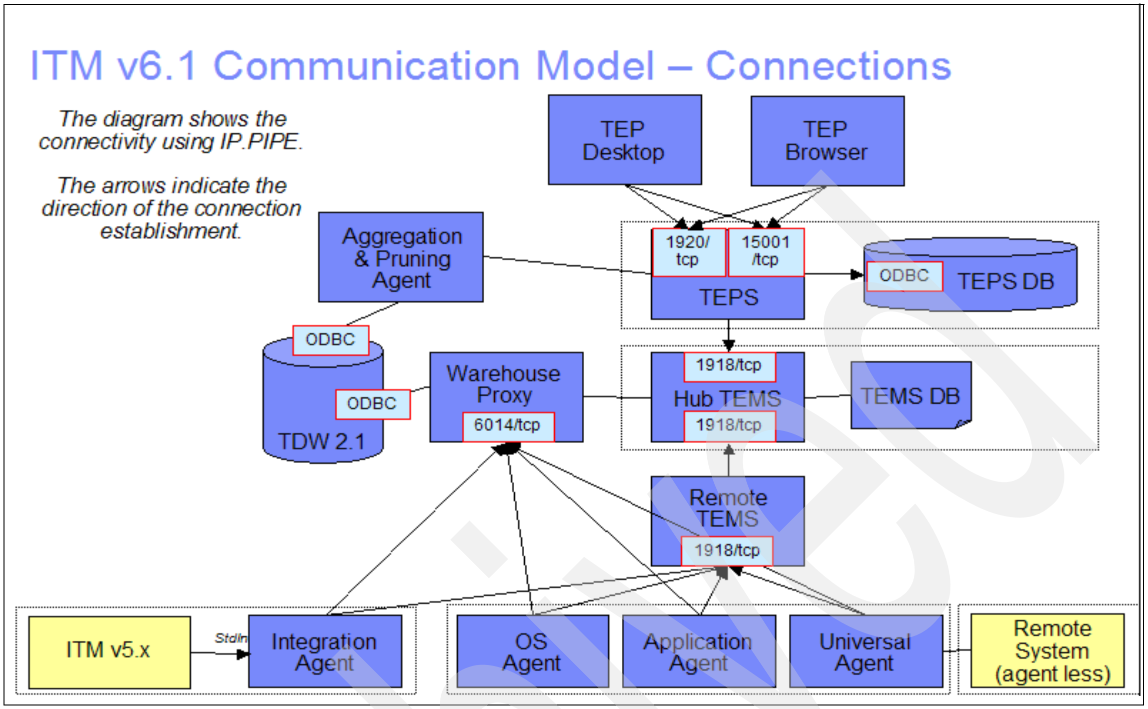


Figure 3-1 IBM Tivoli Monitoring V6.1 communication flow using IP.PIPE

3.2 How your network firewall design relates to IBM Tivoli Monitoring

For all successful IBM Tivoli Monitoring deployments, you must understand the network infrastructure. When planning your IBM Tivoli Monitoring deployment, each of the following are important considerations for deployment:

- Locations of firewalls
- Flows between each firewall
  - uni-directional
  - bi-directional
- Whether you have network address translation (NAT)

In the following sections, we discuss these considerations in more detail.

3.2.1 Simple firewall design

IBM Tivoli Monitoring supports most common firewall configurations, including those that use address translation. To enable this support, IBM Tivoli Monitoring uses the IP.PIPE and IP.SPIPE socket address family, a TCP-based protocol that opens a single port on the firewall for communication by IBM Tivoli Monitoring components.

If your target environment includes a firewall between any IBM Tivoli Monitoring components, you must specify IP.PIPE or IP.SPIPE as your communication protocol during configuration. No other special configuration is needed unless your firewall also uses NAT.

> **Tip:** Refer to *IBM Tivoli Monitoring Installation and Setup Guide*, GC32-9407, for expert advice about simple firewall scenarios. That publications provides several excellent examples of using firewalls involving the monitoring server, remote monitoring server, monitoring agent, portal server and portal clients, both with and without Network Address Translation (NAT).

3.2.2 Complex firewall design

In this section we describe a more complex firewall setup that allows traversal of complex firewall configurations using the IBM Tivoli Monitoring V6.1 Firewall Gateway feature introduced in IBM Tivoli Monitoring V6.1 Fix Pack 2. This new feature is commonly known as the ITM6.1 Gateway Proxy.

Use the firewall proxy gateway when you have *any* of the following scenarios:

1. A single TCP connection cannot be made to span between IBM Tivoli Monitoring components. An example would be that there are multiple firewalls between these components, and a policy that does not allow a single connection to traverse multiple firewalls.
2. Connection requirements do not allow the IBM Tivoli Monitoring default pattern of connections to the hub monitoring server. An example would be agents residing in a less-secure zone connecting to the monitoring server residing in a more-secure zone. Security policy would only allow a connection to be established from a more-secure zone to a less-secure zone, but not the other way round.
3. You must reduce open firewall ports to a single port or connection. For example, rather than opening the port for every system being monitored, a customer would like to consolidate the ports to a single “concentrator.”
4. You must manage agent failover and monitoring server assignment symbolically at the hub monitoring server end of the gateway. Because

gateway connections are made between matching service names, an administrator can change the failover and monitoring server assignment of downstream gateway agents by changing the client proxy bindings next to the hub monitoring server. So, you can control the agents during failover by changing where the gateway proxy is connected. If the Gateway proxy fails over to the backup Tivoli Enterprise Monitoring Server, then the agents will also fail over.

3.2.3 Firewall Gateway Proxy

IBM Tivoli Monitoring V6.1 Gateway Proxy (GWP) can be described as a “black box” that will relay two main communications of IBM Tivoli Monitoring from components in non-secure network to those in a secure network, or vice versa.

This lack box” comprises two components:

- One component is an upstream GWP that is located in a secure network.
- The other component is downstream GWP located in a non-secure network.

The main characteristic of these two components is that the communication is only one-way from the upstream GWP to the downstream GWP. Even though the communication is one-way, all components use this to send data in both directions, secure to non-secure networks and non-secure to secure networks, as illustrated in Figure 3-2.

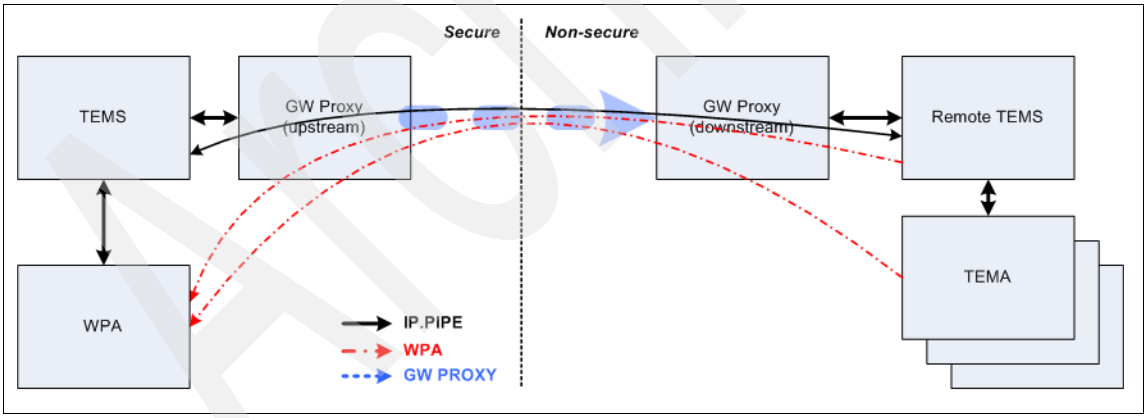


Figure 3-2 Firewall Gateway Proxy data flow

Firewall Gateway Proxy scenario 1

The first scenario of a the firewall environment is when remote monitoring server is located in a secure network, but the monitoring agent in a non-secure network, as illustrated by Figure 3-3 on page 46.

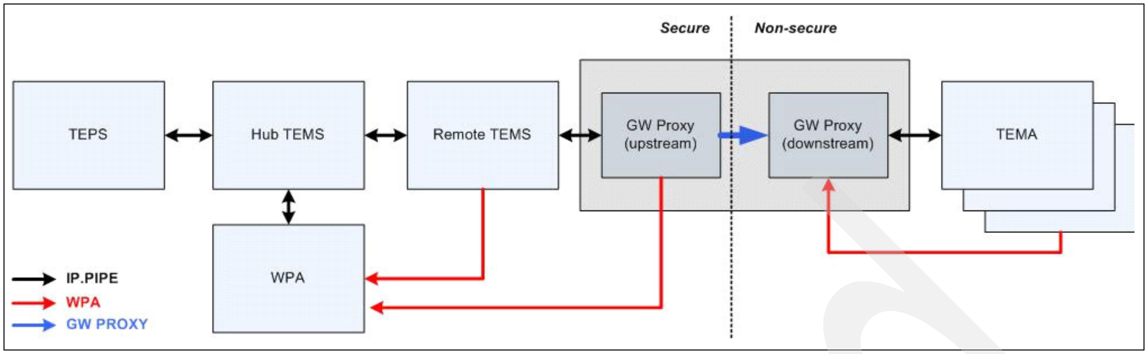


Figure 3-3 Firewall scenario 1: monitoring agent in non-secure network

Firewall Gateway Proxy scenario 2

In the second scenario, both the remote monitoring server and the monitoring agent are located in a non-secure network, but the rest of the components are in a secure network, as illustrated in Figure 3-4.

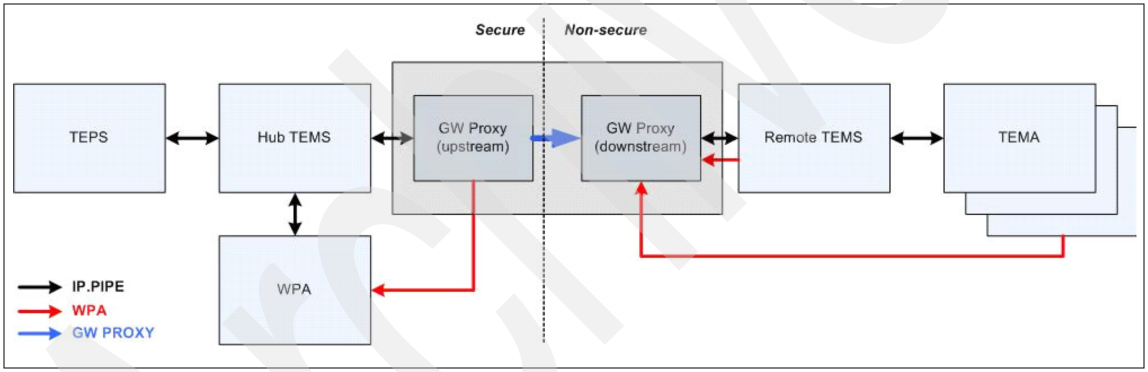


Figure 3-4 Firewall scenario 2: remote monitoring server and monitoring agent in non-secure network

In this scenario, both remote monitoring server communication and monitoring agent are broken from the hub monitoring server and WPA. As such, the upstream GWP is required in the secure network to relay data to the hub monitoring server and WPA. The downstream GWP in the non-secure network is used by the remote monitoring server as its hub monitoring server and WPA, and by the monitoring agent as its WPA.

Firewall Gateway Proxy implementation

Gateway Proxy is implemented as an Operating System agent which could be in Windows or UNIX. The upstream GWP is usually implemented in an OS agent running together with the hub monitoring server (Scenario 2) and the remote monitoring server (Scenario 1). The supported downstream GWP is installed in a separate machine.

The GWP configuration is stored in an XML file (described in detail in “Firewall Gateway Proxy configuration” on page 48). The XML file is placed in one directory and is loaded by the OS agent with the help of the following environment variable:

```
KDE_GATEWAY=<XML file>
```

This variable is loaded by the OS agent. From that moment onward, the OS agent is running as GWP.

When the OS agent is installed in a downstream GWP, its monitoring server must be configured to point to its own IP address. This is a paradox in a downstream GWP because in one machine there is only one OS agent running, but it has two roles: one is being a monitoring server, and the other is being a monitoring agent—and this monitoring agent has to be configured to connect to itself as its monitoring server.

There is one recommendation regarding WPA port configuration in a firewall environment. The port used by WPA to listen to historical data export communication is recommended to be set to the highest port available for IBM Tivoli Monitoring 6.1; that is, 63358.

> **Note:** The reason for recommending that you have the WPAs bound to the highest of the 15 possible ports when the well-known algorithm (1918 + (n * 4096)) is used is because, when multiple agents are installed, the startup order can cause the WPA to bind to a different port than that configured on the monitoring agents.
>
> If you bind the WPA to the highest possible port, then it can never deviate from this port, and therefore you can configure the agents with more confidence.

This can be achieved by changing the following variable in WPA configuration.

```
KDC_FAMILIES="ip.pipe port:1918 SKIP:15 ip use:n ip.spipe use:n sna
use:n"
```

SKIP:15 will skip port configuration 15 times, so WPA will use port number with the formula of 1918 + (15 x 4096) = 63358.

> **Note:** KDC_PARTITION is not used in GWP implementation.

The main steps to implement GWP are listed here:

1. Install the OS agent to enable GWP, and configure it to report to itself as monitoring server.
2. The upstream GWP is installed in the monitoring server in the secure network, and the downstream GWP is installed in a separate machine in the non-secure network.
3. Create an XML file for both GWPs and put them in their respective machines.
4. Insert the KDE_GATEWAY parameter in the agent configuration and point it to the appropriate XML file.
5. Configure WPA to listen to port 63358.

Firewall Gateway Proxy configuration

Example 3-5 displays a sample configuration of an upstream GWP.

Example 3-5 Sample configuration of upstream GWP

```
<tep:gateway xmlns:tep=http://xml.schemas.ibm.com/tivoli/tep/kde/
name="upstream_gw">
   <zone name="trusted">
      <interface name="clientproxy" ipversion="4" role="proxy">
         <bind localport="poolhub" service="tems">
            <connection remoteport="1918">HUB_hostname</connection>
         </bind>
         <bind localport="poolwhp" service="whp">
            <connection remoteport="63358">WPA_hostname</connection>
         </bind>
         <interface name="downrelay1" ipversion="4" role="connect">
            <bind localport="10010">HUB_hostname
               <connection
remoteport="10020">downstream_gw</connection>
            </bind>
         </interface>
      </interface>
   </zone>
   <portpool name="poolhub">20000-20099</portpool>
   <portpool name="poolwhp">20100-20199</portpool>
</tep:gateway>
```

The XML elements are explained here, in hierarchical order.

- The <gateway> element in the assigned namespace contains configuration elements. Note that this element *cannot* contain data.
  - The name attribute is required.

- The <portpool> element is used to create a list of local port numbers to be used for outbound connections.
  - The name attribute is required and must match with that of localport attribute of <bind> element.
- The <zone> element is a container of interfaces sharing communication resources. This element *cannot* contain data.
  - The name attribute is required to identify a specific zone instance.
- The <interface> element describes a set of network bindings that exhibit a fixed behavior according to a specified role. It defines an upstream interface when it is enclosed by the <zone> element. It defines a downstream interface when it is enclosed by the <interface> element.
  - The name attribute is required.
  - The ipversion attribute declares the address family to be used. The default value is 4.
  - The role attribute is required, and it describes the behavior of the network bindings. An upstream GWP uses the “proxy” role to represent local connecting endpoints.
- The <bind> element in an upstream proxy represents the local binding to be used for the outbound connection. There are two <bind> elements defined, one for the monitoring server service and the other one for WPA service.
  - The localport attribute for the “connect”-based role can only contain the name of a portpool element defined within the gateway. Two portpools have been defined, one for the monitoring server and the other one for WPA.
  - The service attribute is a character string used to represent a logical connection between downstream proxy and upstream proxy. This service name must match with that defined in the downstream proxy. There is no value restriction.
- The <connection> element is used to supply remote network interfaces as data. In “connect” mode (upstream GWP), the value represents the remote end of the connection.
  - The remoteport attribute supplies the default port number of remote interfaces. In this example, the WPA remote port is 63358.

Another <interface> element with the role of “connect” is defined inside <interface> that defines the downstream interface.

The <bind> element is then created to define the outbound connection. Its value is the “connect” mode interface (the upstream GWP's IP address). The localport attribute (optional) is the port used by the upstream GWP to create a connection to the downstream GWP.

Then another <connection> element enclosed by <bind> is created to define the end target of the downstream GWP. Its value is the IP address of the downstream GWP, and the remoteport attribute defines the port used by the downstream GWP, as illustrated by Figure 3-5.

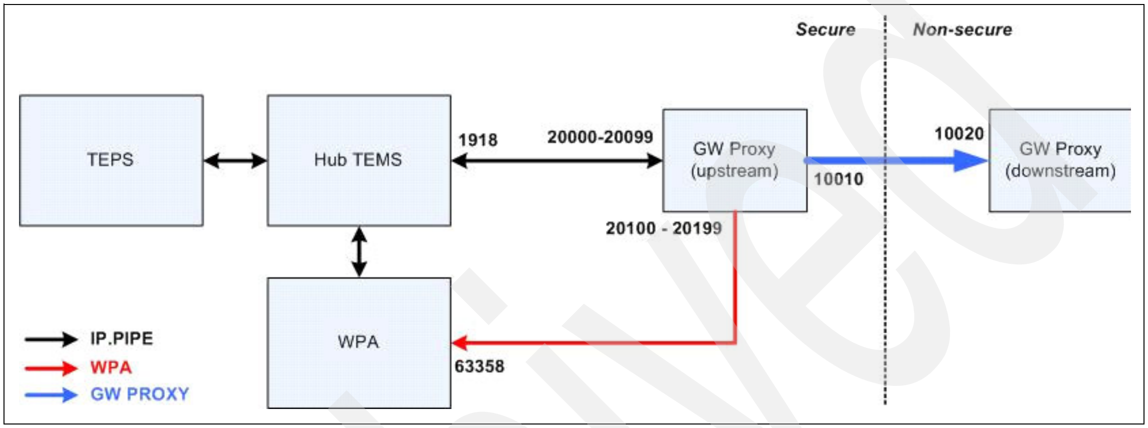


Figure 3-5 Sample upstream Firewall Gateway Proxy configuration

Example 3-6 displays the XML configuration of the downstream GWP.

Example 3-6 XML configuration of downstream GWP

```
<tep:gateway xmlns:tep=http://xml.schemas.ibm.com/tivoli/tep/kde/
name="downstream_gw">
   <zone name="dmz">
      <interface name="uprelay" ipversion="4" role="listen">
         <bind localport="10020">downstream_gw
            <connection remoteport="10010">HUB_hostname</connection>
         </bind>
         <interface name="serverproxy" ipversion="4" role="proxy">
            <bind localport="1918" service="tems"/>
            <bind localport="63358" service="whp"/>
         </interface>
      </interface>
   </zone>
</tep:gateway>
```

The downstream GWP configuration is explained here:

- The <gateway> and <zone> elements are the same as that of the upstream GWP. The only difference is in its names, to differentiate the function.
- The first <interface> defined in the <zone> element has a role of “listen”. The role describes the behavior of the network bindings contained within.
- The <bind> element in the “listen” mode binding represents listening ports on local interfaces. Specific local interface addresses can be supplied as data; the default interface is “any”.
  - The localport attribute defines the listening port used by the downstream GWP. This must match with the remortport attribute in the <connection> element defined in the upstream GWP configuration.
- The <connection> element in the “listen” mode binding represents the list of remote interface addresses that are allowed to make a connection, and its use is optional.
  - The remoteport attribute must match with that of the localport attribute in the <bind> element of the upstream GWP configuration.
- The <interface> element with the role of “proxy” defines the downstream proxy interfaces. It represents local listening endpoints.
- The <bind> element in the “listen” mode binding represents listening ports on local interfaces.
  - The localport attribute must match with the remoteport attribute of the <connection> element in the upstream GWP configuration.
  - The service attribute must match with the service attribute in the <bind> element of the upstream GWP configuration.

Figure 3-6 on page 52 illustrates the configuration of the downstream GWP, together with the upstream GWP.

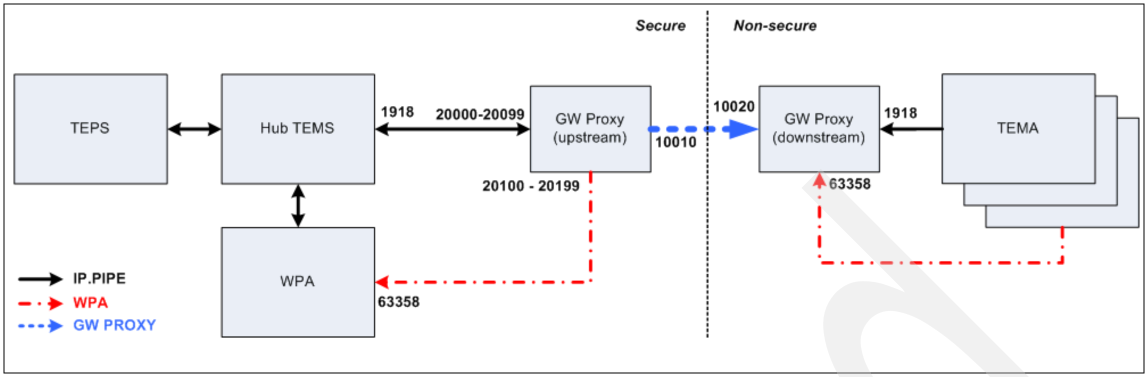


Figure 3-6 Sample upstream and downstream GWP configuration

Firewall Gateway Proxy with Network Address Translation

In a NAT firewall environment, shown in Figure 3-7, the IP address of the GW Proxy interface is translated to a specific IP address (static) or a random IP address (dynamic). In this case, the downstream GWP cannot determine which IP address is used by the upstream GWP to make a connection.

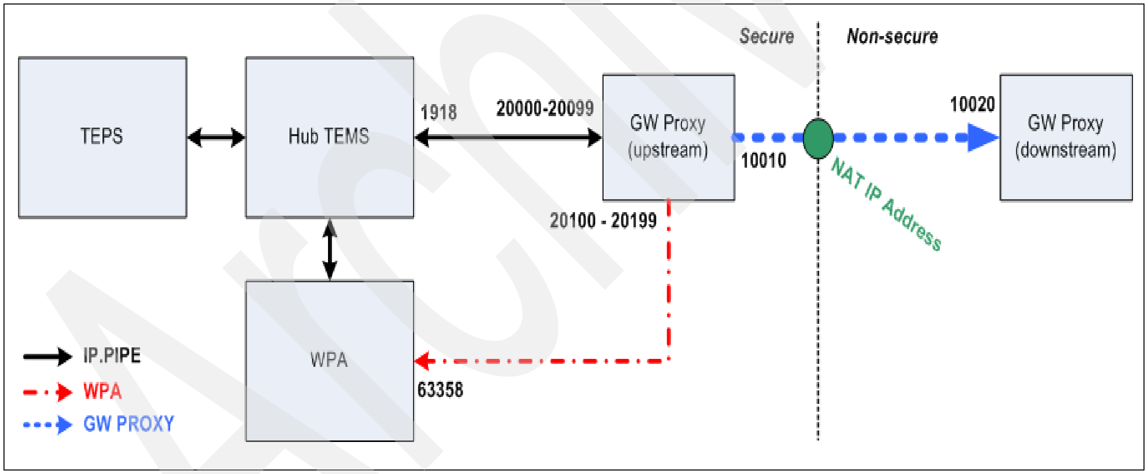


Figure 3-7 NAT firewall environment

There are two ways to configure the downstream GWP to support NAT environment, as described here:

- In a static NAT environment, change the IP address value of the <connection> element of the downstream proxy configuration with the specific IP address provided by the firewall, as illustrated in Example 3-7 on page 53.

Example 3-7 Static NAT environment

```
<tep:gateway xmlns:tep=http://xml.schemas.ibm.com/tivoli/tep/kde/
name="downstream_gw">
   <zone name="dmz">
      <interface name="uprelay" ipversion="4" role="listen">
         <bind localport="10020">downstream_gw
            <connection remoteport="10010">NAT_ip_address</connection>
         </bind>
         <interface name="serverproxy" ipversion="4" role="proxy">
            <bind localport="1918" service="tems"/>
            <bind localport="63358" service="whp"/>
         </interface>
      </interface>
   </zone>
</tep:gateway>
```

- In a dynamic NAT environment, the firewall will assign a random IP address to the upstream GWP. In this case, do *not* specify the IP address of the upstream GWP. Instead, allow the <connection> element to remain empty. See Example 3-8.

Example 3-8 Dynamic NAT environment

```
<tep:gateway xmlns:tep=http://xml.schemas.ibm.com/tivoli/tep/kde/
name="downstream_gw">
   <zone name="dmz">
      <interface name="uprelay" ipversion="4" role="listen">
         <bind localport="10020">downstream_gw
            <connection remoteport="10010"/>
         </bind>
         <interface name="serverproxy" ipversion="4" role="proxy">
            <bind localport="1918" service="tems"/>
            <bind localport="63358" service="whp"/>
         </interface>
      </interface>
   </zone>
</tep:gateway>
```

Firewall Gateway Proxy and a port-mapping firewall

In a port-mapping firewall, one or more IP addresses are assigned to create a connection from a secure network. However, different connections from different source will have the same IP address but a different source port. A different source port is used to differentiate the source of the connection when the result comes from a non-secure network. Figure 3-8 illustrates a port-mapping firewall environment.

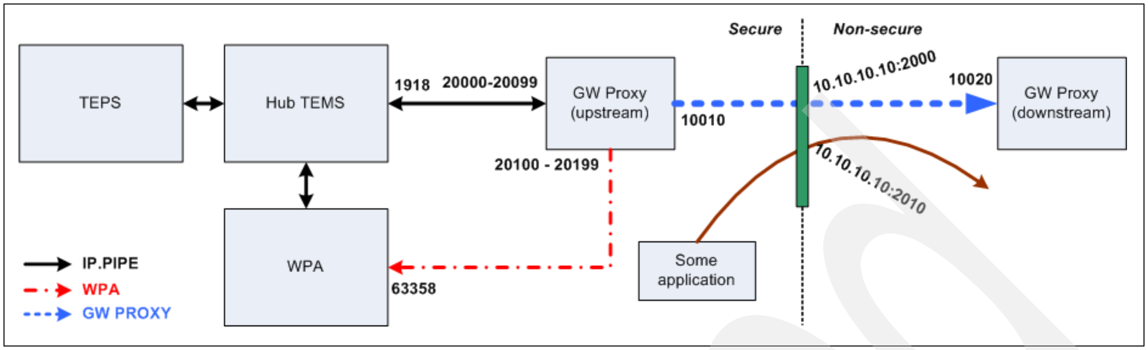


Figure 3-8 Port-mapping firewall environment

A downstream GWP cannot determine which IP address and port number is used by an upstream GWP to make a connection, so the only way is to take the whole <connection> element out of the configuration as illustrated in Example 3-9.

Example 3-9 Port mapping firewall configuration

```
<tep:gateway xmlns:tep=http://xml.schemas.ibm.com/tivoli/tep/kde/
name="downstream_gw">
   <zone name="dmz">
      <interface name="uprelay" ipversion="4" role="listen">
         <bind localport="10020">downstream_gw
            <connection remoteport="10010">HUB_hostname</connection>
         </bind>
         <interface name="serverproxy" ipversion="4" role="proxy">
            <bind localport="1918" service="tems"/>
            <bind localport="63358" service="whp"/>
         </interface>
      </interface>
   </zone>
</tep:gateway>
```

Multiple firewall environments

In multiple firewall environments, the downstream GWP has to be installed configured in each non-secure network segment. The first level downstream GWP (for example, the downstream GWP that is connected directly by the upstream GWP) creates a connection to the next level of downstream proxy.

In this environment, the multiple GWP creates a chain of relay systems and is responsible to relay IP.PIPE and WPA data from each non-secure network segment, as illustrated in Figure 3-9.

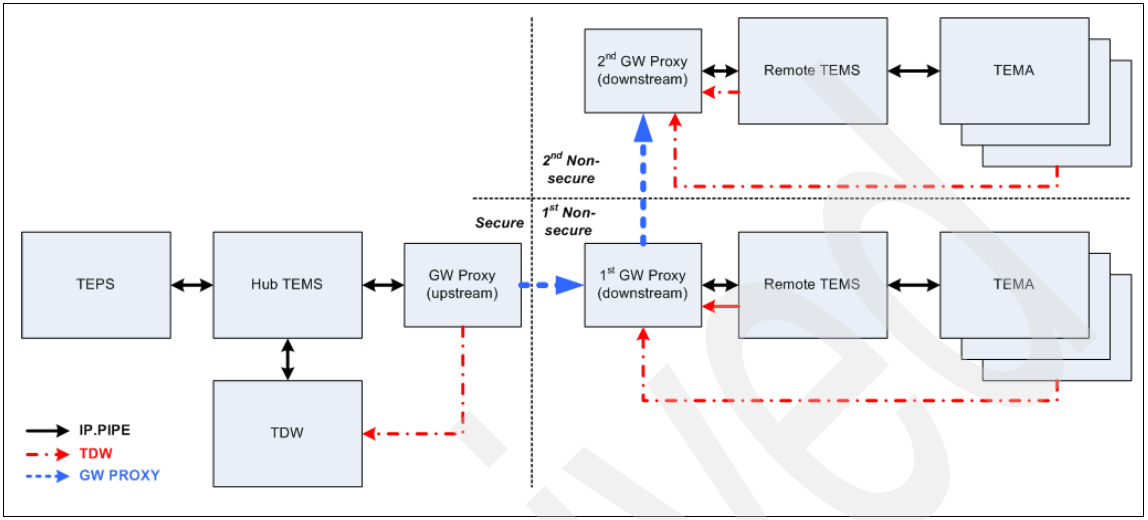


Figure 3-9 Multiple downstream GWP in multiple firewall environment

The first downstream GWP in the first non-secure network creates a GWP connection to the second downstream GWP in the second non-secure network, and extends the relay to reach IBM Tivoli Monitoring components in that network.

When the remote monitoring server in the second non-secure network wants to send data to the hub monitoring server, the second downstream GWP will relay the data to the first downstream GWP and in turn, it will relay the data to upstream GWP which will forward it to the hub monitoring server. It is a similar process to Tivoli Data Warehouse data export.

To continue with the sample configuration, simply change the downstream GWP configuration to include a relay connection to the newly installed second downstream GWP, as illustrated in Example 3-10.

Example 3-10 First downstream GWP

```
<tep:gateway xmlns:tep=http://xml.schemas.ibm.com/tivoli/tep/kde/
name="downstream_gw1">
   <zone name="dmz">
      <interface name="uprelay" ipversion="4" role="listen">
         <bind localport="10020">1st_downstream_gw
            <connection remoteport="10010">HUB_hostname</connection>
         </bind>
```

```
         <interface name="downrelay" ipversion="4" role="connect">
            <bind localport="10010">1st_downstream_gw
               <connection
remoteport="10020">2nd_downstream_gwp</connection>
            </bind>
         </interface>
         <interface name="serverproxy" ipversion="4" role="proxy">
            <bind localport="1918" service="tems"/>
            <bind localport="63358" service="whp"/>
         </interface>
      </interface>
   </zone>
</tep:gateway>
```

The <interface> element is added with the role of “connect” to extend the relay function to the second downstream GWP. The information in this element is very similar to that of the upstream proxy when it needs to create a connection to the first downstream GWP.

The <bind> element is again used to define the local hostname and port, which in this case is the hostname and port of the first downstream GWP. And <connection> is used to define the remote hostname and port of the second downstream GWP.

Example 3-11 The second downstream GWP

```
<tep:gateway xmlns:tep=http://xml.schemas.ibm.com/tivoli/tep/kde/
name="downstream_gw2">
   <zone name="dmz">
      <interface name="uprelay" ipversion="4" role="listen">
         <bind localport="10020">2nd_downstream_gw
            <connection
remoteport="10010">1st_downstream_gw</connection>
         </bind>
         <interface name="serverproxy" ipversion="4" role="proxy">
            <bind localport="1918" service="tems"/>
            <bind localport="63358" service="whp"/>
         </interface>
      </interface>
   </zone>
</tep:gateway>
```

The first <interface> element is defined for the “listen”, and role and <bind> is used to receive a connection from the first downstream GWP with the second downstream GWP hostname and port as local information. In the <connection>

element, the first downstream GWP hostname and port is used as remote information.

And in the interface with the role of <proxy>, localport and service attributes must match with that of the first downstream GWP and upstream GWP.

Figure 3-10 illustrates the overall sample configuration of a multiple firewall environment.

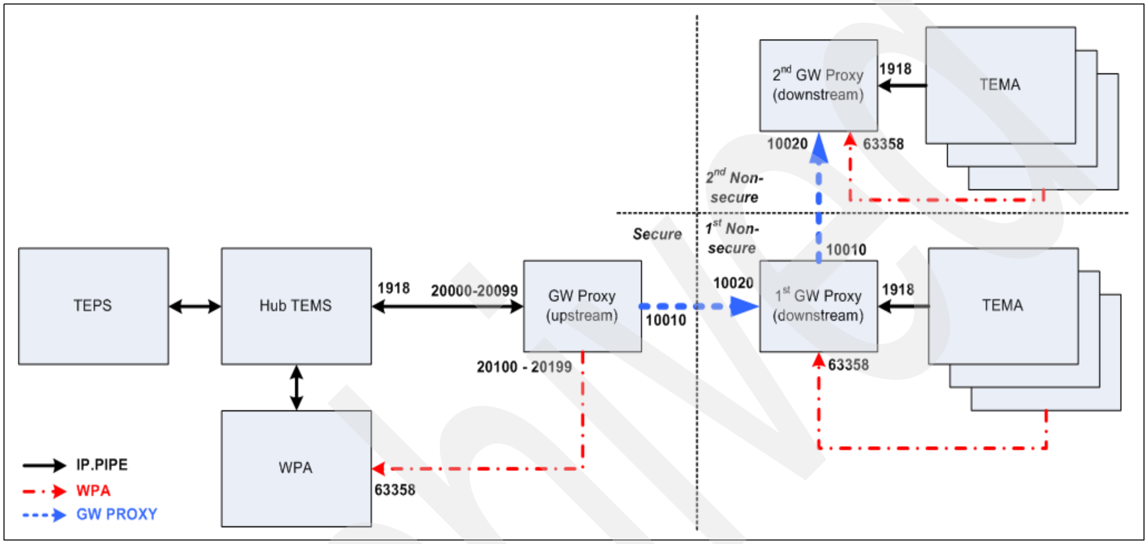


Figure 3-10 Sample GWP in a multiple firewall environment

3.3 Additional considerations

In this section we explain some additional considerations for setting up IBM Tivoli Monitoring V6.1 components across firewalls.

3.3.1 Multiple network interface cards

Whenever an IBM Tivoli Monitoring component starts up, by default it discovers all available network interfaces on the system and actively uses them. This may not always produce the desired results.

Consider, for example, a monitoring server with two networking interface cards (NICs): one interface connected to the main production network, and a second interface connected to a limited network that is used only for server backup.

When a monitoring agent on another system starts up and makes the first connection to the monitoring server using the Global Location Broker, it connects to the monitoring server first interface. Also, assume that the monitoring agent does not have an interface connected to the limited backup network segment.

The monitoring server sends a reply to the monitoring agent that contains the network address on which the monitoring server wants the monitoring agent to connect. This network address may be the NIC that is connected to the backup network. This results in the monitoring agent not being able to connect successfully, even though the initial handshake succeeded.

To avoid this problem, you can specify an environment parameter on all of the IBM Tivoli Monitoring components to force it to use a specific network interface, rather than using any available.

This can be accomplished by passing *either* of these keywords:

- KDCB0_HOSTNAME

  You can specify either the host name corresponding to the NIC to be used, or its IP address in dotted decimal format. If specified, it will take priority over the KDEB_INTERFACELIST parameter.

  Note: Keep in mind, however, that KDCB0_HOSTNAME should be used only in an environment *without* Network Address Translation (NAT), because it will also inactivate the use of the Ephemeral Pipe.

- KDEB_INTERFACELIST

  The NIC must be specified as dotted decimal IP addresses. This keyword is recommended to use when IBM Tivoli Monitoring is installed in a environment with NAT.

3.3.2 Universal Agent firewall considerations

The Universal Agent is similar to any other agent. The Universal Agent connects to the monitoring server using your selected protocol (IP.PIPE, IP.UDP, or IP.SPIPE). However, in addition to the communication between the Universal Agent and the monitoring server, there can also be communications from the Universal Agent to the monitored system.

There are three data providers that frequently have remote connections (ODBC, SNMP, and Socket). You can have remote connections with other data providers (such as the API Data Provider), as well.

When you are in an environment with firewalls, you must ensure that you can establish communications between the monitored system and the Universal Agent.

- For SNMP, the default port is 161. The Universal Agent contacts the managed system on this port.
- For SNMP traps, the Universal Agent listens on port 162.
- For ODBC, the default port varies depending on the database server. For example, the port for DB2 is 50,000.
- For the Socket Data Provider, the data collection program writes to a socket on the Universal Agent using default port of 7500.

3.3.3 Warehouse Proxy agent firewall considerations

The KPX_WAREHOUSE_LOCATION variable can be set on the WPA. It allows a fixed warehouse route via a Warehouse Proxy agent to be delivered to connected agents. The variable contains an optional list of fully qualified network names delimited with semicolons (;).

When specified, this variable disables Proxy monitoring of the Warehouse Proxy location broker registrations. Example 3-12 illustrates the use of this variable.

Example 3-12 KPX_WAREHOUSE_LOCATION variable

```
KPX_WAREHOUSE_LOCATION=ip.pipe:#1.1.1.1[18303];ip:#1.1.1.1[34543]
```

For more information about the KPX_WAREHOUSE_LOCATION variable and Warehouse Proxy location broker registration, refer to 4.4.2, “Configure multiple Warehouse Proxy agents” on page 106.

3.3.4 Service Console ports

The Service Console can be useful when performing simple updates and administration, and for diagnosing each IBM Tivoli Monitoring application remotely. However, using the Service Console across a firewall can be difficult.

When starting the Service Console, users will bind to two ports for HTTP and HTTPS, and these are by default 1920 and 3660. Next, the Service Console application will bind to “opaque” ports, and when a browser is pointed to the base HTTP port of 1920, for example, it is redirected to one of the opaque ports. The same would occur for HTTPS connections. These opaque ports by default use a free port provided by the TCP stack.

Note: An *opaque* port is a random port returned by the TCP stack. It is similar to an *ephemeral* port, except that the definition of the word "ephemeral" infers a brief or short life, but these ports exist for as long as the agent is active.

We can control both the HTTP and HTTPS, as well as all opaque ports used in each agent configuration file, as illustrated in Example 3-13.

Example 3-13 ux.ini file

```
KDC_FAMILIES= IP.PIPE PORT:1918 IP.SPIPE PORT:3660 IP.UDP USE:N IP.TCP
POOL:60000-61023 HTTP:1920 IP.SSL POOL:61024-62047 HTTPS:3660 SNA.PIPE
USE:N
```

4

Planning historical data collection in large scale environments

This chapter discusses several points regarding planning the implementation of historical data collection in large scale environments.

The following topics are addressed:

- Short-term versus long-term historical data
- Infrastructure considerations
- Size the Tivoli historical data collection
- Reduce the number of agents for historical data collection
- Other planning considerations for collection historical data

4.1 Short-term versus long term historical data

IBM Tivoli Monitoring is delivered with an integrated feature for collecting historical data. Before delving into the details of historical data collection, we will clarify some concepts. The following definitions are useful in understanding how historical data is stored and accessed.

Collection location Specified in the dialog box Historical Collection Configuration, this indicates whether the attribute group data should be stored on the monitoring agent (TEMA) or at the monitoring server (TEMS) to which the agent is connected (see Figure 4-1 on page 63).

Short-term historical data Data residing at the collection location (either monitoring agent or monitoring server).

Long-term historical data Data residing in the Tivoli Data Warehouse.

The distinction between short-term and long-term historical data is made by the portal server, and is invisible to the portal client. When a portal client requests historical data from an agent, the portal server retrieves the short-term historical data from the collection location that is configured for the attribute group, and retrieves long-term historical data from the Tivoli Data Warehouse.

4.1.1 Short-term historical data

Short-term historical data is stored in binary files (or in partioned data sets on IBM z/OS systems) on the agent or the directly connected monitoring server, depending on the configuration. If you have activated the historical data collection but have *not* selected a Warehouse interval, no data will be uploaded into the Warehouse database.

In this case, the short-term history files will grow until they eventually fill up the disk. There are *roll-off* programs, described in *IBM Tivoli Monitoring Version 6.1.0 Administrator's Guide*, SC32-9408, which can be used to "prune" the agent files, but this requires action on the part of the administrator.

If you have set the warehouse interval, the short-term historical data will be uploaded into the Warehouse database via the Warehouse Proxy agent. The frequency of this process depends on the selected Warehouse interval, either hourly or daily. There is no command =-line to set this interval; this must be done via the dialog.

Important: If the assigned Warehouse Proxy is unavailable, the short-term historical data will be not deleted, so you can have historical data at the collection location older than 24 hours.

In this case, as long as there is enough space on the local disk, the data will accumulate until it is successfully uploaded.

Figure 4-1 displays the historical data collection dialog box with selections for collecting data from the attribute group NT Memory and selections for summarization and pruning. In this case, the Warehouse interval is switched to Off. That means that for these attribute groups, only short-term data will be collected.

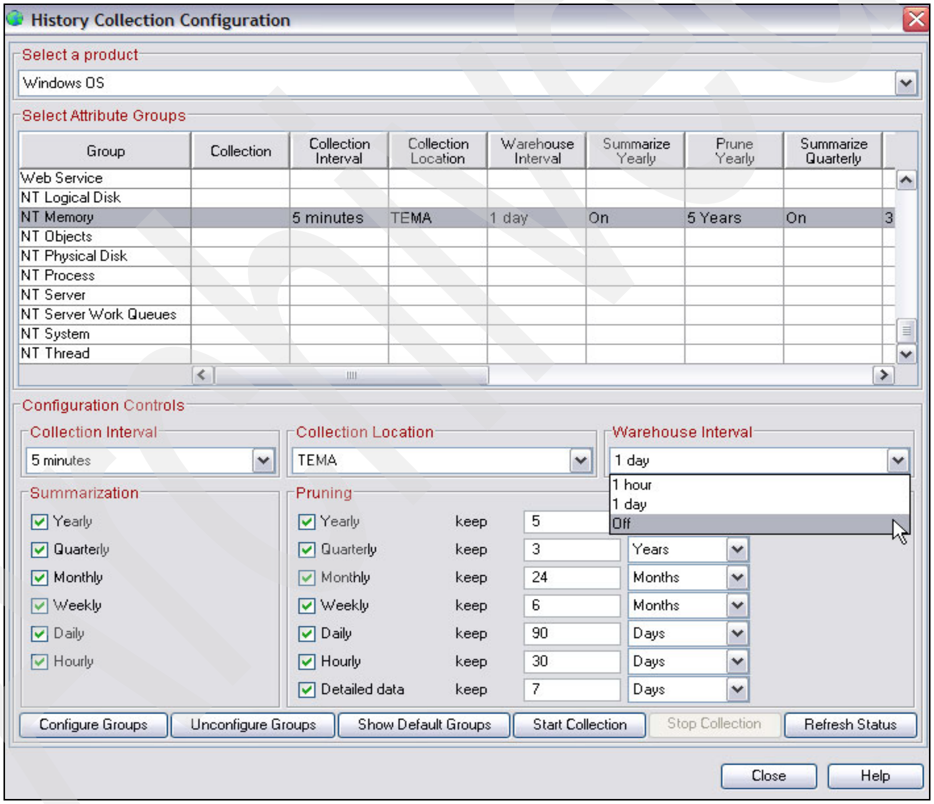


Figure 4-1 Historical data collection dialog box with selection for the Warehouse interval

Short-term data can be configured to store the binary files locally on the monitoring agent (TEMA), or on the monitoring server to which the agent is connected (TEMS). You can configure which directory will be used for storing the data (using the CTIRA_HIST_DIR variable on the agents or monitoring server configuration file). The collection location can be configured by agent type within the Historical Data Collection dialog box. Note that the binary data is considered “short term” because it is only designed for 24-hour access.

It is recommended that you use agent as collection location for the binary files. Selecting monitoring server as the collection location can have performance implications for the data gathering step in large scale environments with several hundred agents connected to one monitoring server.

If there is little space on the logical disk or file system of some agents, then you can use other options like minimizing the Warehouse interval. In an environment with a slow network connection between the agent and the monitoring server, collecting short-term historical data at the monitoring server (rather than at the agent) could be used as a method to improve response time for the portal client requests for short-term historical data.

However, the probability that the short-term data would be viewed by a portal client may be low, and the cost of transmitting the historical data over a slow connection would be incurred regardless of whether the data was ever viewed. Storing short-term historical data at the monitoring server significantly lowers the number of agents that can be managed by a monitoring server.

> **Important:** Performing an accurate plan and assessment stage is imperative for the Tivoli Data Warehouse implementation. Mapping the component topology with the recommended hardware specifications is critical to achieving a highly distributed environment with realistic goals.
>
> Thus, it is important to have a thorough understanding of the monitoring environment before you proceed to implementing any architectural design. It is also important to take into account all the variables within the topology. Substantial consideration must be given to the infrastructure hardware requirements and the underlying network topology, network bandwidth, latency, and firewall restriction require assessment.

4.1.2 Long-term historical data

Long-term historical data is stored in the Tivoli Data Warehouse only. In this case, you have to create and configure a Tivoli Data Warehouse database, configure one or more Warehouse Proxy agents and the Summarization and Pruning agent (optional, but recommended), and select a Warehouse interval in the Historical Data configuration dialog box for one or more attribute groups.

In this case, when the warehousing interval expires, detailed data records are uploaded into the Warehouse database from the collection location. The next time the Summarization and Pruning agent runs, it will summarize the detailed data and create the appropriate aggregate data records.

> **Note:** The location of the data, and whether it is long-term or short-term, is transparent to the portal user.

Figure 4-2 shows all components for the historical data collection data flow and the flow for portal client requests for real-time data, short-term historical data, and long-term historical-data.

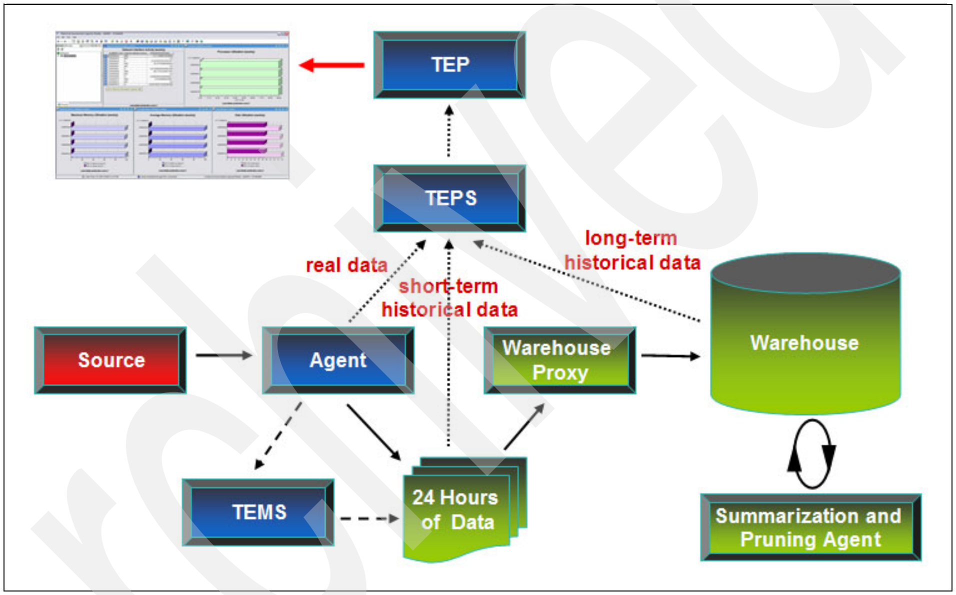


Figure 4-2 Historical data flow

4.1.3 KFW_REPORT_TERM_BREAK_POINT parameter

The portal server environment parameter KFW_REPORT_TERM_BREAK_POINT specifies the point in time, in seconds, offset from the current time where a request for historical data is selected from short-term or long-term history data. The default setting is 86400; that is, short-term history data to be collected from now to 24 hours ago, and long-term from 24 hours onward.

Increasing this value will result in historical data being retrieved from the short-term data store for time periods further in the past. Decreasing this value will result in historical data being retrieved from the Tivoli Data Warehouse for more recent time periods. Set to zero (0) to only select from long-term history data.

You can insert the following before the second paragraph:

The portal server environment parameters are specified in the following files according to operating system:

- Windows

  <ITM HOME>\CNPS\kfwenv

  Example: C:\IBM\ITM\CNPS\kfwenv

- Linux and UNIX

  <ITM HOME>/config/cq.ini

  Example: /opt/IBM/ITM/config

For most environments, the default setting should be used. You might consider modifying this setting in certain cases, such as:

- Storage space on the collection location (agent or monitoring server) is at a premium.
- There is a desire to route more of the portal client historical data requests to the warehouse versus the monitoring agents.

  For example, you may want to set a shorter value for KFW_REPORT_TERM_BREAK_POINT if you have a large amount of data being warehoused using a setting of less than 24 hours for the warehouse proxy scheduling. Alternatively, if the warehouse scheduling is set to a time period greater than 24 hours, then consider increasing the value of this parameter in order to match this.

Note: Be aware that KFW_REPORT_TERM_BREAK_POINT is a global setting that affects *all* history reports and *all* products installed the hub environment. Changing this value will affect response time for the portal client requests for historical data. Whether the change in response time is positive or negative will depend on numerous factors, including the network configuration between the IBM Tivoli Monitoring components and the warehouse configuration.

The portal server KFW_REPORT_TERM_BREAK_POINT parameter is closely related to the KHD_HISTRETENTION parameter, which is specified at the collection location (either in the monitoring server parameter file or the monitoring agent parameter file). See 4.1.4, “KHD_HISTRETENTION parameter” on page 67 for more details.

If KFW_REPORT_TERM_BREAK_POINT is not added to the portal server environment and the KHD_HISTRETENTION variable has been used to customize the number of hours of locally retained historical data, the following error could be seen in the portal server RAS1 logs:

```
"CTSQLEvaluatorODBC_i::Connection::sqlErrorCheck") SQLAllocHandle
STMT rc=-1: SQL_ERRORSQLAllocHandle STMT rc=-1: SQL_ERROR
```

4.1.4 KHD_HISTRETENTION parameter

This historical data collection parameter is used to define, in hours, how long short-term historical data is kept within binary files at the collection location. The parameter value is specified in the environment parameter file for the monitoring agent or monitoring server, whichever is configured as the collection location.

Note: KHD_HISTRETENTION does not apply to z/OS systems.

KHD_HISTRETENTION is only used when warehousing is *enabled*. This setting is checked after data has been successfully uploaded to the warehouse. The default setting is to keep 24 hours of short-term historical data.

If warehousing is not being used, then other steps need to be taken to prune the short-term historical files; otherwise, these files will eventually fill up the disk. Refer to 4.1, “Short-term versus long term historical data” on page 62 for more information.

For most environments, the default KHD_HISTRETENTION setting should be used. The setting can be changed if necessary.

Possible reasons for changing this parameter might be:

- Storage space on the collection location (agent or monitoring server) is at a premium.
- There is a desire to route more of the TEP client historical data requests to the warehouse versus the monitoring agents.

  The KHD_HISTRETENTION setting and the portal server KFW_REPORT_TERM_BREAK_POINT parameter setting are related, and great care should be taken if you decide to change them from the default values.

  Keep the following points in mind in setting these values:

  – KFW_REPORT_TERM_BREAK_POINT is a *global* setting. If you decide to change these values, set KHD_HISTRETENTION to the same value at every historical data collection location (TEMA or TEMS).

  – KFW_REPORT_TERM_BREAK_POINT is set in seconds. By contrast, KHD_HISTRETENTION in hours.

  – If the warehousing interval is set to “daily”, it may not make sense to change the KHD_HISTRETENTION and KFW_REPORT_TERM_BREAK_POINT settings.

  – Ideally, the KHD_HISTRETENTION should specify an interval equivalent to the portal server KFW_REPORT_TERM_BREAK_POINT parameter. For example, if the KHD_HISTRETENTION variable for the agents has been set to 12, then the KFW_REPORT_TERM_BREAK_POINT variable should be set to 43200, which is the number of seconds in 12 hours.

  – If KHD_HISTRETENTION specifies a longer interval than KFW_REPORT_TERM_BREAK_POINT, data will be kept at the collection location that cannot be accessed from the portal client.

  – If KHD_HISTRETENTION specifies a shorter interval than KFW_REPORT_TERM_BREAK_POINT, the portal client will not be able to display data that is older than the KHD_HISTRETENTION setting, but younger than the KFW_REPORT_TERM_BREAK_POINT.

  – There is a 2 GB size limit on the binary files used for short-term historical data collection. If you expect the short-term files to approach the 2 GB limit, setting a shorter KHD_HISTRETENTION interval may be necessary. Refer to “Size the Tivoli historical data collection” on page 80 for more information about estimating the size of the short-term files and reducing the volume of historical data collection.

– If there is a mismatch between the KFW_REPORT_TERM_BREAK_POINT and KHD_HISTRETENTION values, the following error could be seen in the portal server RAS1 logs:

```
“CTSQLEvaluatorODBC_i::Connection::sqlErrorCheck") SQLAllocHandle
STMT rc=-1: SQL_ERRORSQLAllocHandle STMT rc=-1: SQL_ERROR
```

4.2 Infrastructure considerations

In this section we discuss infrastructure considerations for implementing historical data collection in large scale environments.

4.2.1 Tivoli Data Warehouse Version 2.1

The Tivoli Data Warehouse Version 2.1 is a row-based database within a relational database management system (RDBMS). It includes detailed data and summarized data within the same schema, but in different tables.

Table 4-1 lists vendor types and versions for databases that can be used for implementation of the Tivoli Warehouse database.

Table 4-1 Warehouse database vendor and platform types

	IBM DB2 UDB	**Oracle®**	**MS SQL**
	► V8.1, Fix Pack 10 and higher Fix Packs ► V8.2, Fix Pack 3 and higher Fix Packs ► V9.1 (32-bit only) and Fix Packs (since IBM Tivoli Monitoring V6.1 Fix Pack 4)	► V9.2, 10g Release 1 ► 10g Release 2	► 2000 EE ► 2005 (for both: portal server supports connecting to a MS SQL Warehouse database only when the portal server is running on Windows)
Windows	2003 Server	2003 Server	2003 Server
AIX	V5.3	V5.3	-

	IBM DB2 UDB	Oracle®	MS SQL
	► V8.1, Fix Pack 10 and higher Fix Packs ► V8.2, Fix Pack 3 and higher Fix Packs ► V9.1 (32-bit only) and Fix Packs (since IBM Tivoli Monitoring V6.1 Fix Pack 4)	► V9.2, 10g Release 1 ► 10g Release 2	► 2000 EE ► 2005 (for both: portal server supports connecting to a MS SQL Warehouse database only when the portal server is running on Windows)
Sun™	10	10 (see on www.oracle.com for information about installing and configuring)	-
Linux	SUSE Linux Enterprise Server 9 and 10 for Intel, RedHat Enterprise Linux 4 for Intel	SUSE Linux Enterprise Server 9 and 10 for Intel, RedHat Enterprise Linux 4 for Intel	-

Note: A license of IBM DB2 UDB is shipped as part of the IBM Tivoli Monitoring package. This DB2 license should only be used for IBM Tivoli Monitoring purposes.

The Tivoli Warehouse Proxy agent is a special agent that handles warehouse requests from all managed systems in the IBM Tivoli Monitoring environment. It creates one table in the Warehouse database for every collected attribute group. The collected historical record of the agent generates a new row in the table. For access to the database, the Warehouse Proxy agent uses an ODBC or a

JDBC™ connection, depending on the platform where the Warehouse Proxy agent resides.

> **Important:** Starting with IBM Tivoli Monitoring V6.1 Fix Pack 2, multiple Warehouse Proxy agents can be configured in a single IBM Tivoli Monitoring environment. Also IBM Tivoli Monitoring V6.1 Fix Pack 2 provides Linux and AIX support for Warehouse Proxy agents. All configured Warehouse Proxies must export the data into one Warehouse database.
>
> This support must be used in large scale environments, depending on the number of agents and protocols used by agents. Refer to *IBM Tivoli Monitoring Version 6.1 Installation and Setup Guide*, GC32-9407, and the Fix Pack addendum for these requirements.

The Warehouse Proxy agent is a special kind of agent; its product code is hd. The Warehouse Proxy agent files are located in the agents directory. The configuration settings are included in the configuration files, where ITMinstall_dir is the directory where the product is installed:

- (Windows) ITMinstall_dir\TMAITM6\KHDENV
- (UNIX or Linux) ITMinstall_dir/config/hd.ini

The basic configuration settings are set in the agents configuration dialog. On Windows, you can also edit the Warehouse Proxy agent configuration file through the “Manage Tivoli Enterprise Monitoring Services” window. Select the **Warehouse Proxy** in the window and click with your right mouse key **Advanced** → **Edit ENV-File**. Additional options for the Warehouse Proxy agent are discussed elsewhere in the book.

All agent log files are located in the agents log directory. Note that the product code is *not* the identifier; instead, the files can be identified with the identifier Warehouse within the name. Note the following example, where BRUGE is short-name of the system where the Warehouse Proxy agent is installed:

- BRUGE_Warehouse.LG0

Important: In IBM Tivoli Monitoring V6.1 Fix Pack 6, records describing each agent attribute group export are no longer written to the Warehouse Proxy agent operations file (Warehouse.LG0). The same information can be found in the WAREHOUSELOG table within the warehouse.

For example, you can issue the following SQL statement to get the equivalent information formerly found in the Agent Operations file:

```
select ORIGINNODE,OBJECT, EXPORTTIME, ROWSINSERTED, ERRORMSG,
WPSYSNAME from WAREHOUSELOG order by EXPORTTIME desc
```

The results from this query show the inserts performed by the Warehouse Proxy Agent, with the most recent inserts sorted to be at the top of the list.

Table 4-2 on page 73 lists the available platforms for installation of the Tivoli Warehouse proxy.

Table 4-2 Supported platforms for the Warehouse Proxy agent

Windows	UNIX	Linux
► Windows 2000 Server and Advanced Server ► Windows 2003 Server SE(32 bit) and EE (32 bit) with Service Pack 1 or Microsoft® Installer 3.1 (KB893803) ► Windows 2003 on VMWare ESX Server V2.5.2 and V3.0 (with IBM Tivoli Monitoring V6.1 FIx Pack 3) ► XP (for promotion only)	► AIX V5.3 (32-bit and 64- bit) (with IBM Tivoli Monitoring V6.1 Fix Pack 3)	► RedHat Enterprise Linux 3 for intel, zSeries® 31-bit and zSeries 64-bit ► RedHat Enterprise and Desktop Linux 4 intel ► RedHat Enterprise Linux 4 on zSeries 31- bit and zSeries 64-bit ► SUSE Linux Enterprise Server 8 for Intel, z/Series 31-bit and zSeries 64-bit ► SUSE Linux Enterprise Server 9 for Intel, z/Series 31-bit and zSeries 64 bit (all IBM Tivoli Monitoring V6.1 Fix Pack 2) ► RedHat Enterprise Linux 4 for Intel on VMWare ESX Server V2.5.2 (with IBM Tivoli Monitoring V6.1 Fix Pack 3) ► SUSE Linux Enterprise Server 10 for Intel and z/Series 64-bit (with IBM Tivoli Monitoring V6.1 Fix Pack 4) ► Asianux 2.0 for Intel ► Red Flag 5.1 for Intel ► RedHat Enterprise and Desktop Linux 5 Intel ► RedHat Enterprise Linux 5 for z/Series 64-bit (all IBM Tivoli Monitoring V6.1 Fix Pack 5

Note: When configuring one or more Warehouse Proxy agents, connect all of them to the hub monitoring server, not to a remote monitoring server. Connecting a warehouse proxy to a remote monitoring server results in incorrect connections.

4.2.2 Summarization and Pruning agent

The Summarization and Pruning agent is a special agent that manages the data in the Tivoli Data Warehouse using a JDBC connection. This agent summarizes the data, creates the necessary aggregated tables and views tables, and prunes the detailed and summarized data according to the configuration for an attribute group. The agent works only once in every 24 hours, as scheduled in the agents configuration dialog.

Note: In IBM Tivoli Monitoring V6.1 Fix Pack 6 and also in IBM Tivoli Monitoring V6.2, you will be able run Summarization and Pruning agent more than once in every 24 hours by using a flexible scheduling option.

If there are multiple Tivoli Monitoring environments sharing the same Tivoli Data Warehouse, only one Summarization and Pruning agent can be used to manage it.

Important: When you have two distinct IBM Tivoli Monitoring installations using one Tivoli Data Warehouse database, both installations should use the same user ID to connect to the Tivoli Data Warehouse database, because this user ID will then become the first part name of all the Tivoli Data Warehouse tables. Having two user IDs prevents querying with one unique query information for all the systems of the same attribute group in the entire enterprise, because two tables with two different first part will exist.

There is no problem for managing data from different agents of different environments, because the management will be done for attribute groups only.

The product code of the Summarization and Pruning agent is sy. The files are located in the agents directory. The configuration settings are included in the configuration files, where ITMinstall_dir is the directory where the product is installed.

- (Windows) ITMinstall_dir\TMAITM6\KSYENV
- (UNIX or Linux) ITMinstall_dir/config/sy.ini

The basic configuration settings are performed from the agent's configuration dialog. When the configuration file needs to be edited, you can use the Manage Tivoli Enterprise Monitoring Services window. Select the **Warehouse Summarization and Pruning Agent** in the Manage Tivoli Enterprise Monitoring Services window and click with your right mouse key **Advanced** → **Edit ENV-File**. The agent log files are located in the agents log directory with the product code as identifier.

Table 4-3 on page 76 lists the available platforms used for installation of the Summarization and Pruning agent.

Table 4-3 Supported platforms for the Summarization and Pruning agent

Windows	Unix	Linux
► Windows 2000 Server and Advanced Server ► Windows 2003 Server SE (32-bit) and EE (32-bit) with Service Pack 1or Microsoft Installer 3.1 (KB893803) ► Windows 2003 on VMWare ESX Server V2.5.2 and V3.0 (with IBM Tivoli Monitoring V6.1 Fix Pack 3) ► Windows 2000 Professional and XP (for promotion only)	► V5.1 (32- and 64-bit) ► V5.2 (32- and 64-bit) ► V5.3 (32- and 64-bit) ► Solaris™ V8 (32-bit and 64-bit) ► Solaris V9 (SPARC) ► Solaris V10 (SPARC) ► Solaris Zones (with IBM Tivoli Monitoring V6.1 Fix Pack 5)	► RedHat Enterprise Linux 2.1 for Intel ► RedHat Enterprise and Desktop Linux 4 Intel (all with IBM Tivoli Monitoring V6.1) ► RedHat Enterprise Linux 3 for Intel, zSeries 31-bit and zSeries 64-bit ► RedHat Enterprise Linux 4 on zSeries 31-bit and zSeries 64-bit ► SUSE Linux Enterprise Server 8 for Intel, zSeries 31-bit and zSeries 64-bit ► SUSE Linux Enterprise Server 9 for Intel, zSeries 31-bit and zSeries 64-bit (all IBM Tivoli Monitoring V6.1 Fix Pack 2) ► RedHat Enterprise Linux 4 for Intel on VMWare ESX Server V2.5.2 (with IBM Tivoli Monitoring V6.1 Fix Pack 3) ► SUSE Linux Enterprise Server 10 for Intel and z/Series 64-bit (with IBM Tivoli Monitoring V6.1 Fix Pack 4) ► Asianux 2.0 for Intel ► Red Flag 5.1 for Intel ► RedHat Enterprise and Desktop Linux 5 Intel ► RedHat Enterpr. Linux 5 for z/Series 64-bit (all IBM Tivoli Monitoring V6.1 Fix Pack 5)

Important: When using MS SQL server for the Warehouse Summarization and Pruning agent, install the MS SQL 2005 JDBC driver. This is true as well when configuring the Warehouse Summarization and Pruning agent on UNIX and connected to a MSSQL database.

With the previous drivers, the agent might fail to run at the scheduled time on Windows computers because of a limitation of the number of tables it can retrieve. The MS SQL 2005 JDBC driver addresses this limitation.

You can download the JDBC driver from the Microsoft downloads Web site at: `http://www.microsoft.com/downloads`.

Summarization and Pruning agent enhancements

A few functional enhancements were added in Fix Pack 6 to the Summarization and Pruning agent, as described here. These will also be part of the IBM Tivoli Monitoring V6.2 release.

- The Summarization and Pruning agent can be configured to prune old entries from the WAREHOUSELOG and WAREHOUSEAGGREGLOG tables.

  Prior to this change, these logs were not pruned and required user intervention to remove old entries. The retention period for these logs can be specified in the new Log Parameters tab in the Summarization and Pruning configuration dialog, as shown in Figure 4-3 on page 78.

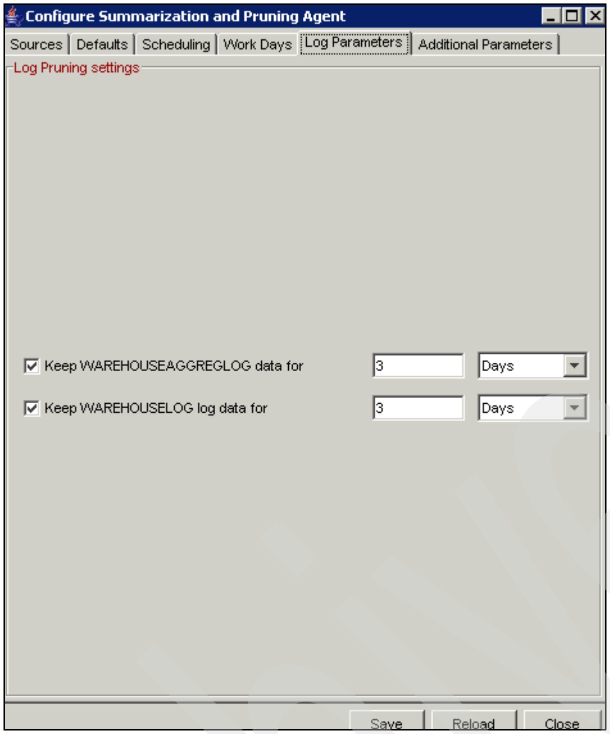


Figure 4-3 Summarization and Pruning agent enhancements

The new log table pruning parameters can also be specified in the Summarization and Pruning Agent configuration file (KSYENV on Windows, sy.ini on Unix/Linux). The following variables need to be added to the configuration files:

– KSY_WAREHOUSELOG_PRUNE=
– KSY_WAREHOUSEAGGREGLOG_PRUNE=

Both variables are expected to be in the format of number.unit, where number is the number of units to keep the data in the database and unit is one of day, month, or year. For example, to keep three days of data, specify:

KSY_WAREHOUSELOG_PRUNE=3.day

KSY_WAREHOUSEAGGREGLOG_PRUNE=3.day

- Prior to IBM Tivoli Monitoring V6.1 Fix Pack 6, the Summarization and Pruning Agent was started using a fixed schedule; for example, once every day starting at 2:00 a.m.

  With IBM Tivoli Monitoring V6.1 Fix Pack 6, a new, flexible schedule option is available. To enable flexible scheduling, the following variables are added to the Summarization and Pruning Agent environment variable file (KSYENV on Windows or sy.ini on UNIX/Linux):

– KSY_FIXED_SCHEDULE

This controls whether to run using the existing mechanism (Y) or the new flexible scheduling (N).

– KSY_EVERY_N_MINS

This controls the frequency of executing the Summarization and Pruning function, and respecting the exception periods provided by KSY_BLACKOUT. This is only used when KSY_FIXED_SCHEDULE=N.

– KSY_BLACKOUT

This lists blackout periods when the Summarization and Pruning function should not run in the format HH:MM-HH:MM, with multiple values separated by a comma (,). This is only used when KSY_FIXED_SCHEDULE=N.

For example, to set up a flexible schedule to run the Summarization and Pruning agent every 60 minutes, but block it from running between 00:00 and 01:59 and between 04:00 and 04:59, you could use the following statements:

– KSY_FIXED_SCHEDULE=N

– KSY_EVERY_N_MINS=60

– KSY_BLACKOUT=00:00-01:59,04:00-04:59

Important: Note the following points:

- “Fixed Summarization and Pruning agent configuration” utilizes the KSY_EVERY_N_DAYS variable.
- “Flexible Summarization and Pruning agent” scheduling utilizes "KSY_EVERY_N_MINS".

Therefore, KSY_FIXED_SCHEDULE should be turned off and thus set to N in the example given.

These values can also be set with the Summarization and Pruning agent configuration dialog on the Scheduling tab, as shown in Figure 4-4 on page 80.

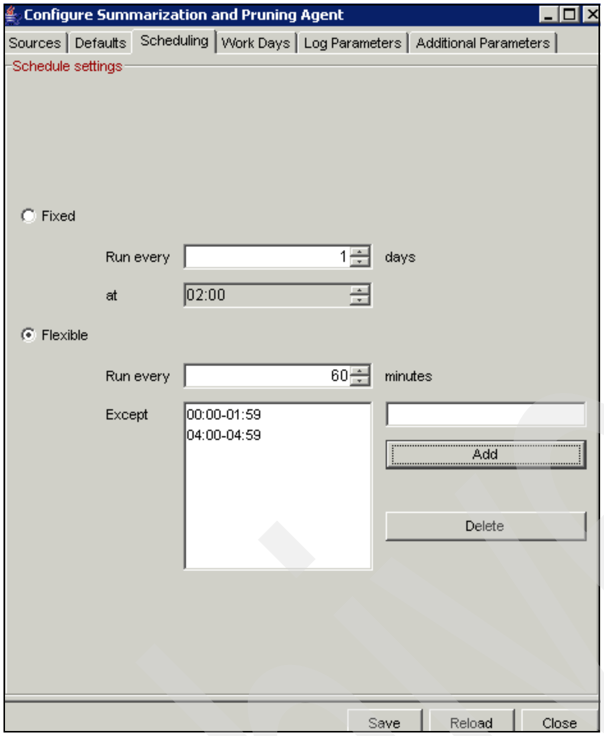


Figure 4-4 Summarization and Pruning agent enhancements

4.3 Size the Tivoli historical data collection

It is very important to estimate the size needed for the Warehouse database for planning the collection of long-term historical data in large scale environments. There are not many differences between the types of database vendors, in terms of size requirements.

There are several important customizations that can affect the database size:

- Attribute groups to be selected for data collection
- How long the data will be kept - pruning settings
- Number of agents
- Data collection interval

The attributes groups that are selected should be based on the reporting and visualization needs, such as the types of historical workspaces or types of reports you have to create. An important aspect is the consideration of the number of instances an attribute group can have. The list of the necessary attribute groups and the summarized data for these attribute groups should go into the calculation of the database size.

The length of time that the data will be kept in the database should also be taken into account. Note that often the most amount of data comes from the detailed data, but after summarized data has been created from the detailed data, usually summarized data (not the detailed data) is used in the reports. However, the detailed data is held in the database for a long time. Therefore, you need to make sure that you set the proper summarization and pruning settings in the historical data collection dialog, as shown in Figure 4-1 on page 63.

Important: We strongly discourage you from turning on the default historical data collection. In fact, in IBM Tivoli Monitoring V6.2, the default settings tab in the Summarization and Pruning agent configuration dialog box will be removed.

The reason for this is because settings for every single attribute group should depend on the reporting and visualization requirements. Therefore, a default value should not be used.

The data collection interval for an attribute group can be specified as either 5, 15, 30 or 60 minutes. (In IBM Tivoli Monitoring V6.2, a 1-minute data collection will also available.)

In configuring historical data collection for an agent attribute group, specify the longest data collection interval that will still provide useful data. More frequent data collection results in more rows to be inserted into the warehouse, more database storage usage, and more work for the Summarization and Pruning agent. Using a 5-minute data collection interval generates three times as many rows as a 15-minute interval, and 12 times as many rows as an hourly data collection interval.

4.3.1 Use the Warehouse Load Projections spreadsheet to estimate and control the Tivoli Warehouse database size

IBM Tivoli Monitoring Installation and Setup Guide, GC32-9407, contains information about how to estimate the disk space requirements for the Tivoli Data Warehouse. As described, the estimation process is executed in these steps:

1. Determine the number of detailed records per day for the attribute group.
   – (60 / collection interval) * 24 * (# instances of each interval)
2. Determine the hard disk footprint (in KB) for the attribute group.
   – (# detailed records) * (attribute group detailed record size) / 1024

3. Determine the amount of detailed data (in MB) for the attribute group.
   – (hard disk footprint) * (# of agents) * (# days of detailed data) / 1024
4. Calculate the amount of aggregate data (in MB) for the attribute group.
   – ((# hourly) + (# daily) + (# weekly) + (# monthly) + (# quarterly) + (# yearly)) * (# instances of each interval) * (attribute group aggregate record size) / 1048576
5. Determine the estimated size of your database.
   – (detailed record size) + (aggregate record size)

The IBM Redbooks publication *Tivoli Management Services Warehouse and Reporting*, SG24-7290, contains a detailed projection with the Warehouse Load Projection spreadsheet that covers all these steps. The spreadsheet can be downloaded from the OPAL Web site for IBM Tivoli monitoring:

http://catalog.lotus.com/wps/portal/tm

At the site, search for the topic "Warehouse Load Projections spreadsheet."

The Warehouse Load Projections spreadsheet has been created to simplify the task of producing the disk space estimate for the Tivoli Data Warehouse. It implements the steps described, and automates the process.

This spreadsheet includes the attribute group information for over 50 different agent types, and allows the user to perform "what-if" exercises to see the effects of different historical data collection configuration options. The spreadsheet includes two predefined charts showing the contribution of each agent type to the total Tivoli Data Warehouse disk space estimate.

Because it is implemented in a standard spreadsheet format, other charts can be generated easily. Projections produced by this spreadsheet should be viewed as rough estimates, but they are useful in making configuration planning decisions and in performing “what-if” exercises.

> **Important:** The Warehouse Load Projections spreadsheet includes the values for the “Bytes per instance (warehouse)” and “Bytes per instance (agent)” in the table “Disk capacity planning for historical data”. These values are taken from the user’s guide for each agent.
>
> These values should correspond to the agent’s version, which is described in the first row of every agent’s tab in the spreadsheet. The “Bytes per instance (agent)”, “Bytes per instance (warehouse)” and Bytes per summarized instance (warehouse)” values are included in the columns AgentsRowSize, DetailRowSize and AggRowSize for every attribute group.
>
> For different agent versions, the values are often different (usually higher, due to the addition of attributes).

In the following examples we l use the Warehouse Load Projections spreadsheet Version 1.3 for estimating and comparing several sizing scenarios for the default attribute groups for the Windows OS agent. We also discuss the use of attribute groups with multiple instances in the context of the Linux agent process attribute group.

Estimate size for Windows OS agent default groups

When you select the agent’s tab in the Warehouse Load Projections spreadsheet (used in Figure 4-5 on page 84), compare the agent’s version located in the first row of the table with the agent’s version actually used in the environment.

If the agent’s version in the spreadsheet is different from the one that is used in the environment, you can compare and update the column DetailRowSize with the current values. DetailRowSize is the value for one data row written for one instance of one agent for this attribute group in the Warehouse database. The column for these values can be found in the agents tab; see Figure 4-5 on page 84.

Total Num. Aggregate Records	(bytes) AgentRowSize	(bytes) DetailRowSize	(bytes) AggRowSize	Agent Tablename
4992000	224	259	878	NETWRKIN
7488000	340	244	827	WTLOGCLDSK
2496000	344	264	1423	WTMEMORY
4992000	196	224	690	WTPHYSDSK
7488000	192	219	646	NTPROCSSR
2496000	320	234	895	WTSERVER
2496000	792	706	1421	WTSYSTEM

Figure 4-5 Row-size data for Windows default attribute groups

> **Note:** The values for the column DetailRowSize can usually be found in the agent's Users Guide (for each agent type within the attribute description detailed within the section "Disk capacity planning for historical data"). Use the values for "Bytes per instance (warehouse)" from the table for each attribute group.
>
> Note that for a few agents, this information is not included. The record lengths for the spreadsheet are derived from the agent ATR files, which is also how they are calculated for the agent's User's Guide.

For this example, first we choose a typical environment, as shown in Figure 5-3.

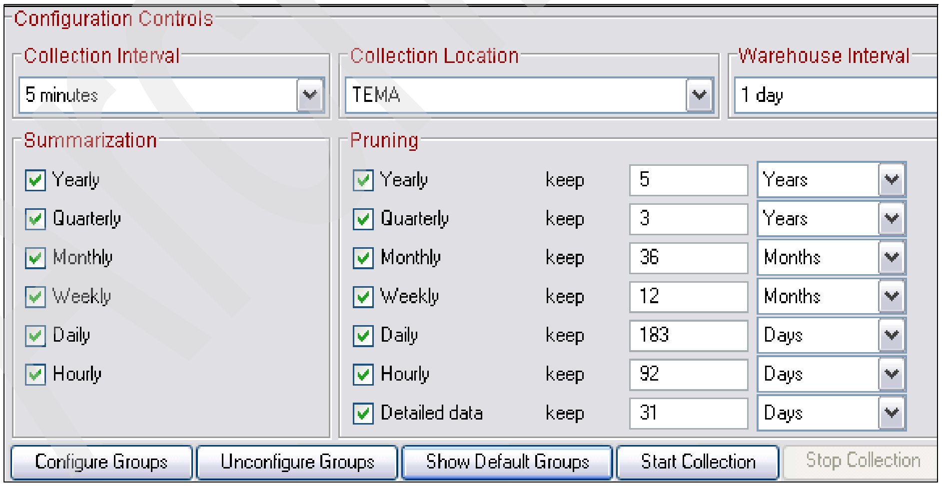


Figure 4-6 Typical settings for Configuration Controls

The reason for choosing these configuration values for collecting, summarizing, and pruning the data is to have detailed values for one month, hourly values for a quarter of a year, daily values for a half year, and so on.

> **Note:** *Collection interval* means a query will be created for the whole attribute group at the time of the interval; each query creates a row of data, as a kind of snapshot. The query works like a situation without a threshold.

Table 4-4 shows the default groups for the Windows OS agent, with the description of each attribute group instance, the assumption for attribute group instance value for this example, and the row-size per instance.

Table 4-4 Default attribute groups for Windows OS Agent with instances and row-size

Attribute group	**Instance description**	**Assumption: number of instances**	**Detailed row-size** (Bytes per instance (warehouse))
Network Interface	1 for each network interface, plus 1 for _Totals	2	259
NT Logical Disk	1 for each logical disk, plus 1 for _Totals	3	244
NT Memory	1 record per interval	1	264
NT Physical Disk	1 for each physical disk, plus 1 for _Totals	2	224
NT Processor	1 for each processor, plus 1 for _Totals	3	219
NT Server	1 record per interval	1	234
NT System	1 record per interval	1	706

Assumptions for choosing the instances are to have Windows-based servers with two processors, two physical disks (mirrored), two partitions. and one network interface, on average.

Table 4-5 shows two examples for choosing values for Summarization and Pruning. Example 1 comes from Figure 4-6 on page 84, and example 2 was created for discussion and comparison. For the entries of the pruning values in the spreadsheet, the number of rows was calculated with the formula in the second column of the table.

Table 4-5 Calculation table for the number of row-size for summarized values

	Calculation for number of rows	**Pruning value example 1**	**Number of rows example 1**	**Pruning value example 2**	**Number of rows example 2**
Yearly	value	**5**	5	**5**	5
Quarterly	value*4	**3**	12	**3**	12
Monthly	value	**36**	36	**36**	36
Weekly	value*52/12	**12**	52	**12**	52
Daily	value	**183**	183	**183**	183
Hourly	value*24	**92**	2208	**31**	744
Detailed	-	**31**	-	**7**	-

If we look at example 1, the calculated number of rows were manually included in the columns F to K of the Windows agents tab in the spreadsheet for the chosen attribute groups. In column B, you need to set the collection interval to 5 minutes. If no collection interval is selected, the attribute group will not go into the calculation. In the “Summary” tab, you need to set the number of agents in the first column for every single type of agents (for this example, 1000 agents).

The results can be seen in Figure 4-7 on page 87.

A	B	C	E	F	G	H	I	J	K
Windows	6.1.0	KNT							
1000	Number of agents (from Summary worksheet)								
156	TDW record inserts/hour/agent								
41.7	TDW KB inserted/hour/agent								
1.1	Agent HDC Disk Space (MB - per agent, across all attribute groups)								
30302.8	TDW Disk Space - Detailed data (MB - total across all attribute groups for all agents)								
26883.9	TDW Disk Space - Aggregate data (MB - total across all attribute groups for all agents)								
274	Average size for detailed records (bytes)								
869	Average size for aggregate records (bytes)								
	Collection	Average							
Attribute Group /	Interval	Rows /	Intervals kept in warehouse (days for Detailed)						
TDW Tablename	(5|15|30|60)	Interval	Detailed	Hourly	Daily	Weekly	Monthly	Quarterly	Yearly
Network_Interface	5	2	31	2208	183	52	36	12	5
NT_Logical_Disk	5	3	31	2208	183	52	36	12	5
NT_Memory	5	1	31	2208	183	52	36	12	5
NT_Physical_Disk	5	2	31	2208	183	52	36	12	5
NT_Processor	5	3	31	2208	183	52	36	12	5
NT_Server	5	1	31	2208	183	52	36	12	5
NT_System	5	1	31	2208	183	52	36	12	5

Figure 4-7 Load projection spread sheet example - Windows tab with summarization and pruning details for the Windows OS agents default attribute groups

The number of agents can be seen in the first column, second row. Furthermore, you can see in the first column the value of needed Warehouse database size for the detailed data in the sixth row (more than 30 GB for this example), and the value of aggregated Warehouse database size in the seventh row (nearly 27 GB). In summary, approximately 57 GB is required for all 1000 Windows OS agents collecting the default attribute groups with the default settings for collecting, summarization, and pruning.

The summary of this data can be seen also in the Summary tab of the Windows OS agents row, as shown in Figure 4-8.

			TDW record	TDW MB	TDW Detailed	TDW Aggregate	TDW inserts
Count	Agent type	Version	inserts/hour	inserted/hour	MB Total	MB Total	/hour/agent
1000	Windows	6.1.0	156000	40.7	**30302.8**	**26883.9**	156

Figure 4-8 Summary example 1 - Windows OS agents default attribute groups with proper pruning settings and 5-minute collection interval

Now we will reduce the values as described in example 2. The reason for this is because the default workspaces do not need detailed data from the warehouse. Two reasons for using detailed data are collecting for summarization (but the summarization should work every day in a large scale environment), and using detailed data for problem determination. Therefore, the detailed data will be kept in the warehouse only for one week (7 days).

The second adjustment in example 2 was done to show the difference of the summarized data. That is the reduction of hourly summarized data from three to one month, 31 days.

Hourly data creates the most rows in the Warehouse database: 24 rows per day for one instance of one agent, and 744 rows per month. You should consider whether the data is really needed. The collection interval is 5 minutes. The estimated number of rows for example 2 is included in Table 4-5 on page 86.

Figure 4-9 now shows the summary for Warehouse database sizing.

Count	Agent type	Version	TDW record inserts/hour	TDW MB inserted/hour	TDW Detailed MB Total	TDW Aggregate MB Total	TDW inserts /hour/agent
1000	Windows	6.1.0	156000	40.7	**6842.6**	**11115.5**	156

Figure 4-9 Summary example 2 - Windows OS agents default attribute groups with smaller pruning settings and 5-minute collection interval

The required size for the same attribute groups and same number of agents is reduced dramatically. The detailed data size is cut down by over 75%. This comes only from the reduction of the detailed data pruning setting from 31 to 7 days. The aggregate data size will be reduced as well by nearly 60%, and this comes from changing the hourly data pruning setting from three to one month. In summary, approximately 40 GB will be saved for the Warehouse database size.

The data can be uploaded hourly into the Warehouse database, but will be stored for 24 hours on the local disk. Another opportunity for reducing the size is to increase the collection interval (thus decreasing the frequency of collection). The decreasing of the collection interval will only shrink the amount of detailed data. This can also be important for the Warehouse Proxy agent's network interface when a large number of agents are being managed.

Figure 4-10 shows example 3 with the same values as example 1 for summarization and pruning, and with a collection interval of 15 minutes.

Count	Agent type	Version	TDW record inserts/hour	TDW MB inserted/hour	TDW Detailed MB Total	TDW Aggregate MB Total	TDW inserts /hour/agent
1000	Windows	6.1.0	52000	13.6	**10100.9**	**26883.9**	52

Figure 4-10 Summary example 3 - Windows OS agents default attribute groups with proper pruning settings and 15 minutes collection interval

You can see that only the size of the detailed data will be reduced, compared with example 1. The effect of this reduction is as high as the reduction of the pruning settings in example 2. The detailed data is cut down cut down to one-third of its original size, because the collection interval is decreased by one-third compared with the first example.

Because every detailed data row represents only a snapshot of a particular moment and can be numerically distant from the rest of the data, the statistical value of the summarized data will be more precise with more rows of detailed data. When the detailed data is cut down by one-third, the data is uploaded into the Warehouse database will be cut down by one-third as well.

This can be observed by looking at the columns TDW record inserts/hour or TDW inserts /hour/agents for a single Windows OS agent. This aspect for planning for the bandwidth utilization is revisited in 4.5.1, “Estimate the size for bandwidth” on page 111.

Finally, example 4 shows the pruning values from example 2 with a collection interval of 15 minutes (see Figure 4-11).

Count	Agent type	Version	TDW record inserts/hour	TDW MB inserted/hour	TDW Detailed MB Total	TDW Aggregate MB Total	TDW inserts /hour/agent
1000	Windows	6.1.0	52000	13.6	2280.9	**11115.5**	52

Figure 4-11 Summary example 4 - Windows OS agents default attribute groups with smaller pruning settings and 15-minute collection interval

In this case, the amount of data is reduced by more than 75% compared with example 1, and the part of the detailed data size is reduced by more than 90%. The needs for visualization, capacity management and problem determination should determine the settings for collection, summarization and pruning estimated with the Warehouse Load Projections spreadsheet. Different settings for single attribute groups can be made.

A summary comparison for all examples (1-4) is shown in Figure 4-12 on page 90.

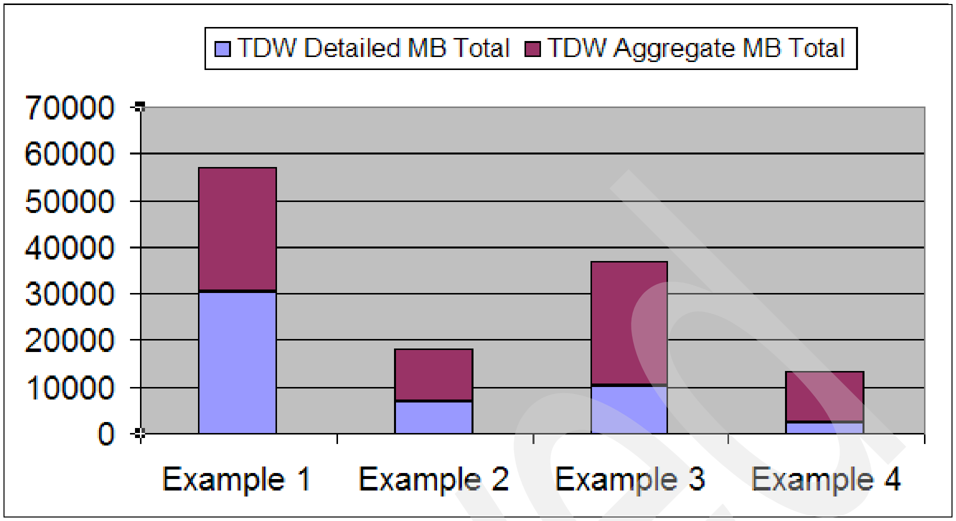


Figure 4-12 Comparison for estimated Warehouse database size with different options

Estimate size for default attribute groups for different agents

Another discussion point is the selection of the so-called default attribute groups for different agents as performed for the preceding Windows examples. Sometimes Tivoli Data Warehouse administrators use the default attribute groups, instead of selecting them individually. This is not a recommended way of customizing the historical data collection, because it could create a significant amount of data, as shown in the next example.

The Warehouse Load Projections spreadsheet shows the attribute groups of more than 50 agent types, with the default attribute groups highlighted. In the fourth column, next to the attribute group's name on every agents tab, is a description of the instance. The column title is Expected number of instances/rows. This is a very important aspect for detailed estimation, and is shown in Figure 4-13 on page 91.

	A	B	C	D	E	F	G
5	58.8	Agent HDC Disk Space (MB - per agent, across all attribute groups)					
6	42970.9	TDW Disk Space - Detailed data (MB - total across all attribute groups for all agents)					
7	129320.9	TDW Disk Space - Aggregate data (MB - total across all attribute groups for all agents)					
8	1275	Average size for detailed records (bytes)					
9	2498	Average size for aggregate records (bytes)					
10							
11		Collection	Average				
12	Attribute Group /	Interval	Rows /		Intervals kept in warehou		
13	TDW Tablename	(5|15|30|60)	Interval	Expected number of instances/rows	Detailed	Hourly	Daily
14	**Linux_CPU**	15	2	Multiple, typically 1-4	7	744	183
15	**Linux_CPU_Averages**	15	2	Multiple, typically 1-4	7	744	183
16	**Linux_CPU_Config**	15	2	Multiple, typically 2	7	744	183
17	**Linux_Disk**	15	2	Multiple, typically 2	7	744	183
18	**Linux_Disk_IO**	15	2	Multiple, typically 2	7	744	183
19	**Linux_Disk_Usage_Trends**	15	2	Multiple, typically 2	7	744	183
20	**Linux_IO_Ext**	15	2	Multiple, typically 2	7	744	183
21	**Linux_Network**	15	3	Multiple, typically 3	7	744	183
22	**Linux_NFS_Statistics**	15	2	Multiple, typically 1-4	7	744	183
23	**Linux_OS_Config**	15	1	Multiple, typically 1	7	744	183
24	**Linux_Process**	15	500	Multiple, typically 100-1000	7	744	183
25	Linux_Process_User_Info			Multiple, typically about 5	7	744	183
26	**Linux_RPC_Statistics**	15	1	1 record per interval	7	744	183
27	Linux_Sockets_Detail			Multiple, typically 10-100	7	744	183
28	**Linux_Sockets_Status**	15		Multiple, typically 10-100	7	744	183
29	**Linux_Swap_Rate**	15	1	1 record per interval	7	744	183
30	**Linux_System_Statistics**	15	1	1 record per interval	7	744	183
31	**Linux_User_Login**	15		Multiple, typically about 5	7	744	183
32	**Linux_VM_Stats**	15	1	1 record per interval	7	744	183

ReadMe / Summary / BarGraph / PieGraph / Unix / Windows / **Linux** / i5-OS / DB2 / Oracle / MSSQL / Syba

Figure 4-13 "Linux" agents' tab with assumed values for number of instances

In the following example (Table 4-6), we select only the agents' default groups. The number of agents is based on an assumption of a typical customer environment. The settings for collecting, summarization and pruning are chosen as in example 4 for Windows (shown in Table 4-6).

The estimated data size calculated by the Warehouse Load Projections spreadsheet can be seen in the spreadsheet tab Summary, or in the BarGraph tab, as shown in Figure 4-14 on page 92.

Table 4-6 Sample environment with settings for collection, summarization, and pruning

Agent Type	Number of agents	Collection Interval	Yearly Pruning	Quarterly Pruning	Monthly Pruning	Weekly Pruning	Dottily Pruning	Hourly Pruning	Detailed Pruning
Windows OS	1000	15	5	12	26	12	183	31	7

Agent Type	Number of agents	Collection Interval	Yearly Pruning	Quarterly Pruning	Monthly Pruning	Weekly Pruning	Dotilly Pruning	Hourly Pruning	Detailed Pruning
UNIX	100	15	5	12	26	12	183	31	7
Linux	100	15	5	12	26	12	183	31	7
i5/OS	10	15	5	12	26	12	183	31	7
Sybase	10	15	5	12	26	12	183	31	7
Active Directory®	3	15	5	12	26	12	183	31	7
mySAP™	3	15	5	12	26	12	183	31	7

Figure 4-14 shows a bar chart with the amount of historical data in a sample customer situation.

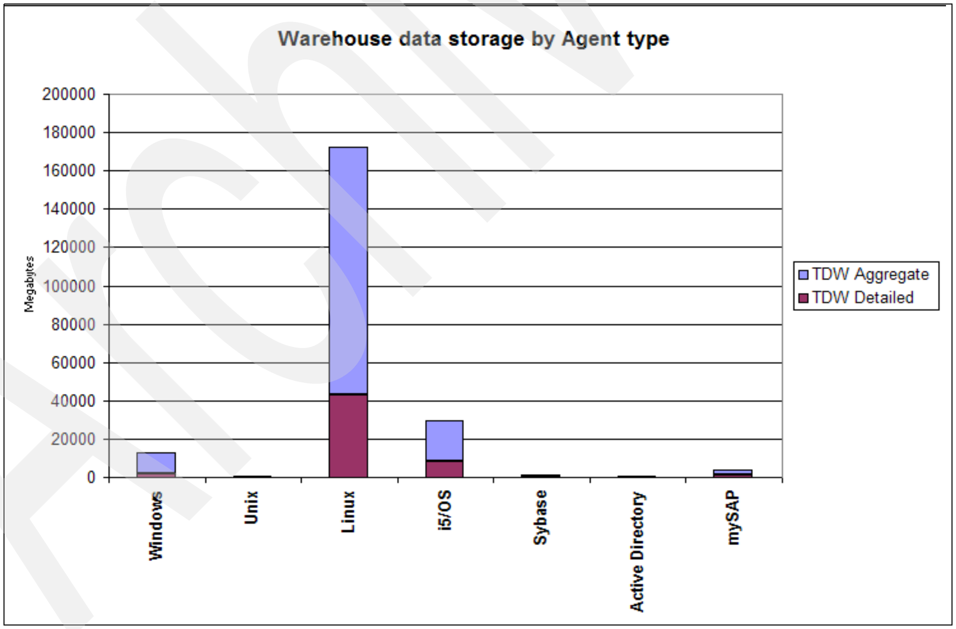


Figure 4-14 BarGraph chart with amount of historical data with default attribute groups in a sample customer situation

Although the number of Linux agents is only 10% of the number of Windows OS agents, the amount of data is approximately 20 times higher. Another important finding is that the number of UNIX agents is the same as the number of Linux agents, but the amount of data for the Linux agents is approximately 238 times higher compared to the UNIX agents.

Furthermore (and this is not shown in the first view, because there is small number of agents), the amount of data for a single i5/OS agent is approximately 1.7 times higher than of a Linux agent.

The amount for the mySAP agent is very high, as well. This is because attribute groups are included in the defaults that create rows in every collection interval for multiple instances. So in the case of the attribute group Linux process, this can be up to 1000 processes, which means 1000 rows.

.

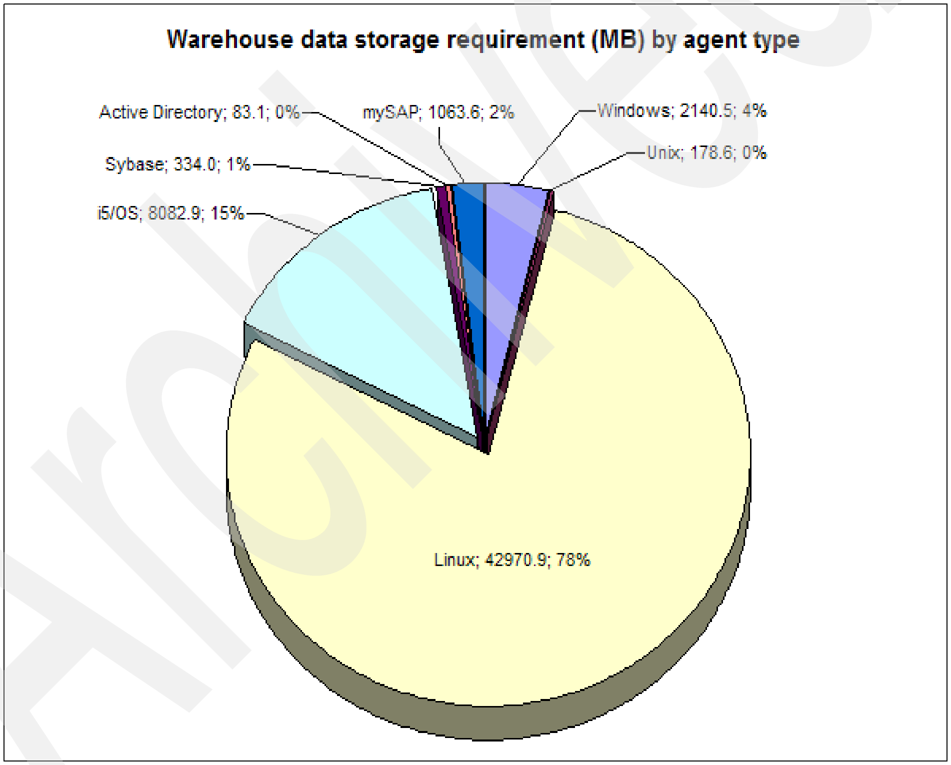


Figure 4-15 PieGraph chart with amount of historical data for default attribute groups

In this example, the estimated number of processes was 500, on average. So you can say that the usage of attribute groups with multiple instances makes the estimation inexact. The complete de-selection of the Linux Process attribute group for this sample environment shrinks the Linux OS agent percentage from 78% to only 1%, as shown in Figure 4-15 on page 93.

> **Important:** Use the Default Group button in the Historical Data Collection dialog box for large scale environments with care, because this could create a significant amount of data. Plan and estimate the historical data collection, based on reporting and visualizing needs.

4.3.2 Estimate agents disk space for data collection using the Warehouse Load Projections spreadsheet

Another aspect for performing a size estimation is how much disk size is needed for the data files of the short-term historical data. The files are located in the following location, where ITMinstall_dir is the directory where the product is installed:

- ITMinstall_dir\TMAITM6\log

The answer to this question is important not only for collecting short-term historical data, but also for uploading the data into the Warehouse. The Warehouse Load Projections spreadsheet can be used for this estimation, as well.

Before using the spreadsheet for estimation, we list the considerations for controlling the amount of data on the agents local disk:

- Determining the attribute groups for collection
- Determining the collecting interval
- Determining the collection location (can be the agents local disk or the disk of the direct connected monitoring server)

> **Important:** When warehousing of historical data is working normally, short-term data will be kept locally for 24 hours. However, if warehousing is not working properly (for example, if the Warehouse Proxy agent or the Tivoli Data Warehouse is not available), then the data is not uploaded to the warehouse. In this case, data will continue to collect on the agent until it is either successfully uploaded (and subsequently removed), or until the local disk is full.
>
> The best practice is to monitor the Warehouse Proxy agent and the Tivoli Data Warehouse availability by a situation.

Choosing the (remote) monitoring server for collection location is usually not recommended for scalability reasons. If the monitoring server is used as the storage location, this selection is for the whole attribute group. The amount of data can be calculated in the same way; you multiply the disk amount required or one agent for that attribute group with the number of agents of that type connected to the monitoring server.

For estimating the amount of data on the agents' local disk, the Warehouse Load Projections spreadsheet can be used as a rough guideline in the same way for estimation for the Warehouse load. But you need to take into account that the data on the agent's local disk is kept in flat files, and the data on the data Warehouse in kept in RDBMS. The size of a data row on the agents' local disk in most cases is higher than the size of the same data row in RDBMS.

The value for one data row written for one instance for this attribute group is seen in the AgentRowSize column. This column can be found behind every agents tab. For the attribute groups, the value of AgentRowSize is usually more than DetailRowSize, because a database application can manage the data (thus decreasing the size of the data).

Note: Another reason for the difference in size is because there is information in the row in the binary file that is not exported to the database. For example, SAMPLES and INTERVAL are columns described in the header file (.hdr), but are never exported to the database.

The value should also be checked when the version of the agent differs from the agent's version within the spreadsheet, as seen in the first row on every agent's tab.

Note: The values for the column AgentsRowSize can usually be found in the agent's user's guide based on the agent type within the attribute description detailed in the chapter "Disk capacity planning for historical data". Use the values of the term "Bytes per instance (agent)" from the table for each attribute group.

In the following example, we use the same values as for the estimation of the warehouse size as in example 1. We use the Windows OS agent default attribute groups, listed in Table 4-4 on page 85, with a collection interval of 5 minutes. The data will be held on the agents disk for 24 hours. The result can be seen in Figure 4-16 on page 96.

TDW inserts /hour/agent	TDW bytes inserted /hour/agent	Agent HDC Disk Space KB/agent	Estimated TDW Table Size Detailed	Hourly	Daily	We
24	6216	**126**	4410.5	3697.6	306.5	
36	8784	**287**	6232.5	5224.3	433.0	
12	3168	**97**	2247.8	2996.4	248.3	
24	5376	**110**	3814.5	2905.9	240.8	
36	7884	**162**	5594.0	4080.9	338.2	
12	2808	**90**	1992.4	1884.6	156.2	
12	8472	**223**	6011.2	2992.2	248.0	

Figure 4-16 List of needed disk space for Windows OS attribute groups

On the agents tab, in this case Windows within the column Agent HDC Disk Space KB/agent, you can see the required size on the local disk for an attribute group/24 hours/one agent. In the Summary tab within the column Agent HDC Disk Space (MB/agent), you can see the summary of all collected attribute groups for one agent for 24 hours, as well in the agents tab in the fourth row (Figure 4-7 on page 87).

The required space in this example is 1.1 MB. This amount can be confirmed with the estimated detailed data from the warehouse examples provided, divided by the number of agents and days for keeping the data in the warehouse. If the data collection will be done on a (remote) monitoring server, with all 1000 agents on one monitoring server, more than 1.1 GB size should be kept on the local disk of the remote monitoring server.

Same considerations can be applied for attribute groups with multiple instances, particularly in regards to many instances in the attribute groups like Linux Process.

4.3.3 Estimate the amount of historical data for Universal Agent applications

Another aspect is to estimate the required size for data in the Tivoli Data Warehouse or for the agents' local disk when using Universal Agent applications. There are two kinds of applications:

- Pre-defined UA applications with predefined attribute groups
- UA applications with self-defined attribute groups

Predefined attribute groups come for data providers POST, HTTP, and SNMP. The self-defined attribute groups for all other data providers (APIS, FILE, SCRP, ODBC, and SOCK) will be defined in the definition files (called *metafiles*).

Table 4-7 lists the pre-defined attribute groups, with a description of a possible number of instances.

Table 4-7 Pre-defined attribute groups for UA application

Data provider	Application name	Attribute group	Instances
POST	MAS	DPPOST	1
HTTP	INTERNET	URL_OBJECTS	1 per URL
HTTP	INTERNET	MANAGED_URL	1 per URL
SNMP	SNMP-MANAGER	TRAP	1 per trap
SNMP	SNMP-MANAGER	MIBNODATA	
SNMP	SNMP-MANAGER	ROUTER	1 per identified router
SNMP	SNMP-MANAGER	NETWORK	1 per every address found in polled network - can be more than 1000
SNMP	SNMP-MANAGER	MANAGEDNODES	
SNMP	SNMP-MANAGER	NETSUMMARY	1 per identified network
SNMP	SNMP-MANAGER	MIBSTATUS	

Determine the value for Bytes per instance (agent)

For predefined and self-defined attribute groups, the method for determining the value for Bytes per instance (agent) is the same. There are no values for the size of a row included in the Warehouse Load Projections spreadsheet or in *IBM Tivoli Universal Agent User's Guide Version 6.1.0*, SC32-9459.

The way to determine the size of a row on the agents local disk (the value for Bytes per instance (agent), that is) is simple and can be used for determining these values for all other attribute groups, too. This is done by following these steps:

1. Start a historical data collection for the selected attribute group.
2. Determine the definition file for attribute group collection (*hdr).

3. Determine the size of a row for one instance.
4. Check the value against a real collected row.

For self-defined attribute groups, do not summarize the single values for bytes for the attributes defined in the metafile. Several default attributes will be added from the agent.

In the following section, we discuss these steps in more detail.

Step 1: Start a historical data collection for the attribute group

When the application is started, the attribute group can be selected within the historical data collection dialog box from the Select a product pull-down menu.

> **Note:** This function is different in IBM Tivoli Monitoring V6.2. In IBM Tivoli Monitoring V6.1, the attribute groups are found in the Universal Data Provider product group.
>
> In IBM Tivoli Monitoring V6.2, every application will have its own product group.

After activation of a data provider, you can select a standard attribute group, as listed in Table 4-7 on page 97. However, this does not work in every case. For example, for the HTTP data provider INTERNET application, a URL must be added for monitoring. It should be active only for one instance (refer to the Instances column in Table 4-7 on page 97).

In any case, it is not possible to activate only one instance. An example would be the data provider SNMP, attribute group NETWORK.

Step 2: Determine the definition file for attribute group collection

Go to an agent where the UA application runs and the data is collected. Go to the agent's LOG directory, where ITMinstall_dir is the directory where the product is installed:

- ITMinstall_dir\TMAITM6\log

Look for the following file, where `[AgentsName]` is the name of the agent:

```
[AgentsName]_UA.LGO
```

Open the file. The file includes the active situations and definitions for historical data collection (which are also situations). For other types of agents, the files are named similarly.

For example, for the DB2 agent it is named `[AgentsName]`_UD.LG0. But for the Windows OS agent, it is called `[AgentsName]`.LG0, where `[AgentsName]` is already the name of the agent.

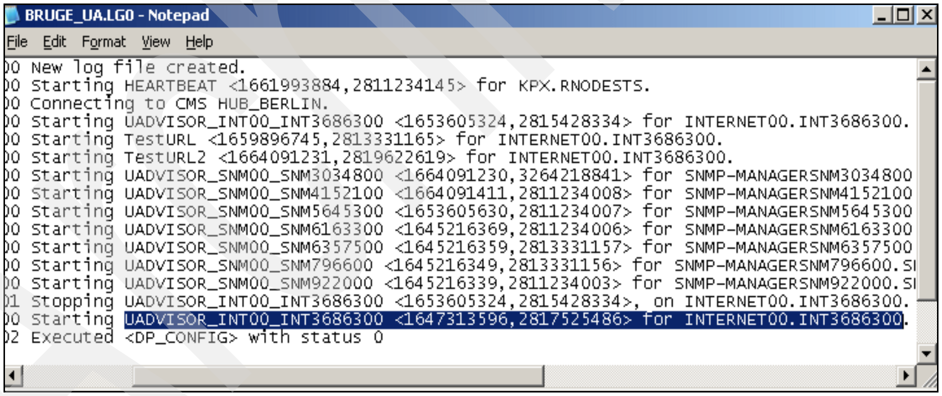
If the file is open, you should see the started situations, where `[SituationsName]` is the name of a started situation; `[ApplicationVV]` is the name of the application, followed by a 2-letter value of the application version; and `[APLnnnnnnn]` is the 3-letter identifier of the application, followed by a 7-letter code:

```
... Starting [SituationsName] <.....> for [ApplicationVV].[APLnnnnnnn]
```

Also, you should see the historical data collection definitions, where `[CollectionsName]` is part of the name of a started collection for historical data;`[ApplicationVV]` is the name of the application followed by a 2-letter value of the applications version; and `[APLnnnnnnn]` is the 3-letter identifier of the application, followed by a 7-letter code.

```
... Starting UADVISOR_APLVV_[CollectionsName] <.....> for
[ApplicationVV].[APLnnnnnnn]
```

For example, for the INTERNET application of the HTTP data provider `[CollectionsName]` is ’INT00’; `[ApplicationVV]` is ’INTERNET00’; and `[APLnnnnnnn]`is ’INT3686300’, as displayed in Figure 4-17.




```
)0 New log file created.
)0 Starting HEARTBEAT <1661993884,2811234145> for KPX.RNODESTS.
)0 Connecting to CMS HUB_BERLIN.
)0 Starting UADVISOR_INT00_INT3686300 <1653605324,2815428334> for INTERNET00.INT3686300.
)0 Starting TestURL <1659896745,2813331165> for INTERNET00.INT3686300.
)0 Starting TestURL2 <1664091231,2819622619> for INTERNET00.INT3686300.
)0 Starting UADVISOR_SNM00_SNM3034800 <1664091230,3264218841> for SNMP-MANAGERSNM3034800
)0 Starting UADVISOR_SNM00_SNM4152100 <1664091411,2811234008> for SNMP-MANAGERSNM4152100
)0 Starting UADVISOR_SNM00_SNM5645300 <1653605630,2811234007> for SNMP-MANAGERSNM5645300
)0 Starting UADVISOR_SNM00_SNM6163300 <1645216369,2811234006> for SNMP-MANAGERSNM6163300
)0 Starting UADVISOR_SNM00_SNM6357500 <1645216359,2813331157> for SNMP-MANAGERSNM6357500
)0 Starting UADVISOR_SNM00_SNM796600 <1645216349,2813331156> for SNMP-MANAGERSNM796600.SI
)0 Starting UADVISOR_SNM00_SNM922000 <1645216339,2811234003> for SNMP-MANAGERSNM922000.SI
)1 Stopping UADVISOR_INT00_INT3686300 <1653605324,2815428334>, on INTERNET00.INT3686300.
)0 Starting UADVISOR_INT00_INT3686300 <1647313596,2817525486> for INTERNET00.INT3686300.
)2 Executed <DP_CONFIG> with status 0
```

Figure 4-17 UA log file with started situations and historical data collections

Thus, the name of the definition file for the historical data collection is called [APLnnnnnnn].hdr (for this example, INT3686300.hdr).

Step 3: Determine the size of a row for one instance

In the same directory, the definition file for historical data collection for these attribute group should exist, as determined in Step 2.

Open the file. The single attribute definitions are shown, where [Attribute] is the name of an attribute; [type] is the type of the attribute; [X] is the starting point in bytes of this attribute; and [Y]is the length of this attribute. As an example, TMZDIFF(int,0,4) or INT3686300.ORIGINAME(char, 20, 32).

```
[Attribute]([type],[X],[Y])
```

Now, look for the last attribute called INTERVALL and summarize the starting point [X] for this attribute to the length [Y] from this attribute. For example, for INTERVAL(int,1664,4) the summary is 1668. This is the value for "Bytes per instance (agent)".

Step 4: Check the value against a real collected row

Similar to the definition file, a collection file for the data collection of this attribute group should exist after the first collection sample, named without an extension (for this example INT3686300). Determine the correct size of the file.

- On Windows, right-click and select **Properties**.
- On UNIX or Linux, use the command `ls -ali`.

The determined size of the file should be a multiple of the value “Bytes per instance (agent)”. If the first collection has been completed and only one instance was collected, the result of this calculation should be 1.

Determine the value for “Bytes per instance (warehouse)”

Now the value for "Bytes per instance (warehouse)" will be determined, but the calculation method depends on the vendor type of the database that will be used.

This example shows a simple method for using DB2 in only three steps. (Note that for Oracle or MS SQL, Step 3 will be different.)

Note: An easier (but not exact) way is to use the value of the agent row size for this estimation. This value will typically be higher than the database row size.

1. Start a historical data collection for the selected attribute group.
2. Determine the table name for attribute group collection.
3. Determine the size of a row for one instance.

In the following section, we explain these steps in more detail.

Step 1: Start a historical data collection for the attribute group

The application and the historical data collection should already be started.

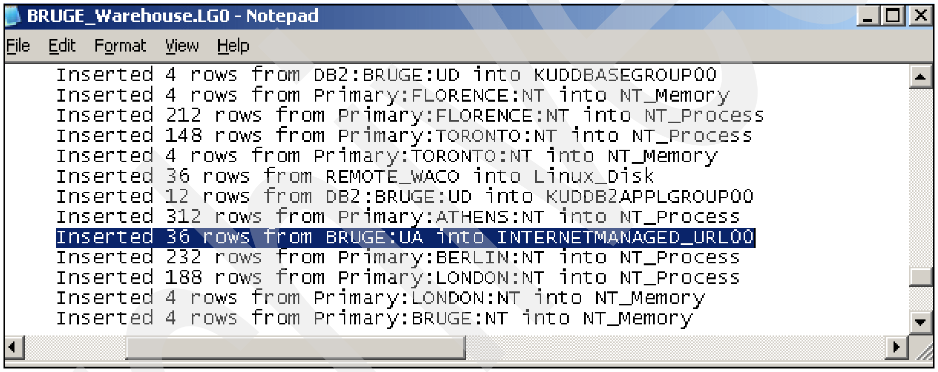
Step 2: Determine the table name for attribute group collection

For this step you only need the name of the application, the name of the attribute group, and the versions number to form the table name. The table will be named:

```
[ApplicationName][AttributeGroupName][Version]
```

Where `[ApplicationName]` is the name of the application (for this example: INTERNET); `[AttributeGroupName]` is the name of the attribute group (for this example MANAGED_URL); and `[Version]` is the two-letter value for the version (for this example, 00). The complete name for the table for this example is INTERNETMANAGED_URL00.

The name can be checked in the Warehouse agent's log file on the Warehouse Proxy agent. There should be an entry after the first insert into the database and one for every another insert. Figure 4-18 shows the insert.




```
Inserted 4 rows from DB2:BRUGE:UD into KUDDBASEGROUP00
Inserted 4 rows from Primary:FLORENCE:NT into NT_Memory
Inserted 212 rows from Primary:FLORENCE:NT into NT_Process
Inserted 148 rows from Primary:TORONTO:NT into NT_Process
Inserted 4 rows from Primary:TORONTO:NT into NT_Memory
Inserted 36 rows from REMOTE_WACO into Linux_Disk
Inserted 12 rows from DB2:BRUGE:UD into KUDDB2APPLGROUP00
Inserted 312 rows from Primary:ATHENS:NT into NT_Process
Inserted 36 rows from BRUGE:UA into INTERNETMANAGED_URL00
Inserted 232 rows from Primary:BERLIN:NT into NT_Process
Inserted 188 rows from Primary:LONDON:NT into NT_Process
Inserted 4 rows from Primary:LONDON:NT into NT_Memory
Inserted 4 rows from Primary:BRUGE:NT into NT_Memory
```

Figure 4-18 Warehouse log file with inserts to the Warehouse database

> **Important:** As of Fix Pack 6, the detailed agent export information is no longer written to the Warehouse Proxy agent log (LG0) file; you will not be able to see this information in the LG0 file if you have installed this Fix Pack.
>
> However, this information is available in the WAREHOUSELOG table. You can use the Control Center and select **Open** for the WAREHOUSELOG table. It will show you 100 records at a time, and you can select **Fetch More Rows** to see the next group of rows. Or you can issue a query for the information.
>
> To avoid the extra effort, you can use the AgentRowSize as an estimate. The values are usually not that far off.

Step 3: Determine the size of a row for one instance

For the third step (only shown for DB2), you should open the table view in the DB2 Control Center. Click **All Databases -> WAREHOUS** (Name of the Warehouse database) **-> Tables**.

Now the table, with the name determined in Step 2, should be found on the right side of the window (in this example, INTERNERTMANAGED_URL00). Select the database and click it using the right mouse key. Choose **Estimate size** from the menu, as shown in Figure 4-19.

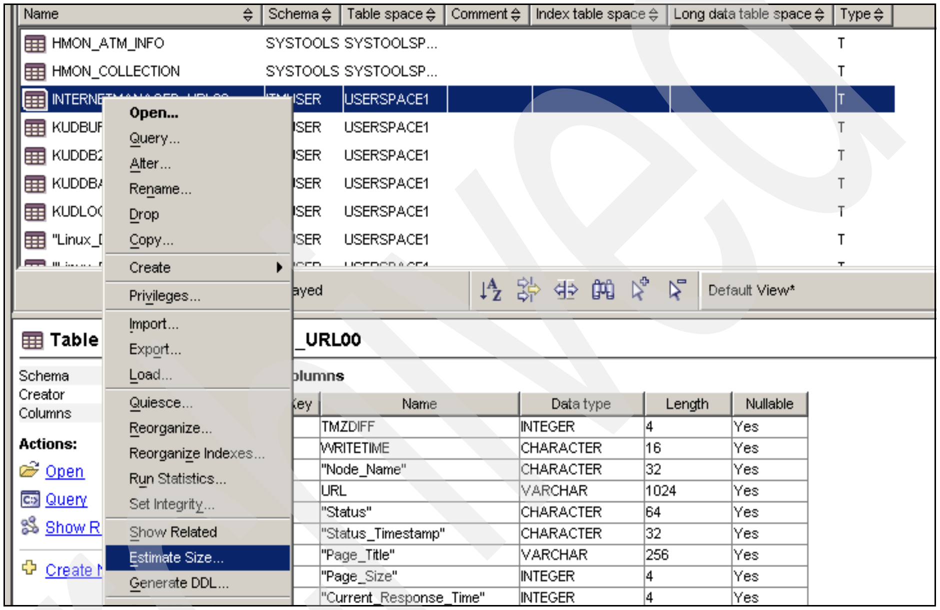


Figure 4-19 DB2 control Center with command for determining row estimation

A relational database management system (RDBMS) can shrink the rows of data with a technology of using indexes for several columns within the table, which is why the row within the database will be smaller than the row in a file stored locally on an agents disk. So the final value for the row size within the database can be determined after a number of rows, after the RDBMS completes the indexing of the table.

Consider multiple instances

For Universal Agent applications, you have the same consideration as the other agents. Attribute groups can create a many rows for a large number of instances. These can be predefined attribute groups, as well, self-defined attribute groups. So the attribute group NETWORK from the SNMP-MANAGER application, which discovers the network, creates only 256 rows for normal class-C-network with the network mask 255.255.255.0. That number increases significantly when other networks, in addition to the local network, will be discovered.

It is difficult to estimate the number of instances when the using the POST, FILE, APIS, ODBC, SOCK, or SNMP data providers. The number of instances depends on the usage of this application (for example, how many KUMPSEND commands, how many lines of log file or how many traps will be received by the agent, and so on).

A special case is the SCRIPT data provider. Using a command-driven execution of the script creates only one data row for one interval. The interval-driven execution creates also one data row for one collection interval. That means one instance, but when the execution interval of the script is a multiple of the collection interval and the time-to-live (TTL) variable is sufficient, another instance will be created (which might to lead to redundant records in the Tivoli Data Warehouse).

On the other hand, when the time-to-live variable is insufficient (too short for the collection interval), then no instance will be created.

4.3.4 Customize the Warehouse Load Projections spreadsheet

The Warehouse Load Projections spreadsheet can be customized, or a similar sheet can be created. However the values for DetailRowSize, AggRowSize, or AgentsRowSize should be checked depending on the agents' version and customized, if necessary. The values can be determined as discussed previously.

For attribute groups of the Universal Agent applications, you can use a tab in the Load Projections spreadsheet for an agent which does not exist in your environment (for example the z/OS tab, if you do not use a mainframe) and customize it with your attribute groups and row size values.

> **Remember:** The key factor for a useful estimate is to take into account the number of instances.

4.4 Reduce the number of agents for historical data collection

There are two ways to reduce the number of agents for collection of historical data:

- By using multiple monitoring servers and suspending any of them from data collection via the Historical Data collection dialog (provided with IBM Tivoli Monitoring Fix Pack 5)
- By using multiple Warehouse Proxies and multiple monitoring servers and excluding any of them via configuration of the Warehouse proxy agents (fully functional with IBM Tivoli Monitoring Fix Pack 5)

Both methods are used for decreasing the number of agents on a remote monitoring server level. You can use both of them at the same time, and they have an important impact on the architecture of the monitoring environment.

> **Important:** You can enable historical data collection of a given agent attribute group for a subset of your monitoring environment. Historical data collection for a given agent type and attribute group is started at the monitoring server level.
>
> For example, if data collection for the Windows OS agent NT_System attribute group is started on a monitoring server, then any Windows OS agents connected to that monitoring server will start collecting data for the NT_System attribute group. If there are five monitoring servers in an environment, and data collection for the NT_System attribute group is only started on one of the five monitoring servers, then only the agents connected to the one monitoring server will collect historical data for the NT_System attribute group.
>
> Assigning agents to a monitoring server based on the historical data collection needs is one way to minimize the amount of historical data collected in an environment, reducing the warehouse size and lowering the processing demands for the Warehouse Proxy agent(s) and Summarization and Pruning agent.

In the following sections we discuss both methods in more detail. In this book we discuss large scale environments, which means thousands of agents, with multiple monitoring servers and Warehouse Proxy agents.

4.4.1 Suspend monitoring server from the Historical Data collection

With IBM Tivoli Monitoring Fix Pack 5, you can now see the number of monitoring servers that a particular collection has been started for. After calling the Historical Data collection dialog box and configuring the chosen attribute group, click the **Start Collection** button to start the collection. The dialog shown in Figure 4-20 on page 105 will be displayed.

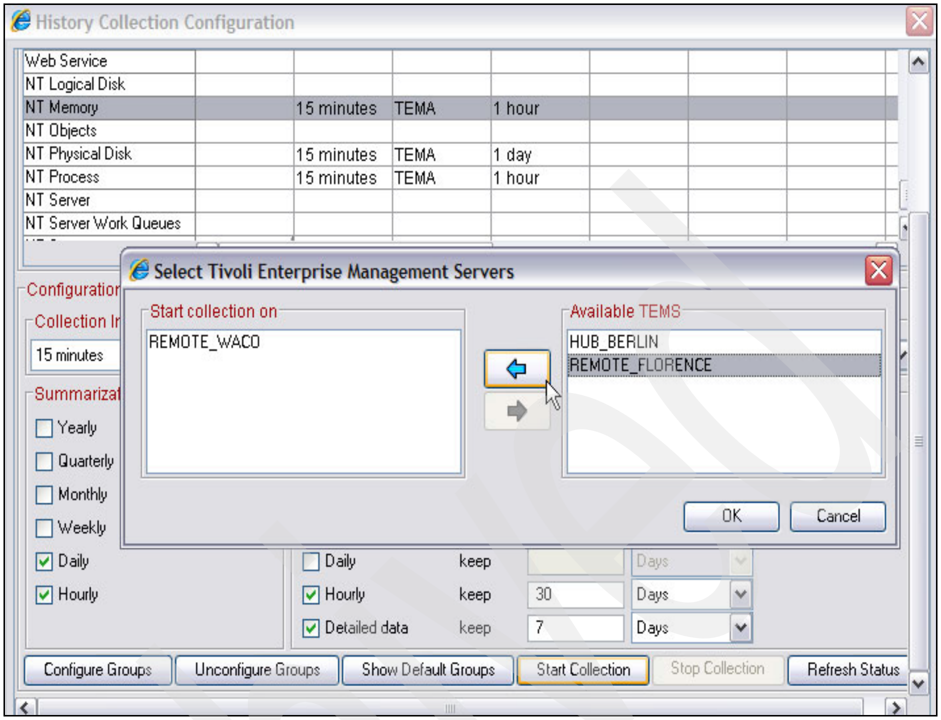


Figure 4-20 Dialog for starting a Historical Data collection

In the dialog, you can select for which of the monitoring servers the data collection will be started. This monitoring server must be selected and transferred from the right side to the left. That means the collection will start for the agents connected to this monitoring server.

The stopping process is similar, the monitoring server for which the collection should be stopped must be selected and transferred to the right side. The starting and stopping processes can be done several times. For every execution, you can see for which monitoring server the data collection is started and for which ones not from the dialog.

Another way to see for which monitoring server the collection is started is to slide the mouse over the second column Collection from the respective attribute group row. A small window will appear listing the monitoring servers, as shown in Figure 4-21 on page 106. This column shows the number of monitoring servers for which the data collection is started. The list can also be sorted by columns.

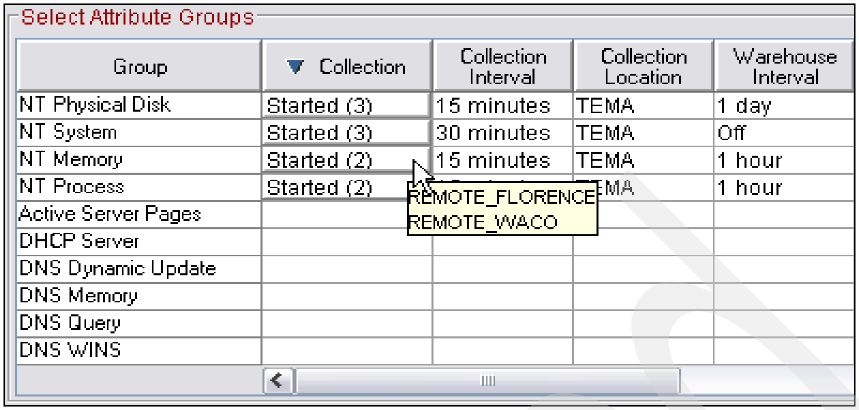


Figure 4-21 Sorted list of attribute groups with number of monitoring server for collection

To benefit from this feature, you have to assign the agents to different monitoring servers when implementing your environment. This feature can also be used to temporarily shift agents to the data collection in certain situations. If you are planning to use different collection parameters for a given attribute group on different monitoring servers, these configuration actions have to be done as separate actions through the historical collection configuration user dialog (that is, not at the same time).

4.4.2 Configure multiple Warehouse Proxy agents

Multiple Warehouse Proxy agents has been supported since IBM Tivoli Monitoring V6.1 Fix Pack 2, but Fix Pack 5 should be installed in order to use all the features described here.

> **Note:** A Technote explaining best practices for configuring multiple Warehouse Proxy agents is available at the following site:
>
> http://www-1.ibm.com/support/docview.wss?rs=0&q1=Configuring+Multiple+Warehouse+Proxy+Agents&uid=swg21268675&loc=en_US&cs=utf-8&cc=us&lang=en

More than one Warehouse Proxy agent could be beneficial for the following reasons:

- Maintenance

  When maintenance procedures (such as adding more memory, replacing a disk, or defragmenting a filesystem) are applied on the machine where one Warehouse Proxy agent is installed, other Warehouse Proxy agents can be used to send the historical data to the Tivoli Data Warehouse database.

- Sharing the load

  A large volume of simultaneous uploaded requests may negatively affect the insertion rate if only one Warehouse Proxy agent is used. 5.5, “Tivoli Data Warehouse performance” on page 151 advises how to set some Warehouse Proxy tuning variables such as the number of export threads, the size of the queue containing the historical exports, the batch option, or the database itself when the Warehouse Proxy agent has to handle large data volumes.

  Note that, if one Warehouse Proxy agent cannot handle all the historical exports despite the tuning, the workload can be shared by installing multiple Warehouse Proxy agents.

- Failover

  Tivoli Monitoring 6.1 FP5 provides the ability to automatically send historical uploads to a failover Warehouse Proxy agent when the communication to the primary Warehouse Proxy agent is unavailable. However, this facility does not provide load balancing or an automatic switch-back to the original Warehouse Proxy agent.

Determining which Warehouse Proxy agent will act for which agent is based on:

- For an agent: which TEMS does it report to
- For a Warehouse Proxy agent: the list of TEMS it is related to for warehousing purposes

The second relationship is set by the KHD_WAREHOUSE_TEMS_LIST environment variable in the Warehouse Proxy agent environment file.

KHD_WAREHOUSE_TEMS_LIST is a space-separated, semi-colon-separated, or comma-separated list of TEMS names.

The primary Warehouse Proxy agent for all agents reporting to a given TEMS is the Warehouse Proxy agent which includes the name of this TEMS in its KHD_WAREHOUSE_TEMS_LIST.

A Warehouse Proxy agent without any KHD_WAREHOUSE_TEMS_LIST variable set, or with a variable containing the string '*ANY' serves automatically all the agents that do not have a primary Warehouse Proxy agent. In this case, it is known as the default Warehouse Proxy agent for all agents.

A default Warehouse Proxy agent should be started last of all the Warehouse Proxy agents, so that agents use their primary Warehouse Proxy agent if it is available.

The relationships indicating which Warehouse Proxy agent is acting for which agent are stored in the hub monitoring server. This is why all Warehouse Proxy agents must be configured to use the hub monitoring server. A Warehouse Proxy agent can never report to a remote monitoring server.

The relationships are replicated to the other monitoring servers based on intervals. It may take up to an hour for those relationships to be synchronized across all monitoring servers.

A monitoring server name should only be specified in the list for a single Warehouse Proxy agent. In other words, do not specify the same monitoring server name in more than one Warehouse Proxy agent monitoring server list. Otherwise, it will be impossible to reliably predict which Warehouse Proxy agent is acting for a given agent.

This is because the same monitoring server name will be registered as a primary monitoring server for multiple Warehouse Proxy agents and depending on the order in which these Warehouse Proxy agents start and register with the hub monitoring server, a different connection topology will exist.

To support the agents prior to IBM Tivoli Monitoring V6.1, the default Warehouse Proxy agent is used. The default Warehouse Proxy agent can be configured in three ways, as explained here:

- KHD_WAREHOUSE_TEMS_LIST variable does not exist
- KHD_WAREHOUSE_TEMS_LIST is empty
- KHD_WAREHOUSE_TEMS_LIST contains the value *ANY

The first time an agent wants to export the historical data, it tries to create a Remote Procedure Call (RPC) route with a Warehouse Proxy agent. The agent will get its preferred route address. Therefore, a list of Warehouse Proxy agents will be contracted, containing the warehouse entries with an annotation equal to the agents' monitoring server name and entries with the default annotation.

But for each, there can be more than one entry because of using different protocols. The agent will try the warehouse entries for exporting the historical data in sequence. When the preferred Warehouse Proxy agent fails, the agent will use the default Warehouse Proxy (if specified). When the preferred Warehouse Proxy is restarted, the agent will reconnect to it.

If one or more monitoring servers are not included in one of the Warehouse Proxy agents lists and if no default Warehouse Proxy agents is specified, the agents connected to these monitoring servers will not export the historical data into the Warehouse database. Unless action is taken, the short-term historical data will eventually fill up the disk at the collection location.

The monitoring server which is not included in the Warehouse Proxy agents' list can be selected within the Historical Data collection dialog box, as shown in

Figure 4-20 on page 105. The short-term historical data for these agents will be collected additionally in certain situations.

But use this option carefully. If the Warehouse Interval is not Off (because you collect long-term historical data for these attribute groups for other agents) for the agents connected to the monitoring server which is not included in the Warehouse Proxy agents list, then the short-term historical data files will grow continuously. The automatic deletion of data that is older than 24 hours (or as specified in the KHD_HISTRETENTION variable) will not work. For this reason, you should pay attention to collecting this data and later, delete it manually.

Important: We recommend that one default Warehouse Proxy agent is configured. For agents connected to a monitoring server which have no annotation to any Warehouse Proxy agent, the short-term historical data will not deleted; the data files will continue to grow.

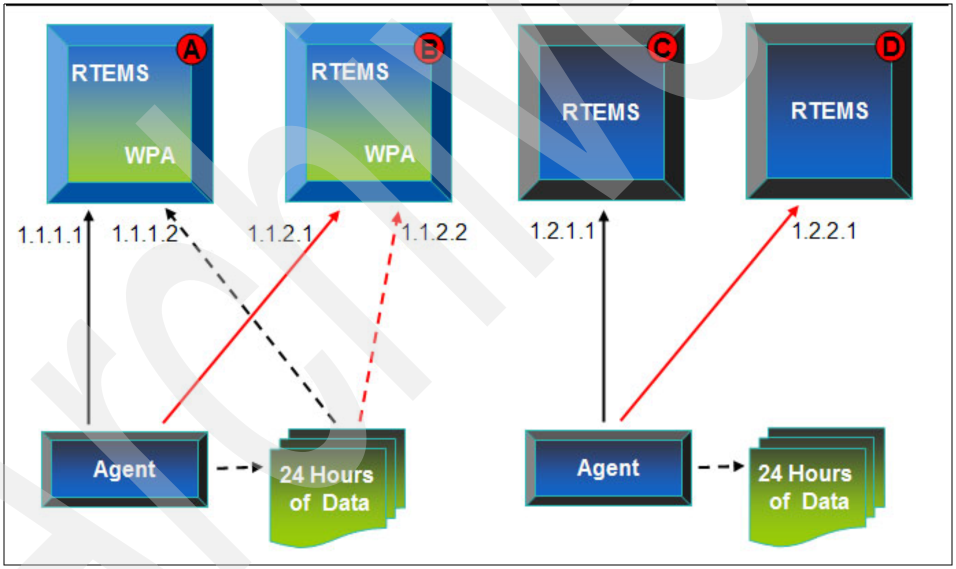


Figure 4-22 Example architecture for using Warehouse Proxies

Figure 4-22 shows the process of using the Warehouse Proxy agents for only some of the agents. The Warehouse Proxy agents are located on the same system where the remote monitoring server resides. There is no default Warehouse Proxy agent.

When such a system fails, these agents will be connected to the backup remote monitoring server using the secondary monitoring server support. For the backup remote monitoring server, an exclusive Warehouse Proxy agent is selected, located on the same system with the backup remote monitoring server. So a backup scenario for these agents that is created.

Figure 4-22 on page 109 shows collecting historical data into the data Warehouse database via the Warehouse Proxy agent located on server A for only a part of the agents. The agents are connected with the remote monitoring server on the same server.

When server A fails, all agents will be connected to server B. The Warehouse Proxy agent on server B will have the connection to the Warehouse database for the connected agents. All agents connected to the remote monitoring server located on server C will not upload historical data into the Warehouse database. Server D can be a backup system for server C and will be connected with the Secondary monitoring server support, as well.

The configuration for KHD_WAREHOUSE_TEMS_LIST looks as follows:

- Server A: KHD_WAREHOUSE_TEMS_LIST = REMOTE_A
- Server B: KHD_WAREHOUSE_TEMS_LIST = REMOTE_B

If you have concerns about the bandwidth utilization when the remote monitoring server and the Warehouse Proxy agent are installed on the same server, you can use the information in 4.5.1, “Estimate the size for bandwidth” on page 111 to estimate the bandwidth utilization.

Configuring for multiple adapters is possible if utilization of a single network adapter turns out to be a bottleneck. The selection of the interface for using the Warehouse Proxy agent can be done with the variable described here:

- KDEB_INTERFACE_CLIENT

This variable can be used to remove an interface and set on the Warehouse Proxy agent. For the example in Figure 4-22 on page 109, this should be set as shown:

```
Server A: KDEB_Interface_List=-1.1.1.1
```

In large scale environments, the following variable could be of interest in this context.

- KPX_WAREHOUSE_LOCATION

This variable is set on the Warehouse Proxy agent and allows a fixed warehouse route via a Warehouse Proxy agent to be delivered to connected agents. The variable contains an optional list of fully qualified network names delimited by semicolons.

When specified, this variable disables the proxy monitoring of the Warehouse Proxy location broker registrations and replaces the routing string currently derived by the proxy via the location broker monitoring, for example:

```
KPX_WAREHOUSE_LOCATION=ip.pipe:#1.1.1.1[18303];ip:#1.1.1.1[34543]
```

It is also important to note that this variable has a maximum length of 200 bytes. If the size of this variable is too short, some design consideration should be taken to move some agents to be connected to a different monitoring server.

To indicate a failover Warehouse Proxy agent when using the KPX_WAREHOUSE_LOCATION variable, add the failover Warehouse Proxy agent address after the primary Warehouse Proxy agent address.

There is no validation checkup on the content of this variable. Setting a Warehouse Proxy agent address is not detected until an export occurs.

- KPX_WAREHOUSE_REGCHK

This variable is also set on the monitoring server and allows to define the time span in minutes for reprocessing the registration of the content from the KHD_WAREHOUSE_TEMS_LIST variable. The default value is 15 minutes.

4.5 Other planning considerations for collecting historical data

One reason for having multiple Warehouse Proxy agents in an IBM Tivoli Monitoring Version 6.X environment is to have an optimum configuration when using no more than 2000 clients to each proxy using the ip.pipe or ip.spipe protocol. We will discuss determining the bandwidth, upload time, security and performance considerations in this context.

4.5.1 Estimate the size for bandwidth

One important question that comes to mind is the required bandwidth for the Warehouse Proxy agent, particularly in regard to using the same system for remote monitoring server and Warehouse Proxy, as shown in Figure 4-22 on page 109. This is a simple calculation. Here the value "Bytes per instance (agent)" (or "AgentsRowSize") should be taken into account when calculating the bandwidth.

For a Warehouse Proxy located on a single system connected to 2000 agents, the data must go in and go out of the warehouse with little overhead. For example, the if clients are all Windows OS agents collecting from default attribute groups with a collection interval of 5 minutes (1,1 MByte per agent (see the example in 4.3.2, “Estimate agents disk space for data collection using the Warehouse Load Projections spreadsheet” on page 94), then the amount of collected will not be more than 5 GB per day.

If the Warehouse Proxy agent has a 100 Mb/s interface, then less than 1% of the bandwidth will be used. But the same consideration holds true as discussed in 4.3, “Size the Tivoli historical data collection” on page 80. That is, if there are attribute groups with multiple instances, then the value can be multiples of this number. In every case, the initial value is the “Bytes per instance (agent)” and this amount of data in most cases goes in and out the Warehouse Proxy agent.

As shown in Figure 4-22 on page 109, you can put the remote monitoring server and the associated Warehouse Proxy agents on one system. The memory and disk requirements have to be added and you should follow the recommendation to connect not more than 1000 agents to the remote monitoring server and so to the Warehouse Proxy agent, as well. The bandwidth for the Warehouse Proxy agent should be estimated roughly.

4.5.2 Determine the upload time

Another question can be how to specify the time that data will be uploaded from the Warehouse database via the Warehouse Proxy agent, particularly when the Warehouse interval is configured to run once a day. There are three considerations in determining this time:

- Adjusting the upload time to run outside the office hours for saving bandwidth
- Synchronizing the upload time of detailed data with the Summarization and Pruning agent
- Synchronizing the upload time with creation of a data mart (if used)

There is no command line interface to configure or start the historical data collection. Only the Historical Data Collection dialog box can be used for start the collection.

If you start a data collection for an attribute group, the data will be written on the local disk. Approximately one hour or one day later (depending on the chosen Warehouse interval), the upload process will start. That means, for example, if there is a Warehouse interval of one day and the collection will be started at 4 a.m., then the first upload will be done at 4 a.m. the next day.

If there are a large number of agents for the selected attribute group, it will take a long time to upload all data from all agents into the Warehouse database. But there is no guarantee that the upload interval for the agents can be adjusted in the early morning time, as in this example. Downtimes of agents, monitoring servers, or the Warehouse Proxy agents over many days could delay the upload time.

> **Note:** The best way to upload historical data continuously for saving summarization and provisioning for a data mart is to use an hourly Warehouse interval.

It is possible to stop all the Warehouse Proxy agents within the peak shift or office hours. But in this case the agents will still attempt the upload, sending a batch of data to the Warehouse Proxy agent machine, only to have the RPC fail, so this might not be a desirable solution.

When you use the daily Warehouse interval for upload, it is a best practice to start the historical data collection initially for a large environment outside of office hours to save bandwidth.

As discussed earlier, collecting short-term historical data on the monitoring server drastically reduces the number of agents that can be supported by the monitoring server. Portal requests for short-term historical data can take longer, because all agent data for a given attribute group is collected into a single file, and additional I/O is required to satisfy the portal request.

This strategy might be considered for environments with slow network connections between the agents and monitoring server, which would spread the upload network traffic more evenly throughout the day, instead of having a large hourly or daily spike for the warehousing upload traffic. This would be a trade-off of the extra memory and disk I/O at the monitoring server versus the data transmission from the agent to the monitoring server.

4.5.3 Considerations regarding performance

The next consideration in this context would be in regard to collecting the short-term historical data on the remote monitoring server. This can be a option for performance (see 5.2, “Tivoli Enterprise Monitoring Server performance” on page 123).

If you configure remote monitoring server and Warehouse Proxy agent on the same system, the data will have to traverse the network only one time, to the monitoring server. The pipe from monitoring server to Warehouse Proxy agent occurs internally on the system (so it does not traverse the network). Another benefit is that the data flow is more continuous.

In this context, we should clarify two points:

- Required local disk size on the remote monitoring server
- What happens to the short-term data when the monitoring server fails

Regarding the first point, you should calculate the agent's row size and multiply with the number of connected agents. For example, in 4.3.2, "Estimate agents disk space for data collection using the Warehouse Load Projections spreadsheet" on page 94, for the Windows OS default groups for 1000 agents you need approximately 2 GB local disk space for collecting data. The number of instances should be taken into account and the disk capacity and the Warehouse Proxy availability should be monitored, as well. The short-historical data files are located in the following location, where ITMinstall_dir is the directory where the product is installed.

- ITMinstall_dir\CMS\

Regarding the second point, if the monitoring server is selected as Collection Location for short-term historical data and this monitoring server fails, the data can be lost for the period of failure.

Only the collected data from the last interval is stored in the agents cache, and it will be sent to the monitoring server again. So when this option will be needed, you should implement the Secondary monitoring server support.

For example, if there is a collection interval of 15 minutes, it is unlikely that any date will be lost. If the collection interval is only 5 minutes, in worst case one interval of data can be lost, because the agent could need up to 10 minutes for reconnect to the Secondary monitoring server by default. An option for bypassing can be to set the CTIRA_RECONNECT_WAIT variable to a smaller value.

Another aspect of performance regarding historical data is to make sure that there is fast access to this data. Therefore, using a data mart can be an option. A *data mart* includes only data for user access for reporting, and it will be built from the Warehouse database. In most cases, this is a daily task. These aspects are discussed in the IBM Redbooks publication *Tivoli Management Services Warehouse and Reporting*, SG24-7290.

4.5.4 Considerations regarding security and availability

The reasons for locating a remote monitoring server and Warehouse Proxy on different locations within the network or to collect short-term historical data on a monitoring server can also be considered in the same context with questions or policies regarding security. These topics are discussed part of Chapter 3, “Firewall considerations” on page 37.

Another consideration is the need for high availability. This topic is discussed in Chapter 10, “IBM Tivoli Monitoring resiliency and high availability” on page 331.

5

Optimizing performance within an IBM Tivoli Monitoring environment

This chapter discusses the performance of several components within an IBM Tivoli Monitoring environment, and provides considerations for optimizing their performance.

> **Note:** This chapter refers to parameters within various environment settings files that you may consider changing as part of the performance optimization process. Be aware that whenever maintenance or reconfiguration takes place in your environment, these changes may be lost and need to be reapplied.

The following topics are addressed:

- Setting the heartbeat frequency
- Tivoli Enterprise Monitoring Server performance
- Tivoli Enterprise Portal Server performance
- Tivoli Enterprise Portal client performance
- Tivoli Data Warehouse performance
- OMEGAMON XE performance

- Event management integration performance
- Query optimization
- Situation optimization

5.1 Setting the heartbeat frequency

Monitoring the heartbeat is essential to monitoring the availability of different components within the Enterprise Software. IBM Tivoli Monitoring V6.1 provides a heartbeat monitoring capability within its architecture.

Several key variables are used within the components of IBM Tivoli Monitoring V6.1 (hub monitoring server, remote monitoring server and monitoring agent) for the heartbeat mechanism, as explained here:

- CTIRA_HEARTBEAT (on the agent) - minutes [Default 10] is the time, in minutes, of how often the monitoring agent sends a status or update to its monitoring server.
- CTIRA_HEARTBEAT (on the remote monitoring server) - minutes [Default 3] is the time, in minutes, of how often the remote monitoring server sends the status or update to its hub monitoring server.

> **Note:** CTIRA_HEARTBEAT (on the hub monitoring server) is also called *self-heartbeat*. With the current implementation, the hub heartbeat means that the hub monitoring server will "self-heartbeat" every 3 minutes by default.
>
> The self-heartbeat is used to determine whether SITMON is healthy. If the timer expires for a self-SITMON heartbeat, the nodes are taken offline. So if the hub's self-heartbeat fails, then all agents connected directly to the hub are taken offline.
>
> This behavior is expected to be changed in IBM Tivoli Monitoring V6.2 and the self-heartbeat will not be allowed to expire.

- CTIRA_RECONNECT_WAIT - seconds [Default 600] is the time, in seconds, that the monitoring agent waits before attempting to reconnect with the monitoring server.
- CTIRA_MAX_RECONNECT_TRIES [Default 720] controls how many times the monitoring agent will try to reconnect before exiting.

Each monitoring agent can have a different CTIRA_HEARTBEAT value, if that is what you want. The remote monitoring server performs the availability monitoring for any nodes connected directly to that remote monitoring server or for subnodes connected to an IRA Manager that is connected to the remote monitoring server.

The monitoring server maintains expiration timers and monitors the nodes it is responsible for. The remote monitoring server reports the availability for itself to the hub monitoring server using its heartbeat, and additionally will report any status changes to the hub for any nodes connected via the remote monitoring server. If the remote monitoring server becomes unavailable, then all the agents reporting through that remote monitoring server will also be marked as OFFLINE.

The heartbeat between the remote monitoring server and the hub monitoring server by default is set to 3 minutes. The heartbeat between the monitoring agent and the remote monitoring server by default is set to 10 mins. The heartbeat flows from down (monitoring agent) to up (hub monitoring server), or from right to left, as illustrated in Figure 5-1.

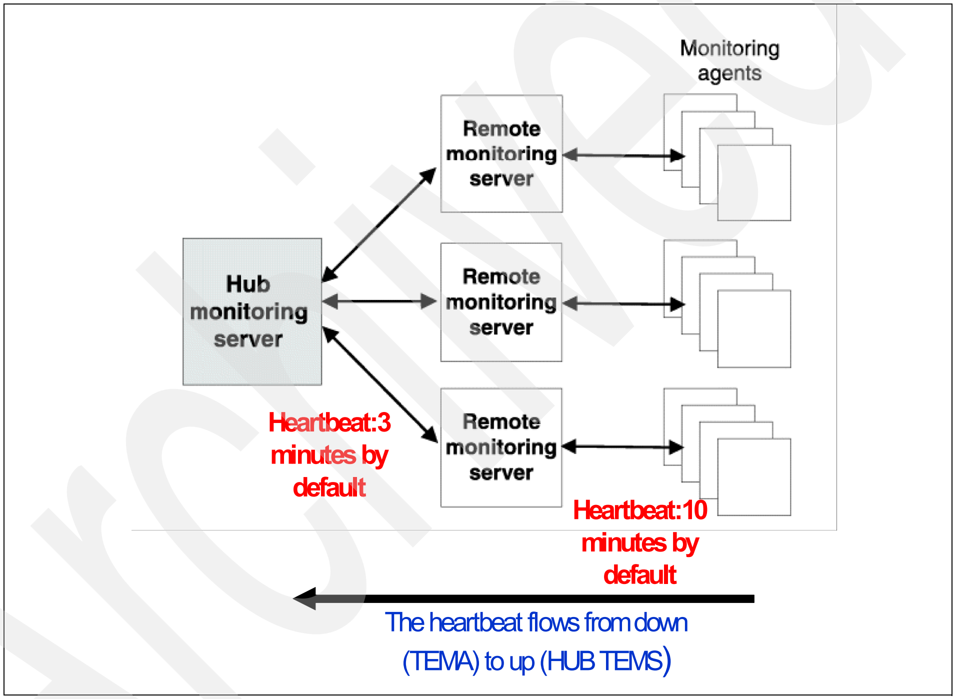


Figure 5-1 IBM Tivoli Monitoring heartbeat architecture

When the monitoring agent logs in to a remote monitoring server, it passes its CTIRA_HEARTBEAT information to the remote monitoring server (in cases where no explicit CTIRA_HEARTBEAT variable has been specified in the monitoring agent configuration, the default value of 10 mins is used).

The remote monitoring server then sets a timer based on that value and waits for the next heartbeat interval. If an agent crashes (that is, it does not send an offline heartbeat), then the remote monitoring server will wait a 3-minute grace period on top of the remaining time for the heartbeat to arrive. After this time expires, the remote monitoring server sets the state to OFFLINE and INSERT the node status record with the OFFLINE notification to the hub monitoring server. The hub monitoring server posts the event to the portal server for display on the console.

So if the heartbeat of the agent is 3 minutes, the following sequence takes place:

1. The agent heartbeats are at 00:00 (relative time).
2. A timer is set for the agent to be current time + heartbeat interval (180 seconds) + GRACE_PERIOD (180) seconds.
3. The agent crashes at 00:01.
4. The timer expires at 06:01 after heartbeat.

At this point, the node tables will be marked to reflect the status and the event is posted to the portal console. The event is delivered immediately to the portal server.

> **Note:** The portal server maintains a number of SQL1 queries open on the hub monitoring server that is used to update the console.
>
> This SQL1 pseudo call is an example:
>
> ```
> SELECT column1, column2 from table name WHERE TIMESTAMP > now.
> ```
>
> This means that there is something new in the hub monitoring server; that is, a node change state is reported back to the portal server when the data is written into the hub monitoring server table.
>
> When an agent goes offline, the node status table is updated, and this SQL1 data record is reported back to the portal server, and then presented to the user.
>
> When a situation fires, the same thing happens. The portal server keeps an outstanding SQL1 on the node status and situation status tables, and that is how the console is updated. So when the event happens at the hub monitoring server, it is sent to the portal server. (Depending on the workload, there might be slight delays in this sequence.)

The MS_Offline situation has a sample interval of 1 minute. This situation samples the node tables to see if the node state is OFFLINE. If the sample took place in the above timeline, at 6:00 it would not fire. It could take up to 7 minutes after a node crashed (with a 3-minute heartbeat) for it to be put up on the situation event console.

Now considering the default heartbeat time for an agent is 10 minutes, that would push the situation firing for an offline node out to 14 minutes after the crash.

If an agent crashes, and there is an open situation for that node, then it can take 3 times the sample period for the hub monitoring server to close the situations.

This is because there are 3 retries by proxy for the sample. So if this situation has a 15-minute sample rate, then it could take 45 minutes from the first sample to close the situation after a crash. For example:

- **1:00** first sample true - fire
- **1:15** sample true - no change in status
- **1:15:01** kill agent
- **1:30** proxy sends last sample
- **1:45** proxy sends last sample
- **2:00** proxy gives up. Return no sample.

The monitoring agent, when configured with multiple remote monitoring servers, only sends updates to its primary remote monitoring server and ignores the secondary remote monitoring server. If the primary monitoring server is not available, after a timeout the monitoring agent attempts to communicate to the next monitoring server listed in CT_CMSLIST.

If that also times out, it switches back to the original monitoring server. That continues until one of the monitoring servers responds. The time delay is CTIRA_RECONNECT_WAIT, default 600 seconds (10 minutes).

If the monitoring agent fails to connect to a monitoring server, it will keep trying. By default, it will wait 10 minutes between attempts and try for 5 days before giving up. When it gives up, the monitoring agent shuts down and ends. The total number of attempts (default) is 720. If there were 2 entries in CT_CMSLIST, then each monitoring server would be contacted 360 times before a failure was declared.

You can modify the frequency and number of retries by monitoring agent to its monitoring server. Options listed here will allow the monitoring agent to try to reconnect every 5 minutes for 2.5 days before giving up.

CTIRA_MAX_RECONNECT_TRIES=720

CTIRA_RECONNECT_WAIT=300

The exact workload that a heartbeat signal generates on the remote server depends on which transport protocol the agent is using to communicate with the server. The heartbeat workloads driven by IP.PIPE and IP.SPIPE transports typically exhibit better average performance than workloads driven by the IP.UDP transport.

The heartbeat workload remains manageable for all transports as long as large numbers of node status changes do not occur. This assumes an agent heartbeat interval of 10 minutes and no status changes.

Each heartbeat signal generates about 2 kilobytes of network traffic between the parent and child. For a remote server monitoring 1000 agents that have heartbeat intervals of 10 minutes, this generates about 12 megabytes of network traffic over an hour.

If a monitoring agent does generate a status change, either by coming online or going offline, then the workload for the remote server increases. The remote server must communicate the status change to the hub. In addition, the portal server responds to the status change by updating the monitoring agent's status and generating messages indicating the status change has occurred.

The workload generated by a single status change is minimal for all components, but if hundreds or even thousands of status changes occur simultaneously, the cumulative workload can cause significant impact to each component. This is particularly true for the portal clients, which display status messages in the graphical message log for each status change.

> **Important:** The following guidelines should be followed when deciding whether an agent's heartbeat interval should be changed or not:
>
> - Consider lower heartbeat intervals for agents monitoring critical systems. If a system is not critical, a 10-minute heartbeat interval is probably sufficient.
> - In large environments with thousands of agents and monitoring servers handling a large volume of situation event workload, it may be desirable to change the default heartbeat interval for agents or remote monitoring servers to an appropriately higher value. Be aware, however, that increasing the heartbeat interval will increase the time to know that an agent or other component might have failed.
> - When increasing the heartbeat interval, be careful not to exceed the “idle connection” timeout values for any intervening firewalls.

5.2 Tivoli Enterprise Monitoring Server performance

The Tivoli Enterprise Management Server (monitoring server) manages and processes monitoring and automation tasks for any attached agents, applications, adapters, or gateways.

This section discusses considerations for optimizing the monitoring server performance, namely:

- General monitoring server performance considerations
- Monitor server parameter tuning performance considerations

In any logical monitoring server configuration, there is exactly one hub monitoring server and any number of attached remote monitoring servers connecting to the hub monitoring server. The number of remote monitoring server connected to a given hub provides some measure of horizontal scalability load balancing as each remote monitoring server controls its own workload on attached resources.

The hub monitoring server handles all propagated events originating at attached resources, that is monitoring agents, reporting either through the remote monitoring server or connected directly to the hub. As a general practice, attached resources should report via a remote monitoring server and not directly to the hub monitoring server thereby allowing the hub monitoring server more bandwidth for evening workloads originating from each remote monitoring server.

Smaller configurations with very few attached resources may only require a hub monitoring server if the event workloads are sufficiently constrained. Performance and response time for monitoring, availability, and automation events flowing from resources through remote and hub monitoring server subsequently used by portal server or other monitoring facilities, such as adapters or gateways, are also affected by vertical performance considerations.

Vertical performance and response time concerns the tuning at each monitoring server for the best use and practices of workload management. Such tuning is generally controllable by end users through sampling intervals given to individual situations, limiting situation monitoring objects necessarily deployed to achieve end-user desired results, and defining filtering criteria for situations that return only necessary information to monitor a particular condition.

Situations that have matching monitoring attributes and monitoring intervals may be internally combined into single monitoring objects. The combining of situation objects into a single filtering object is called *situation synchronization* or *duperization*. Situation synchronization can have both beneficial and detrimental influences on work load performance. A monitoring server that contains address space resident applications tends to benefit the most from synchronization, because sampling is occurring in the monitoring server runtime environment. A monitoring server that has mainly IRA-based agents attached to it will benefit far less or may even suffer response time degradation with synchronization applied to enough situation objects. Situation synchronization is discussed in further detail in 5.9.5, “Situation synchronization” on page 192.

5.2.1 Monitoring server parameter tuning

This section provides information about parameters you may consider editing to improve either hub or remote monitoring server performance. The parameters are set in the following files according to operating system:

- Windows

  <ITM HOME>\cms\KBBENV

  Example: C:\IBM\ITM\cms\KBBENV

- Linux and UNIX

  <ITM HOME>/config/<tems-hostname>_ms_<temsname>.config

  Example: /opt/IBM/ITM/config/edinburg_ms_labtems.config

- z/OS

  &shilev.&rtename.RKANPAR(KDSENV)

  Example: ITM.SYP1.RKANPAR(KDSENV)

> **Note:** The &shilev and &rtename variables correspond to high level qualifiers of the RKANPAR(KDSENV) partitioned dataset. These variables can take 1 to 8 characters.

Be aware that whenever maintenance or reconfiguration takes place in your environment, these files may be recreated and the changes lost and need to be reapplied.

> **Note:** Because some of the monitoring server performance parameters might depend on the agent parameters, we list the configuration parameters on agents as well:
>
> - Windows: <ITM HOME>\tmaitm6\K<ProductCode>ENV
> - Linux and UNIX: <ITM HOME>/config/<ProductCode>.ini

The following lists the settings that may affect the monitoring server performance.

- KDS_NCSLISTEN

  This parameter governs the number of concurrent remote procedure calls (RPCs) allowed in execution and has a direct effect on the system utilization. The default value is 10 and although this may be enough for modest systems, larger enterprises may want to increase this value to improve concurrency performance. The maximum value is 256 and large scale deployments should set this value for optimum performance.

- KDCFP_RXLIMIT

  This parameter establishes the maximum number of 1 KB packets which may be transmitted to this endpoint in a single RPC request or response. The default is 4096 KB (4 MB); the minimum is 1024 KB (1 MB); there is no maximum.

  If the remote endpoint (session partner) exceeds this limit (that is, sends more), then the RPC request will be failed with status KDE1_STC_RXLIMITEXCEEDED.

  The intent of RXLIMIT is to prevent memory overrun by placing an upper limit on a single RPC request or response. If sufficient capacity exists in a large scale deployment, consider setting this value to 8192.

  To increase the buffer size to 8 MB, include the following environment setting:

  KDCFP_RXLIMIT=8192

- CMS_DUPER

  This parameter enables or disables situation synchronization of common filter objects actually monitored by agents or endpoints. Enabling this setting in monitoring server environments with predominantly z/OS address space applications (for example, OMEGAMON XE for CICS or sysplex) improves performance and response time by limiting data collection samplings on behalf of running situations.

  You enable it by setting the value to YES. You disable by setting the value to NO. By default, this parameter is not enabled. Situation synchronization is discussed in further detail in 5.9.5, "Situation synchronization" on page 192.

- EVENT_FLUSH_TIMER

  This parameter is used to specify, in minutes, how long the monitoring server should wait before resetting distribution and database event requests to an initial state, thereby freeing held resources by the request if no event information has been able to get processed in the specified time.

  The default setting is 2 minutes. If event requests are not responding within 2 minutes, it may be desirable to allow for a higher minutes setting in order to give requests more time to process, particularly in larger, more complex environments.

- EIB_FLUSH_TIMER

  This parameter is used to specify, in minutes, how long the monitoring server should wait before resetting distribution and database event requests to an initial state, thereby freeing held resources by the request if no event information has been able to get processed in the specified time.

The default setting is 2 minutes. If event requests are not responding within 2 minutes, it may be desirable to allow for a higher minutes setting in order to give requests more time to process, particularly in larger, more complex environments.

- DELAY_STOP

  This parameter is used to specify, in seconds, how long to delay monitoring server shutdown for UNIX and Linux monitoring servers, as invoked by the **`itmcmd server stop`** ***<temsname>*** command. The default value is 60 seconds.

  The delay is used to allow network connections to close prior to an immediate restart of the monitoring server with the **`itmcmd server start`** ***<temsname>*** command. If you do not immediately restart the monitoring server after shutting it down, this parameter can be set to a lower value to cause the monitoring server shutdown to complete more quickly.

- KGLCB_FSYNC_ENABLED

  This parameter was added in IBM Tivoli Monitoring V6.2 for the monitoring server on UNIX and Linux platforms. This parameter is not available on Windows systems, and is not available on IBM Tivoli Monitoring V6.1 systems.

  For Linux and UNIX platforms, this variable can be used to specify whether the fsync() system call should be invoked after writes to the filesystem. This configuration variable may be set in the standard configuration file for the monitoring server. By default, for maximum reliability, fsync() will be called.

  If—and *only* if—the following line is added to the monitoring server configuration file, fsync() calls will be omitted:

  KGLCB_FSYNC_ENABLED='0'

  The default behavior is to call fsync() after writes, which is equivalent to the following setting:

  KGLCB_FSYNC_ENABLED='1'

  The fsync() system call flushes the filesystem's dirty pages to disk and protects against loss of data in the event of an operating system crash, hardware crash, or power failure. However, it can have a significant negative effect on performance because in many cases it defeats the caching mechanisms of the platform file system.

  On many UNIX platforms, the operating system itself synchronizes the entire filesystem on a regular basis. For example, by default the syncd daemon that runs on AIX syncs the filesystem every 60 seconds, which limits the benefit of fsync() calls by application programs to protecting against database corruption in the most recent 60-second window.

5.3 Tivoli Enterprise Portal Server performance

The Tivoli Enterprise Portal Server (portal server) is a key component in the IBM Tivoli Monitoring environment. It acts as a conduit for Tivoli Enterprise Portal clients requesting data for analysis from monitoring agents and other components within the enterprise.

The portal server connects directly to the hub monitoring server, which it queries for enterprise information, and the portal server receives updates as they occur. Because it is responsible for handling large amounts of data, the portal server can be a potential bottleneck within the IBM Tivoli Monitoring environment. The following considerations for optimizing portal server performance are detailed here:

- Portal server database tuning
- Configure an external Web server for large environments
- Deploy multiple view-only servers
- Maintain multiple view-only portal servers
- Portal server parameter tuning
- Portal server database memory model on AIX

5.3.1 Portal server database tuning

The portal server database stores information related to the presentation of monitored data at the portal client, including definitions related to users, workspaces, and views. After IBM Tivoli Monitoring V6.1 Fix Pack 3, the portal server also stores information related to events and event attachments in the portal server database.

Prior to IBM Tivoli Monitoring V6.1 Fix Pack 3, the portal server database previously required little or no tuning. In environments with a moderate to high rate of events, however, the portal server database may require some tuning to optimize performance. In particular, the KFWTSIT table, which is used to store events, can grow large.

If you are using DB2 for the portal server database, consider the following tuning recommendations:

- The default buffer pool size is 250 4K pages. Increase the size of this buffer pool to minimize disk I/O activity.

  The following commands illustrate how to increase the size of the bufferpool to 2000 4K pages (from a DB2 command prompt):

```
CONNECT TO TEPS;
ALTER BUFFERPOOL IBMDEFAULTBP IMMEDIATE SIZE 2000;
CONNECT RESET;
```

- Because the KFWTSIT table is the most active table, use the runstats and reorg facilities on a regular (for example, daily) basis.

  The following commands illustrate how to run the RUNSTATS facility on the KFWTSIT table (from a DB2 command prompt):

  ```
  CONNECT TO TEPS;
  RUNSTATS ON TABLE ITMUSER.KFWTSIT ON ALL COLUMNS ALLOW WRITE ACCESS
  ;
  COMMIT WORK;
  CONNECT RESET;
  ```

  Note that ITMUSER is the instance owner.

- The following commands illustrate how to run the REORG facility on the KFWTSIT table (from a DB2 command prompt):

  ```
  CONNECT TO TEPS;
  REORG  TABLE ITMUSER.KFWTSIT;
  REORG INDEXES ALL FOR TABLE ITMUSER.KFWTSIT ALLOW WRITE ACCESS;
  COMMIT WORK;
  CONNECT RESET;
  ```

These tuning changes can also be made using the DB2 Control Center.

5.3.2 Configuring an external Web server for large environments

If there will be more than ten portal clients connected to the portal server concurrently, consider configuring an external Web server to work with the portal clients. An external Web server will offload processing from the portal server process (KfwServices), and offer enhanced scalability.

Instructions for configuring an external Web server to work with the Tivoli Enterprise Portal can be found in the chapter “Additional Tivoli Enterprise Portal configuration” in *IBM Tivoli Monitoring Installation and Setup Guide Version 6.1.0*, GC32-9407.

The steps for configuring an IBM HTTP Server for Windows to work with the Tivoli Enterprise Portal are similar to those for IBM HTTP Server for Linux. The only difference is that the DocumentRoot value should be set to itm_installdir/CNB, where itm_installdir is the directory where IBM Tivoli Monitoring is installed.

Here is an example:

DocumentRoot "c:/IBM/ITM/CNB"

5.3.3 Deploying multiple view-only portal server

The traditional recommended approach when planning and deploying an IBM Tivoli Monitoring installation is to insist on a single portal server for the environment from which all clients connect through to access data. Because the portal server is responsible for maintaining information about the users, their privileges, workspace details, and customizations, having more than one portal server communicating with the hub monitoring server could cause instability and serious problems in the IBM Tivoli Monitoring environment.

Typical examples could be loss of workspace data, changes made by one user lost because the change conflicted with data held on another portal server, or breakdown in the administration of users and their access rights. Additionally, any creation or update activity made from a client to the situations or monitoring agents are written to the hub monitoring server database. If there is more than one portal server online and able to update the hub monitoring server database, this may result in a potential clash and crash the monitoring server.

Deploying only a single portal server, however, can cause a number of problems, both logistically and performance-wise, particularly for large scale environments, as explained here:

- There is a limit (around 50 portal clients at the time of writing) to the number of Tivoli Enterprise Portal clients (both desktop and browser versions) that can connect to the portal server before performance is adversely affected. Slower performance could result in alerts being missed within the monitored environment.
- Large scale environments are more likely to be deployed across several geographically dispersed sites and perhaps across several time zones. Factors such as bandwidth latency and when to schedule offline maintenance sessions may need to be considered.
- There could be problems caused by firewall policies. It may be that most portal clients are situated outside of a firewall and there is strict enforcement of network security restrictions on what can and cannot pass through. Network administrators may not want to grant access to hundreds of clients to allow them access to important resources.

To address these issues, one or more additional portal servers may be installed and configured to communicate directly with the hub monitoring server. The difference with these additional portal server instances is that they must be configured to be *view-only*. That is, no users connected via these portal server may modify, for example, workspaces or other properties stored at the portal server. The original portal server should be designated as the *primary* or *master* portal server, and it alone will allow updates and administrative activities to be performed.

Although not allowing any client connected to a view-only portal server the ability to make updates sounds like a unwanted restriction, in practice, it can be a benefit by enforcing user access control. You may consider giving only administrative users IDs for the primary portal server. Regular non-administrative users do not need to be aware of the primary portal server, so they connect only to a defined view-only portal server.

Note: With multiple portal servers and a large number of portal clients, the increasing rate of portal client requests processed by the monitoring servers can impact the monitoring server performance. Monitoring the performance of the portal servers and monitoring servers is highly recommended, especially before and after adding portal clients, or when adding monitoring agents.

Scenario - Deploying view-only portal server

To illustrate the use of a view-only portal server, we describe a simple scenario comprising of a hub monitoring server, portal server and one or more remote monitoring servers. This multiple portal server solution requires additional portal servers to be installed and connected to the same hub monitoring server, as illustrated in Figure 5-2.

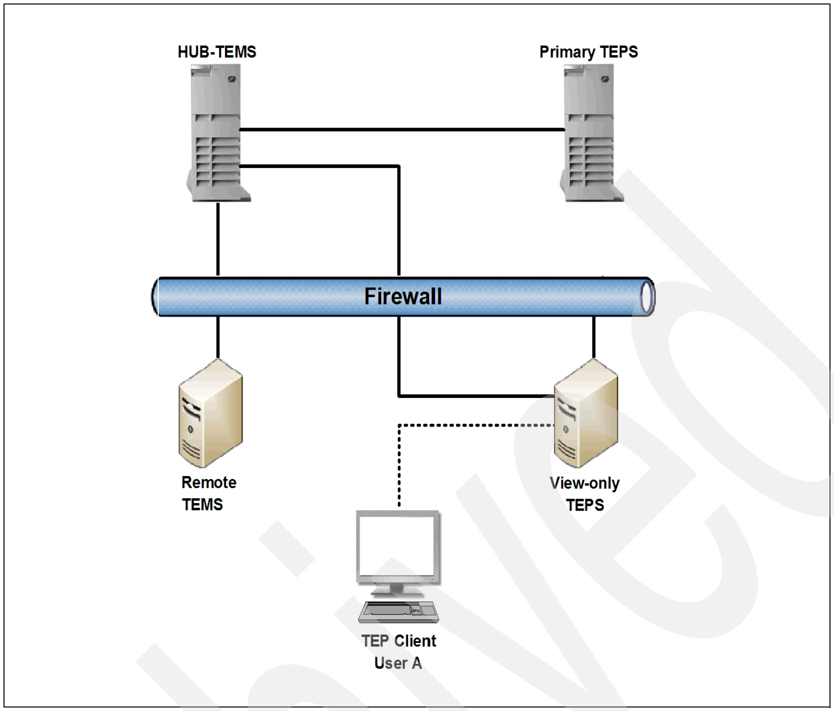


Figure 5-2 Sample environment deploying view-only portal server

As shown in Figure 5-2 on page 132, we have deployed a second portal server outside of the firewall from which the hub monitoring server and our original primary portal server are hosted. Perhaps this is because of restrictions forced on us by network administrators, or maybe we want to place this second portal server in a geographically different location. So, instead of being forced to allow all clients such as User A access through the firewall, we now simply need to enable our new portal server to connect to the hub monitoring server, and clients can now connect directly to this.

Because our original portal server has been designated as the primary, user permissions need to be set to view only for any users created on the view-only portal server by following these steps:

1. Log on to a portal client connected to view-only portal server using an administrator user ID (for example, “sysadmin”).
2. Open the Administer Users dialog box as shown in Figure 5-3 on page 133.

   Click **Edit -> Administer Users...**

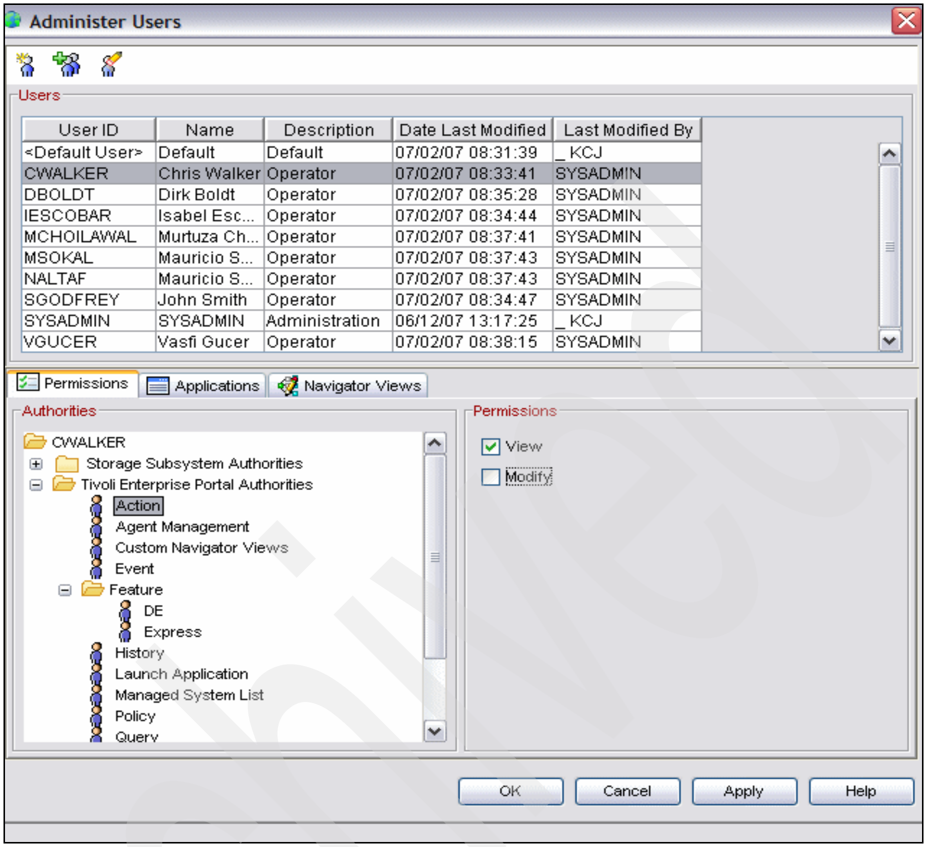


Figure 5-3 Configuring Administer Users

3. For each defined user, select the user name from the Users list and modify the permissions to match those defined in Table 5-1 on page 134. In general, the permissions needed are similar to those typically assigned to a system operator, an individual who can monitor and manage events, and use workspaces (whether product-provided or customer-defined) for problem determination purposes.

   However, the one exception here regards the *attach permission* associated with the Event authority. Normally an IBM Tivoli Monitoring operator would have the ability to use the attach feature to include relevant information about a given event that is being handled by that operator. Because all these attachments are stored at the portal server, they would not be shared across all of the configured portal server environments connected to the same hub monitoring server. For this reason, you may want to additionally remove that permission from the Event authority for view-only portal server environments.

Table 5-1 Portal server user authority reference

Authority	Permission
Action	View
Agent Management	Start/Stop
Custom Navigator Views	None
Event	Close, View, Acknowledge
History	None
Launch Application	Launch, View
Managed System List	View
Policy	View, Start/Stop
Query	View
Situation	View, Start/Stop
Terminal Script	View
User Administration	None
Workspace Administration	None

4. Click **OK**.

Note: Do *not* modify any permissions for the “sysadmin” user. This user should remain as is to allow an administrator to log on and make further changes when necessary.

At this point, regular non-administrator users can log on to their portal clients, accessing the view-only portal server and view data, alerts, and events as before.

Note: User administration within IBM Tivoli Monitoring V6.2 introduces the concept of user groups. This will allow an administrator, for example, to establish a user group definition with a set of permissions that represent some particular user persona.

There are two product-provided user groups: ADMINISTRATOR and OPERATOR. The OPERATOR group will have all of the permissions shown in Table 5-1. So, for an IBM Tivoli Monitoring V6.2 user, a portal administrator could create a set of users with no initial permissions, and then simply add them to the OPERATOR group. Users in the OPERATOR group would then be positioned, from a permissions perspective, to connect with a view-only portal server.

5.3.4 Maintaining multiple view-only portal servers

As updates to portal server properties, for example workspaces, are only now permitted on the primary portal server, any changes made must be migrated across to *all* view-only portal servers such that all users will be able to view the replica. The simplest way to accomplish this is by using the scripts provided by IBM Tivoli Monitoring to first export details out of the primary portal server, and then to import these into each view-only portal server available.

To export the data, run the script **migrate-export.sh** (on Windows, run **migrate-export.bat**) from the portal server subdirectory:

```
migrate-export.sh
```

Note that the portal server *must* be running when you run this script. The result of a successful export is an SQL file called saveexport.sql.

To import the data to another portal server, copy the file saveexport.sql to the machine running the view-only portal server and run the **migrate-import.sh** (on Windows, run **migrate-import.bat**) from the portal server subdirectory:

```
migrate-import.sh /tmp/saveexport.sql
```

If you do not supply a path to the file to import, then the path is assumed to be the same as used by the migrate-export script.

Note: These scripts will export all data from the primary portal server database including user administration details, and they will replace what may be contained on the importing portal server database. So make sure that you are not overwriting important data, particularly if you have different users defined on each portal server instance.

If you want to only export or import a subset of the portal server data, then consider creating a customized version of this script to extract only the required data from the portal server database.

An alternative to the scripts provided by IBM Tivoli Monitoring could be to use the migration tools provided by DB2 to extract the relevant data from the portal server database. This should be discussed with the database administrator.

The frequency of replication is dependent on the organization and the frequency with which portal server data changes. For example, if new workspaces are defined or edited regularly, or new users are added at several times during a week, then consider daily replications. In more stable environments where change is less common, the replication could be part of a weekly maintenance schedule or performed on demand.

5.3.5 Portal server parameter tuning

This section provides information about parameters that you may consider editing to improve portal server performance. The parameters are set in the following files according to operating system:

- Windows

  <ITM HOME>\CNPS\kfwenv

  Example: C:\IBM\ITM\CNPS\kfwenv
- Linux and UNIX
- <ITM HOME>/config/cq.ini

  Example: /opt/IBM/ITM/config

Again, be aware that whenever maintenance or reconfiguration takes place in your environment, these files may be recreated and changes lost and need to be reapplied.

Note: For parameter changes made in the portal server configuration file to take effect, the portal server must be stopped and restarted.

- KFW_CMW_EVENT_SLEEP

  In a complex environment, you may have a number of events occurring simultaneously. This variable specifies a time, in seconds, that the portal server waits between processing similar events. Consider setting this variable to a value of less than 10 seconds if you are experiencing slow performance (such as portal client refresh), as a result.

- KFW_CMW_SITUATION_ADMIN_SUPPRESS

  When a situation is stopped, a message is written to the situation event console for each system for which that situation had been distributed to. You may also see an event pop-up on the portal client workspace.

  The messages are generated because the state of the monitored resource is now unknown and event is alerting the user to this change in state. If you are stopping a large number of situations or if a situation is distributed to a large number of managed systems, there may be an excessive number of messages to deal with. If this behavior is undesirable, you can suppress the messages by setting this parameter to Y.

- KFW_REPORT_TERM_BREAK_POINT

  This variable specifies the point in time, in seconds, offset from the current time where a request for historical data is selected from short-term or long-term history data. The default setting is 86400; that is, short-term history data to be collected from now to 24 hours ago, and long-term from 24 hours onward.

  Increasing this value will result in historical data being retrieved from the short-term data store for time periods further in the past. Decreasing this value will result in historical data being retrieved from the Tivoli Data Warehouse for more recent time periods.

  > **Important:** Special care and consideration should be taken if this value is changed from the default, because this is a global parameter that affects all historical data requests and all monitoring products.
  >
  > Refer to 4.1, “Short-term versus long term historical data” on page 62 before deciding whether to change this parameter.

- KFW_INTERFACE_CNPS_SSL

  Secure Socket Layer (SSL) provides an additional level of data encryption for communication between IBM Tivoli Monitoring components and the portal server. The increased security, however, is at the expense of additional performance overhead. By default, SSL is disabled. To enable SSL, change the value of this parameter from N to Y.

> **Note:** Be careful when editing this variable on UNIX or Linux. The default name of the variable is KFW_INTERFACE_cnps_SSL, and is case-sensitive. The `cnps` part of the variable refers to the interface defined.
>
> Ensure you use the correct name if a different name has been specified.

On Windows, the portal server Interface definitions can be altered additionally via the GUI. From the Manage Tivoli Enterprise Monitoring Services, select the TEPS service and choose **Advanced** → **Configure TEPS Interfaces...**

Event information is stored in the KFW tables in the portal server database. This table will grow over time as more events are opened during the lifetime of the IBM Tivoli Monitoring environment, therefore it is automatically pruned so that older events are discarded under the presumption they are no longer relevant.

The following parameters can be modified to manage event pruning:

- KFW_EVENT_RETENTION

  The number of days to keep a closed event. The default value is 1 day. For example, to prune an event 2 days after it is closed, specify 2.

- KFW_PRUNE_START

  The time of day to start pruning data, specified in 24-hour notation (hh:mm). The default value is 00:00 (midnight). For example, to begin pruning data at 11:00 PM, specify 23:00.

- KFW_PRUNE_END

  The time of day to stop pruning data, specified in 24-hour notation (hh:mm). The default value is 04:00 (4 AM). For example, to end pruning data at 1:00 AM, specify 01:00.

You can control the size of file attachments for events either at the individual client level or at the monitoring environment level, by changing environment variables in the portal server environment file. Consider changing these if you want to restrict the number of large attachments to be held at the portal server, as explained here:

- KFW_ATTACHMENT_MAX

  Use this parameter to control the maximum size of all the files attached to an acknowledgement. Consider editing this parameter if the size of event attachments are too large and are causing network issues, or alternatively you have excess capacity that may be used and attachments have been discarded. Enter the maximum size in bytes, such as 1000000 for 1 MB. The default value is 10000000 (10 MB).

- KFW_ATTACHMENT_SEGMENT_MAX

  This parameter enables you to set a size limitation for individual files attached to an acknowledgement. Enter the maximum size in bytes, such as 250000 for 250 KB. The default value is 1000000 (1 MB).

> **Note:** Settings made at the client level take precedence over those at the monitoring environment level defined here. Note the following points:
>
> - If you have specified a setting at the client level, then any event variables defined within the portal server configuration are ignored.
> - If you have not specified a setting at the client level, the portal server variables are used.
> - If you have not specified any settings, the default values are used.
>
> See 5.4.6, “Portal client parameter tuning” on page 150 for details about parameters defined for the portal client.

5.3.6 Portal server database memory model on AIX

In large environments, two symptoms might be observed when a portal server on AIX is started:

- The system stops while initializing KfwServices. This happens when the KFW_STARTJVM environment variable setting in the cq.ini file is N. Other similar symptoms are possible.
- The system goes into a loop when initializing KfwServices. This happens when the setting for the KFW_STARTJVM environment variable is Y.

KfwServices on the portal server is linked with the default memory model. The default data and stack size of 256 MB in the default memory model causes this problem.

In smaller environments, this problem might not occur at startup. But at some later point, as more virtual storage is required, the same situation can be observed.

To determine whether your portal server is likely to encounter this problem, enter **`topas`** from the command line on the portal server AIX system where the portal server is running. If the output of this command shows that KfwServices has a `PgSp` value of 180-250 MB, then you should take steps to prevent this failure. In smaller environments, even if the value for this parameter is near 180, this is an indicator that the problem might occur when the system processes large queries.

Apply this workaround to systems that use the DB2 small memory model to prevent these types of failures. This workaround requires that you modify the KfwServices load header, the portal server configuration, and the DB2 configuration.

If the changes are not made in *both* applications at the same time, the portal server log will show DB2 SQL errors of `SQLSTATE=55032`.

> **Note:** The directory names in the instructions that follow are typical, but you should use the directory locations appropriate to your system.

1. Make these changes to the portal server configuration files.
   a. Stop the portal server by using these commands:
   ```
   cd /opt/IBM/ITM/bin
   ./itmcmd agent stop cq
   ```
   b. Issue the following commands to reset the `maxdata` value:
   ```
   cp KfwServices KfwServices.orig /usr/ccs/bin/ldedit
   -bmaxdata:0x80000000 KfwServices
   ```
   c. To verify that the `maxdata` value has been reset, issue the following command:
   ```
   dump -ov KfwServices
   ```
   d. This command causes the `maxdata` value in KfwServices to be displayed, as shown in Example 5-1:

Example 5-1 Maxdata output for portal server process

```
maxSTACK maxDATA Nbss magic modtype
0x00000000 0x80000000 0x0003 0x010b 1L
```

   e. Change directories as indicated:
   ```
   cd /opt/IBM/ITM/config
   ```
   f. Use any AIX text editor to add the following line at the end of the `cq.ini` file:
   ```
   EXTSHM=ON
   ```
   g. Save the edited `cq.ini` file.
2. Make these changes to the DB2 configuration files from the DB2 installation user ID (the default is db2inst1):
   a. Stop the DB2 server, if not already stopped, by using these commands:
   ```
   cd /db2inst1/sqllib/adm
   ```

```
db2stop
```

b. Issue the following commands:

```
export EXTSHM=ON
db2set DB2ENVLIST=EXTSHM
db2set -all
```

c. Use any AIX text editor to add the following lines at the end of the file `/db2inst1/sqllib/userprofile`:

```
EXTSHM=ON
export EXTSHM
```

d. Save the edit userprofile file.

3. Restart DB2 by using these commands:

```
cd /db2inst1/sqllib/adm
db2start
```

4. Restart the portal server by using these commands:

```
cd /opt/IBM/ITM/bin
./itmcmd agent start cq
```

> **Note:** For information about this workaround, see Technote 1258694 at:
>
> http://www-1.ibm.com/support/docview.wss?uid=swg21258694

5.4 Tivoli Enterprise Portal client performance

The Tivoli Enterprise Portal (portal) client, as the primary interface for a user to interact with the IBM Tivoli Monitoring 6.1 environment, plays a key part in ensuring that data is requested and displayed to the user. As a Java-based GUI, it is responsible for issuing requests to the Tivoli Enterprise Portal Server and rendering the data retrieved as part of the query.

Depending on the choice of installation, the portal client can be started as a desktop application or as an applet embedded in a Web browser. This section discusses in detail the following considerations for optimizing portal client performance:

- Portal browser client versus portal desktop client
- Tuning the portal client JVM™
- Structuring the enterprise
- Creating workspace links

- Workspace refresh interval
- Portal client parameter tuning

5.4.1 Portal browser client versus portal desktop client

The portal desktop client traditionally has a faster response time than the browser client. The difference at times, depending on the topology of the IBM Tivoli Monitoring environment and network considerations, can be noticeable up to a few seconds. Code changes in successive releases of IBM Tivoli Monitoring have reduced this discrepancy; however, the portal desktop client remains generally faster.

Many customers, however, prefer the portal browser client because of the ease of installation and maintenance over the portal desktop client. Table 5-2 lists the various advantages and disadvantages of each client type. Select the most appropriate client for your environment.

Table 5-2 Advantages and disadvantages of each portal client

Portal client	Advantages	Disadvantages
Browser client	► Does not need to be installed on client machine. ► No need to for maintenance to be applied for each client user. ► Workspaces can be referenced using URLs.	► Slow initial download. ► Requires tuning of Java settings. ► Only available on Windows using IE browser.
Desktop client	► Faster startup and performance over browser client. ► Supported on Linux and Windows.	► Needs to be installed on each client machine. ► Requires maintenance to be installed individually on each client machine. ► Mismatch of versions between portal server and the client is not permitted.

Portal client	Advantages	Disadvantages
Java Web Start client	► Similar performance to desktop client. ► Faster download startup time than Browser client. ► Supports multiple JREs. ► Centralized administration for maintenance.	

IBM Tivoli Monitoring V6.1 Fix Pack 4 introduced Java Web Start capability for administering the desktop client. This approach combines many of the advantages of both the browser and desktop clients (for example, the better performance of the desktop client with the ease of administration and maintenance of the browser). Java Web Start will allow the portal desktop client to be deployed over the network, ensuring the most current version is used.

Tip: For more information about downloading and deploying portal desktop client with Java Web Start, navigate to the IBM Tivoli OPAL (Open Process Automation Library) Web site. Then search under “Java Web Start Enablement Kit for the Tivoli Enterprise Portal Desktop Client”.

Note that this applies to users of IBM Tivoli Monitoring V6.1 only. Web Start enablement is provided as part of IBM Tivoli Monitoring V6.2.

5.4.2 Tuning the portal client JVM

The behavior of a portal client connecting to a large scale environment can exhibit unsatisfactory performance, especially if the utilized Java Virtual Machine (JVM) is not correctly tuned. When refreshing a workspace, client processing is usually the biggest contributor to the response time. Response time can also increase if the portal client has to wait long periods for data from the portal server. Most of the workspaces generate manageable data requests, even in large scale environments. Some however, request large amounts of data from the portal server and take extended periods of time to process and present the data.

The memory usage of the portal client JVM increases as the size of the monitoring environment increases. If the maximum Java heap size setting is too low, the JVM will spend a significant amount of time performing garbage collection. This can result in poor portal client response time and high CPU utilization. The IBM Tivoli Monitoring documentation suggests increasing the maximum Java heap size (using the –Xmx parameter) if the JVM uses more than 5% of its CPU time performing garbage collection.

Another performance parameter important to the JVM memory heap is the initial Java heap size, set by specifying the –Xms parameter. The heap is allocated the amount of memory specified by the –Xms parameter when the JVM is created. The default initial heap size is one megabyte. A larger initial heap size will reduce the time it takes for the portal client to initially start, and also reduce the amount of times the heap will need to be extended as more and more memory is allocated to support portal client processing.

We recommend that if using the browser version of the portal client, that you increase the default Java plug-in parameters for heap management.

The third important setting that affects performance is the size of the JRE™ cache. By default, only 50 MB is allocated for the cache, which is insufficient for working with the portal browser client. The effect of not increasing the cache will mean slower performance, particularly at startup if the client applet has to be downloaded again because there was not enough space to store it from the previous download.

Because the portal client applet is large, you may find other applications affected because the cache is always full. We recommend increasing the cache size to an unlimited size, if possible. If that is not possible, then increase to at least 100 MB.

Note: The level of Java used will be dependent on the release of IBM Tivoli Monitoring you are using:

- IBM Tivoli Monitoring 6.1 requires IBM Java 1.4.2
- IBM Tivoli Monitoring 6.2 requires IBM Java 5.0

Tuning IBM Java 1.4.2 settings

If using IBM Tivoli Monitoring V6.1, you can change the Java settings by following these steps:

1. Open the Windows Control Panel.
2. Double-click the **Java Plug-in** icon.
3. Click the **Advanced** tab.
4. Select the appropriate level of Java used by IBM Tivoli Monitoring from the Java Runtime Environment list.
5. In the **Java Runtime Parameters** field, as shown in Figure 5-4 on page 145, enter: `–Xms64m -Xmx256m`

6. Click the **Cache** tab.
7. Ensure the **Unlimited** radio button is selected for size in **Cache Settings.**
8. Click **Apply**.

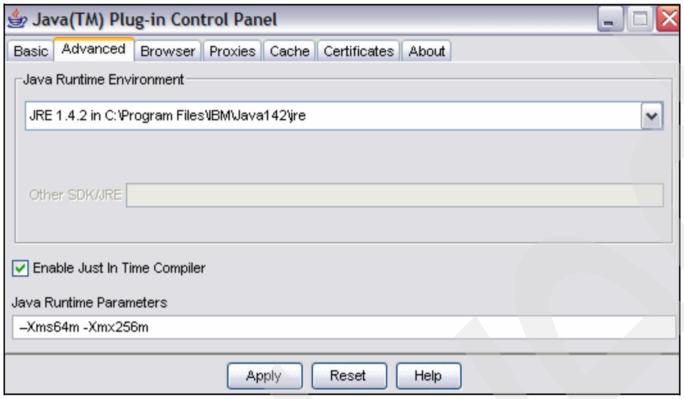


Figure 5-4 Java 1.4.2 plug-in settings

Tuning IBM Java 5.0 settings

If using IBM Tivoli Monitoring V6.2, you can change the Java settings by following these steps:

1. Open the Windows Control Panel.
2. Double-click the **Java(TM) Control Panel** icon.
3. Click the **Java(TM)** tab.
4. Click the **View...** button for **Java(TM) Applet Runtime Settings**.
5. Select the row containing the appropriate level of Java used by IBM Tivoli Monitoring from the Java Runtime Environment list.
6. In the **Java(TM) Runtime Parameters** field, as shown in Figure 5-5 on page 146, enter: `–Xms64m -Xmx256m` and then click **OK**.
7. Click the **General** tab.
8. Click the **Settings...** button for **Temporary Internet Files**.
9. Ensure the **Unlimited** radio button is selected for **Amount of disk space to use** and click **OK**.
10. Click **OK** to close the control panel and apply the settings.

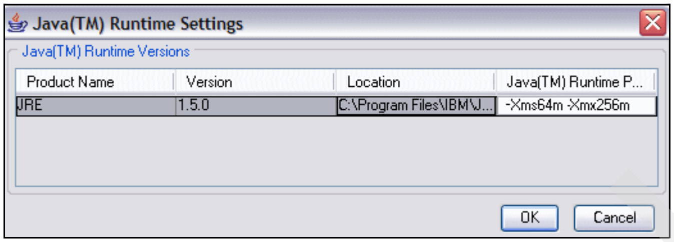


Figure 5-5 Java 5.0 plug-in settings

Tuning Java Web Start client settings

If using the Java Web Start client, you can modify the initial heap size and maximum heap size settings by modifying the tep.jnlp file which launches the client. Modify the j2se tag by changing the initial-heap-size and max-heap-size elements as shown in Example 5-2.

Example 5-2 Content of tep.jnlp

```
<!-- JNLP File for Tivoli Enterprise Portal V6.1.0 -->
<?xml version="1.0" encoding="utf-8"?>
<jnlp
  spec="1.0+"
  codebase="http://localhost:1920///cnp/kdh/lib/"
  href="tep.jnlp">
  ...
<resources>
    <j2se version="1.4+" initial-heap-size="64m" max-heap-size="128m"
java-vm-args="-showversion -noverify" />
    <property name="kjr.trace.mode" value="LOCAL"/>
    <property name="kjr.trace.params" value="ERROR"/>
    <property name="ORBtcpNoDelay" value="true"/>
    <property name="ibm.stream.nio" value="true"/>
```

If response time remains slow and displays high CPU activity for an extended period, the maximum heap size may still be insufficient. Consider increasing the maximum heap size of the JVM to 512 megabytes.

The desktop version of the portal client starts with initial heap parameter values suitable for environments of moderate size. The default values used are –Xms64m (initial heap size set to 64 megabytes) and –Xmx256m (maximum heap size set to 256 megabytes). If you want to increase the initial heap size and maximum heap size for the desktop portal client, modify the appropriate batch or shell script used to start the portal client used. Find the values -Xmx and -Xms within the file and edit the value as required.

When modifying the maximum Java heap size, ensure there is enough physical memory available for the heap to remain resident in memory. If there is not enough physical memory available, the JVM performance will be affected by excessive paging activity.

5.4.3 Structuring the enterprise

In a large enterprise there are personnel, in various roles, who need to look at your systems in various ways. IT managers often need to see a high-level overview of how their department and the systems they are responsible for are performing. There may be an interest in how different geographies are doing. Another common requirement is to know how business-critical applications performing and how they are impacted by components that are unavailable or performing badly. Those different requirements can be met by using *navigator views* that allow you to structure your enterprise in ways that are useful for your users.

Using IBM Tivoli Monitoring V6.1, you have different ways to view data from managed systems.

For example, you can use the *physical* view, which is the system view of your environment hierarchically structured by operating system or application. This is a product-provided view. The physical view, however, is not recommended for use as the default view when monitoring large scale environments. Large amounts of data will be requested by the portal client when viewing the enterprise though this view, and for many users much of the data displayed will be extraneous to their requirements.

As an alternative, we recommend creating your own views that will meet the requirements of the company. This is possible using the Navigator Editor function. Examples of possible ways you may decide to structure your enterprise business views include the following:

- Geographic views
- Systems views
- Application views
- Operational views
- Combinations

Figure 5-6 on page 148 shows the navigator editor window. You can drag and drop navigator items between the trees.

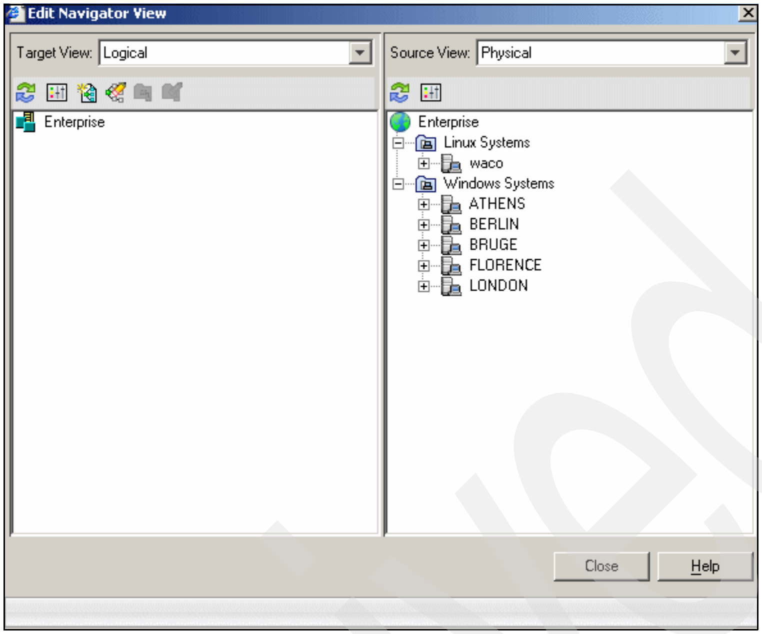


Figure 5-6 Navigator editor

Note: All users are assigned access to the Physical and Logical views available at install time. If new views are made available, then users will need to be assigned access in order to use it. You can assign this access by using the Navigator Views tab found within the User Administration window in the portal client.

5.4.4 Creating workspace links

Linking is a powerful way of navigating from one workspace to another and thereby accessing related or more detailed information. IBM Tivoli Monitoring 6.x allows you to quickly access information that may be meaningful as a substitution for the information displayed in the source workspace.

You can create links across different applications or attribute groups. A useful addition to simple links between workspaces is to pass values to the new workspace in order to build it based on either filtered or modified values. Links can be executed from a right-click menu on a navigator item, an object in a graphic view, or a table row.

You can also define default links that can be executed with a single left mouse click on a target. For table rows, a link indicator can be included to indicate the presence of a Link. The Link Wizard is shown in Figure 5-7.

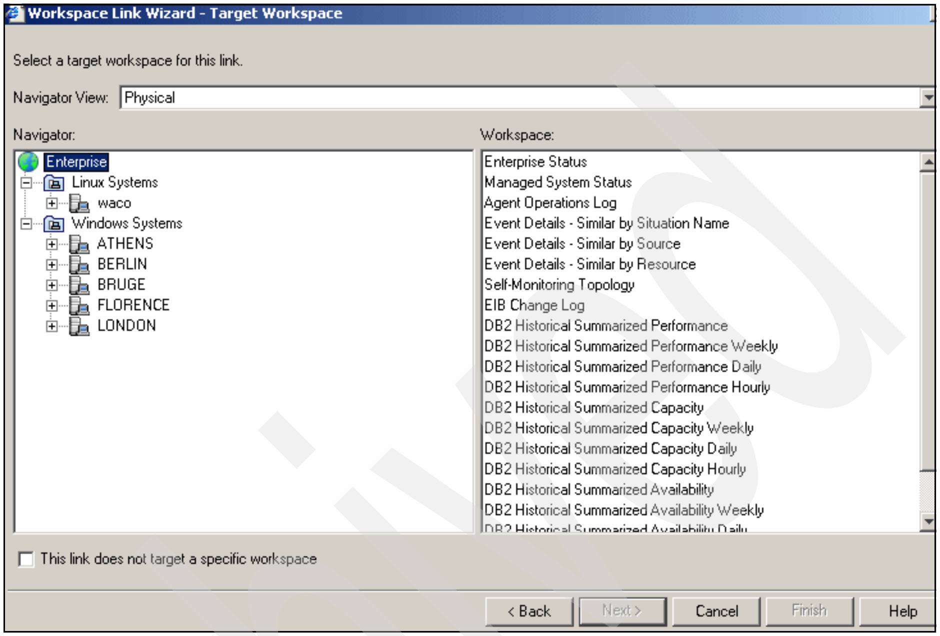


Figure 5-7 Workspace link wizard

5.4.5 Workspace refresh interval

When a workspace is opened that contains charts or tables, a request is made via the Tivoli Enterprise Portal Server to the appropriate monitoring agent for new data samples. The returned data is processed and displayed on screen. The portal client can be configured to automatically refresh information for its active views with the following intervals: 30 seconds, 60 seconds, 5 minutes, 15 minutes, or 60 minutes. It can also be refreshed on demand.

Workspace refreshes drive workloads across the IBM Tivoli Monitoring infrastructure including the Tivoli Enterprise Portal Server and Tivoli Enterprise Monitoring Server. The overhead of the workload depends both on the refresh interval and the complexity of the workspaces being updated. We recommend that you evaluate how often a workspace needs to be refreshed and tune the settings within the portal client. Longer refresh intervals will reduce the overhead for data traffic across the IBM Tivoli Monitoring environment. Refresh intervals of less that five minutes should be avoided and where possible, using refresh on demand is a recommended good practice.

Note: Regardless of the workspace interval refresh chosen, the navigator view and event views such as the message log and event console will always refresh automatically. This provides you with instantaneous alerts from events and situations from which you can then choose to navigate to workspaces with actual data.

5.4.6 Portal client parameter tuning

This section provides information about parameters you may consider editing to improve portal client performance and usability. The parameters are set in the following files according to operating system:

- Windows

  <ITM HOME>\CNP\cnp.bat (or cnp_<instance name>.bat)

  Example: C:\IBM\ITM\CNP\cnp.bat

- Linux

  <ITM HOME>/<OS Specific Directory>/cj/bin/cnp.sh (or cnp_<instance name>.sh)

  Example: /opt/IBM/ITM/li6243/cj/bin/cnp.sh

Be aware that whenever maintenance or reconfiguration takes place in your environment, these files may be recreated and changes lost and need to be reapplied.

- cnp.databus.pageSize

  Number of rows to fetch in single logical page for any workspace table. By default, 100 rows are returned. Although there is no limit to what you can set here, the larger the page size, the more memory is required at the client and portal server.

  Consider editing this value if you have a reasonable idea of the number of rows are to be returned by query such that a manageable number of rows are displayed.

- cnp.attachment.total.maxsize

  Use this parameter to control the maximum size of all the files attached to an acknowledgement. Consider editing this parameter if the size of event attachments are too large and are causing network issues, or alternatively you have excess capacity that may be used and attachments have been discarded.

  Enter the maximum size in bytes, such as 1000000 for 1 MB. The default value is 10 MB.

- cnp.attachment.segment.maxsize

  This parameter enables you to set a size limitation for individual files attached to an acknowledgement. Enter the maximum size in bytes, such as 250000 for 250 KB. The default value is 1 MB.

> **Note:** Settings made at the client level take precedence over those at the monitoring environment level. Note the following points:
>
> - If you have specified a setting at the client level, then any event variables defined within the portal server configuration are ignored.
> - If you have not specified a setting at the client level, the portal server variables are used.
> - If you have not specified any settings, the default values are used.
>
> Refer to 5.3.5, "Portal server parameter tuning" on page 136 for details about parameters defined for the portal server.

5.5 Tivoli Data Warehouse performance

The Tivoli Data Warehouse is the most intensely utilized component of the IBM Tivoli Monitoring 6.1 infrastructure. The warehouse must support huge amounts of data transactions during every warehousing period, daily summarization and pruning of the warehoused data, and multiple queries that often returning result sets that are thousands of rows long.

The processing power required to support the warehouse database and the Summarization and Pruning agent can be significant. This section discusses processes for optimizing the performance of the warehousing process, including the Warehouse Proxy and the Summarization and Pruning agents. The following topics are covered:

- Tuning the Warehouse Proxy agent
- Tuning the Summarization and Pruning agent
- Using data marts
- Database tuning

5.5.1 Tuning the Warehouse Proxy agent

The Warehouse Proxy agent provides a means of translating and transferring historical data collected by other monitoring agents to the Tivoli Data

Warehouse. The amount of historical data generated can be huge, particularly in environments with tens of thousands of monitoring agents or when instance intense attribute groups are enabled. Properly configuring Warehouse Proxy agents can ensure that all historical data is smoothly transferred to the warehouse. An improper configuration can cause poor performance of the proxy and historical data build up on the storage locations. The parameters are set in the following files, according to operating system:

- Windows

  <ITM HOME>\TMAITM6\khdenv

  Example: C:\IBM\ITM\TMAITM6\khdenv

- Linux and UNIX

  <ITM HOME>/config/hd.ini

  Example: /opt/IBM/ITM/config/hd.ini

Be aware that whenever maintenance or reconfiguration takes place in your environment, these files may be recreated and changes lost and need to be reapplied.

Warehouse Proxy internals

A basic understanding of how the Warehouse Proxy agent collects and transfers data to the warehouse is required before either can be properly configured. The internal structure of the proxy is comprised of three distinct components; the IRA communication framework, the work queue and the exporter threads. These three components work together to collect, translate and transmit historical data to the warehouse. Historical data flows from one component to the next, undergoing specific processing at each step before being passed on.

Figure 5-8 on page 153 shows how data is transferred from the agents, through the proxy components, and then into the warehouse. It also lists the primary factors impacting the performance and scalability of that component during normal operation. You can also refer to "Warehouse Proxy Agent component flow" on page 485 in Appendix B for a more information.

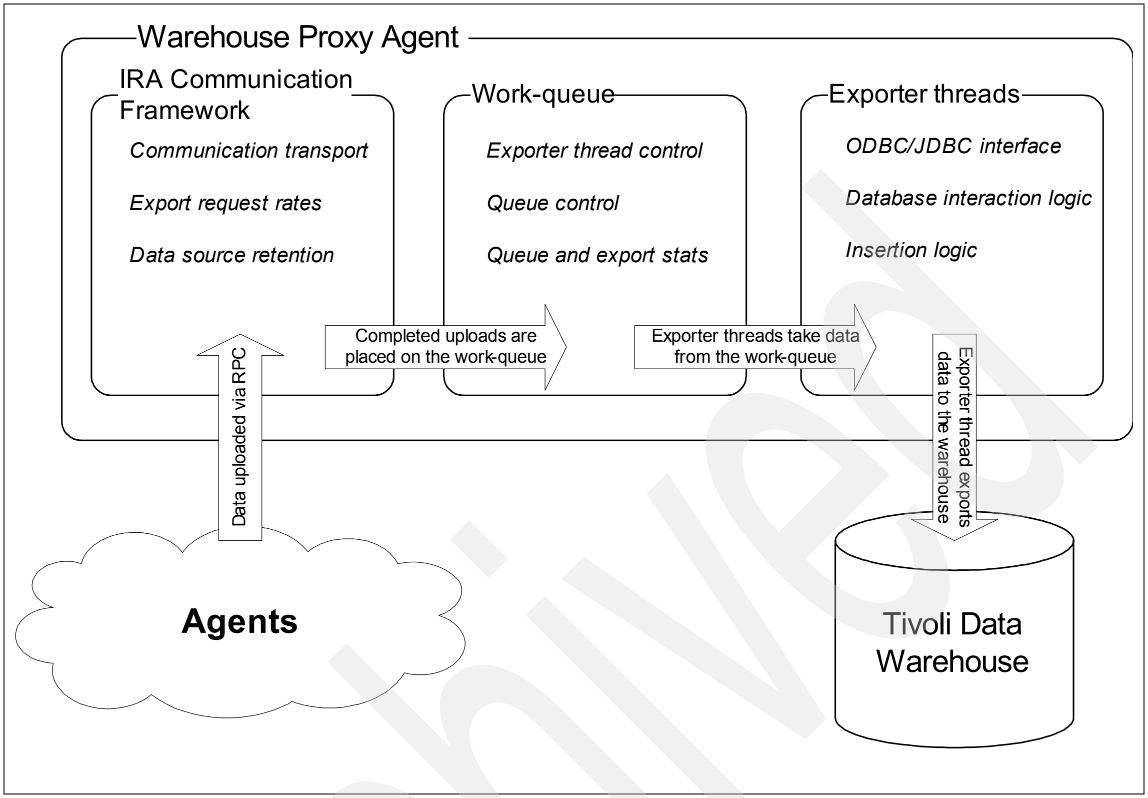


Figure 5-8 How data is transferred from the agents

1. The data for each attribute group is initially stored in a binary file at the monitoring agent or at the monitoring server, depending on the configuration.

> **Note:** In Figure 5-8 it is assumed that the binary files are stored on agents, which is the preferred way in most cases.
>
> If the data is stored on the monitoring server, then the binary files will be much bigger, because each file (for one attributed group) will have the data from multiple agents.

2. If configured to do so, the data from each attribute group is uploaded from multiple agents directly to the Warehouse Proxy agent via RPC at an internal of either 1 to 24 hours.

   NCS listen threads are responsible for establishing the RPC communication. The number of these threads is determined by the CTIRA_NCSLISTEN variable. The maximum amount data transmitted per RPC request is 1000 rows. If the data for one attribute group to transmit is larger than 1000 rows, there will be n more than 1 RPC request for that same attribute.

3. Completed uploads are put on the work queue. The queue length is determined by the f KHD_QUEUE_LENGTH variable.
4. Export threads perform the following steps:

> **Note:** The number of export threads is defined by the KHD_EXPORT_THREADS variable.
>
> KHD_EXPORT_THREADS setting should be *equal* to the KHD_CNX_POOL_SIZE number that sets the number of database connection pools, because each worker thread requests an ODBC/JDBC connection from the connection pool.
>
> The default number of both KHD_EXPORT_THREADS and KHD_CNX_POOL_SIZE is 10.

   a. Get the data from the queue.
   b. Create the necessary SQL steps to insert the data into the database via JDBC or ODBC bridge.
   c. Start the transaction. Insert the data into the database.
   d. Update the Warehouse Proxy log.
   e. Commit the transaction.
   f. Inform the agent.

> **Note:** There are two parameters that affect the timeout of the operation:
>
> - KHD_STATUSTIMEOUT (Client timeout)
>
>   An export request will be re-sent if a status has not been received from the Warehouse Proxy before this timeout. The default is KHD_STATUSTIMEOUT = 900secs (15 min). It is set on the monitoring agent.
>
> - KHD_SRV_STATUSTIMEOUT (Server timeout)
>
>   This is set on the Warehouse Proxy agent. The default is KHD_SRV_STATUSTIMEOUT = 600secs (10 min). Note that KHD_SRV_STATUSTIMEOUT should always be less than KHD_STATUSTIMEOUT by at least 60secs (1min).

The number of active exporting threads should not exceed the KHD_CNX_POOL_SIZE; otherwise, those threads will be idle.

5. The agent gets the message and updates its khdexp.cfg file. In this binary file there is one record for each attribute group, and that shows the time that the latest warehouse upload was successful for this attribute group.
6. All the data in the binary file that has been successfully warehoused and older than 24 hours (by default) is deleted.

Protocol selection

Just like any other monitoring agent within IBM Tivoli Monitoring, the Warehouse Proxy agent communicates with other components (in this case, the hub monitoring server) via remote procedure calls (RPC). However, other agents do not need to handle the same volume of RPC calls as the Warehouse Proxy agent. For this reason, tuning the communication variables is very important.

The most reliable and usually the fastest protocol for historical data throughput is ip.pipe, because each client maintains a connection with the Warehouse Proxy agent. The drawback with ip.pipe are the additional system resources needed to maintain the separate socket that each connected client requires. Even so, unreliable protocols such as IP.UDP are more likely to result in broken exports and the need for the client to retransmit data.

For this reason, we recommend that you only choose IP.UDP as the communication protocol between agents and the Warehouse Proxy if you have a reliable network and the resources required to support ip.pipe are not available.

The IP.SPIPE protocol behaves similarly to IP.PIPE, but imposes encryption and decryption on the transferred data. This will have an associated impact on the speed of data transfer, and you should decide whether or not this is an important factor within your environment when choosing a communication protocol.

NCS listen thread management

Performance and reliability can also be affected by the number of threads allocated to process incoming RPCs. These are known as Network Computing System (NCS) listen threads.

> **Note:** Remote procedure call (RPC) is a protocol that allows a computer program running on one host to cause code to be executed on another host.

By default, 10 NCS listen threads are created for all IBM Tivoli Monitoring agents, but this may be insufficient for the Warehouse Proxy agent. Ideally, the number of threads would depend on the number of processors on the proxy's host. However, in practice it depends more on the number of created export threads; that is, the number of worker threads allocated for the purpose of exporting the data from the proxy to the data warehouse.

To set this option, edit the Warehouse Proxy configuration file and modify the variable CTIRA_NCSLISTEN to the value required. Enabling 10 NCS listen thread per single export thread is a good starting place. Increase the ratio if the proxy's insertion rate is adequate but there are numerous RPC errors found in the proxy's log file.

Work queues

Each Warehouse Proxy agent contains a data structure for holding export requests from clients called the work queue. Exports are processed in order by the exporter threads. The exporter threads become dormant if no more work is available on the queue. When the queue reaches its maximum size, the proxy will no longer accept export requests from the clients.

The size of the work queue can be set using the KHD_QUEUE_LENGTH variable in the Warehouse Proxy configuration file. By default, the work queue has a size of 1000. The queue size should be set equal to the number of clients that regularly upload data to a particular proxy. This will ensure that each client is able to place at least one export buffer on the queue during a collection interval, assuming no export errors occur.

If the value of KHD_QUEUE_LENGTH is less than the number of clients exporting data to the proxy, perhaps as a result of more monitored systems being added to the environment, there is an increased chance of rejected requests due to the queue reaching its maximum size. If the Warehouse Proxy log files show a significant number of rejected requests, then consider increasing the value of KHD_QUEUE_LENGTH.

> **Note:** It is possible to set the value of KHD_QUEUE_LENGTH to zero (0), meaning there is no enforced queue limit. However, this should be employed with caution because as the majority of the memory used by the Warehouse Proxy agent is allocated for storing the historical data in the export buffer, modifying this value will change the maximum amount of memory the Warehouse Proxy will require. A work queue with no limit could therefore consume a large amount of resources if not closely monitored.

Exporter threads and database connection pool size

The exporter threads remove export buffers from the work queue, prepare SQL statements appropriate for the data, and insert data into the warehouse using a JDBC or ODBC bridge supported by the warehouse database. After the export has finished, the export thread notifies the client of the export status.

Selecting the proper number of exporter threads, the number of database connections, and the insertion logic will impact the rate at which data is inserted into the warehouse.

The number of exporter threads can be configured using the variable KHD_EXPORT_THREADS within the Warehouse Proxy configuration file. The default number is 10 and should serve for most configurations. The system resources required by the 10 export threads and the associated database connections are minimal, and are well worth the extra performance that the concurrent exports provide.

If the value of KHD_EXPORT_THREADS is changed, be sure to also change the KHD_CNX_POOL_SIZE to the *same* value. The excess connections will not be used, and they will consume system resources unnecessarily.

> **Tip:** As a best practice, the KHD_EXPORT_THREADS setting should always be equal to the KHD_CNX_POOL_SIZE number. KHD_CNX_POOL_SIZE defines the number of database connections.

If you have multiple warehouse proxy agents installed, each will have its own pool of export threads for sending data to the warehouse database. This number will increase, which could cause performance bottleneck problems at the warehouse when significant amounts of data is being uploaded. If this scenario exists, consider reducing the number of exporter threads at each proxy to avoid overloading the Tivoli Data Warehouse.

Batch inserts

Use this option to improve the performance when the proxy and the warehouse are located on different hosts. The Warehouse Proxy agent can then submit multiple execute statements to the Tivoli Data Warehouse database as a batch (in batches of 1000 rows). Using batch inserts is recommended in all configurations, but they will place increased load on the data warehouse. By default, this property is not enabled.

- Windows

  To set this option, edit the KHDENV file and add the variable KHD_BATCH_USE=Y.

- Linux and UNIX
  - Batch processing is one of the features provided with the Java Database Connectivity 2.0 (JDBC 2.0) application programming interface (API). The JDBC driver that is selected must support the batch feature to use this option.
  - To set this option, select the **Use Batch** check box on the configuration panel. This updates the KHD_BATCH_USE variable to Y in the hd.ini file.

Tuning the JVM

Warehouse Proxy agents using the JDBC interface to communicate with the warehouse database start a JVM to support JNI™ calls to JDBC. If the heap size of the JVM is set to a low value, performance can be degraded by frequent garbage collections.

Setting KHD_JNI_VERBOSE=Y in the configuration file will enable logging of the garbage collector's actions. If the Java log contains an excessive number of garbage collection entries during a single warehousing interval, typically more than one collection per minute, then consider increasing the size of the Java heap. To set this option, edit the configuration file and modify the variable:

```
KHD_JAVA_ARGS=–Xms512m –Xmx768m
```

Using multiple Warehouse Proxy agents

A strategy particularly suitable to large scale environments is to deploy multiple Warehouse Proxy agents. Each proxy will be configured to export data from a subset of the remote monitoring server instances found in the environment.

To specify a subset, edit each Warehouse Proxy configuration file and modify the variable KHD_WAREHOUSE_TEMS_LIST. This is a space-delimited list of monitoring server instances. Any monitoring server listed in this variable may select the configured proxy to be its monitoring agent's export server. For example, we configured one of our Proxy agents with the setting:

```
KHD_WAREHOUSE_TEMS_LIST=remote_melbourne remote_edinburgh
```

Upon startup, a proxy will attempt to register itself with the hub monitoring server. It will make this attempt every five minutes or until it succeeds, after which it will re-register its information with the hub monitoring server every hour. Every remote monitoring server will then query the hub to determine which Warehouse Proxy agent it is associated with.

By default, the remote monitoring server will recheck this every 60 minutes. This interval can be modified using the KPX_WAREHOUSE_REGCHK environment variable in the monitoring server configuration file. This value specifies the number of minutes to wait between rechecking.

To verify that the proxy is registering with the hub monitoring server, add the (UNIT: khdxrpcr STATE) trace entry to the proxy's environment file. This setting will print the value of KHD_WAREHOUSE_TEMS_LIST and show any errors associated with proxy registration.

To determine which Warehouse Proxy a particular monitoring server selects for its agents, add the (UNIT: kpxrwhpx STATE) trace entry to the monitoring server environment. This setting will log entries in the monitoring server's RAS log whenever a registration change occurs, displaying the address of the new proxy.

For more information about multiple Warehouse Proxy agents, refer to 3.2.3, "Firewall Gateway Proxy" on page 45.

Starting Warehouse proxy agents

There are two parameters that affect the startup of the Warehouse proxy agents. These are KHD_CNX_WAIT_ENABLE and KHD_CNX_WAIT.

The KHD_CNX_WAIT_ENABLE parameter enables the Warehouse Proxy to wait in between attempts to connect to the database. By default, this variable is set to Y. If you do not want the Warehouse Proxy to wait, then change the variable to N. However, this can generate a large log file if the connection to the database fails with each attempt.

The KHD_CNX_WAIT parameter defines the amount of time, in minutes, that the Warehouse Proxy waits between attempts to connect to the database. The default value is 10 minutes, which is a reasonable setting for most environments.

5.5.2 Tuning the Summarization and Pruning agent

Some of the DB2-related performance information in this section is based on a white paper entitled "Relational Database Design and Performance Tuning for DB2 Database Servers" written by Edward Bernal, IBM USA. You can download this paper from the following site:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/fe582a1e48331b55852 56de50062ae1c/546c74feec117c118625718400173a3e?OpenDocument

The Summarization and Pruning (S&P) agent is a multi-threaded, Java-based application. It interacts with the warehouse using a JDBC driver appropriate to the warehouse's database. The number of worker threads available and the heap size of the JVM affect the performance of the Summarization and Pruning agent and the length of time of its processing runs.

The installation location of the Summarization and Pruning agent is another important aspect of Summarization and Pruning agent performance tuning. On Windows, the Summarization and Pruning agent configuration file is called KSYENV. On Linux and UNIX, the configuration file is called sy.ini.

The parameters are set in the following files, according to operating system:

- Windows

  <ITM HOME>\TMAITM6\KSYENV

Example: C:\IBM\ITM\TMAITM6\KSYENV

- Linux and UNIX

  <ITM HOME>/config/sy.ini

  Example: /opt/IBM/ITM/config/sy.ini

Be aware that whenever maintenance or reconfiguration takes place in your environment, these files may be recreated and changes lost and need to be reapplied.

Summarization and Pruning agent internals

A basic understanding of how the Summarization and Pruning agent works is required before configuring the agent for performance optimization.

Figure 5-9 shows the Summarization and Pruning agent workflow. This assumes that Summarization and Pruning agent and the Tivoli Data Warehouse are installed on separate systems.

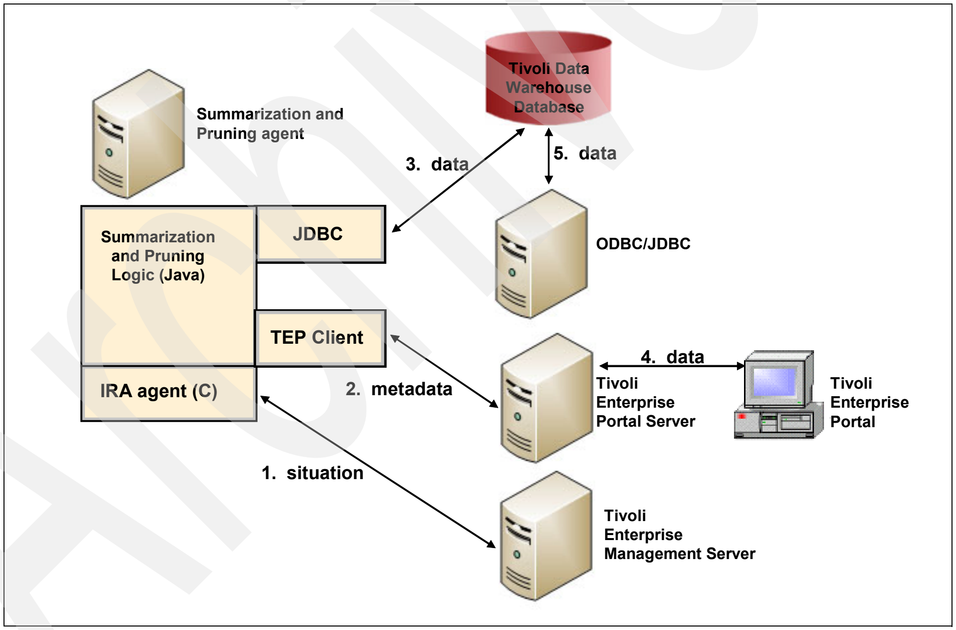


Figure 5-9 Summarization and Pruning agent workflow

1. A situation runs every 5 minutes and instructs Summarization and Pruning agent to check whether it is time to run according to the scheduled time.
2. The Summarization and Pruning agent Java component uses the portal client interface to get metadata in about attribute groups, attributes from the portal

server ODI files. It also fetches the current Summarization and Pruning agent configuration settings from the portal server.

3. The Summarization and Pruning agent communicates with the Tivoli Data Warehouse to create summarization tables, get short-term historical data, summarize it, and stores it back into the Tivoli Data Warehouse.

   Pruning is performed by using SQL DELETE statements. Summarization and pruning tasks are performed by worker threads, each of which works on one attribute group at a time.

4. When the user chooses to view the summarized data from the portal client, the portal server provides the data to portal client to display after the summarization is complete and data is gathered at the portal server.

5. The portal server communicates with the Tivoli Data Warehouse via ODBC/JDBC to provide the requested data. If the user uses the time span dialog to indicate they want to see the summarized data, then portal server automatically builds the SQL statement that needs to be executed against the Tivoli Data Warehouse.

Installation optimization

The Summarization and Pruning agent should be installed on the *same* host as the warehouse in order to minimize network communication. Multiple Summarization and Pruning agent agents are not supported, so distributing the processing loads across multiple hosts is not possible.

Installing the Summarization and Pruning agent with the warehouse eliminates wait times induced when performing SQL calls across a network. The benefit to installing the Summarization and Pruning agent on a separate machine is that additional processing resources can be dedicated to the task.

If the Summarization and Pruning agent cannot be installed on the warehouse database server, ensure that there is a high-speed network connection between them (100Mbps or higher). Note that Summarization and Pruning agent does not communicate with the other agents; it only communicates with Tivoli Data Warehouse and the portal server.

Optimizing the number of available worker threads

The Summarization and Pruning agent creates a pool of worker threads during initialization for the purpose of performing the summarization and pruning tasks. Each worker thread operates independently of all others, concentrating on one attribute group and its associated warehouse tables.

After a worker thread finishes an attribute group, it locates the next attribute group scheduled for processing. If there are no more attribute groups to process, the thread ends, leaving the remaining threads to finish their work.

Although this threading model provides some level of concurrency, it will not take advantage of multiple CPUs, if the Summarization and Pruning agent is only working on a single attribute group. This means that extremely long tables will require lengthy processing time, even on systems with multiple CPUs.

To set this option, edit the configuration file and modify the variable KSY_MAX_WORKER_THREADS to the value required. The default is 1.

> **Note:** The default value will be changed to 2 in IBM Tivoli Monitoring V6.2.
>
> Also in IBM Tivoli Monitoring V6.2, KSY_MAX_WORKER_THREADS will be customizable from the Summarization and Pruning agent configuration GUI. In IBM Tivoli Monitoring V6.1, it must be edited in the configuration file.

The suggested number of worker threads is 2 or 4 times the number of processors on the host system. It is a good idea to use the latter number, especially if you install the Summarization and Pruning agent on a dedicated machine away from the Tivoli Data Warehouse it is working with.

In a large scale environment, even for a single processor system, you should set the KSY_MAX_WORKER_THREADS number to 4. Note that configuring more threads than attribute groups will not decrease the processing time, because each thread works on one attribute group at a time.

> **Note:** In IBM Tivoli Monitoring V6.2, there will be several performance and functionality enhancements to the Summarization and Pruning agent that might improve performance. These are:
>
> - New scheduling options are added (run every N minutes with blackout periods).
> - Pruning of the log tables (WAREHOUSELOG and WAREHOUSEAGGREGLOG) is provided.
> - Instead of a single row, a range of rows are processed at a time.
> - Key columns added to indexes to improve performance.
> - VARCHAR is used for any columns greater than 16 characters.

Tuning the JVM

The JVM heap size should be increased as the daily amount of data collected by the Warehouse Proxy Agent increases. We recommend values of at least 512 megabytes for collection rates of 10 gigabytes a day, and 1536 megabytes for collection rates of 25 gigabytes a day.

To set this option, edit the configuration file and modify the variable:

KSZ_JAVA_ARGS=–Xms512m –Xmx768m

Tip: After you enable the historical collection, do not wait long to set up and run the summarization and pruning, because the longer you wait, the larger the Tivoli Data Warehouse tables will be.

If you do not set up and run the summarization and pruning right away after enabling the historical collection, you might end up with several months' worth of data built up in the Tivoli Data Warehouse database. If that happens, then the first time it is run, the Summarization and Pruning process can run for several days.

5.5.3 Using data marts

Data marts are specialized subset versions of a data warehouse. They contain a snapshot of operational data just like a the data warehouse, which helps customers make decisions based on analysis of past trends and experiences.

The key difference between a data warehouse and a data mart is that the creation of a data mart is predicated on a specific, predefined need for a certain grouping and configuration of select data. Data marts can satisfy many objectives, such as departmental or organizational control, easier and quicker design and build, fast payback, and faster query response times.

The data mart therefore typically contains a subset of corporate data that is of value to a specific business unit, department, or set of users. This subset consists of historical, summarized, and possibly detailed data captured from the IBM Tivoli Monitoring environment. It is important to realize that a data mart is defined by the functional scope of its users, and not by the size of the data mart database. Most data marts today involve less than 100 GB of data; some are larger. However, it is expected that as data mart usage increases, data marts will continue to increase in size.

The primary purposes of a data mart can be summarized as follows:

- Provides fast access to information for specific analytical needs
- Controls (or limits) user access to the information
- Represents the user view and data interface to the data warehouse
- Stores pre-aggregated information for faster response times

Reporting against data marts has better performance compared to reporting against the main data warehouse because data marts only have a subset of the data available in the data warehouse. This reduces the level of processing required and because the filtering has already been applied at the mart extract, transform, and load (ETL), the reporting queries become much smaller. Smaller queries that are performing on a small subset of data are, of course, easier to tune.

Consider using data marts if you find you have a large data warehouse and there is a constant need to retrieve summarized and detailed data from the warehouse. Data that is known to be needed can then be made available via a data mart.

Data marts also may be suitable if you have a number of geographically distinct data sites, and each site has known specific sets of data they require. A data mart can then be deployed locally and remain available for use even if the network connection between the main data center (where the main data warehouse resides) and the local data center is down.

For further information about data marts and assistance on how to create them, see Chapter 4, “Tivoli Data Warehouse tuning”, in the IBM Redbooks publication *Tivoli Management Services Warehouse and Reporting*, SG24-7290.

5.5.4 Database tuning

The warehouse database can be implemented using one of the three supported relational databases: DB2, Oracle or Microsoft SQL Server™. It is critical to make sure that the database supporting the warehouse is configured correctly. Changing the database configuration from a reasonable to an ideal configuration might yield a small performance improvement, but a badly configured database can seriously impede the capabilities of the warehouse.

In this section, we cover fine-tuning tips for database configuration. Note that we focus only on the actions (or settings) that are applicable in a Tivoli Data Warehouse environment, because a Tivoli Data Warehouse user does not have control over all the possible performance optimization actions (such as changing the table structure or creating indexes) in this environment.

Also note that the latest versions of commercial RDBMS programs, such as DB2, have some form of self-tuning functionality. This means that the RDBMS observes what is running on itself, and automatically makes internal adjustments which, for the most part, keep the databases running as optimally as possible given the tasks at hand and the given hardware.

> **Important:** Before implementing any of these suggestions in your environment, proper back-out procedures should be followed, such as saving a backup copy of database configuration, database manager configuration, and registry parameters.

Database schema design

For most users, the database design or schema as created by the warehousing tools will not require any additional tuning or modification. In large scale installations and environments where it is known that large amounts of data will be collected and possibly retrieved by client users, it may be desirable to modify the structure of the database to optimize performance.

Under normal operation, the database schema is created when the historical data collection is configured within the portal client. The default tablespace for most attribute groups is 4 KB. There are two possible consequences of this that a database admistrative (DBA) may want to change:

- Not all attribute groups may warehouse the same amount of data due to differences in collection interval or the amount of data (rows) generated at each collection interval.
- The 4 KB tablespace enforces a maximum table size of 64 GB. If pruning is not performed before a table reaches this size, there could be warehouse problems.

If the DBA wants to change the structure of the schema such that more space is allocated to attribute groups that are expected to provide more data to the warehouse, or reduced for those pruned before table limits are reached, they should first extract the schemas as created by the Warehouse Proxy agent or Summarization and Pruning agent, make a copy, and then make the appropriate changes. The tables created with the original schema should then be dropped and recreated with the modified schema.

Many customers use RAID for warehouse storage. A DBA may wish to consider how to structure the data warehouse to optimize both the warehousing performance and data retrieval rate from RAID such that overall performance is not affected by problems within the warehouse should disk failure occur.

In this section we present configuration parameters for DB2 that may be applicable to customers using RAID for their warehouse. When planning the data warehouse structure, these settings can be combined with other techniques, such as multiple warehouse proxy agents (discussed in 5.5.1, “Tuning the Warehouse Proxy agent” on page 151), to ensure that performance is optimized for attributes that require the fastest data retrieval rate.

For further information about IBM Tivoli Data Warehouse internals and discussion about schema design, refer to Chapter 2, “Tivoli Data Warehouse internals and deployment configurations”, in the IBM Redbooks publication *Tivoli Management Services Warehouse and Reporting*, SG24-7290.

DB2 parameters

This section provides information about how to tune a DB2 database in the context of parameters.

- LOGFILSIZ

  This parameter defines the size of each primary and secondary log file. The size of these log files limits the number of log records that can be written to them before they become full and a new log file is required.

  The default value is 1000.

  The default value is quite a small size; therefore, start with a value of 8192, which means 8192 x 4 KB pages of data.

  > **Tip:** You must balance the size of the log files with the number of primary log files; a small number of primary logs associated with a log file size too big will demand a significant amount of I/O when archiving that log.

  The value of the LOGFILSIZ should be increased if the database has a large number of update, delete, or insert transactions running against it, which will cause the log file to become full very quickly. This will generate a lot of error messages indicating log full.

  A log file that is too small can affect system performance due to the overhead of archiving old log files, allocating new log files, and waiting for a usable log file.

  > **Tip:** Making this parameter too large can slow things down. The LOGBUFSZ memory comes out of memory for DBHEAP.

- LOGPRIMARY

  This parameter allows you to specify the number of primary log files to be preallocate.

  The default value is 3.

  The default value is not recommended. Consider changing it to a higher number, at least 10.

- LOGSECOND

  This parameter specifies the number of secondary log files that are created and used for recovery log files, when the primary log files become full. The secondary log files (of the same size as specified in LOGFILSIZ) are allocated one at a time as needed, up to a maximum number as controlled by this parameter.

The default value is 2.

Change the default value to a number close to 80% of the number in the LOGPRIMARY parameter in small environments that do not have many transactions. Alternatively, change the default value to at least three times more than the LOGPRIMARY parameter to environments with a significant number of transactions.

Tip: Use secondary log files for databases that have a periodic need for large amounts of log space.

- DBHEAP

  This parameter contains control block information for tables, indexes, table spaces, buffer pools, and space for the log buffer. The value corresponds to the number of pages (4 KB).

  The default value is:

  – UNIX: 1200
  – Windows database server with local and remote clients: 600
  – Windows 64-bit database server with local clients: 600
  – Windows 32-bit database server with local clients: 300

  For large scale environments, start with a value of 8000 or at least twice the size of the LOGBUFSZ parameter.

- DFT_DEGREE

  This parameter specifies the default value for the CURRENT DEGREE special register and the DEGREE bind option.

  The default value is 1.

  A value of 1 means no intrapartition parallelism.

  Note: *Intrapartition parallelism* basically means parallel processing (simultaneous execution of multiple instances of programmed instructions and data on multiple processors in order to obtain results faster) on a single symmetric multiprocessing server.

  A value of -1 means the optimizer determines the degree of intrapartition parallelism based on the number of processors and the type of query. For a multi-processor machine, set this to -1 (ANY), and allow intrapartition parallelism for this database.

- INTRA_PARALLEL

  This parameter specifies whether the database manager can use intrapartition parallelism.

The default value is NO.

You should change it to YES.

> **Restriction:** The INTRA_PARALLEL parameter does not work without changing the DFT_DEGREE parameter.

- LOCKLIST

  This indicates the amount of storage that is allocated to the locklist.

  The default value is automatic allocation.

  Consider changing this value to 1500.

- LOGBUFSZ

  This specifies the amount of database heap (defined by the DBHEAP parameter) to use as a buffer for log records before writing these records to disk. These indicate the number of pages (4 KB per page).

  The default value is 8.

  The default value is too small, therefore you should change it. Start with a value of 512 for small environments and 4000 for large scale environments.

- NUM_IOCLEANERS

  This specifies the number of asynchronous page cleaners for a database.

  The default value is automatic allocation.

  Consider changing to a minimum value of 10 or the number of physical disk drives you have available.

- NUM_IOSERVERS

  This specifies the number of I/O servers for the database.

  The default value is automatic allocation.

  Consider changing to a minimum value of 12 or the number of physical disk drives you have available plus two.

- SORTHEAP

  This defines the maximum number of private memory pages to be used for private sorts, or the maximum number of shared memory pages to be used for shared sorts.

  The default value is automatic allocation.

  This may not be desirable in an OLTP environment; if this is your case, change it. Start with a value of 1024. If changing this parameter, consider also changing the SHEAPTHRES database manager parameter additionally.

- DB2_PARALLEL_IO

  This specifies whether DB2 can use parallel I/O when reading or writing data to and from table space containers (even in situations where the table space contains only one container).

  The default value is NULL.

  If you use multiple HDs (data-striping, multiple containers, RAID), set this parameter to DB2_PARALLEL_IO=*.

- DB2_STRIPED_CONTAINERS

  This specifies whether the table space container ID tag will take up a partial or full RAID disk stripe.

  The default value is NULL.

  If you use multiple HDs (data-striping, multiple containers, RAID), set this parameter to DB2_STRIPED_CONTAINERS=on.

- DB2_BINSORT

  This enables a new sort algorithm that reduces the CPU time and elapsed time of sorts.

  The default value is YES.

  The default value is recommended.

- DB2_ANTIJOIN

  This specifies whether the optimizer should search for opportunities to transform NOT EXISTS subqueries into anti-joins, which can be processed more efficiently by DB2.

  The default value for an ESE environment is NO.

  The default value for a non-ESE environment is YES.

  Set this parameter to YES.

- DB2_HASH_JOIN

  This specifies whether a hash join can be used when compiling a data access plan.

  The default value is NO.

  Enable this feature; set the parameter to YES.

- DB2_PRED_FACTORIZE

  This specifies whether the optimizer searches for opportunities to extract additional predicates from disjuncts.

  The default value is NO.

  Enable this feature; set the parameter to YES.

- DB2_CORRELATED_PREDICATES

  This specifies whether the optimizer should use the KEYCARD information of unique index statistics to detect cases of correlation, and dynamically adjust the combined selectivities of the correlated predicates, thus obtaining a more accurate estimate of the join size and cost.

  The default value is YES.

  Keep the default value.

- DB2_RR_TO_RS

  Next key lock is used to make sure that no phantom insert or delete happens while you are scanning data; when you turn on DB2_RR_TO_RS, you will not be able to guarantee the isolation level RR.

  The default value is NO.

  Keep the default value; otherwise, your application may exhibit strange behaviors. For example, when DB2_RR_TO_RS is on, scans will skip over rows that have been deleted but not committed, even though the row may have qualified for the scan.

- DB2_DIRECT_IO and DB2NTNOCACHE

  This specifies whether file system caching is performed on AIX and Windows respectively.

  The default value is OFF for both operating systems.

  Enable this feature, setting the parameter to ON so that the system eliminates caching and allows more memory to be available to the database so that the buffer pool or SORTHEAP can be increased.

Oracle parameters

This section provides information about how to tune a Oracle database in the context of parameters.

- SGA_MAX_SIZE

  This specifies the maximum size of the SGA for the lifetime of the instance. A good starting value is 800 MB.

  The shared_pool_size parameter is very hard to determine before statistics are gathered about the actual use of the shared pool. In Oracle 9i, however, it is dynamic, and the upper limit of shared pool size is controlled by the sga_max_size parameter.

  For systems with more than 1 GB: shared_pool_size = 128 MB and sga_max_size = 30% of real memory.

  Starting with Oracle 9i, SGA managing is dynamic. It means that the DBA simply has to set the maximum amount of memory available to Oracle (sga_max_size) and a initial value (shared_pool_size), and it will automatically grow or shrink as necessary. Also, we advise you to use the lock_sga parameter to lock the SGA in real memory when you have large amounts of it.

- SGA_LOCK

  This locks the entire SGA into physical memory, ensuring no SGA memory is paged to the disk.

  Set to true.

- DB_CACHE_SIZE

  This specifies the size of the DEFAULT buffer pool for buffers with the primary block size.

  Start with a value of 500 MB. If you have no idea about how large to set this parameter (which is typical on a new system until you start running tests), then set it so that the SGA is roughly 40% to 50% of memory.

- OPEN_CURSORS

  This specifies the maximum number of open cursors (handles to private SQL areas) a session can have at once. You can use this parameter to prevent a session from opening an excessive number of cursors.

  Default value is 50.

  The default value is usually too small for most of DSS and some OLTP environments; therefore, set it to 500.

- SHARED_POOL_SIZE

  This specifies the size of the shared pool, in bytes. The shared pool contains shared cursors, stored procedures, control structures, and other structures.

  Start with a value of 148 MB.

- CURSOR_SHARING

  This determines what kind of SQL statements can share the same cursors.

  The default value is EXACT.

  You can increase performance when you are running select statements by setting CURSOR_SHARING to SIMILAR.

- PGA_AGGREGATE_TARGET

  This specifies the target aggregate PGA memory available to all server processes attached to the instance. The recommended starting value is 1 GB. The WORKAREA_SIZE_POLICY will be set to AUTO, if the PGA_AGGREGATE_TARGET parameter is set to any value.

- LOG_BUFFER

This specifies the amount of memory (in bytes) that Oracle uses when buffering redo entries to a redo log file. This buffer speeds up the database performance by allowing transactions to record the updates to the database but not send nearly empty log records to the redo log disk.

If there are many transactions added to the log buffer faster than they can be written to disk, then the buffer can get filled up. This can impact performance quite negatively.

The recommended starting value is 1 MB; you can also set the size of this parameter to be equal to your redo log file's size.

MS SQL Server parameters

This section provides information about how to tune an SQL Server database in the context of parameters.

- Affinity mask

  This limits SQL Server execution to only a certain set of processors defined by the bit mask. It is useful for reserving processors for other applications running on the database server.

  The default value is 0; execute on all processors.

  There is no need to alter this setting if your server is a dedicated database server with only one instance of SQL Server running. However, if you have multiple SQL Server instances, you might want to change the parameter to assign each SQL Server instance to particular processes on the server.

  Table 5-3 shows the affinity mask values for an 8-CPU system.

Table 5-3 Affinity mask values for an 8-CPU system

Decimal value	Binary bit mask	Allow SQL Server threads on processors
1	00000001	0
3	00000011	0 and 1
7	00000111	0, 1 and 2
15	00001111	0, 1, 2 and 3
31	00011111	0, 1, 2, 3, and 4
63	00111111	0, 1, 2, 3, 4, and 5
127	01111111	0, 1, 2, 3, 4, 5, and 6
255	11111111	0, 1, 2, 3, 4, 5, 6, and 7

- Affinity I/O mask

  This limits SQL Server I/O threads execution to only a certain set of processors defined by the bit mask.

  The default value is 0; execute on all processors.

  The affinity I/O mask option is defined according to the same conditions as the affinity mask option. It is best to use the default setting of 0.

- Lightweight pooling

  This controls fiber mode scheduling. It primarily helps large multiprocessor servers that are experiencing a high volume of context switching and high processor utilization.

  The default value is OFF.

  In general, the lightweight pooling option is not recommended, because it results in only minor performance improvements. This option is typically used for benchmark workloads that are extremely uniform.

  > **Restriction:** This parameter is not supported for Microsoft Windows 2000 and Microsoft Windows XP, because today only Windows Server® 2003 provides full support for lightweight pooling.

- Priority boos

  This option specifies whether the SQL Server should have a higher scheduling priority than other processes on the same machine.

  The default value is 0.

  There is no need to alter this setting if your server is a dedicated database server, because you might drain CPU from other processes such as networking and even I/O. This can cause failure situations and transaction rollbacks, so you might be gaining performance on one side, but losing it on the other.

  > **Important:** Raising the priority too high may drain resources from essential operating system and network functions, resulting in problems in shutting down the SQL Server or using other operating system tasks on the server.

- max degree of parallelism

  This limits the number of processors considered for parallel plan execution to a specified value.

The default value is 0; execute on all processors, no limit. This default setting may help some complex SQL statements, but it can take away CPU cycles from other users during high online usage periods.

Set this parameter to 1 during peak OLTP periods. Increase the value of this parameter during periods of low OLTP and high batch processing, reporting, and query activity.

> **Note:** Index creation and re-creation can take advantage of parallelism. Therefore, it is advisable to enable parallelism through this setting when planning to build or rebuild indexes.

Performance tests on some of the batch processes showed that parallelism can result in very good performance. If you do not want to toggle this value based on the type of load, you can set the value to 1 to disable this setting.

However, you may want to explore some middle ground by setting this option to a higher value (for example, 2), which may help some complex batch jobs and also online performance. The value can be changed to a even higher number up to 64.

However, do not think that if you increase this value to the maximum, it will exponentially increase the overall performance of your SQL statement. This will not be true for 98% of the SQL statements. In fact, it may do the opposite; that is, a simple query may take longer than expected.

Note that if the computer has only one processor, then the max degree of parallelism value is ignored.

> **Important:** If the affinity mask option is not set to the default, it may restrict the number of processors available to SQL Server on symmetric multiprocessing (SMP) systems.

- Cost threshold for parallelism

  This specifies the cost threshold in seconds that needs to be met before a query is eligible to be executed with a parallel query execution plan.

  The default value is 5.

  Most of the SQL statements of Tivoli are simple in nature and do not require parallel query execution plans. Consider increasing the value to 60 so that only true complex queries are evaluated for parallel query execution plans. Therefore, if you set this value to 60, this means that the Query Optimizer will not consider parallelism for any query that it thinks will take less than 60 seconds to run.

> **Important:** The SQL Server ignores the cost threshold for parallelism value under the following conditions:
>
> - Your computer has only one processor.
> - Only a single CPU is available to SQL Server because of the affinity mask configuration option.
> - The max degree of parallelism option is set to 1.

- AWE enabled

  Enable this parameter to take advantage of memory above 4 GB. This is applicable only for 32-bit operating systems.

  In Microsoft SQL Server 2005, you can use the Address Windowing Extensions (AWE) API to provide access to physical memory in excess of the limits set on configured virtual memory.

  The default value is 0 (Disabled).

  To enable AWE, set awe enabled to 1.

> **Important:** Using AWE enabled and max server memory can affect the performance of other applications or SQL Server running in a multi-instance or cluster environment.

- Max server memory

  This specifies the maximum and minimum memory, respectively, in megabytes allocated to an SQL Server instance.

  The default values are 2147483647 and 0, respectively.

  Max server memory/min server memory:

  The SQL Server can adapt its memory consumption to the workload. The SQL Server allocates memory dynamically within the range of the real memory. Use this setting for a dedicated database server that does not leverage AWE.

5.6 OMEGAMON XE performance

OMEGAMON agents reside on the z/OS platform. Most of the performance tuning tips provided in this chapter apply equally to OMEGAMON XE as well, such as situation tuning, query optimization and workspace layout design.

Indeed, correct tuning of the IBM Tivoli Monitoring environment may be considered even more important when managing z/OS-based agents from a distributed hub monitoring server due to overhead caused by the EBCDIC/ASCII data conversion performed whenever data is transferred either way between the host and distributed platforms.

This section discusses additional considerations that may be appropriate for customers with large scale installations of OMEGAMON agents, including:

- General performance guidelines
- Short-term historical collection at agent
- Use backup and archiving jobs sparingly
- Tuning ITMS:Engine

5.6.1 General performance guidelines

In this section, we suggest generic performance guidelines that may be applicable to large scale installations of OMEGAMON agents. These guidelines are:

- Minimize ASCII-to-EBCDIC conversions.

  Whenever data passes between z/OS and distributed machines, the *receiver* converts the data to ASCII or EBCDIC as required. These conversions can occur for the following cases:

  – A z/OS-based remote monitoring server connects to a distributed-based hub monitoring server.

  – A z/OS-based hub monitoring server connects to a portal server.

  – Warehouse proxy agent connects to a z/OS-based agent or z/OS-based monitoring server.

  – Distributed agents connect to a z/OS-based monitoring server.

  In all cases, it is advisable to limit the data conversion for all required processing. Because the warehouse proxy agent and portal server are available only on ASCII-based platforms (Windows, UNIX, Linux, including Linux on zSeries), any z/OS-based data must go through the conversion process at least once.

  Examples of redundant data conversion that can be avoided by having a distributed-based Tivoli Enterprise Monitoring Server include:

  – Distributed agent connected to z/OS-based monitoring server and retrieved by portal server.

  - Warehouse proxy connected to z/OS-based Tivoli Enterprise Monitoring Server for retrieving data by portal client.
- Use one or more remote monitoring server deployed on the host environment to connect agents in a large environment. The benefits of using this structure are:
  - You can offload agent management processing from the hub monitoring server so it can perform hub-related work.
  - It provides support for more agents to be deployed relative to using a single monitoring server.
  - It provides a heartbeat concentrator to minimize network traffic for hub monitoring server (crossing the host to distributed link). The remote monitoring server instead sends a heartbeat summary to the hub monitoring server.
  - IT shortens the distance traveled for agent heartbeat and other agent-related processing. For example, a geographically local remote monitoring server is more responsive to agent processing. This topology can be useful for geographically dispersed enterprises. Also, policies that run on these remote monitoring server are more responsive and hub tolerant.
- Using individual address space adds little to CPU and memory requirements and can improve problem determination by compartmentalizing the problem. A typical recommendation is to run z/OS-based agents in their own address spaces, if possible. A single monitoring agent per address space provides the following benefits:
  - It reduces problem source identification time because you can more easily isolate and identify the source of problems when the address space contains only one component.
  - With multiple address spaces, you can make a small change or upgrade the agent and cycle one address space instead of quiescing all the others. If you have a problem that is causing your agent address space to terminate, you lose one component instead of multiple components.
  - Startup time for each address space is significantly reduced. When a monitoring server or monitoring agent starts (or restarts), many services are engaged, including the historical persistent data store. These services all have to initialize before startup is complete.

    If you have the historical persistent data stores in a different address spaces, the effect on monitoring server startup time is reduced. If you have multiple-engine hardware, they can happen concurrently. Many address spaces preload all of their code. If you have multiple products, it has to load multiple products; if you have one product, it only loads that one plus the ITMS:Engine.

- Remove duplicate data collection and redundant address spaces. Processing overhead such as OMEGAMON II® CUA® address spaces do not have to be active for most of the data that is collected by OMEGAMON XE.

5.6.2 Short-term historical collection at agent

Short-term historical data collected for OMEGAMON agents is stored in Persistent Data Stores (PDS), a set of VSAM files created at agent configuration time. It is possible for short-term history to be stored at the monitoring server rather than the agent. However, we recommend that for performance purposes, PDS files at the agents are used. If the hub monitoring server was not based on the host but a distributed platform such as Windows, there would be an expensive overhead of EBCDIC/ASCII data conversion required each time a data sample is taken for historical collection.

By default, the IBM Configuration Tool (ICAT) forces users to define their PDS local to the monitoring agent for most OMEGAMON agents. The result of this is that if a monitoring agent is configured to run within its own address space, then the PDS will be defined in that address space.

If the monitoring agent is configured to run in the same address space as a monitoring server, either hub or preferably remote, then the PDS must be within that address space.

The exceptions to this are OMEGAMON XE for CICS, OMEGAMON XE for IMS™, and OMEGAMON XE for DB2. However, it is good practice for most environments to configure the PDS local to the monitoring agent.

> **Note:** A persistent data store will need to be configured at the hub monitoring server if it is z/OS-based to store short-term historical data for both z/OS and non-z/OS agents. This topology is beyond the scope of this book, which is covering an environment with a distributed hub monitoring server.

5.6.3 Use backup and archiving jobs sparingly

When defining the PDS using ICAT, you are asked to specify the amount of cylinders to be allocated to each group for storage. Ideally, you want to allocate enough space so that maintenance procedures need to run no more than once a day without allocating excessive space that remains unused.

Over time, the appropriate allocation for your environment can be determined through observing how often the maintenance procedures are running and adjusting space according. The amount of storage needed is dependent on the number of attribute groups have been designated for historical collection multiplied by the data collection interval.

After a PDS becomes full, the automatic maintenance process begins. A check is made to see if there are any empty PDSs remaining to continue data collection. If no empty data sets remain, then data collection is temporarily suspended while maintenance is applied to the oldest data set.

The maintenance job will either back up the PDS to another location (such as DASD or tape), or it will export the data into a flat file suitable for external programs to post process the data. The maintenance job carries a large overhead and if started will slow the entire historical collection process down.

We recommend that application of the maintenance jobs should be minimized. For most users, the Tivoli Data Warehouse via the warehouse proxy agent should be used as the backup procedure, rather than archiving onto tape or DASD with a recommended warehousing occurring every 24 hours.

If the maintenance job is found to be running several times within a single 24-hour period due to the PDS becoming full, it is strongly recommended that the PDS size should be increased until the job runs no more than once in any 24-hour period.

5.6.4 Tuning ITMS:Engine

The ITMS:Engine provides common functions such as communications, diagnosis (dumps), and logging (RKLVLOG) for z/OS-based monitoring agents, OMEGAMON II components of OMEGAMON XE products, plus z/OS-based monitoring server. It has startup parameters defined with appropriate defaults for most customer environments though for some customers with large amounts of data to be transferred these defaults may be insufficient.

These parameters are defined in the data set pointed to by the RKLVIN DD statement in the started task procedure for the monitoring agent or monitoring server. Typically this would be as follows, where xx is replaced by the product code for the monitoring agent or monitoring server:

```
&rhilev.&midlev.RKANPARU(KxxSYSIN)
```

If you find a message similar to Example 5-3 on page 180 in the RKLVLOG for an OMEGAMON agent, it means an error has occurred when a monitoring agent tried to return data for the table specified by the table variable and failed. The most common cause of this error is a query that returns a large number of rows of data, causing an out-of-memory condition.

Example 5-3 RKLVLOG error message for failed request

```
KN3I008E CALL TO ADDDATA FUNCTION FAILED, RETURN CODE=retcode, FOR
TABLE=table
```

To address this issue, you could modify the query as described in 5.8, "Query optimization" on page 182 so that it returns fewer rows of data or change the LIMIT and MINIMUM values in the ITMS:Engine parameters:

- The LIMIT parameter can be used to specify the largest block of primary storage or extended storage that can be allocated. The syntax for setting the largest block of extended storage is as shown:
  - LIMIT(n,X)

  This value is specified in bytes, as a power of 2. For example, if n is 22, the largest block that can be allocated is 4 MB. If the LIMIT value is too small and a process in ITMS:Engine attempts to allocate a block of storage larger allowed by LIMIT, then a program interruption U0100 or U0200 results. When managing a large number of resources, a value of LIMIT (25,X) is recommended.

  > **Note:** Modifying the limit value for primary storage, LIMIT(n,P), is not recommended.

- The MINIMUM parameter is used to specify the minimum amount, in KB, of extended storage that can be allocated. This value is specified with the following syntax where n represents the minimum amount of extended storage in KB that can be allocated:
  - MINIMUM(n,X)

  For example, to specify a 16 MB above-the-line region, specify MINIMUM(16384,X). In a large scale environment, consider specifying the value as MINIMUM (500000,X).

When the LIMIT and MINIMUM are increased for a monitoring agent, the LIMIT and MINIMUM should also be increased on the corresponding remote monitoring server, if applicable.

5.7 Event management integration performance

This section describes tuning parameters that may be appropriate for users of IBM Tivoli Enterprise Console (TEC).

5.7.1 Tivoli Enterprise Console tuning parameters

The Tivoli Enterprise Console tuning parameters listed in Table 5-4 are for UNIX servers running Tivoli Enterprise Console with portmapper.

> **Note:** These parameters are *not* to be used for Windows or UNIX servers with the `tec_recv_agent_port=5529` turned on, which will bypass portmapper and make the Tivoli Enterprise Console server process listen on port 5529/tcp.

These parameters are important for sending events from the monitoring server to the Tivoli Enterprise Console server, so the Tivoli Enterprise Console server will not get overloaded.

These parameters will need to be updated on the Tivoli Enterprise Console server in the following configuration file:

```
$BINDIR/TME/TEC/.tec_config
```

Table 5-4 Tivoli Enterprise Console tuning parameters

Parameter	Description
getport_timeout_seconds	Specifies the number of seconds to wait before resending the UDP call for a port, if no response is received. The UDP call is transmitted until the RPC call times out. The default value is zero (0).
getport_timeout_usec	Specifies the number of microseconds to add to the value specified with the getport_timeout_seconds option. The default value is 50,000 microseconds.
getport_total_timeout_seconds	Specifies the number of seconds to wait for a response from a call to the portmapper. The default value is zero (0).
getport_total_timeout_usec	Specifies the number of microseconds to add to the value specified with the getport_total_timeout_seconds option. The default value is 50,000 microseconds.

> **Tip:** After you update *any* of these parameters, you must stop and start the Tivoli Enterprise Console server process:
>
> ```
> wstopesvr
> wstartesvr
> ```

5.8 Query optimization

This section discusses tuning the queries that are processed to display the tables, charts, and graphs that make up workspace views within the Tivoli Enterprise Portal.

5.8.1 Query processing

The query assigned to a chart or table view requests data from a particular attribute group. It executes when you open or refresh the workspace. Queries make up the processing load for on-demand data collection. You can reduce the frequency and amount of data sampling by:

- Customizing the query to filter out unwanted data. This reduces the number of selection criteria (rows) and attributes (columns) collected.
- Applying the same query to other views in the workspace. This reduces the number of data samples required: one query uses a single sample for multiple views.
- Disabling automatic refresh of workspace views or adjust the refresh rate to longer intervals. This causes Tivoli Enterprise Monitoring Agent data to be collected less frequently.
- Consider how you wish to display the data returned by a query. A graphical view workspace may require significantly less data compared to a table view, because it uses only the truly numeric data and leaves out the character data.

> **Note:** Do not confuse custom queries with view filters from the Filters tab of the Query Properties editor. View filters fine-tune the data after it has been retrieved by the query and do not reduce network traffic, data collection processing, or memory demands.

The following general recommendations and observations should be considered as well:

- Some attributes are more expensive to retrieve than others. One expensive column in a table will make any workspace view or situation that references that table more expensive. An example of an expensive attribute is one that must run long storage chains to determine its value, such as using a process table to look for looping tasks. Where possible, ensure you only retrieve attributes that are required for the monitoring process.
- Column function (such as MIN, MAX, AVG, and so on) requires post-processing of the query results set after data is returned to Tivoli Enterprise Monitoring Server.

- Use more efficient data manipulating functions, such as substring instead of string scan. If you know the position of the string to search, do not scan the whole string to check for the value.

Post-filters versus pre-filters

Performance will be improved for each view when you pre-filter the required view information and only send to the portal client what is needed in that view. You thereby reduce the amount of data sent through the network, and do not have to post-process it either. However, there is one exception to the rule.

Think of a workspace that contains many views. Each of those views has a query associated with it, which will be issued when the workspace is accessed. This might result in many queries that have to be processed in parallel.

A better way of doing this might be to create one query that returns all data that is required in the workspace. In this case, the query will only be issued once and the data can then be post-filtered for each view to only display information as it applies to each view.

One important consideration, however, is that queries are saved globally and are not user ID-dependent. This means that only administrators will be able to modify queries in most installations. For the end user to be able to modify filters, the preferred method might be the filters applied in the view properties filters tab.

5.8.2 Defining custom queries

Custom queries reduce network traffic, processing at the agent and Tivoli Enterprise Monitoring Server, and memory usage at the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal client. Custom queries accomplish this by limiting the number of rows and columns passed from the Tivoli Enterprise Monitoring Agent to the Tivoli Enterprise Monitoring Server.

Most of the predefined, product-provided queries request all columns and all rows of an attribute group, of which only a few may be of interest to you. Removing the unwanted columns (attributes) will reduce the amount of data transferred from monitoring agent to Tivoli Enterprise Monitoring client via the portal server, hub monitoring server and remote monitoring server. Additionally, in the case of OMEGAMON monitoring agents that reside on z/OS, you will also reduce the amount of EBCDIC/ASCII conversion required when data is passed between mainframe and distributed platforms.

It is recommended that you tune any queries servicing workspaces that are frequently executed or return large quantities of data. Query execution always requires resources, and intermittent extremely large reports will cause a spike in memory requirements and network resources.

- Restricting the number of rows

  Most predefined queries return *all* rows and columns. You can create a custom query to filter out the irrelevant or uninteresting metrics. Not only does this make it easier to read the report, but it saves Tivoli Enterprise Portal Server memory, client memory, and CPU because there are fewer rows to translate, sort, and transmit.

  You may be using view filters inappropriately. View filters work only on the current page returned from the query. For example, if page size is 100 lines and the filter reduces it to five lines on page 1 and similarly on subsequent pages, the row set cannot be seen on one page. Do not increase the workspace page size to see everything on one page. Increased page size actually increases Tivoli Enterprise Portal client memory requirements.

  Instead, avoid this condition by creating a custom query that filters the data at query execution time.

- Restricting the number of columns

  Most predefined queries return *all* columns (or attributes). A particular attribute group may contain 50 columns, yet all you need is five. Creating a custom query to retrieve just the desired five attributes will reduce Tivoli Enterprise Portal Server and client CPU and memory.

  For example, the OMEGAMON XE for z/OS product-provided DASD_MVS_Devices query returns 32 columns for approximately 5,100 rows of data in the ITSO environment. Many of the default columns and many of the rows of data may not be of interest so we made a copy of the query and modified the specification.

  The modified version of the query, as shown in Figure 5-10 on page 185, removes unwanted columns and selects only devices with an I/O rate greater than zero and contains only 17 columns and 20 rows of data, which is a drastically smaller amount of data.

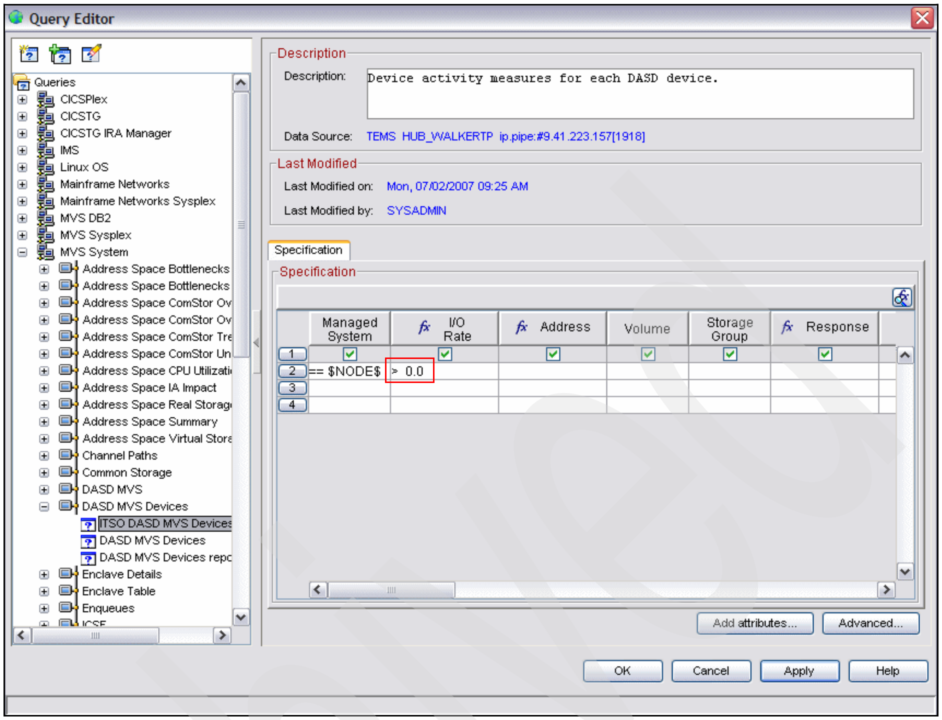


Figure 5-10 ITSO DASD MVS™ query

- Use the same query in a workspace

  If you have multiple views in a workspace requiring data from different attribute groups, you will need a different query for each group. If, however, the views have data from the same attribute group, use a single query to accommodate both even if the attributes (columns) each view requires differs.

  Two unique queries will each drive data collection at the agent and increase resource overhead. For each workspace, have one query that can be shared by all views using that attribute group. Remember that the entire results set for each query is stored on the Tivoli Enterprise Portal Server, this technique will avoid duplicate result sets being stored.

- Collect agent data less frequently

  A good practice is to avoid the use of workspace automatic refresh when possible. The Navigator view and event views (message log, event console, and graphic console) will refresh automatically. This provides you with instantaneous alerts, and you can navigate to their event workspaces with actual data. The graphic view, which is the graphical equivalent of the Navigator which shows alerts but no data, affects client memory but not the portal server.

- Reduce the distribution of queries

  A query can be assigned to a particular managed system or to a group of systems by defining a Managed System List (MSL). The default MSL will usually include all known instances for that application or operating system.

  It may be that you want this query to be applied to only certain managed systems and therefore, distribution to all managed systems is an unnecessary waste of resource. Modify MSLs to reduce the distribution of the query. Also remove the MSLs for queries that are of no interest to the user. Even if you are not viewing the results of the query, there may be a use of system resources that can be avoided by restricting the distribution of the unneeded queries.

5.9 Situation optimization

A *situation* is the mechanism in IBM Tivoli Monitoring that generates an alert when an unusual or abnormal condition occurs within the monitored environment. The situation evaluates system attributes and decides whether a potential problem condition is occurring. The situations are evaluated on a regular interval. This section covers analyzing situation processing and reducing its processing requirements.

Several important considerations for situation processing that can greatly affect the performance of the IBM Tivoli Monitoring system are discussed here:

- Situation processing
- Situation distribution
- Situation predicate processing
- Situation sampling interval
- Situation synchronization
- Situations and policies
- Embedded situations
- Using policies to manage time-dependent situations

5.9.1 Situation processing

As mentioned, situations are a mechanism in IBM Tivoli Monitoring that automatically performs monitoring for you. Situations collect performance indicators and subject them to conditional checking, which is associated with a threshold.

When the condition is true, the situation is raised or fires and an event is generated. These state changes are passed to the Tivoli Enterprise Monitoring Server. The end user is alerted to the event by the Tivoli Enterprise Portal client.

The frequency of state changes affects hub and remote Tivoli Enterprise Monitoring Server capacity. It also affects storage requirements because these events are saved in memory and written to disk. The sampling intervals themselves affect the Tivoli Enterprise Monitoring Server capacity.

5.9.2 Situation distribution

All situations are stored in QA1CSITF, a database file for distributed-based Tivoli Enterprise Monitoring Server or RKDSSITF, a VSAM file for z/OS-based Tivoli Enterprise Monitoring Server. This file shows every user-defined situation, including those that are not distributed or are distributed but not started.

Situations can be autostarted or manually started. Situations assigned to managed systems, whether autostarted or not, are distributed to their Tivoli Enterprise Monitoring Agents at Tivoli Enterprise Monitoring Server startup time. This requires processing and affects Tivoli Enterprise Monitoring Server startup time.

If a Managed System List (MSL) is updated, then all the situations that use the MSL are redistributed to all managed systems in the list that require processing at the Tivoli Enterprise Monitoring Server and the affected Tivoli Enterprise Monitoring Agents. For example, when a new CICS region comes online for the first time, the *CICS Managed System List is updated. If the default distribution assignments have not been modified, this will cause all product-provided CICS situations to be redistributed to all Tivoli Enterprise Monitoring Agents that are monitoring CICS regions. Therefore, you should remove the distribution assignment from any situation that is not autostarted or manually started.

When you copy predefined situations and give them site-specific names to protect them from changes that IBM might make in future releases, make sure you disable the original situation by turning off autostart and removing its distribution list. This is critical for many distributed products that distribute all of their predefined situations.

At Tivoli Enterprise Monitoring Server startup, autostarted situations must be distributed to all of their agents. Note that subnodes are not unique distributions. Because situations also use CPU at the agents, Tivoli Enterprise Monitoring Servers that have agents installed on them are also affected.

To stop a situation from autostarting, open the situation editor and select the **Formula** tab, which is the default. If it is checked, remove the check mark from the **Run as startup** specification, as shown in Figure 5-11. This does not stop a situation that is currently started, because it will not take effect until the next startup of the affected Tivoli Enterprise Monitoring Agent. If required, manually stop the situations.

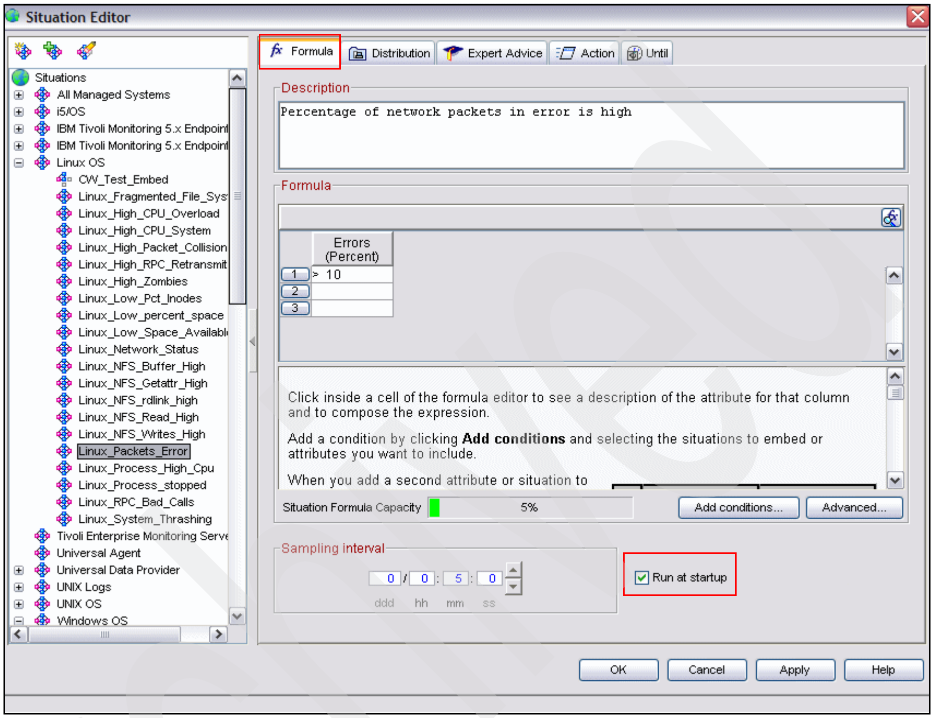


Figure 5-11 Modifying a situation's run at startup setting

To modify a situation's distribution, open the situation editor and select the **Distribution** tab. Select the Managed Systems, Managed System Lists, or both in the Assigned window. Then use the Add and Remove arrows to remove the assigned managed systems, as shown in Figure 5-12 on page 189. At the next Tivoli Enterprise Monitoring Agent startup, this situation will not be distributed.

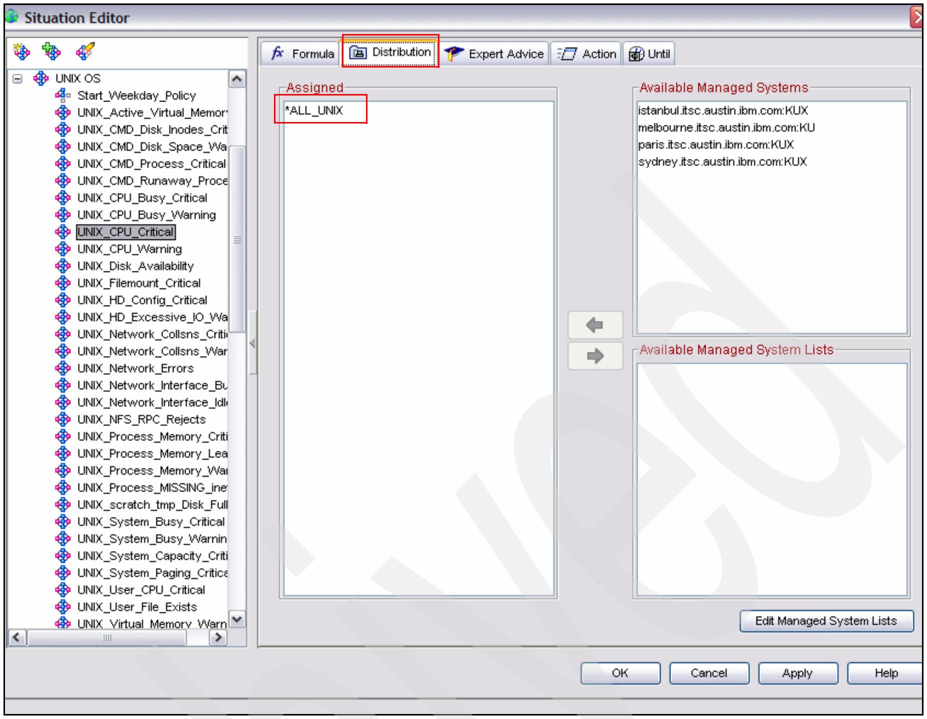


Figure 5-12 Modify situation distribution

IBM Tivoli Monitoring automatically creates MSLs for situations to monitor all discovered instances of a monitored operating system or application. For example, the *CICS MSL includes all CICS regions known within this IBM Tivoli Monitoring environment. The *ALL_UNIX MSL, as shown in Figure 5-12, includes all monitored UNIX systems.

Many customers configure production regions and subsystems to a production hub monitoring server and portal server, and test regions or subsystems to a separate set. If you have systems for which you do not want to perform situation analysis or collect historical data, create user-defined Managed System Lists such as CICS_Production, which will contain only the list of regions you want monitored.

Tip: Scripts have been developed to automate the process of stopping all autostarted product-provided situations and also to remove them from distributing across all monitoring server instances. You may find this convenient to use, especially when configuring support for some monitoring agents (such as OMEGAMON XE for CICS) that have a very large number of product-provided situations.

The scripts are available via IBM Tivoli Open Process Automation Library (OPAL) Web site. Search under IBM Tivoli Monitoring Situation Distribution.

5.9.3 Situation predicate processing

Each situation contains *predicates*, in which the monitoring attributes to be evaluated are specified. Different predicates generate different processing requirements. It is important to understand how predicates are processed in order to analyze them.

Predicates are processed sequentially from first to last. If you have multiple predicates in a situation, order them in such a way as to eliminate as many rows as possible as early as possible, with the least expensive or most restrictive filter as the first predicate. For example, on Windows, to check whether process XYZ uses more than n amount of real storage, you have to test two attributes: process name and real storage use.

If you first test on real storage use, the result set may contain multiple rows; then you check whether our process name is among the returned rows. It would be more efficient to first test on process name (the result will be one row), followed by the test on the storage usage, just on this single row.

Here are other predicate processing tips:

- Numeric attributes are processed more quickly than text attributes.
- String checking with substring (STR) is more efficient than the string scan (SCAN), especially for long strings. If you know the exact location of the text or characters to be evaluated, use a substring.
- A column function is evaluated at the Tivoli Enterprise Monitoring Server, so all available rows will be sent back to Tivoli Enterprise Monitoring Server from the agent, with high network usage, memory, and processing requirements as a result. Try to avoid the predicates such as MAX, MIN, AVG on a column.

5.9.4 Situation sampling interval

Each predefined situation has a default sampling interval defined. However, these defaults may not be appropriate for your installation. Adjust the intervals for each situation to fit the performance standards at your environment.

For example, situations that are part of a *health check* process can use a longer interval, such as 30 minutes. Situations that detect a *resource shortage* condition probably require a more frequent interval.

With IBM Tivoli Monitoring, you can set different sampling intervals for every situation. Be aware that picking different intervals will cause higher processing overhead. Some products, such as IBM Tivoli OMEGAMON XE for CICS on z/OS, have preselected that all critical severity situations for stall and crash prevention are automatically started by default. However, they do not autostart any warning situations.

For some customers, the warning alerts are not important and will be ignored, so it would be a waste of processing to distribute and start these situations. Other customers pay close attention to warning alerts, in which case the situations should be started.

The predefined situations may have more frequent sampling interval than required by your environment. Recognize that not all data is equally important, nor is it equally expensive to collect. Some higher processing situations may not be crucial to availability in your environment. Predefined situations attempt to have low overhead defaults. However, no vendor can know your environment or what is relevant to your business applications.

IBM Tivoli Monitoring was designed to provide in-depth analysis, as you will need these capabilities to deal with problems when they have been identified. However, awareness of situations is not necessarily minimized by selectively disabling data collection. It is best to disable the items that you can categorize as having low benefit coupled with high overhead.

In some cases, disabling a situation is not an option if it provides high benefit even at a higher processing level. Such situations should be tuned or substituted. Some products or subsystems may have more than one way to detect the same problem, and not all situations have the same processing requirement.

IBM Tivoli Monitoring provides Boolean logic and multiple ways to detect complex situations. For example, a CICS transaction rate of zero (0) with high CPU indicates that a single transaction is consuming cycles and not allowing the dispatcher to schedule other transactions. This particular CICS-related situation can be replaced by evaluating the CICS time-of-day clock (TOD) not being updated. Numerous data collection parameters can be adjusted in scope and frequency to reduce overhead.

5.9.5 Situation synchronization

One of the biggest savings in situation processing is to perform *situation synchronization*, which is sometimes also referred to as *duperization* or *situation grouping*. Situation synchronization enables multiple situations to collectively perform a single data collection or take a sample, and the result set is evaluated from a single data sample.Situation synchronization enables more situations to be evaluated with a constant processing requirement. To be eligible for situation synchronization, the conditions listed in Table 5-5 must be fulfilled.

Table 5-5 Synchronization eligibility requirements

Situation definition	**Needed**
Same attribute group required	Yes
Same time interval required	Yes
Can be restarted	Yes
Must be autostarted (except UADVISOR cannot have autostart set to SYN)	Yes
New / update situation supported (requires Tivoli Enterprise Monitoring Server to be recycled)	Yes
Can combine situations with different distribution lists	Yes
Maximum of 10 expressions per situation	Yes
Display Item (atomize) supported	No
Take action allowed	No
Until clause permitted	No
Can use *Check for Missing Items* function	No
Can be used in a policy	No
Can use STR (return a subset of string) or SCAN (scan for string within string) function	No
Can use group functions: AVG (average), COUNT (count), MAX (maximum), MIN (minimum), SUM (sum)	No
JOIN allowed between two single-row attribute groups	No
Event persistent situations permitted	No

Based on the requirements in Table 5-5 on page 192, the following common principles apply:

- It is inadvisable to arbitrarily change the evaluation interval of situations, because this may hinder the situation synchronization processing. Setting a longer interval does not necessarily mean that processing requirements are reduced.

  For example, when you change the interval of one situation, you schedule a new sampling and double the processing. If you take a third situation and set it to yet another interval, you could triple the original overhead.

- Some complex situations cannot be synchronized because the predicate length exceeds the record length in RKDSRULD. The RKDSRULD file contains the situation synchronization rules. The actual limitation is the size of the rule.

  Typically, eight to ten conditions can be built into a single scheduled sampling, and anything that was previously combined is uncombined when you changed it. Note that some Tivoli Enterprise Monitoring Agents are intelligent enough to automatically collect information once for multiple autostarted situations with the same interval.

Situation synchronization can have both beneficial and detrimental influences on workload performance. A z/OS monitoring server that contains address space resident applications, such as OMEGAMON agents, tend to benefit the greatest from duperization since sampling is occurring in the monitoring server runtime environment. A monitoring server that has mainly Intelligent Remote Agent (IRA)-based agents attached to it will benefit far less or may even suffer response time degradation if applied to enough situation objects.

In order to enable situation synchronization, modify the monitoring server environment file and add the following line:

```
CMS_DUPER=YES
```

The monitoring server will need to be recycled following this edit. Set the value to NO if you want to ensure no duperization takes place. Refer to 5.2.1, “Monitoring server parameter tuning” on page 125 for further details about this parameter.

Situation synchronization is performed by monitoring server during its startup. If the monitoring server finds a number of situations that are eligible for synchronization, it will create a new internal situation that will perform the data collection at the specified interval.

All synchronized situations will then compare their criteria to the data returned by the internal situation. These internal situations only exist for the duration of the monitoring server run. They get an internal name that starts with _Z_ and the full name will be built from the following parts: _Z_, table name, sequence number.

For example, on Windows, when synchronizing situations on table WTPROCESS, the synchronized situation will be called _Z_WTPROCESS0. These situations are not added to the permanent situation tables in monitoring server (such as TSITDESC). However, because they are only temporary, they can only be seen in situation temporary tables such as TSITSTSC. For more information about this topic, refer to "Situation event processing component flow" on page 480.

To verify whether any synchronized situations are created, you can run a SQL statement from a portal view, using custom SQL, as shown in Example 5-4.

Example 5-4 SQL statement from a portal view

```
SELECT SITNAME,
       ATOMIZE,
       DELTASTAT,
       LCLTMSTMP,
       NODE,
       ORIGINNODE,
       RESULTS,
       SITCOUNT,
       TYPE
FROM O4SRV.TSITSTSC;
```

The synchronization occurs only at monitoring server startup, so any new situations or modifications will not benefit from grouping until the monitoring server is restarted.

On z/OS, to determine whether a situation has been included in synchronization, you can browse the RKDSRULD dataset on the Tivoli Enterprise Monitoring Server that the Tivoli Enterprise Monitoring Agent is connected to. Figure 5-13 on page 195 shows the RKDSRULD file for z/OS.

```
DITTO/ESA for MVS             VB - VSAM Browse

CATALOG
RBA 262144       Key _Z_CICSROV1___________________   Col 1       Format CHAR
VOLSER OMXE3C   Type KSDS   DSNAME OMEGAMON.SHR3.RKDSRULD

RBA        Len     <===5===10====5===20====5===30=>..5...40....5...50....5...60
262144     3032    _Z_CICSROV1_____________________..SELECT CICSROV.AIDS, CICSR
265176     3032    _Z_CICSROV3_____________________..SELECT CICSROV.AIDS, CICSR
268208     3032    _Z_CICSSTOR0____________________.aSELECT CICSSTOR.AREA, CICS
271240     3032    _Z_ICSF2________________________..SELECT ICSF.AVGWAIT, ICSF.
274272     3032    _Z_ICSF4________________________..SELECT ICSF.AVGWAIT, ICSF.
277304     3032    Crypto_CKDS_Access_Disabled_____.nSELECT ICSF.AVGWAIT, ICSF.
0          3032    Crypto_CKDS_80PCT_Full__________.~SELECT ICSF.AVGWAIT, ICSF.
3032       3032    Crypto_Internal_Error___________..SELECT ICSF.AVGWAIT, ICSF.
6064       3032    Crypto_Invalid_Master_Key_______.~SELECT ICSF.AVGWAIT, ICSF.
9096       3032    Crypto_Invalid_PKA_Master_Keys__.lSELECT ICSF.AVGWAIT, ICSF.
12128      3032    Crypto_No_Coprocessors__________..SELECT ICSF.AVGWAIT, ICSF.
```

Figure 5-13 RKDSRULD dataset contents

Figure 5-13 contains the definition that is being sent to the Tivoli Enterprise Monitoring Agents, including predefined and custom situations. Any duperized situations appear at the beginning of the file, with names start with _Z_.

Looking at the list of synchronized situations, you may see several _Z_ entries with different suffixes. These are the different synchronized collections that are performed for that data source. The example in Figure 5-13 shows two synchronized collections of CICSROV the table. This can be the result of too many predicates for one situation, or of different collection intervals.

You can also check the messages from the RKLVLOG message, KRAIRA000, which indicate which synchronized (duperized) situations are being started. Example 5-5 shows a sample message.

Example 5-5 RKLVLOG message for synchronized situation

```
KRAIRA000, Starting _Z_ICSF4 <3145945,1048781> for KM5.ICSF.,
Producer(IRA Constructor)
```

5.9.6 Situations and policies

Policies can be a significant source of processing in situation overhead. Policies can more than double the processing requirements of collecting data for situations. This is because they duplicate the collection for the situation to drive alerts to the Tivoli Enterprise Server, and are not eligible for situation synchronization.

For example, if you have three situations that are eligible for synchronization in one synchronized data collection, and you add them to policies, the result will be additional scheduling of the collector. The original situation is not eligible for synchronization but shares the collected data with the policy.

In a policy, the `Evaluate a Situation Now` activity does not actually start the situation; it simply performs one-time sampling of data. However, the `Wait Until A Situation is True` activity *does* start the situation.

5.9.7 Embedded situations

Embedded situations are inserted into the formula of another situation, called the *embedded* or *parent* situation. The properties of the parent situation override those of the embedded situation. Thus, the parent situation uses its interval to drive a take sample for the data required from the embedded situation. You might use embedded situations to look for time-dependent workload issues.

The embedded situation runs independent from the original (non-embedded) form. When the same situation is embedded in several different other situations, they are triggered independent of each other, which can cause excessive situation evaluation and unnecessary processing. However, when embedding several product-specific situations into a single parent situation, each situation runs only once (assuming none of the embedded situations are autostarted).

An example of embedded situation usage is to provide a time-sensitive situation. Instead of changing all situations to include time checking, you can use embedded situations.

There are two approaches to this. The following example illustrates the differentiation of weekday and weekend situations:

- Create a generic Weekday situation that evaluates the day of the week and is triggered if the day is determined to be greater than or equal to Monday and less than Saturday. Now embed this situation into all weekday-sensitive situations. These situations will now only trigger if the embedded situation is also true (that is, it is a weekday). This means that if you have 20 situations embedding the Weekday situation, then the Weekday situation will be executed each time the parent situation is triggered. This can mean unnecessary processing.
- Another alternative have the Weekday situation embed all situations that must be evaluated on a weekday. This removes the overhead of the additional 20 Weekday situations evaluation. However, the generated event

will be a generic one, called Weekday. (You could use a more meaningful name for Weekday, such as UNIX_Weekday_Alerts, as shown in Figure 5-14).

This situation checks to see whether the day of the week is Monday through Friday, and if true, it evaluates the situation UNIX_System_Busy_Critical.

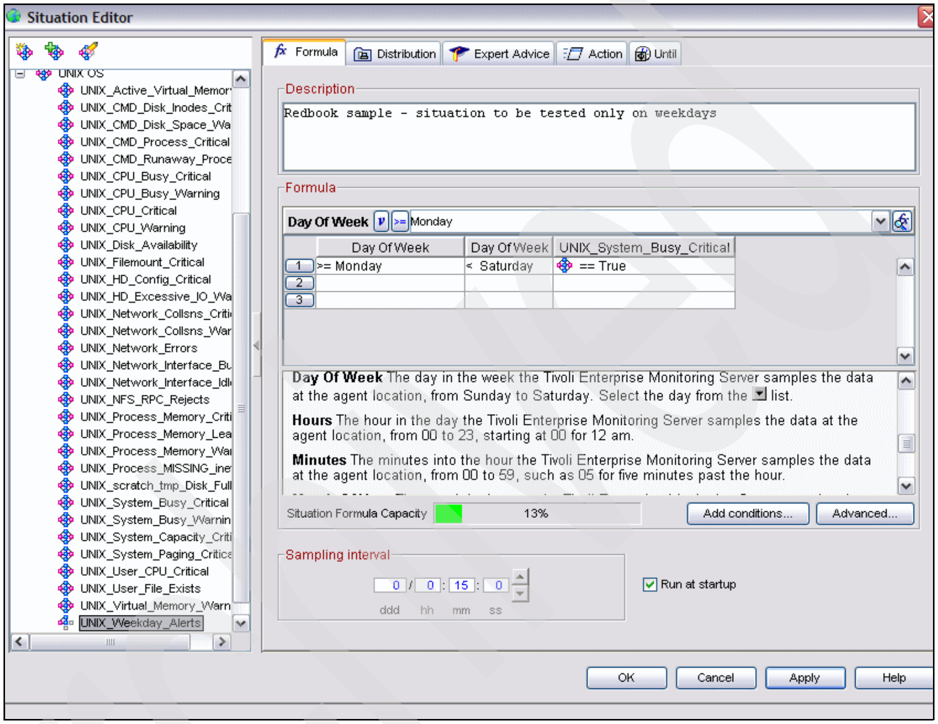


Figure 5-14 Embedded situation UNIX_Weekday_Alert

Note: Any time you use embedded situations, neither the parent or the child are eligible for situation synchronization. This may cause the product-specific situation to run more often and increase the processing requirements for the additional data collections that are being scheduled.

Here is another example. In this case we will choose three product-provided situations and embed them in a situation created by us such that the situations are only “active” Monday to Friday between 9 a.m. and 5 p.m.

1. To make the situation “active” Monday to Friday between 9 a.m. and 5 p.m., follow these steps:

   a. Click the **Edit -> Situations editor**.

b. Expand All Managed Systems, then right-click **Weekday** and select **Create Another**.

c. Specify a Name and a Description.

d. Click **Add conditions**.

e. In the Select condition dialog, select the Attribute Item **Hours** and click **OK** as shown in Figure 5-15. Repeat this step one more time, so that you will have 2 Hour attributes in the situation (to specify the time frame from 9:00 a.m. to 5 p.m.).

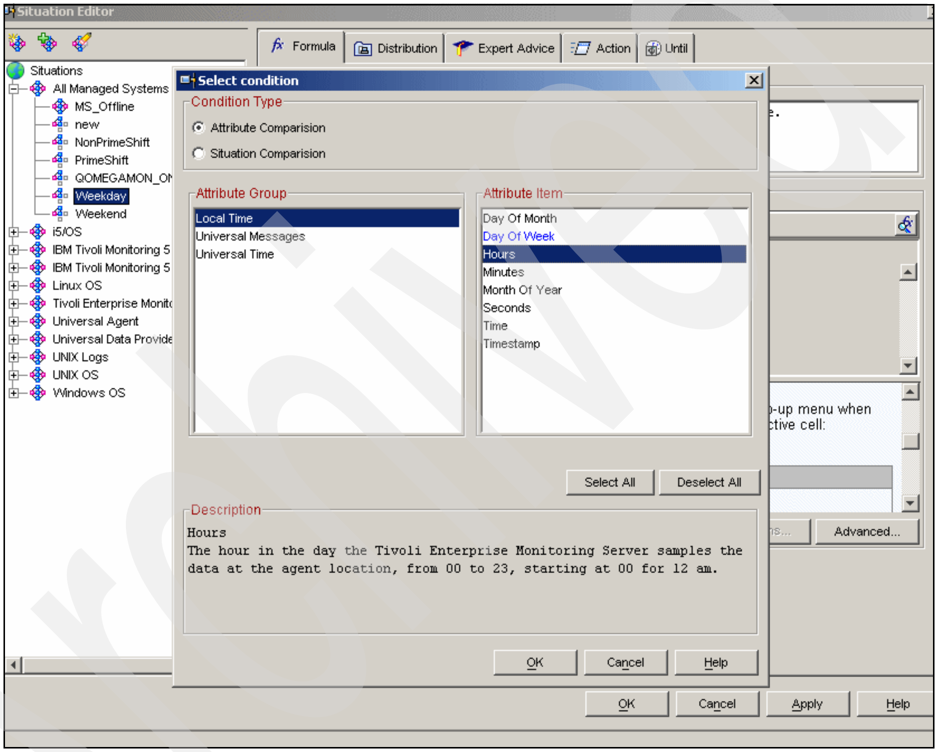


Figure 5-15 Select condition dialog

f. Specify “>=9” and “<=17” to make the situation true between 9 a.m. and 5 p.m.

2. To embed the situations, follow these steps:

a. Click **Add conditions**.

b. In the Select condition dialog, select **Situation Comparison** as the Condition Type.

c. Select the three situation names you have chosen by holding the Ctrl key. In this example we use NT_Log_Space_Low, NT_Process_CPU_Critical, and NT_Process_CPU_Warning.

d. Specify the right expressions to make the situation true between the specified timeframe; see Figure 5-16.

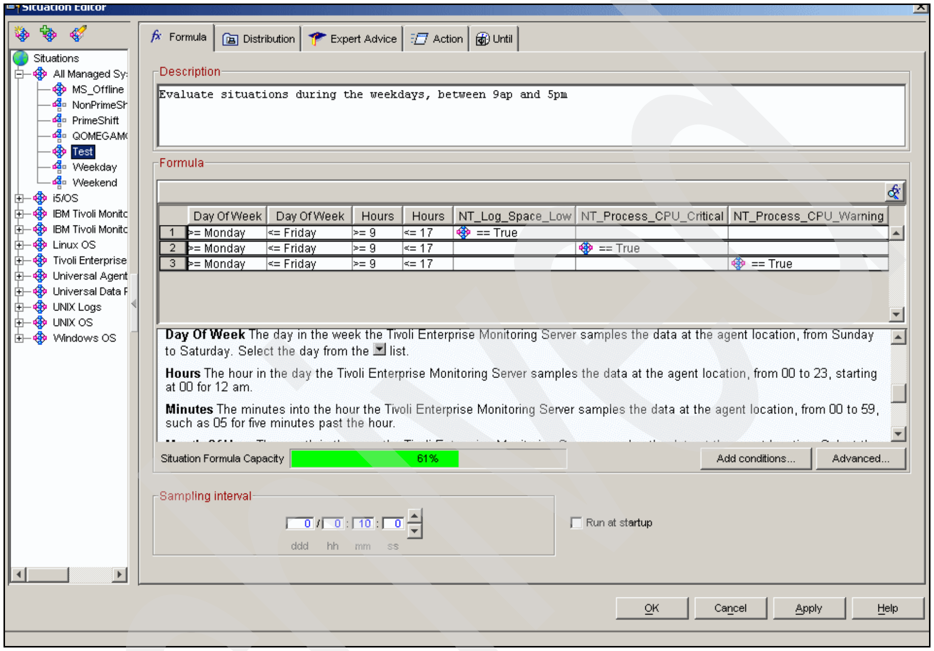


Figure 5-16 Embedded situation example

5.9.8 Using policies to manage time-dependent situations

An alternative for reducing the required processing for time-sensitive alerts is to create a policy to control false alerts based on time. This has its benefits in that all situations will be eligible for situation synchronization and the Tivoli Enterprise Monitoring Server situation monitor will not have to process (filter out) alerts during the time frame when they are not applicable. Having all situations synchronized will reduce the total number of situations, which will also reduce both monitoring server and agent overhead.

These steps outline the procedure for creating time-dependent policies to reduce the agent and processing requirements:

1. Create two situations that check the day of the week.

   – Weekday situation: check for weekday DAYOFWEEK>=”Monday” and DAYOFWEEK<“Saturday”

– Weekend situation: checking for weekend DAYOFWEEK==”Sunday” or DAYOFWEEK==”Saturday”

2. Create two policies:

   – Weekday_Policy that, based on the active date situation (Weekday), will start all the situations that should be active.

   – Weekend_Policy policy that, based on the inactive period, will stop all the situations you started in the first policy.

   By default, policies are set to autostart. Be sure to maintain that setting for both of these policies. Do not rely on the policy to start the situations the first time, because they will *not* be eligible for situation synchronization processing unless they are autostarted. See Figure 5-17 for an example of the Policy definition.

   InFigure 5-17, the policy executes when situation Weekday is true. At that time, the policy will start four situations. The started situations can cover all monitored applications.

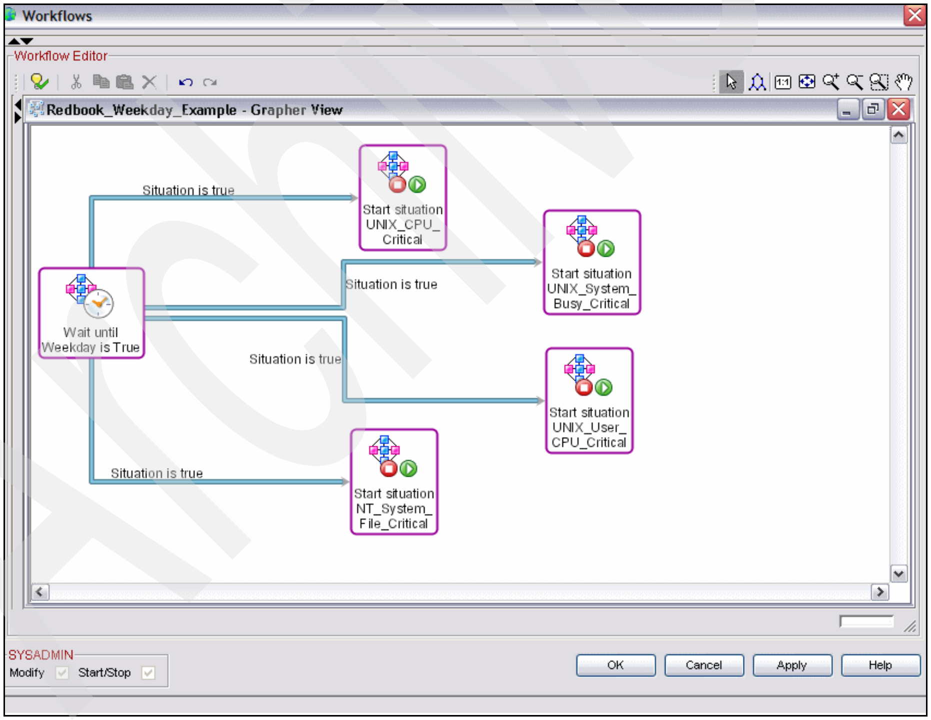


Figure 5-17 Using policies to start time-dependent situations

3. Make sure that all product situations are autostarted so they will become synchronized. You do not need to start the time/date dependent situations, because that will be done automatically by policies.

This same concept can be useful for eliminating false alerts for scheduled maintenance windows. For example, if all CICS regions are brought down every night from 2:00 a.m. until 4:00 a.m., you want to avoid the alerts associated with this planned event.

You achieve this by creating a Begin_Maintenance_Window policy with processes to stop all situations that generate false alerts. You would also have a corresponding End_Maintenance_Window policy. These policies could be manually started and stopped by the operator, instead of acknowledging all of the alerts that are normally generated.

5.10 Platform-specific recommendations

In this section we present platform-specific recommendations.

5.10.1 Disable TCP delayed acknowledgements on AIX systems

On AIX systems, the default behavior for TCP connections is to allow delayed acknowledgements (Ack packets). When tcp_nodelayack is set to zero (0), which is the default setting, TCP delays sending Ack packets by up to 200ms. This allows the Ack to be piggy-backed onto a response and minimizes system overhead.

Setting the tcp_nodelayack parameter to 1 causes TCP to send immediate acknowledgement (Ack) packets to the sender. Setting tcp_nodelayack to 1 will cause slightly more system overhead, but it can result in much higher performance for network transfers if the sender is waiting on the receiver's acknowledgement. Measurements of communication between IBM Tivoli Monitoring components have shown that setting tcp_nodelayack to 1 can significantly improve performance.

To make the parameter setting, issue the following commands:

```
# no -p -o tcp_nodelayack=1
Setting tcp_nodelayack to 1
Setting tcp_nodelayack to 1 in nextboot file
```

The `-p` flag makes the change persistent, so that it will still be in effect at the next boot. This is a dynamic change that takes effect immediately.

5.10.2 VMware configuration setting to improve tacmd response time

If you experience slow response time when invoking **tacmd** while running on a VMware ESX guest machine, consider disabling virtualization of the Time Stamp Counter (TSC). To make this change, add the following setting in the .vmx configuration file for the virtual machine where the tacmd is being invoked.

```
monitor_control.virtual_rdtsc = false
```

This parameter is described in the paper “Timekeeping in VMware Virtual Machines” on the VMware Web site:

http://www.vmware.com/pdf/vmware_timekeeping.pdf

Measurement experience has shown that this setting can significantly improve tacmd response time on VMware ESX systems.

6

Monitoring best practices

This chapter describes best practices and a methodology for implementing a monitoring solution in your environment. It also demonstrates a situation creation example to explain these concepts in action.

Although the information presented in this chapter is applicable to any type of IBM Tivoli Monitoring installation, regardless of size, in large scale implementations, with thousands of agents and situations, applying these best practices and the methodology is especially important.

This chapter addresses the following topics:

- Who owns the tool
- Defining the need for monitoring
- Building the monitoring
- IBM Tivoli Monitoring 6.2 enhancements for event integration

6.1 Who owns the tool

Before discussing a real-life application of the tool, we address a basic question: who will own the tool?

An organization performing Enterprise Systems Management can be identified as somewhere along an evolutionary process for distributed computing. The host arena has maturity. Distributed environments, however, generally can be expected to be less mature. This is not to say that the less mature environment is somehow qualitatively less desirable than a more mature environment; it is merely the recognition of a current state. The location on the maturity scale is a recognition of process maturity, technical achievement, and financial investment.

At a lower level of maturity, we can expect that the Tivoli team will own the monitoring solution and development of monitors for the enterprise. An increasingly common movement among more mature enterprise customers is to turn their developed Tivoli products into services. The Tivoli team manages infrastructure and sets direction, trains for the use of the services, and provides support for those trying to use the services. In this context, for IBM Tivoli Monitoring V6.1, the creation of monitors is left to the administrators and application teams while overall tool function and architecture is held by the Tivoli team.

What are the criteria for determining whether your implementation should be managed this way?

- IBM Tivoli Monitoring V6.1 is already installed in the environment and is stable (including implementation processes). The Tivoli team has acquired IBM Tivoli Monitoring V6.1 skills.
- Administrator teams and application teams have skills required for programming (general skills).
- You are prepared to educate the administration and application teams about how to build their monitors, and provide documentation to support this.
- Precedence exists for application teams and administration teams to “own” their own monitors, and they are ready to “own” their monitors (recognizing the resource requirements within their own teams).
- Change control to the production environment is a fairly mature process, and the application teams and administration teams will follow the process.
- Development and QA environments will be available for the application and administration teams to develop their monitors.
- A commitment to a QA process will be followed prior to introduction of the monitors into the production environment.

If these criteria cannot be met in your environment, your organization will be much better served by holding control, development, and management of the tool in a single group.

6.2 Defining the need for monitoring

This section discusses the methodology to be used in turning a need for monitoring into a technical solution to answer that need. Non-technical points (process-related) will be made in addition to the stated technical details.

Five distinct types of activities are advocated in this methodology for defining the need for monitoring:

- “Identification of monitoring need” on page 211 discusses how the nomination should come from either a business problem or from the Problem Management activity within your organization.
- “Identify the target audience” on page 212 discusses the importance of ownership of the events. The development of monitoring should always be in coordination with the system/application administrators and the help desk.
- “Identify and refine the possible events list” on page 213 discusses how to create a complete solution by investigation of the event request.
- “Meet with the target audience for approval” on page 214 is a very important activity. At this point, all parties should agree to the escalation process for this event and its owner if changes are required.
- “Create, test, and implement the monitor” on page 215 discusses the process that should surround these activities.

Without undertaking these activities, the overall success of any event creation will be hampered in some way by acceptance, value realization, or satisfaction of need.

Before discussing this need for monitoring, however, we explain some concepts related to the product that must be understood. These concepts are important to understanding the behavior of the product as it is observable at the console and therefore the use of the tool in your environment (enterprise impact).

6.2.1 Understanding the terms

For this discussion, you need to understand the following terms as defined in *IBM Tivoli Monitoring, Version 6.1.0, User’s Guide*, SC32-9409.

Event An event is an action or some occurrence, such as running out of memory or completing a transaction, that can be detected by a situation. The event causes the situation to become true and an alert to be issued.

Event indicator The event indicator is a colored icon that displays over a Navigator item when an event opens for a situation.

Monitor interval A monitor interval is a specified time, scalable to seconds, minutes, hours, or days, for how often the monitoring server checks whether a situation has become true. The minimum monitor interval is 30 seconds; the default is 15 minutes.

Pure event A pure event occurs automatically, such as a paper-out condition on the printer or writing of a new log entry. Situations written to notify of pure events remain true until they are manually closed or automatically closed by an UNTIL clause.

Sample A sample is the data that the monitoring agent collects for the server instance. The interval is the time between data samplings.

Sampled event Sampled events happen when a situation becomes true. Situations sample data at regular intervals. When the situation is true, it opens an event, which gets closed automatically when the situation goes back to false (or you can close it manually).

Situation A situation is a set of conditions that are measured according to criteria and evaluated to be true or false. A condition consists of an attribute, an operator such as greater-than or equal-to, and a value. It can be read as `If - `*`system condition`*` - `*`compared to`*` - `*`value`*` - is true`. An example of a situation is `IF - CPU usage - > - 90% - TRUE`. The expression `CPU usage > 90%` is the situation condition.

State A state refers to the severity of the situation event: critical, warning, or informational. It is indicated by a colored event indicator, state is set in the Situation Editor, and can be different for different Navigator items.

Status The status is the true or false condition of a situation.

View A view is a windowpane, or frame, in a workspace. It may contain data from an agent in a chart or table, or it may contain a terminal session or browser, for example. A view can be split into two separate, autonomous views.

6.2.2 Pure versus sampled events with impact to the IBM Tivoli Monitoring V6.1 console user

The concept of a pure event can be understood as a stateless event. There is no recognition of a state of a resource in a pure event. An easily understood example is when an event is read from a Windows log file.

In the case of the log file monitor, the agent is merely matching lines from the log that it has been configured to forward to the monitoring server. The agent is not keeping track of anything to compare events, and there is no evaluation other than matching.

There is also no concept of an interval when building a situation to detect pure events, although there is some configuration of a time interval possible for most agents that detect pure events in most cases. However, that is for all operation of the agent and not on a situation-by-situation basis as with sampled events.

The sampled event, in contrast, has a state. The current state of the resource at sample time has a value and a state against which it is being measured. If instead of reading the log for an event we evaluated the current status of the storage application process for up/down, this would be a sampled event.

We evaluate the status (what is it) and compare against some criteria (up/down). When the monitor determines that the criteria have been met, the sampled situation becomes true and thus appears on the event console. When it is resolved (or no longer true), it is false.

When a pure event comes to the console, it is there until acted on by a human operator (if you are managing the events from the console) unless you include an UNTIL setting to expire it at a later time.

In contrast, when a situation reports a sampled event and it comes to the console, it is there until the conditions of the situation change that would result in it going back to false. Sampled events cannot and should not be closed by the console, by IBM Tivoli Enterprise Console, or by the UNTIL tab in the situation editor.

A small business support team (or any organization without an enterprise view tool) using the events console would need a view created in a workspace for the console operators that displays only pure events for events they are required to act on. A second view might display sampled events to alert the console operator to the fact that they cannot close the events, but might want to investigate situations that are currently visible in that view. Figure 6-1 on page 208 illustrates such a console.

This is done via the creation of a custom workspace for the console operator.

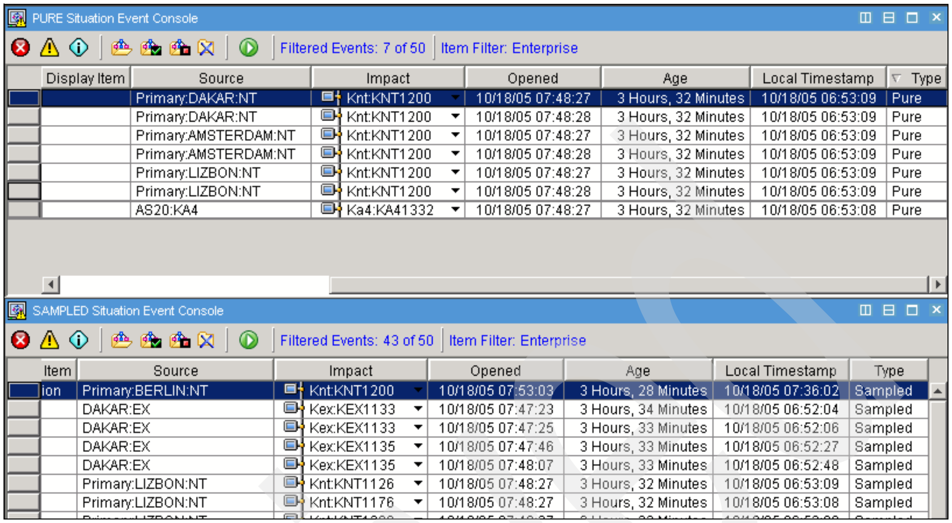


Figure 6-1 A custom workspace with separate pure and sampled event views

> **Note:** IBM Tivoli Monitoring V6.2 will provide a new event console called the common event console. See 6.4.2, "Common event console" on page 231 for more information on this new and much improved console.

6.2.3 Pure versus sampled events and customer impact

If you are an enterprise customer with a central event console such as IBM Tivoli Enterprise Console or IBM Tivoli Netcool/OMNIbus™, pure events are presented and treated in a way that is most closely related to the events that you had received from the Tivoli Enterprise Console Logfile Adapter, the Windows event log adapter, the SNMP adapter, and other related adapters. There is no schedule for the events. They arrive according to the logic of the agent that is sending them. The sampling, in that sense, is done at the agent's discretion and may vary from agent to agent.

Note: The information presented in this chapter for Tivoli Enterprise Console is also applicable to IBM Tivoli Netcool/OMNIbus. In order to send events to IBM Tivoli Netcool/OMNIbus IBM Tivoli Monitoring V6.1, you need to install a TEC Event Adapter probe on a system and modify the IBM Tivoli Monitoring om_tec.config file to point to the server running the probe. The fields to be modified in the om_tec.config file are:

<tme10tecad probe host> = hostname of where the Tivoli EIF Probe was installed

<tme10tecad probe port number> = is the probe listening port (default 5530) from install

Also see 6.4, "IBM Tivoli Monitoring V6.2 enhancements for event integration" on page 229 for the new functionality in the event management integration area in IBM Tivoli Monitoring V6.2.

What to do with these events is easily recognized and fits within our traditional notion of enterprise monitoring. Integration with IBM Tivoli Enterprise Console can result in pure events creating IBM Tivoli Enterprise Console events.

Important: It is permissible for pure events to be closed by rules from IBM Tivoli Enterprise Console. It is not advisable (although currently there is nothing in our evaluated version of the product that disables it from doing so) to close sampled events. As will be mentioned later, this causes problems as the event is closed while the situation remains true and it does not recover until the situation becomes false and then true again.

Although one strategy might be to not generate an IBM Tivoli Enterprise Console event for a sampled situation (not forwarding the events to IBM Tivoli Enterprise Console), this hardly is a maintainable strategy in an enterprise environment, especially because sampled events are probably as common as pure events (or more so).

For sampled events, it is less clear for an enterprise customer (with IBM Tivoli Enterprise Console or another enterprise console product), especially if the intent is to try to use the IBM Tivoli Monitoring V6.1 event console for some adjunct purposes to IBM Tivoli Enterprise Console. The questions start to rise: What is our policy for exploring these sampled situations? When do we decide that a sampled event becomes an IBM Tivoli Enterprise Console event or trouble ticket?

Note: If the decision is made to not use the IBM Tivoli Monitoring V6.1 event console as some adjunct tool to an existing Tivoli Enterprise Console implementation (that is, not installing the IBM Tivoli Enterprise Console event synchronization), the point is moot and all sampled situations should send IBM Tivoli Enterprise Console events and then be set to expire in the UNTIL tab (for pure). You *do* have to install the BAROC file for IBM Tivoli Monitoring V6.1 at the IBM Tivoli Enterprise Console server. You must set up the monitoring server to use the IBM Tivoli Enterprise Console event integration facility so that the events are sent to IBM Tivoli Enterprise Console.

If no one looks at the IBM Tivoli Monitoring V6.1 console, the current status of the situation is not an issue. At that point, two-way communication between IBM Tivoli Enterprise Console and IBM Tivoli Monitoring V6.1 is not required or desired. The one-way communication delivers the events.

For customers who choose to use the IBM Tivoli Monitoring V6.1 console in an enterprise environment, we can offer the best practices from some OMEGAMON XE customers. Many who used the OMEGAMON XE product chose to implement some logic for sampled events where possible. This logic is a suggestion about how you could choose to deal with these sampled events.

Your organization should explore the concept of these events with the administrators of the application and systems involved in the sampled events to decide whether this is the appropriate course of action. It is possible that by the time the sampled event indicates an issue and the person arrives on the scene, the sampled event could have become false again. They need to understand how the product functions in order to reconcile the current state with the fact that they were paged or got a trouble ticket or whatever is the action. We discuss using the console as an evaluative tool later in this chapter.

These steps outline the suggested best practice for dealing with sampled events:

1. When the sampled situation becomes true, attempt to resolve the issue via automation. You will not want operators to try to resolve these issues (as indicated before) until you are sure that the sampled situation will not become false within your tolerance limits.
2. Via policy, wait for the next sample to determine whether the automation has resolved the situation. Even if the event that the situation addresses has been resolved, it will not make the situation false until it samples again.

   To make this most efficient, the time between samples should allow for the automation to have a chance to resolve the issue. In other words, if the automation will require stopping and restarting a process and it takes x seconds for the process to recycle, the sampling interval should not be less than x seconds.

3. If the situation has not become false after the next iteration, you will want to take action (such as generate a trouble ticket or open an IBM Tivoli Enterprise Console event).

If automation is not possible, you want the policy to notify someone immediately so that a resolution process can be started as soon as possible (according to agreed response times negotiated in your Service Level Agreements.)

6.2.4 Identification of monitoring need

This identification of a need can include things such as recognizing the need for application monitoring for new applications, improving existing monitoring solutions through the introduction of newer, more sophisticated monitoring, or correlation among existing known monitoring items. This identification should identify the criticality of the request and the impact of the situation that this monitoring is to address in order to prioritize development activities across the available resources.

One fairly straightforward way to discover the need for monitoring is through the ITIL concept of a Problem Management function within your organization.

When a situation occurs that is of enough significance in your organization and that could be detected via some technical means, the Problem Management function should request via the application or system owner that some mechanism be created to proactively identify and resolve the issue, identify it and alert the required parties, or take some automated action to attempt to resolve the issue before it becomes an incident.

We advocate using the complete ITIL Service Level Management process set including Help Desk, Incident Management, Change Management, and Release Management, which are processes that should be observed in the monitoring space and should be recognized as a part of this activity.

The second way that the need for monitoring is discovered should be through the requirement for application management by the organization. All new developed and purchased applications in the environment should be monitored as a part of systems management to the level deemed necessary for the importance of the system or application being monitored.

In the case of “off-the-shelf” software, the vendor should be able to provide error conditions for trapping and advice as to which conditions are most critical. For developed software, the developer should build the software with monitoring in mind. This means that alerting to possible problems prior to failure should be a function of the application.

Monitoring that is merely reactive (identifying post-failure) can never be proactive in assisting the administrators of the application in preventing failure. Process-down monitoring is important, but far more valuable is a set of alerts to recognize that conditions exist that may precipitate or precede a failure.

> **Note:** As an example of this idea, would you rather have a tornado warning given to you or just wait for it to hit your house to react? As sure as you are of your answer to this question, it is very common for administrators to say to you "don't bother me until it actually goes down." While this does assure them that there is something they can do, it comes at the expense of the service which at that point is compromised.

In general, you will want to go through a planning session to design your monitoring and dashboard views. The following approach is advocated.

Define the objectives of the solution. Make sure to state the business objectives. Define the scope of the solution. Identify the most critical applications.

Examine the existing technology infrastructure to see what needs can be met with the existing solution. If parts will not be met by the existing solution, describe the additional monitoring pieces that have to be created. Identify the information and functions you need to design for the user communities.

Additionally, a gatekeeping function should exist for the team who develops or implements the monitoring solution. This gatekeeper might be the manager of the team or a business analyst role within the team.

The five advocated areas are the criteria that should be used by the gatekeeper to make sure that the resources that are required for building, maintaining, and operating the monitoring solution are appropriately managed by this request (and the other requests that have been or will be made.)

The criticality of the request helps drive the schedule for design and implementation of the monitoring. This is also determined by the impact of the outage or failure that this monitoring addresses and the likelihood of failure. Mission-critical systems and applications should be monitored more than and differently from systems whose impact is low. If an event happens every day and it is related to a mission-critical system, this event should be prioritized much higher for monitoring development than something that has never been known to have occurred, even though it is related to a significant event.

6.2.5 Identify the target audience

Every event that will be monitored in the enterprise should have some known value and an agreed-on course of action. It should be agreed to by all parties from the operations person who views the event to the administrator who receives the problem management ticket and is expected to act on the event. The target audience for this event must fully accept and own the implications of this event; otherwise, monitoring for this event is a waste of time to even build.

Even so-called “best practices” events that may be implemented by a solution provider or the vendor of the monitoring solution are not best practices in your organization if they fail to achieve recognition as such. For example, the administrator at Company A may fully believe that Event X (an out-of-the-box, solution-implemented event) is very important and act on it every time it occurs. The administrator at Company B, who does not, promptly disabled the monitor after it alerted him twice in one day. Both experiences and activities may be valid, depending on the needs of the organization and their system administrator’s experience and perceptions.

As an example, a monitor is created that monitors the level of activity of the computer system by examining the queue lengths, processor activity levels, and so forth. The event is generated and results in a problem management ticket. Per our advocated item, then first, the administrator of that system must agree that activity level is something they want to be alerted about and indeed it is a problem that they act on.

If the administrator who receives this event is not willing to act on the event (knows the value in being notified and agrees to act on it), the reception of the event is a nuisance. In environments where the events were not agreed to be important and understood by all parties, there is no clear value in the monitoring solution that can be realized or recognized by the organization.

Such an organization usually has many monitoring solutions, each owned by a different faction of the organization, and each believed to deliver the value that the faction seeks, regardless of the ability of a single tool to meet the needs of the entire organization.

6.2.6 Identify and refine the possible events list

The first step of design is to map the monitoring, managing, and integration facilities to the discovered needs. When you have this mapping design, move into organizing the presentation of the information. What business views, links, workspaces, and reports are required and for which communities?

Plan the event management and automation to solve monitoring needs, always keeping in mind the changes in volume of events and the impact on the support and operations areas. Experts from each area should be pulled in to help with this design.

For every event that is generated by every application and every system in the environment, there likely exists some relationship with other events or occurrences in the environment. Consider a request to monitor this event:

```
EVENT_ID 20503        The application WYSIWYG has failed with error -9.
```

This can be seen as a possible mistake to alert when it is found to be a symptom of the WYSIWYG application on server Y failing and server X being unable to contact it. You cannot rely on the vendor to tell you these relationships and often they will be found only with experience. Even though picking up these alerts for the purpose of validation can be a worthwhile endeavor, it should be understood what the event means so that an appropriate correlation can be made and the action taken to resolve the correct issue.

As important as identifying the specific indicators is to determine the approximate event volume in the environment. It is important to make sure that your activity does not overwhelm the monitoring solution that exists (an excessive number of events or time interval) or require significant resources beyond a sustainable or desirable level.

The monitoring team usually works with the team supporting the enterprise tools to assure that their desired monitoring solution is technically feasible and supportable. As an example, an application developer who feels that event 20503 must be trapped to correlate might need some help to understand the impact of doing so if they throw these events at a rate of 100/second. In this case, doing so requires some changes to the architecture and possibly other components.

6.2.7 Meet with the target audience for approval

Prior to implementation of the monitor in the enterprise environment, all activities listed above should be completed. This involves such sign-offs as:

- A defined business value has been stated.
- An owner has been identified and they agree to the ownership.
- An appropriate SME has been identified and consulted and concurs with the request as a valid measurement.
- The action (possible) that is taken in response to the reception is known.
- The actor responsible for the action is identified and is aware of the event and agrees to act on it when notified (could be automated, if possible.)

At this point, the monitor does not exist. Prior to the allocation of resources to develop the solution, all of these sign-offs must be complete. Failure to do so will likely result is dissatisfaction and the inability to recognize or achieve the value of the solution.

6.2.8 Create, test, and implement the monitor

First and foremost, separation of duties should be utilized: the creator of the monitor is *not* the tester, the implementer is not the tester, and so on. All of these parties should sign off on their responsibilities for this monitor.

Some small amount of work is likely to remain when the solution becomes ready for production and for final user acceptance. This is likely to be customization to fit final or late identification of needs. Remember to develop documentation and training for the users.

> **Important:** A development environment is required to develop the monitors. A test environment is required to test the monitors. The way that the product functions, you are making changes in the production environment whenever you are working to create monitors if you are not working on an isolated system. Doing so invites problems at some point and is definitely *not* recommended.
>
> The requirements for your test and development environments are not high and could be met with virtual machines. If necessary, test and development could be one and the same; however, it is not a best practice to have them be the same. If they are different, it also gives a test to the import of them into another environment such as you are going to do in production.

Security in the production environment should be locked down to the point that only a very select group can make changes to situations running in production. Any changes that must happen in the production environment must be approved by the change control process or through an emergency change proviso.

Release management should also play a role in development. As you build your monitors, you should export your work and save it in a directory structure that enables you to version your work. That way, you will be able to restore the work to some previous point if you should accidentally introduce some problem at a future point to a known good save.

The facility that enables you to export the situations from your test environment to reimport later or to import into your production environment is a set of command-line commands:

```
tacmd viewSit {-s|--situation} SITNAME [{-m|--system} SYSTEM]
[{-e|--export} [FILENAME] ]

-s|--situation: Specifies the name of the situation to view.
-m|--system: Specifies the managed system to view the situation definition
for.
```

```
-e|--export: Exports the situation definition to a file of the name
specified.
```

The reimport of the created XML file into the test environment later or into the production environment is done through the facility to re-import these XML files:

```
tacmd createSit {-i|--import} [FILENAME]

-i|--import: Specifies the situation definition to import.
```

In the next section, we explore an example of this in detail.

6.3 Building the monitoring

In building the monitoring within IBM Tivoli Monitoring V6.1, there is a choice between the situation editor and the command line. First we discuss building situation monitors in the editor and then the GUI, because when we look at the GUI, we can see the formula that we would be building with the command line. It will be easier to understand the options if we explore the situation editor first.

Building a complete situation involves the following considerations:

- Naming the situation
- Selecting attributes for comparison
- Editing the formula
- Selecting targets of distribution: MSL, systems, or both
- Writing expert advice to explain the situation event
- Setting the action
- Using the Until tab

6.3.1 Naming the situation

The situation editor is launched from the Tivoli Enterprise Portal browser or desktop by issuing the key sequence Control-E. The Situation Editor window (Figure 6-2 on page 217) appears and provides a list of situations in your monitoring environment.

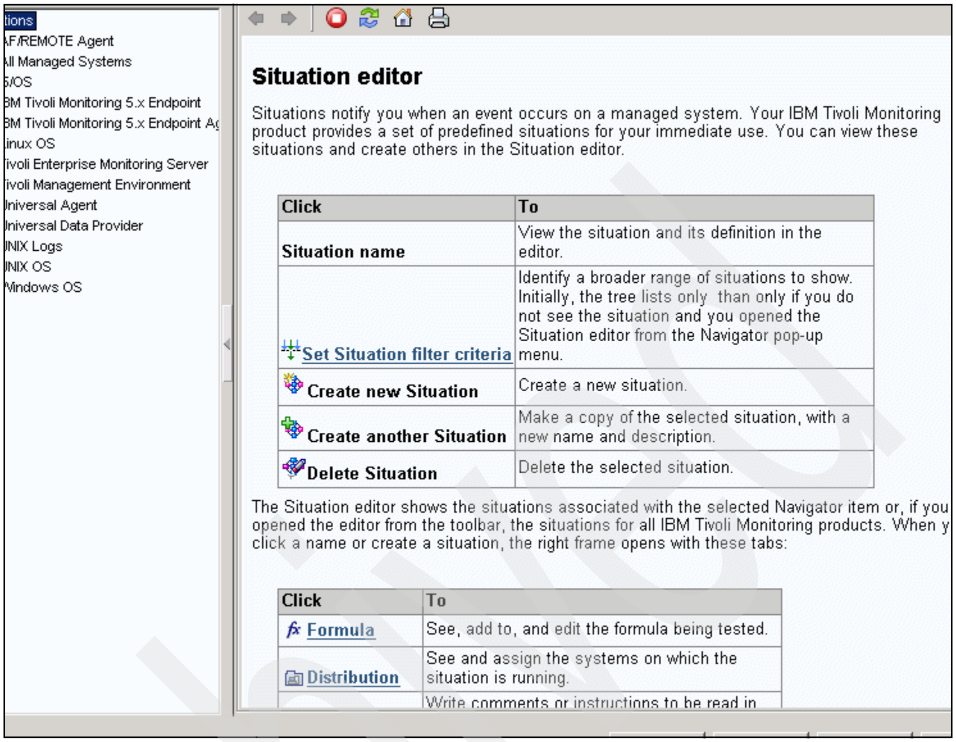


Figure 6-2 The Situation editor screen

It is useful to explore the types of situations that exist out of the box.

> **Tip:** It is possible to reach the situation editor in several ways from the Navigator other than by using Ctrl + E. If you right-click the Enterprise item and select the situation editor icon, you see the situations that were distributed to the monitoring server. If you right-click a system item to see the situation editor icon, you see all situations that were distributed to that managed system. If you right-click an agent item (or if the agent has subagents), you see all situations that have been distributed to the monitoring agent. At the attribute level, you can see situations distributed to that managed system but only those that were created with related attribute groups.
>
> With Ctrl + E, you see the realm of options in the environment. You are able to build situations that are not associated with a Navigator icon by doing this.

Scenario 1

In our first scenario, we have been charged with coming up with an alerting mechanism to indicate that a drwtsn32 alert has been generated on a Windows system. A mission-critical application for our environment has been found to be suffering from some problem that results in this failure. Although we would prefer to have the application programmers create a more proactive way to alert us to the problem, this is the only existing way that the situation can be recognized (in our scenario). Additionally, we understand that other key applications might fail and generate these events. We will refine this concept as we move through this discussion to add new features to this monitoring solution.

The first step is to name the situation (see Figure 6-3). We consider that the Windows OS agent will be used to make the monitoring happen. Therefore, we click the plus icon to expand the Windows OS category. We then click the Windows OS title for the category and right-click the icon above to create a new situation.

Because in our environment we want to generate an IBM Tivoli Enterprise Console event, we name the situation with _CRITICAL on the end of the name. This works with the IBM Tivoli Enterprise Console integration of IBM Tivoli Monitoring V6.1 to automatically set the severity of the event to critical. Our other options include using _WARNING or to manually map the criticality of the event in the mapping file. We might choose Drwtsn32_CRITICAL.

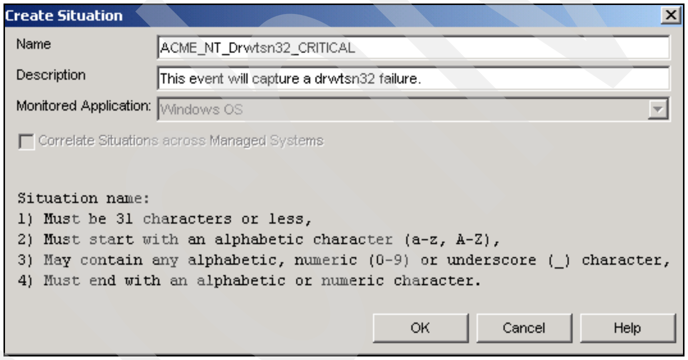


Figure 6-3 Naming the situation

> **Note:** In IBM Tivoli Monitoring V6.2 you will be able to set the event severity in the Situation Editor, so appending the severity level to the situation name will not be necessary. See 6.4.1, "Sending events to multiple destinations and setting event severity" on page 230 for a brief description of this functionality.

For our first iteration of this situation, we simply look for the existence of a process. When a Dr. Watson alert is caused, a pop-up appears on the screen and an entry (multiple-line) is made in the log. While the pop-up is present on the screen, a process appears in the process table that we can identify. The method that we choose for identifying the problem is to look for this process. Obviously, there are some flaws in the methodology but we use it for the purpose of this example. We will continue to refine our sophistication throughout the examples.

When we name the situation, certain practices will be useful. First, look at the situation names that are defaulted with the product. Note that the NT alerts begin with NT_. It is helpful to make all of your situations stand out by naming them with the name of your company at the beginning.

An example format for a standard is:

```
<Customer>_<Component>_<Description>_<Severity>
```

Revising our name under this format, if our company was Acme and the component we are interested in is the NT component, we might name it ACME_NT_Drwtsn32_CRITICAL. The Description field that you populate in this dialog will populate the fly-over text of the situation in the navigator.

You might also name the situation with a responsibility group in the name. SYSADMIN_NT_Drwtsn32_CRITICAL might be a choice, as might NASYSADMIN_... (North America) or EURSYSADMIN_... (Europe) for other examples. The reason for doing this might be that your organization cannot standardize on situations (highly recommended as one size can fit all if you can make them agree.) Naming them with the same prefix will make them appear in the situation navigator in alphabetical order grouping as well.

The example format might be:

```
<Responsibility>_<Component>_<Description>_<Severity>
```

You are limited to 31 characters for situation names, so you might have to create abbreviations. For example, instead of ITSO_REDBOOKS_PUBLICATIONS_NT, you should choose ITSORBP_NT. Also, the Tivoli Enterprise Console integration _CRITICAL and _WARNING can be abbreviated to _CRIT and _WARN if necessary to meet the 31-character limit.

Tip: If you save the name by closing this dialog box, you will be forced to copy the situation if you want to rename it. The function of duplicating a situation (and other parts of this GUI use a similar approach) is called *Create Another*. This menu option essentially creates a duplicate copy. The first thing it does in the situation editor is provide you with a dialog box to name the copy (and thus you can rename). You will then have to delete the version with the name you wanted to change.

This is a particularly good reason to develop your naming standards *before* you write the first monitor situation. Otherwise, you will be forced to create duplicates for all existing custom situations in order to rename them, then forced to delete all the incorrectly named copies. This can involve a significant amount of work for someone later when your standard is created.

6.3.2 Selecting attributes for comparison

After we have finished naming our situation, we select attributes for comparison in our situation. Figure 6-4 shows that the named situation has appeared in the list of situations. We click **Add conditions** to gain access to this list of attributes.

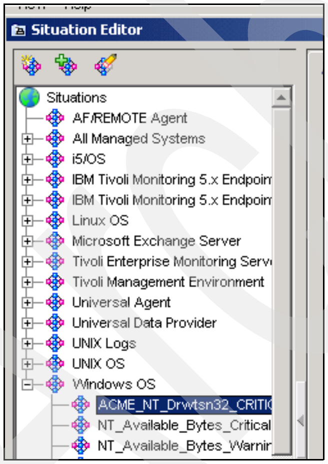


Figure 6-4 Our situation name appears in the situation editor list

Note: If the name of the situation is incorrect or it is located in the wrong place in the situation list, delete it and create it again in the correct location.

We have previously discussed that we will look for a process name in the process table. Therefore, we select the attribute group for NT_Process and the individual attribute as Process Name (Figure 6-5). For the purposes of our simple example, this is sufficient.

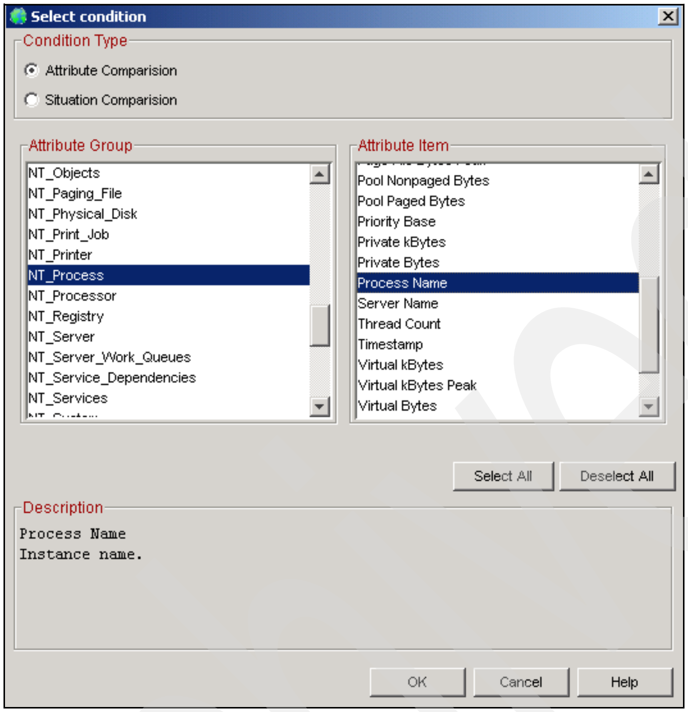


Figure 6-5 Attribute comparison selection

When we click **OK**, the editing window appears.

6.3.3 Editing the formula

Important: You are limited in the number of attributes that you can select from this list by the size of the formula that is created with the selection and creation of criteria for those attributes. The number of attributes that you can have used to be a hard stop of 10 in the OMEGAMON product. That restriction has been lifted and replaced with a status bar that indicates the current size of the formula against the total size possible. The current formula limit is 1020 characters.

It is still recommended, however, that if you have multiple situations using the same attribute group, that you limit the number of attributes to 10 for performance reasons.

Figure 6-6 shows the formula editing window. We will refer to the labelled points in the figure to explain what you will need to know.

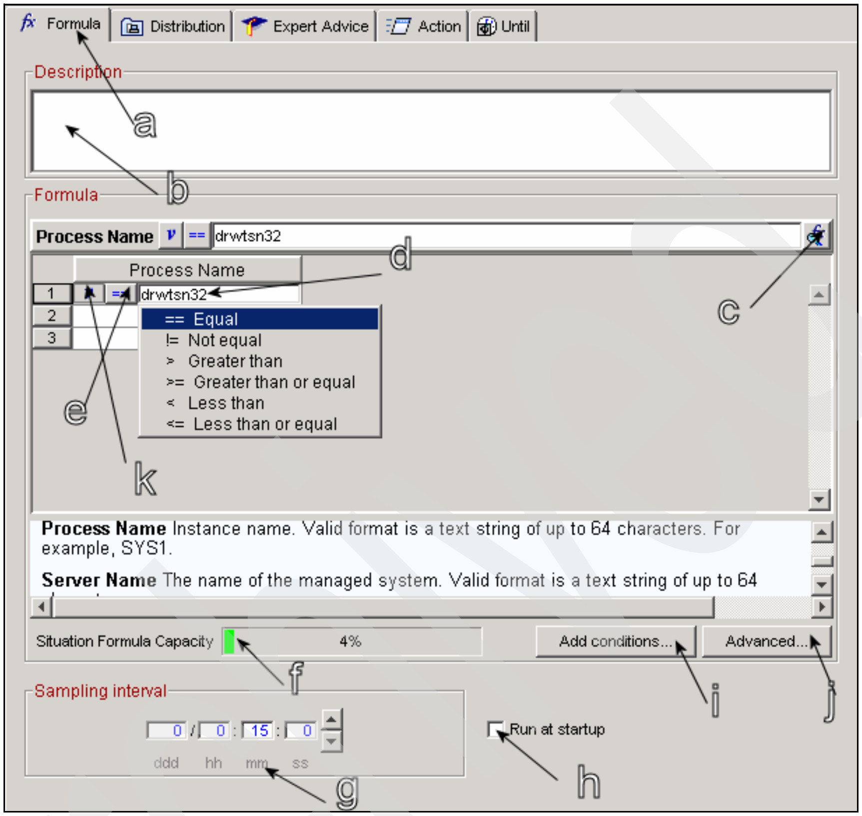


Figure 6-6 The formula editing window

a. Note that the formula tab is selected.

b. Type a short description here. You are limited to 31 characters.

c. This button will show the completed formula.

d. In this cell we type the name of the process we are going to match.

e. This button brings up the equality choices list (not equal, equal, greater than, and so on)

f. This shows the formula capacity. The total limit of characters is 1020 for the formula and this bar shows your percentage of that in the formula. Refer to button "c".

g. The sampling interval is set here. By default it is 15 minutes. See the previous discussion about considerations in item 2 on page 210.

h. Run at startup is checked by default. If this box is checked, as soon as you save the situation it will start to run to all subscribed systems.

i. Add conditions. We have already used this. This brings up our attribute selection dialog.

j. Advanced. This brings up two more options. One is persistence. The other is called display item.

k. Functions. The list of available functions varies by the type of attribute. The default is value of expression. For a complete list of choices, refer to the online help or to *IBM Tivoli Monitoring, Version 6.1.0, User's Guide,* SC32-9409.

> **Important:** The persistence setting requires that you wait a selected number of occurrences of TRUE evaluation prior to creating an event. In IBM Tivoli Monitoring 5.x terms, this is a choice of the number of occurrences; however, you are limited to zero holes. This means that you cannot be as sensitive to outliers. (If it happens nine times over the course of 10 sample cycles and the situation is measured as false on the 10th measurement, it would not alert you unless there are 10 in a row.)
>
> The display item choice opens a dialog that lets you pick from only certain attributes to continue checking for in the sampling and create other alerts. If it were possible, for instance, to have multiple Dr. Watson pop-ups on the screen (and thus multiple processes in the process table), you could use this dialog to cause separate events for each one on the system. The default behavior would only give you a single event in this scenario.

For our example, we have the process name drwtsn32, set our sampling interval to 30 seconds, and set it to run at startup.

6.3.4 Selecting targets of distribution: MSL, systems, or both

Next we look at the Distribution tab for our options. Figure 6-7 shows the Distribution tab.

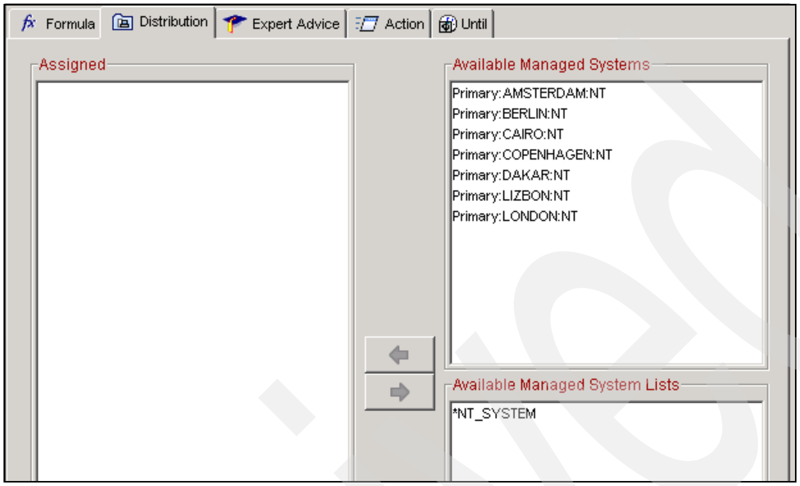


Figure 6-7 The Distribution tab dialog

The Distribution tab offers the choice of single systems or managed system lists. These managed system lists are groups of servers and are defined by the edit button. The MSLs that are predefined for you show up with an asterisk (*) at the beginning of the name. While the systems automatically go into platform-managed systems lists by platform, you will have to move the systems into MSLs manually.

In our example, we pick the MSL for *NT_SYSTEM so that all new NT systems added to the environment get this situation by default.

6.3.5 Writing expert advice to explain the situation event

The next tab in the Situation Editor is the Expert Advice tab (Figure 6-8).

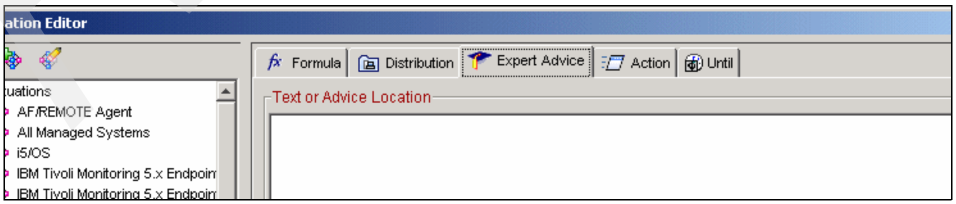


Figure 6-8 The Expert Advice tab

You can write text in this Expert Advice tab and it will display in the Expert Advice pane as text.

You can also write HTML code in this space. It will be handled appropriately. In this space, you can use the variable expression syntax exactly as you can in the link wizard, including input such as Example 6-1.

Example 6-1 Variable expression example

```
situation name            $EVENT:ATTRIBUTE.ISITSTSH.SITNAME$
monitoring server name    $EVENT:ATTRIBUTE.ISITSTSH.NODE$
managed system name       $EVENT:ATTRIBUTE.ISITSTSH.ORIGINNODE$
display item              $EVENT:ATTRIBUTE.ISITSTSH.ATOMIZE$
global timestamp          $EVENT:ATTRIBUTE.ISITSTSH.GBLTMSTMP$
local timestamp           $EVENT:ATTRIBUTE.ISITSTSH.LCLTMSTMP$
status                    $EVENT:ATTRIBUTE.ISITSTSH.DELTASTAT$
```

If you want to redirect the user to an existing Web page, simply put the URL in the pane without any other text:

```
http://www.cisco.com/warp/public/78/drwtsn32.shtml
```

The browser will appear and launch to this page. You also can use variables if you want to launch the Web page in the context of some variable, as in passing information to a search engine.

> **Important:** Due to the limitation of 512 characters, you will likely want to use an external Web file as the product does for the built-in situation expert advice.
>
> This does not allow you to use the variable substitution. If you want to use variable substitution, you must stay within the 512-character limit.

This procedure could be used to build expert advice using an external HTML file:

1. In the Expert Advice box, enter the same information that exists in one of the default situations (Figure 6-9):

   ```
   ADVICE("knt:"+$ISITSTSH.SITNAME$);
   ```

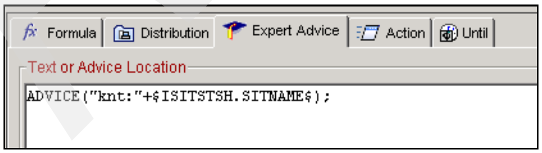


Figure 6-9 Populate the Expert Advice tab like this

2. Copy one of the IBM Tivoli Monitoring V6.1 expert advice HTML files (Figure 6-10) and create one with the name matching your situation. In our case, we create `ACME_NT_Drwtsn32_CRITICAL.htm`. It should be placed in the same directory as the other HTML files.

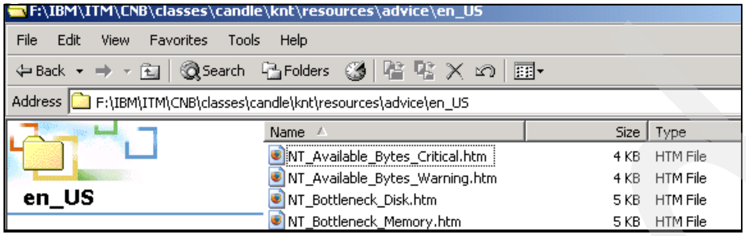


Figure 6-10 Default location of expert advice files on Windows TEPS server

3. Edit the expert advice file to contain your desired expert advice information. Save the HTML file.
4. Click **Preview** to see how it looks (Figure 6-11).

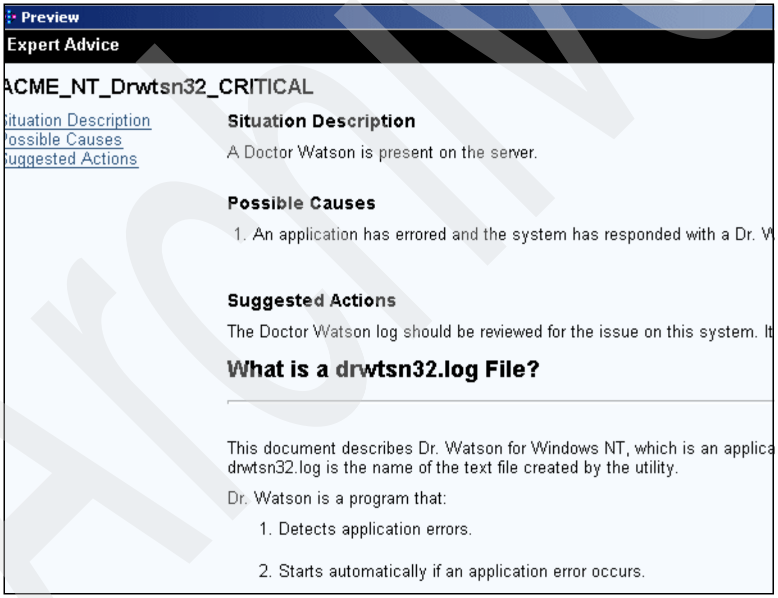


Figure 6-11 Our custom expert advice for the ACME event

6.3.6 Setting the action

Next, set the actions for when the situation becomes true. The other options in the Action tab are described following Figure 6-12.

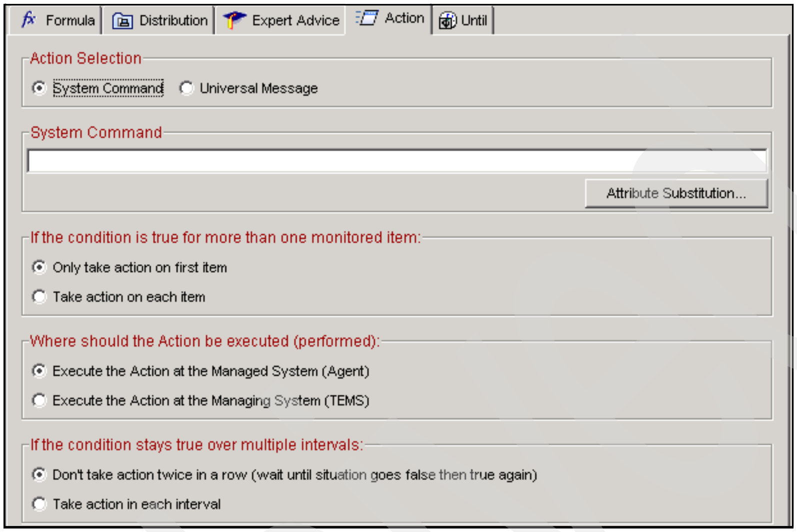


Figure 6-12 The Action tab

The Action tab offers two main options: System Command and Universal Message. Figure 6-12 shows the System Command choice. Figure 6-13 shows the changed part of the screen if Universal Message is selected.

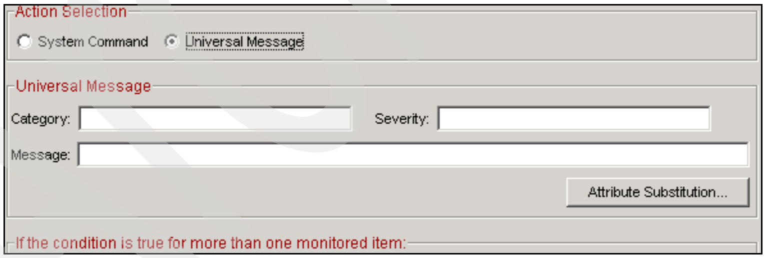


Figure 6-13 Action tab with Universal Message selected (the differences only)

Under the System Command selection, any system command can be issued. There is a button for attribute substitution that enters a variable so that when the situation is true, appropriate values will be used.

As an example, the system command might be something like this:

```
net stop oserv
echo "There is a serious error on the system" | mail root
```

Of course, the system command must be platform-appropriate. An example of using the variable substitution is to have the process name be generic and thus have a more multipurpose situation. Our next example explores this further.

Other options on the Action tab enable you to customize the response, such as:

- Only take action on first item.
- Take action on each item.

The differentiation is where possibly there could be multiple matches to the same attribute condition check (thus returning a result of multiple rows of data). The selection of Take action on each item will cause your system command to be executed for each line. We also look at this in our next situation.

The next group of actions on the tab are to select where the action is issued: on the agent machine or on the monitoring server machine. This might be useful if, for example, you want to build a log file of these entries in a single file on the monitoring server:

```
echo " &NT_Process.Timestamp There has been a Dr Watson on
&NT_Process.Server_Name" >> /logs/DrWatsonRollup.log
```

The final section controls how many times you might run the system command; that is, whether it issues the command twice in a row or waits for the situation to evaluate false again (symptom gone), or if it should issue the command at every sample that is true.

Important: Pay attention to sample interval here (as previously discussed). For example, if you know that a command takes more than 30 seconds to complete, it does not make sense to set your sample interval to 15 seconds and then pick fire every time, because it will issue the command at least two times before the first attempt has a chance to be successful.

Under the Universal Messages selection, you define a universal message for the universal message log. You define the category that the message should fall under, such as Non-essential or SiteCritical. Type a one-word term of up to 16 characters to categorize the message.

Severity is the importance of the message, such as High or 1. Type a one-word term of up to eight characters that conveys the severity of the message. Message is the text to display in the UML when the situation occurs. Type a message of up to 245 characters. You can include an attribute value in the message by selecting Attribute Substitution.

6.3.7 Using the Until tab

The Until tab contains two separate functions for a single purpose. The purpose of this tab is to designate when to close the event automatically. If you never want it to be closed automatically, do not choose anything on this tab.

The first function is to close the event when another event also becomes true. The situations available are for the same type of agent that this situation uses or that use the common attributes. The second function is to close the event at some time interval.

Important: Do not use the Until tab for situations that are sampled. The sampled events, as you may recall, should not be manually closed. The result of having them close is that the situation goes to a status of closed. It is not until the situation would recover (become false) and then suffer again (become true) that you will see this event again.

As an example, we set the NT_Missing_Process to look for Notepad on our system. Then we set an until for 30 seconds. With Notepad not running, in 30 seconds, the event closes and soon after disappears. Although our “critical” situation is still technically existing, the tool cannot alert us until somebody starts Notepad again and then closes it.

On the other hand, closing a sampled event using another situation is better than embedding situations or time predicates. For example, you may want to write situations that will alert the support people of a condition only during prime time. The best way to code such a situation is to say "UNTIL Off_Hours" where Off_Hours is a time based situation that becomes true at those time when the support people do not want to be alerted of some problems.

6.4 IBM Tivoli Monitoring V6.2 enhancements for event integration

IBM Tivoli Monitoring V6.2 provides several enhancements in the area of event management, such as sending events to multiple destination and setting the event severity in the Situations Editor and Common Event Console.

6.4.1 Sending events to multiple destinations and setting event severity

If your monitoring environment includes the Tivoli Enterprise Console Event Server or the Tivoli Netcool/OMNIbus Object Server, and situation event forwarding has been configured on the hub monitoring server, you can forward situation events to either or both types of event servers. You can also send a situation event to more than one event destination.

Figure 6-14 on page 230 shows how to configure the event destination per situation. You can access the EIF tab during a situation creation or by editing it. You can set the event severity in the EIF Severity tab.

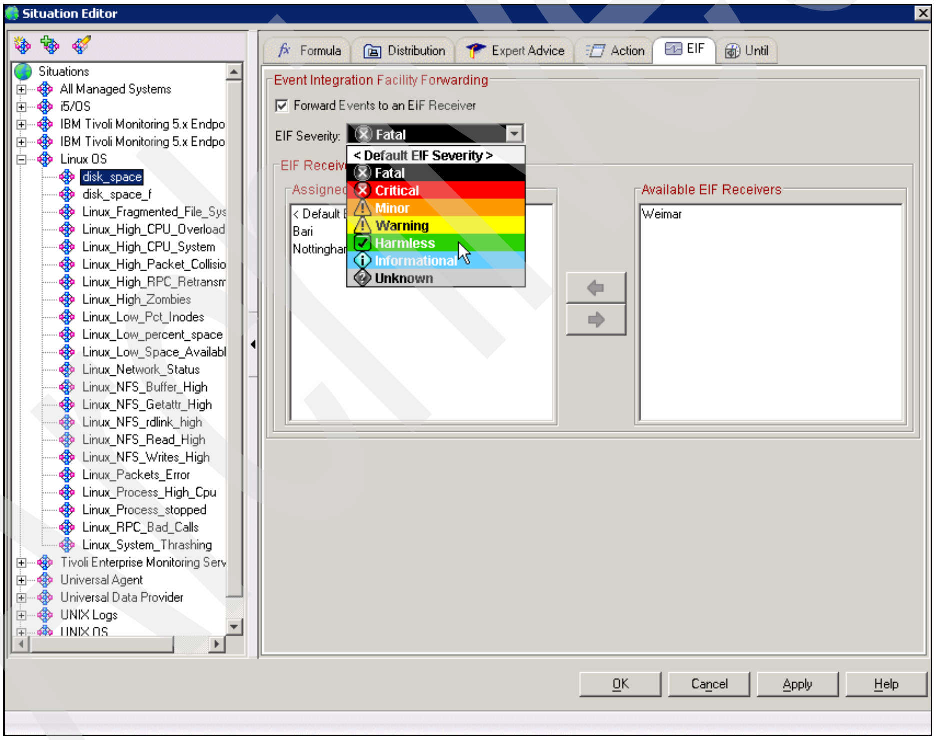


Figure 6-14 EIF configuration dialog

6.4.2 Common event console

The common event console is a Tivoli Enterprise Portal view that provides a single, integrated display of events from multiple event systems. In one table, the common event console presents events from the event systems, and users can sort, filter, and perform actions on these events. The following event systems are supported:

- IBM Tivoli Monitoring
- IBM Tivoli Enterprise Console
- IBM Tivoli Netcool/OMNIbus

A common event connector (frequently called a *connector*) is software that enables the integrated display of events from multiple event systems in the common event console. A connector retrieves event data from an event system and sends user-initiated actions to be run in that event system.

For example, if you perform an action on a Tivoli Enterprise Console or Netcool/OMNIbus event in the common event console, the associated common event console connector sends that action to the originating event system (Tivoli Enterprise Console or Netcool/OMNIbus) for execution. To have the events from a specific event system displayed in the common event console, you must configure a connector for that event system.

To give you an idea of what this functionality will be, we took the following screenshots from our test IBM Tivoli Monitoring V6.2 (beta at the time of writing) system. They illustrate how to configure TEC and OMNIbus connections for the common event console.

Figure 6-15 shows the Common Event Console-TEC connector configuration window.

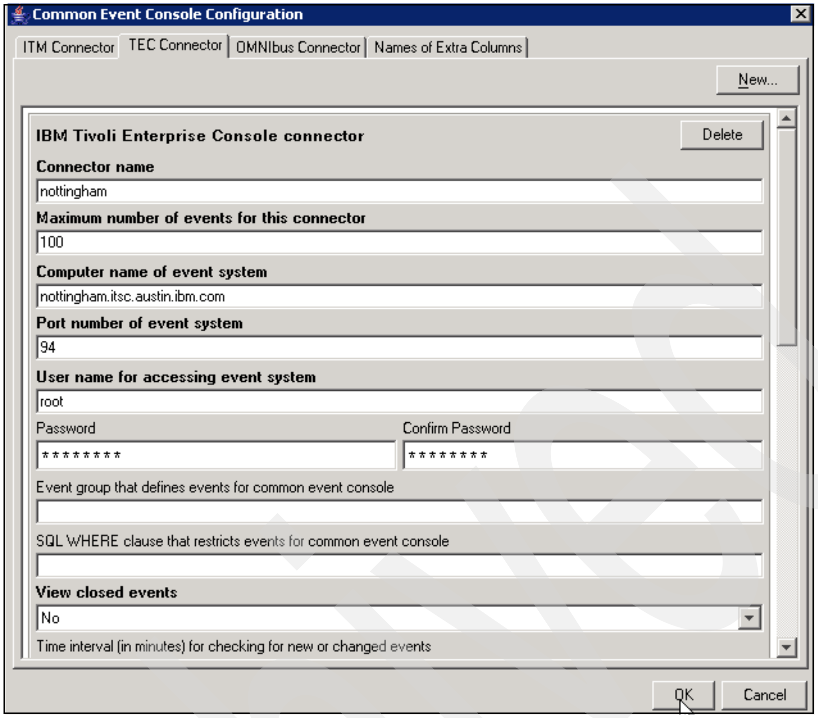


Figure 6-15   Common Event Console-TEC connector configuration

Figure 6-16 on page 233 shows the Common Event Console-IBM Tivoli Netcool/OMNIbus connector configuration window.

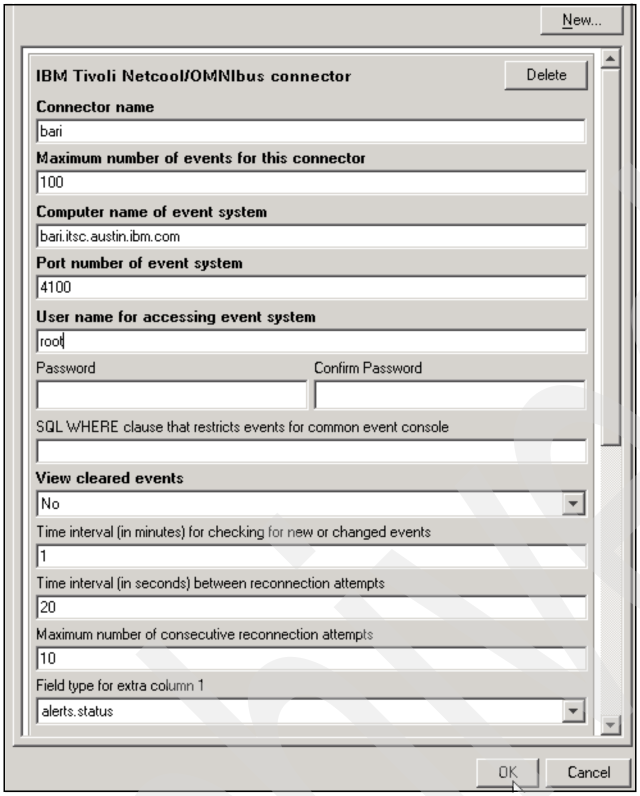
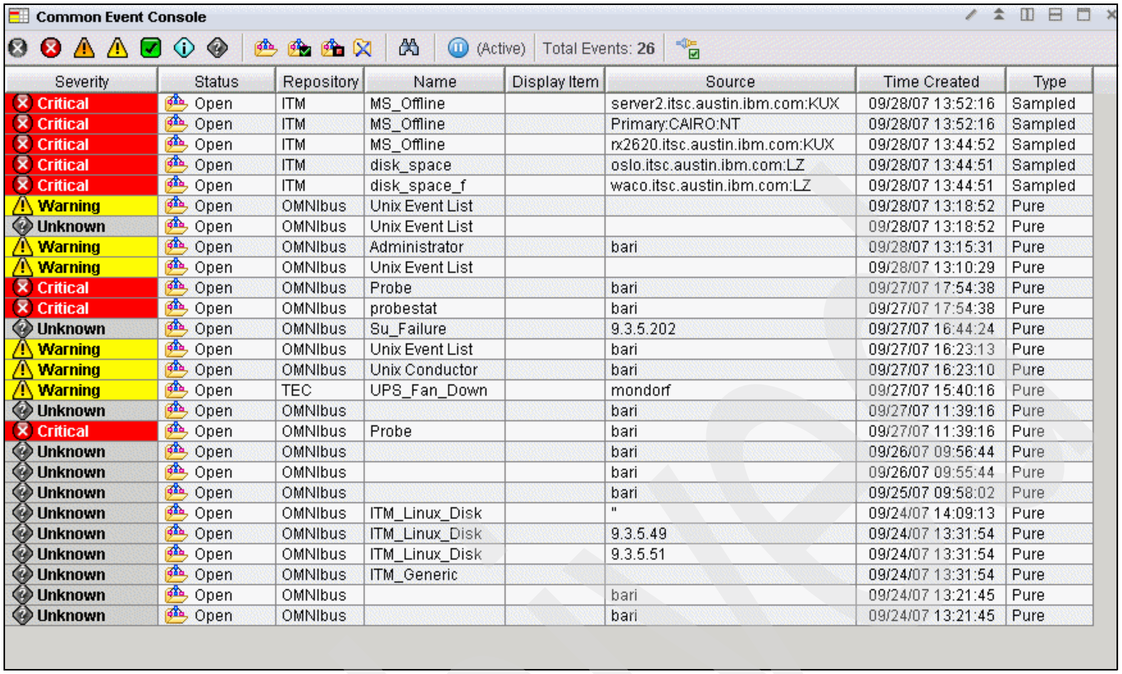


Figure 6-16 Common Event Console-IBM Tivoli Netcool/OMNIbus connector configuration

Figure 6-17 on page 234 shows the Common Event Console.



Common Event Console

(Active) Total Events: 26

Severity	Status	Repository	Name	Display Item	Source	Time Created	Type
Critical	Open	ITM	MS_Offline		server2.itsc.austin.ibm.com:KUX	09/28/07 13:52:16	Sampled
Critical	Open	ITM	MS_Offline		Primary:CAIRO:NT	09/28/07 13:52:16	Sampled
Critical	Open	ITM	MS_Offline		rx2620.itsc.austin.ibm.com:KUX	09/28/07 13:44:52	Sampled
Critical	Open	ITM	disk_space		oslo.itsc.austin.ibm.com:LZ	09/28/07 13:44:51	Sampled
Critical	Open	ITM	disk_space_f		waco.itsc.austin.ibm.com:LZ	09/28/07 13:44:51	Sampled
Warning	Open	OMNIbus	Unix Event List			09/28/07 13:18:52	Pure
Unknown	Open	OMNIbus	Unix Event List			09/28/07 13:18:52	Pure
Warning	Open	OMNIbus	Administrator		bari	09/28/07 13:15:31	Pure
Warning	Open	OMNIbus	Unix Event List			09/28/07 13:10:29	Pure
Critical	Open	OMNIbus	Probe		bari	09/27/07 17:54:38	Pure
Critical	Open	OMNIbus	probestat		bari	09/27/07 17:54:38	Pure
Unknown	Open	OMNIbus	Su_Failure		9.3.5.202	09/27/07 16:44:24	Pure
Warning	Open	OMNIbus	Unix Event List		bari	09/27/07 16:23:13	Pure
Warning	Open	OMNIbus	Unix Conductor		bari	09/27/07 16:23:10	Pure
Warning	Open	TEC	UPS_Fan_Down		mondorf	09/27/07 15:40:16	Pure
Unknown	Open	OMNIbus			bari	09/27/07 11:39:16	Pure
Critical	Open	OMNIbus	Probe		bari	09/27/07 11:39:16	Pure
Unknown	Open	OMNIbus			bari	09/26/07 09:56:44	Pure
Unknown	Open	OMNIbus			bari	09/26/07 09:55:44	Pure
Unknown	Open	OMNIbus			bari	09/25/07 09:58:02	Pure
Unknown	Open	OMNIbus	ITM_Linux_Disk		"	09/24/07 14:09:13	Pure
Unknown	Open	OMNIbus	ITM_Linux_Disk		9.3.5.49	09/24/07 13:31:54	Pure
Unknown	Open	OMNIbus	ITM_Linux_Disk		9.3.5.51	09/24/07 13:31:54	Pure
Unknown	Open	OMNIbus	ITM_Generic			09/24/07 13:31:54	Pure
Unknown	Open	OMNIbus			bari	09/24/07 13:21:45	Pure
Unknown	Open	OMNIbus			bari	09/24/07 13:21:45	Pure

Figure 6-17 Common Event Console

7

Deploying IBM Tivoli Monitoring agents in a large scale environment

This chapter discusses the deployment of IBM Tivoli Monitoring V6.1 agents in a large scale environment. It provides various best practices recommendations, depending on your environment.

The following topics are addressed:

- Choosing the most suitable method for deploying agents
- Agent depots
- Deploying the monitoring agents using the tacmd command
- Deploying the agents using Tivoli Configuration Manager or Tivoli Provisioning Manager

7.1 Choosing the most suitable method for deploying agents

The deployment of monitoring agents is an important consideration in a large scale environments. Prior to start deploying the IBM Tivoli Monitoring V6.1 agents, it is very important to identify the most appropriate deployment method for your environment.

Depending on your network design, bandwidth, firewalls, security concerns and the allocated time frame for deployment, you can use the product built-in features for deployment, or you can use a specialized software deployment product.

The built-in feature, the **`tacmd`** command, is a very powerful tool. You can use this command to deploy all agents in your environment. However, before choosing to use this method you need to be aware of the characteristics of this command. For example, you cannot use the **`tacmd`** command to deploy more than one agent simultaneously (by providing a list of servers).

A second consideration involves bandwidth control. This command does not have built-in bandwidth utilization control, so if the deployment is performed during busy work hours, it may impact the performance of other applications that are using the same WAN network, depending on the available bandwidth.

As an option, you can make use of a software deployment tool such as IBM Tivoli Configuration Manager or IBM Tivoli Provisioning Manager, which both have robust capabilities to perform massive deployment of software. Using Tivoli Configuration Manager or Tivoli Provisioning Manager, you can deploy of hundreds of agents at same time *and* control bandwidth utilization. You also have restart capabilities from where you left off, if network failures should occur.

Installing the agents via Tivoli Configuration Manager or Tivoli Provisioning Manager is discussed in 7.7, “Deploying agents using Tivoli Configuration Manager (or Tivoli Provisioning Manager)” on page 254.

7.2 Command line interface terminology

Before explaining how to install agents by using the **`tacmd`** command, this section defines command line terminology.

Agent or deployment depot The agent or deployment depot is an installation directory on the monitoring server from which you deploy agents and maintenance packages across your environment.

Bundle A bundle is the agent installation image and any prerequisites.

Agent A monitoring agent is a data collector. There are two types of monitoring agents:

Operating System (OS) agents These are other agents (referred to as application agents or non-OS agents)

System Monitoring agents are installed where you have resources you want to monitor. The agents and the operating system, subsystem, or computer they are running on are referred to as managed systems.

Node The node is the installation directory for all agents. The name of a node includes the computer where the OS agent is installed and the product code for the OS agent.

7.3 Agent deployment architecture

The agent deployment architecture, shown in Figure 7-1, consists of several entities.

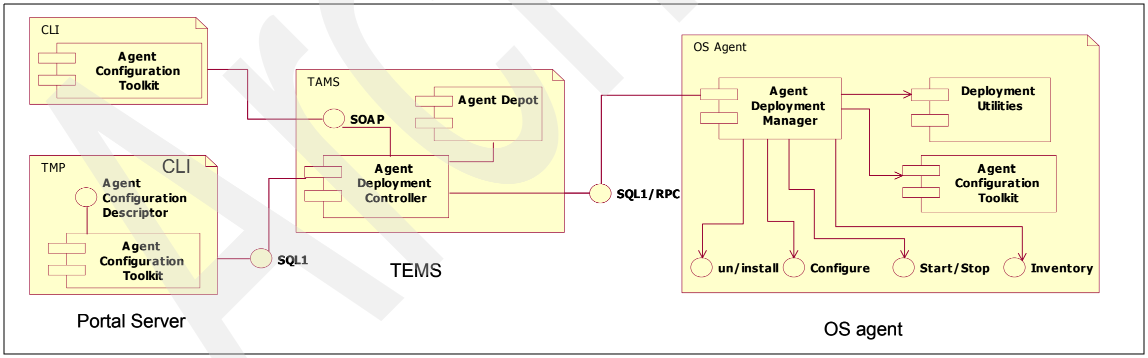


Figure 7-1 Agent deployment architecture

The deployment controller, a service on the Management Server, acts as the driver for the deployment. It handles the user requests and manages the deployment activities. The deployment controller queries the agent depot contents and transfers agent bundles using Remote Procedure Calls (RPC). All other tasks are initiated by making SQL1 calls. Agent deployment requests are made using SQL1 calls to a Management Server. The deployment controller provides the ability to initiate deployment commands from an SQL1 interface.

Deployment controller commands can be targeted to a specific system or to a managed system list. The deployment controller manages the interaction with the management agent (OS agent). The controller manages receiving and aggregating results from multiple targets and provides forwarding of requests to the appropriate management server, as well as queuing of requests for scalability. The following processes can be initiated: install, uninstall, and upgrade.

> **Note:** Deployment requests are asynchronous; when a request is received it is queued up for processing.

Agents vary greatly in how they are configured depending on the agent type and the OS platform. The agent configuration toolkit collects and transfers configuration data. It provides a set of utilities that enable the agent deployment to configure agents. The Agent Configuration Toolkit and the deployment controller communicate via SOAP (Simple Object Access Protocol).

A typical agent deployment process has the following steps, assuming that an OS agent is installed on the system where the monitoring agent will be installed:

1. A deployment request is received at the hub monitoring server by an SQL1 call from the command line (`tacmd` command) or portal client.
2. A deploy request is forwarded to the appropriate remote monitoring server, if necessary. A request is queued on the remote monitoring server and processed when a thread is available.
3. The following checks are performed before initiating the installation:
   a. Get the target platform.
   b. Find the bundle in the depot.
   c. Check whether if the bundle is already installed.
   d. Check the prerequisites
4. Installation and configuration files are transferred to the system where the monitoring agent will be installed via RPC.
5. A silent installation command is invoked via an SQL1 call to the OS agent. The command waits for status feedback asynchronously.
6. A configuration command is invoked via an SQL1 call to the OS agent. The command waits for the status feedback asynchronously.

7. A start command is invoked via an SQL1 call to the OS agent, to start the monitoring agent. The command waits for the status feedback asynchronously.

There are three main parameters that might affect the performance of the deployment process. These parameters are set on the monitoring server's KBBENV (or kbbenv.ini) files.

- **DEPLOYQUEUESIZE**

  This is the maximum number of requests that the request queue on the agent deployment controller will handle. It has a default of 100, if you do not override with a specific setting.

  Note that OS agent deployments that are performed with `createNode` do not use this setting, because these deployments do not go through the deployment controller. Instead, they communicate directly with the target system. This parameter only works when deploying application agents via `addSystem`.

- **DEPLOYTHREADPOOLSIZE**

  This is the number of threads that are available to the deployment controller. The default is 10 unless otherwise overridden. It controls how many active asynchronous threads the deployment controller uses to push the depot files down with.

- **DATACHUNKSIZE**

  This is the size of the data chunk that will be passed to RPC for transfer. The default is 500 KB unless it is overridden. If you have sufficient network bandwidth, you might want to increase this value, but be careful if this bandwidth is shared with critical production applications (especially when the deployment is done during business hours).

7.4 Considerations for installing OS and non-OS agents remotely

As mentioned earlier, you can install both OS and non-OS agents remotely, but there are some differences.

7.4.1 OS agents

OS agents must be installed before you can install a non-OS agent (such as a DB2 monitoring agent) on a target computer. OS agents can be installed remotely using the command line interface (CLI).

An OS agent deployment is performed against a machine that does not have IBM Tivoli Monitoring code at all; therefore, the hub monitoring server or remote monitoring server that performs the installation needs to contact the remote node using one of following mechanisms:

- SMB (server message block)
- SSH (secure shell)
- REXEC (remote exec)
- RSH (remote shell)

After the connection is established between the two machines, you can use a CLI to create the remote node. There is no GUI option to perform this operation, because the agent is not yet in the hub monitoring server node table and therefore not listed in the portal.

7.4.2 Non-OS agents

This section presents an example of a deployment executed by the GUI.

1. We are installing a Universal Agent on a system that the Windows OS agent has installed; see Figure 7-2.

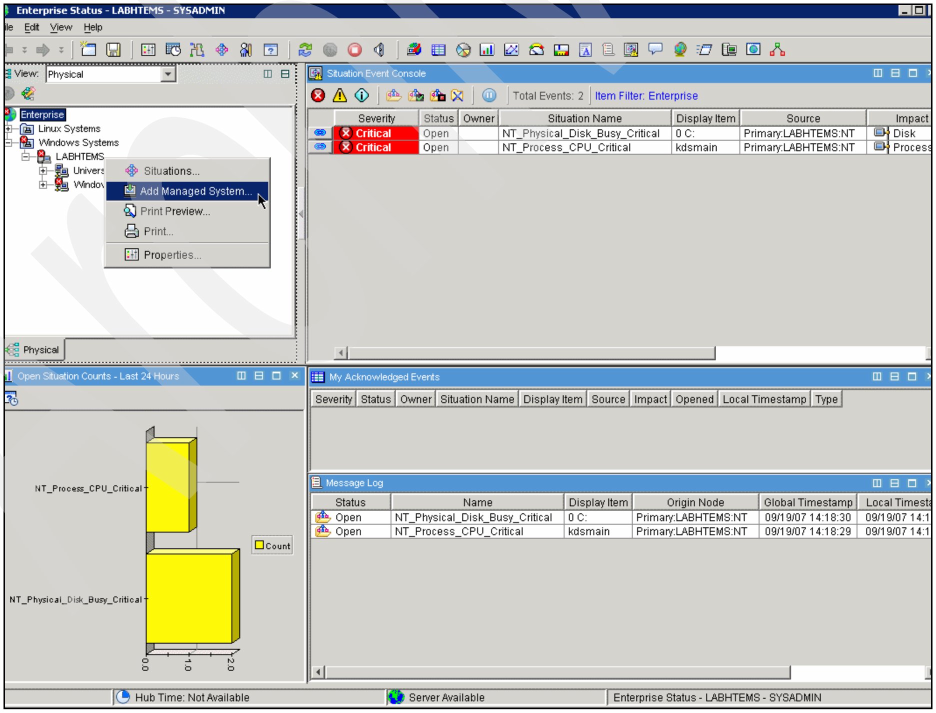


Figure 7-2 Installing a non-OS agent

2. The Select a Monitoring Agent window, shown Figure 7-3, is where we select the agent to be installed. It is important to note that agent's bundle should exist in the depot (see "Populating agent depots" on page 246).

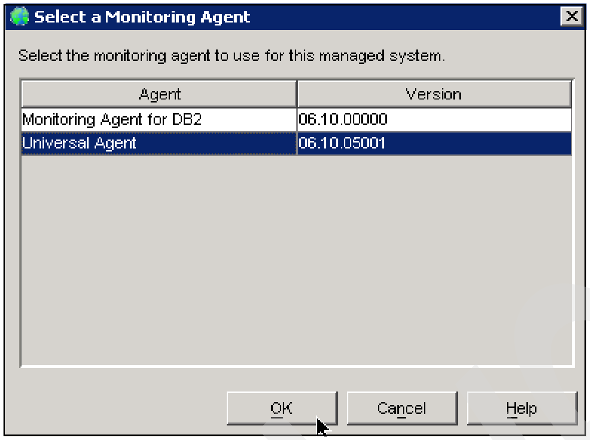


Figure 7-3 Selecting the agent to be installed

3. In the New Managed System Configuration panel (see Figure 7-4), we select the configuration options. These are different for each agent.

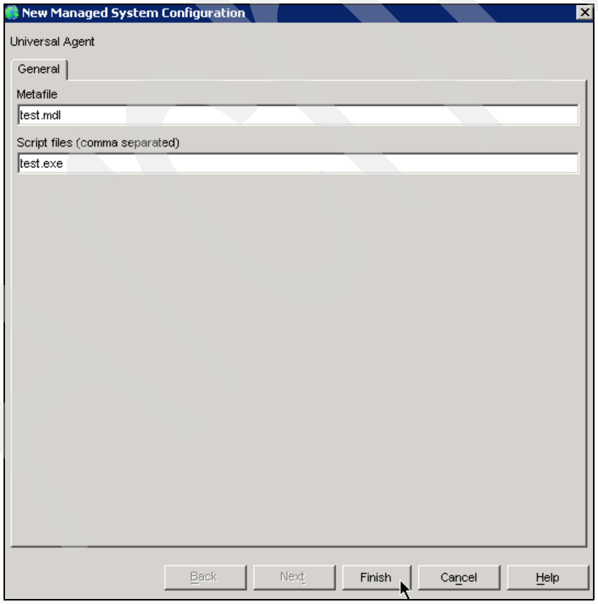


Figure 7-4 New Managed System Configuration panel

4. After we click **Finish**, the system starts installing and configuring the agent remotely, as shown in Figure 7-5.

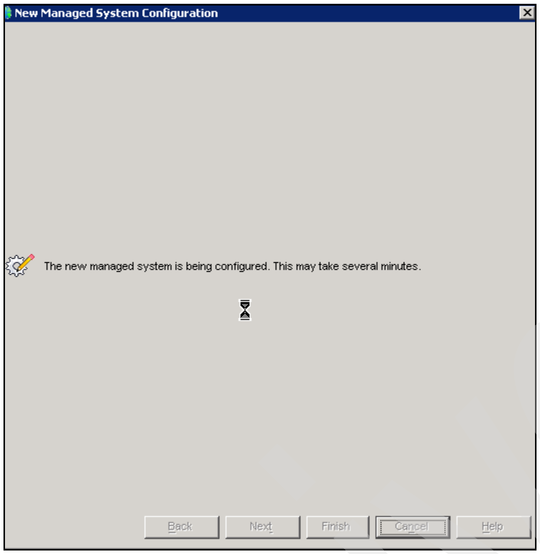


Figure 7-5 Installing the agent

7.5 Agent depots

A depot for IBM Tivoli Monitoring V6.1 is basically a place in your file system where you put all necessary files to deploy the agents. In other words, it is a repository of agent bundles that a management server can access during agent deployment. The agent depot can also contain metafiles (MDL files) and scripts used in the deployment of the Universal Agent.

Each agent bundle in the agent depot can be identified by its product id, version, and platform support. Each monitoring server used for deployment must have access to an agent depot. The depot can be on a remote file system.

The agents are loaded into the agent depot at install time, or through the **`tacmd addbundles`** command.

The agent depot is stored in the PACKAGES directory under the depot home directory, which is configurable via the DEPOTHOME environment variable. The DEPOTHOME environment variable is defined in the KBBENV environment file.

The file KBBENV is placed in the following location:

- On Windows, in <itm_installdir>\CMS

- On UNIX, in <itm_installdir>/tables/<tems_name>

If you do not have this environment variable in the file, the IBM Tivoli Monitoring will use the default home directory:

- On Windows, <itm_installdir>/CMS/depot
- On UNIX, <itm_installdir>/tables/<tems_name>/depot

> **Tip:** Place the agent depot on a separate directory from your IBM Tivoli Monitoring installation directory, which allows cleaner backup and restore operations. Typically, be sure to allocate at least 2 GB of disk space for the depot.

7.5.1 Agent depot solution

For a large scale environment, and for the reasons discussed in Chapter 2, “Planning considerations - large scale deployment of Tivoli Monitoring V6.1 components” on page 21, we recommend that you have one or more remote monitoring servers installed in your environment, depending on your solution.

In a decentralized solution, managing depots might be a problem because you need to keep the depots on all of your remote monitoring servers up-to-date, and the problem here is that you cannot populate the remote depots from the hub monitoring server. You need to do this locally.

> **Tip:** If you have a slow network, consider placing remote monitoring servers at remote locations so that the agent code is only pushed once over any slow link.

Instead of having a decentralized depot solution, you have the option of having a central depot using the operating system features, sharing a file system, and accessing the depot from all monitoring servers.

To do this on UNIX, you can use a NFS to export a central filesystem and mount it on all your monitoring servers. After that, you need to edit the file KBBENV and change the value of DEPOTHOME environment variable to point it to the new location.

On Windows systems, you can use a shared file system. In the path specification you must use the UNC designation (such as \\computer_name\shared_folder). Also, you must set the service id for the Windows monitoring server to a user id that are able to authenticate into the file system (you can use “Administrator”).

Note: Shared agent depots may be the most appropriate solution for large scale environments if you have high speed communication between your monitoring servers. If you decide to use a shared depot solution, ensure that you upgrade all remote monitoring servers before performing upgrades on the agents that are connected to these remote monitoring servers.

Alternatively, if the shared depot is not an option for you, you can use a mechanism to synchronize the data between your monitoring servers. One option would be to use **`rsync`** on UNIX. For Windows, you can use Windows Distributed File System (DFS™).

7.5.2 Populating agent depots

Before you can deploy agents by using the **`tacmd`** command, you need to populate the depot with the necessary bundles. A *bundle* is an installation image and any prerequisites with the necessary information to perform the installation successfully. IBM Tivoli Monitoring provides different bundles for each operating system. Depending on the types of agents, you may need to add bundles for all different operating systems you have in your environment.

There are two different methods for populating agent depots:

- The first method is by using the product installer for each operating system (GUI method).
- The second method is by using the **`tacmd`** command. This section only explains the **`tacmd`** command method.

The parameters that you can use with the **`tacmd`** command to manage your depots are listed and explained in Table 7-1.

Table 7-1 Parameters to manage depots and bundles

Command	Functionality
addBundles	Use the **`tacmd addBundles`** command to add one or more deployment bundles to the local agent deployment depot.
removeBundles	Use the **`removeBundles`** command to remove one or more deployment bundles from the local deployment depot.
listBundles	Use the **`listBundles`** command to display the details of one or more deployment bundles that are available for deployment to the local deployment depot.

Command	Functionality
viewDepot	Use the **tacmd viewDepot** command to display the types of agents you can install from the deployment depot on the server where you are logged in, or the specified remote server.

To see the full syntax of the **tacmd** command, refer to *IBM Tivoli Monitoring 6.1 Command Reference,* Version 6.1, SC23-6045.

Example 7-1 shows the output of using the **tacmd** command to add a bundle of Monitoring Agent for Windows OS into a depot, using the following command line:

```
tacmd addBundles -i C:\temp\ITM_6.2\WINDOWS\Deploy -t nt
```

In this command, the **-t** parameter specifies the product code for the bundle that needs to be added, and the **-i** parameter specifies where IBM Tivoli Monitoring will search for the installation files.

Example 7-1 tacmd addBundles command output

```
C:\IBM\ITM\BIN>tacmd addBundles -i C:\temp\ITM_6.2\WINDOWS\Deploy -t nt

Are you sure you want to add the following bundles to the
C:\IBM\ITM\CMS\depot\
depot?

Product Code : NT
Version      : 062000000
Description  : Monitoring Agent for Windows OS
Host Type    : WINNT
Host Version : WINNT
Prerequisites:
Enter Y for yes or N for no: Y

Adding bundles to the C:\IBM\ITM\CMS\depot\ depot. The time required to
complete
 this operation depends on the number and size of the added bundles.

The following bundles were successfully added to the
C:\IBM\ITM\CMS\depot\ depot
:
Product Code : NT
Version      : 062000000
```

```
Description  : Monitoring Agent for Windows OS
Host Type    : WINNT
Host Version : WINNT
Prerequisites:
C:\IBM\ITM\BIN>
```

During this process, IBM Tivoli Monitoring reads the descriptor file of the specified bundle you are adding and identifies (among all of the files in the installation media) which files need to be copied on the agent's depot in your monitoring server. The files loaded are copied onto the path specified in the DEPOTHOME environment variable, defined in the KBBENV environment file.

You need to execute this process only once for each bundle. Subsequently, every time you install a new agent, IBM Tivoli Monitoring uses these files, which are already loaded.

> **Note:** If you do not have a central shared depot, you cannot populate the other remote monitoring servers that you may have in your environment from the hub monitoring server. You need to login on each remote monitoring server and populate them individually.

After you have run the command, you can check whether the depot is properly populated with all bundles that are needed by using the following command:

```
tacmd viewDepot
```

Example 7-2 shows the output of the command.

Example 7-2 tacmd viewDepot command output

```
C:\IBM\ITM\BIN>tacmd viewDepot
Product Code : NT
Version      : 062000000
Description  : Monitoring Agent for Windows OS
Host Type    : WINNT
Host Version : WINNT
Prerequisites:

Product Code : OQ
Version      : 062000000
Description  : Monitoring Agent for Microsoft SQL Server
Host Type    : WINNT
Host Version : WINNT
Prerequisites:
```

```
Product Code : OR
Version      : 062000000
Description  : Monitoring Agent for Oracle
Host Type    : WINNT
Host Version : WINNT
Prerequisites:

Product Code : OY
Version      : 062000000
Description  : Monitoring Agent for Sybase
Host Type    : WINNT
Host Version : WINNT
Prerequisites:

Product Code : UD
Version      : 062000000
Description  : Monitoring Agent for DB2
Host Type    : WINNT
Host Version : WINNT
Prerequisites:

Product Code : UM
Version      : 062000000
Description  : Universal Agent
Host Type    : WINNT
Host Version : WINNT
Prerequisites:

C:\IBM\ITM\BIN>
```

At this point, as you can see in Example 7-2 on page 248, you have the depot properly populated and are now ready to deploy the agents onto the servers.

7.6 Deploying the monitoring agents by using the tacmd command

IBM Tivoli Monitoring V6.1 makes use of agents installed locally on each resource. These agents are frequently communicating with your monitoring server, sending a significant amount of information that will give you a better view of your enterprise.

A very common mistake is to think that each agent corresponds to one server.

In actuality, however, one agent means basically one *resource* of your server—your server probably has not only an operating system, but also a database or an application that also needs to be monitored.

In IBM Tivoli Monitoring V6.1, you have one specialized agent for each resource you monitor. That means it is not uncommon to have servers with two or more agents installed. These are some of the specialized agents you may have:

- Operating system
- Cluster (Microsoft Cluster)
- Database (DB2, Oracle, Microsoft SQL Server and Sybase)
- Application (SAP®)
- Virtual Machine (Citrix, Microsoft Virtual Server and ESX)
- Messaging and Collaboration (Lotus® Domino® and Microsoft Exchange Server)

Next, we describe how to install and maintain some of these agents by using IBM Tivoli Monitoring V6.1 facilities.

7.6.1 Installing the operating system agents

IBM Tivoli Monitoring V6.1 provides a built-in method to deploy agents on the servers. This can be done from a central location. You can also use the remote agent deployment function to configure already deployed agents and install maintenance fixes. There are only two types of monitoring agents:

- OS agents, using **`tacmd createNode`**
- Non-OS agents, using **`tacmd addSystem`**

To perform the installation, the deployment mechanism makes use of a descriptor (.dsc) file that contains all of the information required to deploy and install the product. This file is stored in a depot when you add the bundles, and each agent has its own descriptor file.

In Example 7-3 on page 251, you can see the output of using the **`tacmd`** command to create a new node, deploying a Monitoring Agent for Windows OS using the following command (which should be run at the monitoring server):

```
tacmd createNode -h smb://toronto -u Administrator \
-p SERVER=IP.PIPE://9.3.5.103:1918
```

The command **`tacmd createNode`** is used to deploy an OS agent onto a remote computer. The parameter **`-h`** specifies what network service should be used during the deployment and location of the remote host where the agent will be installed. The **`-u`** parameter specifies the user that will be used during this operation. The **`-p`** parameter provides parameters to configure the agent.

Tip: You can also specify the password in the same command line by using the parameter **-w**. Otherwise, the installation program will ask you to type the password during command execution.

In this case, the **-p** parameter provides the following information to the installation program:

- The protocol that the agent will use to communicate with the monitoring server (IP.PIPE)
- The monitoring server that the agent will communicate to
- The port number

Note: For large scale environments, we recommend that you set the secondary monitoring server when you are installing the monitoring agent. Define in the parameter **-p** the attribute BSERVER, exactly as you did to the attribute SERVER.

Example 7-3 tacmd createNode command output

```
C:\IBM\ITM\BIN>tacmd createNode -h smb://toronto -u Administrator -p
SERVER=IP.PIPE://9.3.5.103:1918
KUICCN001I Initializing required services...
KUICCN005I Enter the password for Administrator.

KUICCN039I Attempting to connect to host toronto ...
KUICCN050I Distributing file 90 of 90 (144 MB / 144 MB)...
KUICCN002I Beginning the installation and configuration process...

KUICCN057I The node creation on host toronto was successful.

KUICCN065I The node creation operation was a success.

C:\IBM\ITM\BIN>
```

Tip: Remember to specify the networking service that should be used to deploy the agent (smb, ssh, rexec or rsh). Otherwise, the command will try all services.

This command is used to install the basic operating system monitoring agent, but you can have many agents for different kinds of applications on the same system.

7.6.2 Installing non-operating system agents

After the node has been created and OS agent is running, you can install other non-OS agents quite easily. This can be done by using the Tivoli Enterprise Portal or again, by using the t**acmd** command with another parameter. For more information about installing additional agents using Tivoli Enterprise Portal, refer to *IBM Tivoli Monitoring 6.1 Installation and Setup Guide*, GC32-9407.

To install an additional agent, you need first to know the correct name of the ragent. You can check the agent name using the following command (which should be run at the monitoring server):

```
tacmd listSystems
```

This command displays all nodes and agents, as shown in the output in Example 7-4.

Example 7-4 tacmd listSystems command output

```
C:\IBM\ITM\BIN>tacmd listSystems
Managed System Name          Product Code Version     Status
DB2:ATHENS:UD                UD           06.20.00.00 Y
Primary:ATHENS:NT            NT           06.20.00.00 Y
DB2:BRUGE:UD                 UD           06.20.00.00 Y
BRUGE:Warehouse              HD           06.20.00.00 Y
waco.itsc.austin.ibm.com:LZ  LZ           06.20.00.00 Y
REMOTE_WACO                  EM           06.20.00.00 Y
Primary:BERLIN:NT            NT           06.20.00.00 Y
Primary:TORONTO:NT           NT           06.20.00.00 Y
Primary:BRUGE:NT             NT           06.20.00.00 Y
Primary:FLORENCE:NT          NT           06.20.00.00 Y
Primary:LONDON:NT            NT           06.20.00.00 Y
BERLIN:UA                    UM           06.20.00.00 Y
REMOTE_FLORENCE              EM           06.20.00.00 Y
HUB_BERLIN                   EM           06.20.00.00 Y

C:\IBM\ITM\BIN>
```

To install a non-OS agent (in this case, a DB2 monitoring agent on a Windows machine), use following command:

```
tacmd addSystem -t ud -n Primary:ATHENS:NT -p INSTANCE=”DB2”
```

Example 7-5 on page 253 shows the output from this command.

The command **tacmd addSystem** installs the agent defined with the parameter **-t**. This value corresponds to the product code for the DB2 monitoring agent.

The target agent is indicated with the parameter **-n**. This value should correspond to the agent name listed in the output of the **tacmd listSystems output** command. The **-p** parameter is sending to the agent the configuration (in this case, the DB2 instance name).

If you deploy the DB2 agent on an UNIX system, you need to use the following command line:

```
tacmd addSystem -t ud -n waco:LZ -p INSTANCE=”db2inst1” \
_UNIX_STARTUP_.Username=”db2inst1”
```

The output is shown in Example 7-5; note that it is same for Windows or UNIX systems.

> **Note:** To install the non-OS agent remotely, IBM Tivoli Monitoring uses the OS agent to transfer the necessary data, so ensure that you have the OS agent installed and running properly.

Example 7-5 tacmd addSystem command

```
C:\IBM\ITM\BIN>tacmd addSystem -t ud -n Primary:ATHENS:NT -p
INSTANCE="DB2"

The agent type UD is being deployed.

The product type UD on https://berlin:3661 is now being managed.

C:\IBM\ITM\BIN>
```

The deployment method for different types of agents is the same. Depending on the agent, you may have different variables to specify with the **-p** parameter to set up the agent. For all the variables, you need to refer the user’s guide of the agent you are deploying.

7.6.3 tacmd return codes

Table 7-2 on page 254 lists the return codes for the **tacmd** command.

> **Note:** These return codes are applicable to all **tacmd** commands, not just for the ones related to the installation of agents.

Table 7-2 tacmd return codes

Code	Category	Description
0	Success	Indicates the command was successful.
1	Syntax Error or Help	Indicates either that the help command was given, or that the syntax used was incorrect.
2	No Permission	Indicates that the user does not have permission to issue the command.
3	Version Mismatch	Indicates that the version of the server was not what was expected.
4	Communication Error	Indicates that an error occurred in the communications with the server.
5	Timeout	Indicates that an operation waiting for data did not receive it within the time it was expected.
6	Input Error	Indicates that the input to the command was not what was expected.
7	Server Exception	Indicates that an error occurred on the server that caused the command to fail.
8	Command Error	Indicates that an internal error occurred while executing the command.
9	Invalid Object	Indicates that a specified object does not exist.

7.7 Deploying agents using Tivoli Configuration Manager (or Tivoli Provisioning Manager)

It is possible to deploy the monitoring agents by using Tivoli Configuration Manager or Tivoli Provisioning Manager, as explained here.

Note: The remainder of this chapter addresses using Tivoli Configuration Manager to deploy monitoring agents—but keep in mind that everything discussed here is also applicable to Tivoli Provisioning Manager or Tivoli Provisioning Manager for Software.

Refer to the IBM Redbooks publications *Deployment Guide Series: IBM Tivoli Provisioning Manager Version 5.1*, SG24-7261, or *IBM Tivoli Configuration Manager and IBM Tivoli Provisioning Manager for Software Coexistence Considerations*, SG24-7426, for detailed information about these products.

Because many IBM Tivoli Software customers already have an investment in the Tivoli Management Framework V4.1.1 and Tivoli Configuration Manager V 4.2, it makes sense for them to use Tivoli Configuration Manager for this purpose as well, as this provides all of the benefits of Tivoli Configuration Manager (such as bandwidth control; centralized monitoring of the deployment; the ability to set prerequisite packages; integration with the Inventory database; the ability to perform user-initiated install pull versus the push method, and so forth).

Tivoli Configuration Manager uses a package format called Software Package Block (SPB). The SPB contains both the binaries and the definition of how to install these on the target systems.

Also important is the Software Package Definition (SPD), which is a text file that describes the package. An SPD along with a CD or DVD image of the agent may be used to create the SPB file.

At the time of writing, the disk images for the product are not provided in a form that can directly be imported into a Tivoli Configuration Manager or Tivoli Provisioning Manager environment. Also at this time, the disk images do not include SPDs that could be used to create the software packages for those products either.

Initial SPDs for the General Availability (GA) code were developed as part of a project resulting in the creation of the IBM Redbooks publication *Getting Started with IBM Tivoli Monitoring 6.1 on Distributed Environments*, SG24-7143.

However, as Fix Packs for the product were released, clients were unsure of what needed to be changed in order to accommodate the new image. The SPDs provided with this document have been revised to accommodate the Fix Pack 5 disk images. You can download these SPDs from the ITSO Web site. Refer to Appendix C, “Additional material” on page 487 for instructions about how to download these files.

The changes made to the installation structure have been relatively minimal to this point, but there is no guarantee that they will work for future releases.

The agent deployment solution using Tivoli Configuration Manager is comprised of one SPD file for each available IBM Tivoli Monitoring V6.1 agent, and a set of files containing silent response files and scripts used by the install and uninstall processes.

Note: The SPDs provided as part of this document are on as-is basis.

The SPD contains user-defined variables needed to build and deploy the package. These variables can be categorized into two sets:

- Variables needed to convert the SPD in SPB that the user must pass in. Note that these variables have a default value which could be invalid in your environment.
- Variables used to configure the install. A default variable for them is provided, and it represents the same default variable that comes with IBM Tivoli Monitoring V6.1 agents. Review them and perform the appropriate changes as per your configuration.

The following variables belong to the first set:

source_dir A local directory that contains the Agent binaries

silent_files_dir A local directory that contains the reworked response files and scripts

The following variables belong to the second set:

CandleHome Destination install directory. Default: /opt/IBM/ITM

CandleEncryptionKey Encryption key.

Default: IBMTivoliMonitoringEncryptionKey

Others The variables needed for initial configuration, such as TEMS server, protocol, and port number.

Note: To shrink the package size for AIX, Solaris, and HP-UX, there are three SPD options presented:

- One that only contains the information for a 32-bit install
- One that only contains the information for a 64 bit install
- One that contains information for both a 32-bit and a 64-bit install.

This provides some flexibility in the deployment. If adequate space is available, or if the operating mode of the target machine is unknown, the combined SPD may be used. If space or network utilization is a bigger concern, the targeted SPDs may be used.

8

Managing IBM Tivoli Universal Agent in a large scale deployment

This chapter describes how to manage IBM Tivoli Universal Agent in a large enterprise environment. It also explains how to deal with the Universal Agent versions, and the use of the Metafile Server.

The following topics are addressed:

- Introduction to Universal Agent
- Dealing with Universal Agent versions
- Metafile Server for the Universal Agent

8.1 Introduction to IBM Tivoli Universal Agent

The IBM Tivoli Universal Agent (Universal Agent) is a generic agent of IBM Tivoli Monitoring V6.1. This agent is capable of using data from a wide variety of data providers.

You create data definitions that describe the source and structure of the data supplied by the data providers, and you store the data definitions in metafiles (metafiles are illustrated in Figure 8-1).

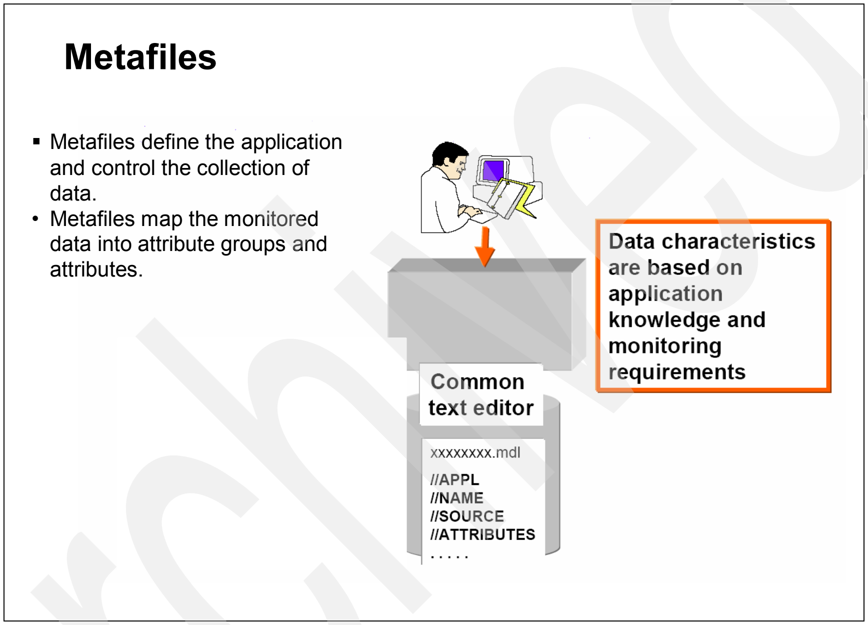


Figure 8-1 Metafiles

The data providers relay the monitoring data and the metafile statements to the Universal Agent, which then sends the monitoring data to the Tivoli Enterprise Monitoring Server. The Universal Agent uses the metafile statements to dynamically generate and upload application definition files, which represent the definitions you supplied in the metafile, to the Tivoli Enterprise Monitoring Server and the Tivoli Enterprise Portal.

Note: Everything Universal Agent “knows” about an application comes from the metafile. You can have multiple metafiles active in a single Universal Agent system. The KUMPCNFG file stores the active metafiles.

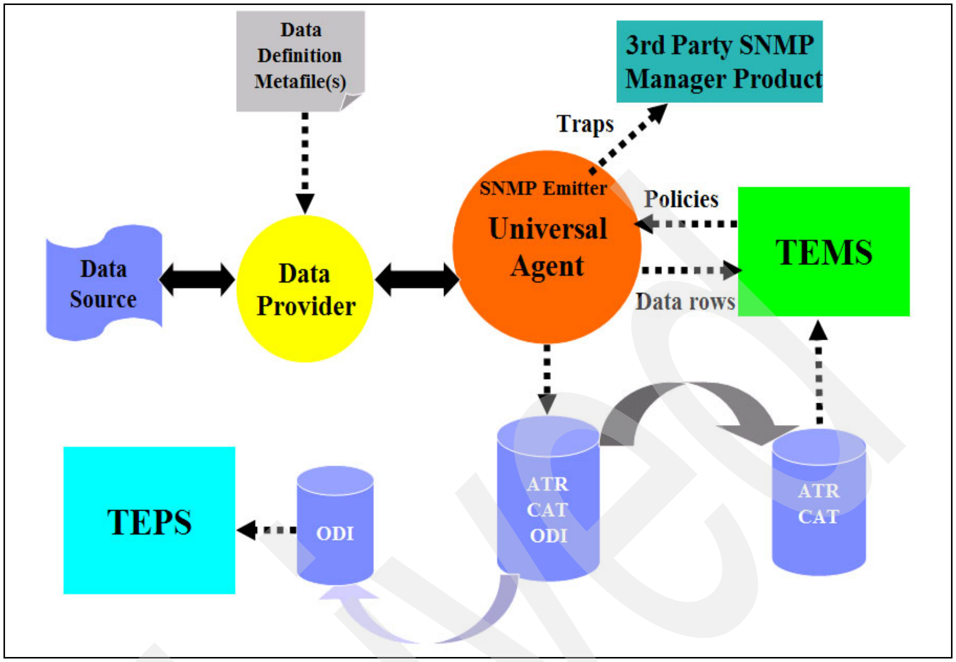


Figure 8-2 Universal Agent dataflow

Figure 8-2 illustrates the sequence in which the Universal Agent works.

1. The mdl metafile is created by a user, using some data provider.
2. The mdl file is imported to the Universal Agent.
3. As soon as the mdl file becomes imported, it creates following files:
   a. cat, atr on the monitoring server to which it is connected.
   b. ODI (object data interface) files in the portal server.
   c. cat, atr and kumawxxx files are also placed on Universal Agent where the mdl file was imported.

Note that Universal Agent-generated ODI files have a different naming standard than the dockxx ODI files that regular IRAs deliver. They are named as xxxODI##.

The ODI file upload occurs when a Universal Agent application first registers with the Tivoli Enterprise Monitoring Server, or when a Universal Agent application version changes. Every attribute group in a Universal Agent metafile equates to an application table, and the table becomes a workspace in the portal Navigator. The cat/atr files by Universal Agent metafile are used by the monitoring server.

Note: When the metafile is being imported for the first time to the Universal Agent, it is important to have the Universal Agent connected to the hub monitoring server. This is because the hub monitoring server only has the capability to push the contents to portal server, so that the portal server can generate the ODI file for the metafile dynamically. You can then reconfigure the Universal Agent to connect to the relevant remote monitoring server.

In a large environment with hundreds of Universal Agents, if they are all using the *same* set of metafiles, you need to reconfigure only one Universal Agent to the hub monitoring server to dynamically upload the CAT and ATR files.

The option is to manually copy the CAT and ATR files for the metafile applications to the hub monitoring server and then recycle the hub, but in large environments, it might be inconvenient to restart the hub monitoring server every time a new metafile is being imported.

The Attribute Group, and its attributes as specified in the metafile, now becomes available to the portal server in a Query Editor as well as a Situation Editor, where users can create custom queries to view or extract data from the agent and situations to monitor the desired behavior on agents.

8.2 Dealing with Universal Agent versions

Metafiles are assigned both version and modification numbers. When you first import a metafile into the Universal Agent, it is assigned a version number of 0 and a modification number of 0.

Subsequently, each time you change the metafile and refresh it on the Universal Agent, either the version number (major changes) or the modification number (minor changes) is incremented by one, depending upon the type of change you make.

Using the tools and techniques described in this chapter, you can reset the version and modification numbers.

Tip: In order to reset the Universal Agent version, you need to *stop* the monitoring and portal servers. Because this might be a problem in large production environments, always test your Universal Agent thoroughly in a test site before the initial deployment, so that there will be no need for a frequent version change.

Also, always consider using the Agent Builder first, before attempting to solve a monitoring requirement with the Universal Agent. Using the Agent Builder is a better choice for the following reasons:

- There is no Universal Agent versioning consideration.
- There is more efficient data collection. Data is cached for 30 seconds by default, and data is only collected when requested by the portal server, situation, or historical collection.
- The Agent Builder allows multiple data collectors to be built into a single agent.

Agent Builder is discussed in detail in the IBM Redbooks publication *Deployment Guide Series: IBM Tivoli Monitoring V6.2*, SG24-7444.

When the version number of a metafile changes, the managed systems, managed system lists, and attribute groups based on the previous version number go offline and new ones with the new version number come online.

The managed system name change means that a new Tivoli Enterprise Portal Navigator tree entry is inserted for the new metafile application version, and report workspaces are positioned below the entry.

Version numbers changes are considered “major” because you cannot simply restart situations distributed to a previous version of a managed system. Instead, you must create new situations or modify the old ones to use the new attribute group names, and then distribute the situations to the new versions o f the managed systems or the managed system list. For these reasons, it is often necessary to reset the version numbers.

Note: You cannot turn off Universal Agent versioning.

8.2.1 Types of changes

This section describes explains changes that constitute major, minor, or no change actions.

Changes that do not affect modification or version number

You can make the following changes without changing the modification or version number of the metafile:

- TTL value
- A change to the SOURCE statement
- Data type from P, S, or K to any of P, S, or K
- Delimiter specified in the ATTRIBUTE statement
- A change to the RECORDSET statement
- A change to the CONFIRM statement
- A change to an attribute FILTER parameters
- A change to the SQL statement

Note: TTL indicates how long the data provided by data providers is considered valid for evaluation. The TTL value you choose affects how situations are evaluated and what data rows you see in the Tivoli Universal Agent application workspaces.

The RECORDSET statement, which is for File and Script Data Provider metafiles only, enables the data provider to extract attribute data from multiple records.

P, S, E and E are called “nature of the data”:

P for **Polled (default):** Polled data becomes available periodically and only the latest collected data row is available for situation monitoring and reporting.

S for **Sampled:** Sampled data behaves in the same way as polled data except that sampled data is cumulative, meaning that more than one set of attribute data values can be available for use.

E for **Event:** Event data occurs unpredictably and is reported in asynchronous fashion as soon as the data becomes available.

K for **Keyed:** Keyed data behaves in the same way as sampled data, but allows you to correlate events. You can designate up to five attributes in each group as key attributes. If no attributes are designated as KEY, then the first attribute in the group is assumed to be the KEY.

For example, an application checks system status such as CPU utilization and network data traffic rate every 30 seconds. This is polled or sampled data, or possibly even keyed data if any of the attributes function as unique keys or indexes.

By contrast, network alerts or console messages, which occur at unpredictable intervals, should be defined in the metafile as event data.

Changes that affect the modification number (minor changes)

The following changes cause the modification number to be incremented:

- Adding a new attribute to the end of the attribute list for an attribute group
- Adding a new attribute group at the end of the metafile
- Adding, removing, or changing help text
- Atomizing an existing attribute
- Adding, removing, or changing Scale or Precision values
- Adding, removing, or changing Caption values
- Adding, removing, or changing Warehouse or Aggregation parameters
- Adding, removing, or changing HistoricalTimestamp or PrimaryKey options

Note: The Tivoli Universal Agent allows you to atomize attributes in your metafiles. *Atomizing* an attribute permits the use of the display item option during situation definition. The display item option allows you to do the following:

- Generate separate events for a single true condition
- Easily display the value that caused the situation to become true

Scale and Precision are used for floating point numbers. The Tivoli Enterprise Portal uses the Scale value to know how many positions to shift the decimal point to the left, and it uses the Precision value to determine the overall width of the floating point number.

Using a CAPTION parameter in your attribute definition provides a more readable version of an attribute in the workspace column headings.

Changes that affect the version number (major changes)

The following changes cause the version number to be incremented:

- Renaming or deleting an existing attribute
- Changing the type of an attribute
- Changing the length of an attribute
- Changing the name of an attribute group
- Adding a new attribute anywhere other than the end of a list of existing attributes
- Changing the order of attributes
- Changing a data type from E *to* P, S, or K
- Changing a data type from P, S, or K *to* E
- Adding a new attribute group anywhere other than the end of a metafile

> **Important:** The version number increases by 1, starting at 00, when certain changes described here are made. It is important to note that the *entire* 00 is the version number.
>
> Some people interpret that 00 represents, respectively, the version number and the modification number. This is *not* correct, however, because you will never see the first 0 changing on its own (for example, from 00 to 10). It is always incremented by 1 (for example, from 01 to 02).

8.2.2 Resetting the Universal Agent version

To reset the version back to 00, follow these steps. You may want to reset the version back to 00 if you are editing the metafile by adding or editing the attributes, and you keep re-importing (refreshing) the metafile.

1. Make sure your monitoring server and portal server are up and running.
2. Stop the Universal Agent where your MDL file is installed.
3. Log into your Tivoli Enterprise Portal Client.
4. From the Tivoli Enterprise Portal client Physical Navigator views, right-click Enterprise and select **Workspace** → **Managed System Status**. The managed System Status workspace is displayed.
5. Highlight all of the IBM Tivoli Universal Agent Managed Systems where your MDL file is installed.
6. Right-click and select **Clear off-line entry**. This will remove all of the selected Universal Agent entries from that table.
7. Logout of the Tivoli Enterprise Portal Client.
8. Stop your monitoring server and portal server.
9. Log into your Universal Agent System, open a command prompt, and go to the <ITM_INSTALL>/TMAITM6 directory.
   - If this is a Windows system:
     - Run the command **`um_cleanup.bat IBM\ITM UA`** to reset the versioning in the Universal Agent (this assumes your installation directory is C:\IBM\ITM on Windows and you used the default Work directory for the Universal Agent).
   - If your system is a UNIX system:
     - Run the command **`um_cleanup.sh <ITM Install Path> UA`**.

> **Important:** The command `um_cleanup bat` (or `um_cleanup.sh`) resets the version numbers of *all* applications—and not simply on a specific application on the system on which it is run. It deletes *all* Universal Agent-generated catalog, attribute, and ODI files for the various Universal Agent applications that you have activated.
>
> For example, if you imported a metafile with an application name of TST, you should have a TSTCATR00, a TSTCAT00, a TSTODI00, and a kumawTST file in your Universal Agent work subdirectory.
>
> If you run `um_cleanup` with the UA argument, it will delete every ???ATR??, ???CAT??, ???ODI??, and kumaw??? file in your work directory.

10. Log into your each monitoring server system, and open up a command prompt.
    - If this is a Windows system:
      - Run the command `um_cleanup.bat IBM\ITM CMS` to reset the versioning in the monitoring server (this assumes your installation directory is C:\IBM\ITM on Windows).
    - If your system is a UNIX system:
      - Run the command `um_cleanup.sh <ITM Install Path> CMS`.
11. Log into your portal server system, and open up a command prompt.
    - If this is a Windows system:
      - Run the command `um_cleanup.bat IBM\ITM CNPS` to reset the versioning in the portal server (this assumes your installation directory is C:\IBM\ITM on Windows).
    - If your system is a UNIX system:
      - Run the command `um_cleanup.sh <ITM Install Path> CNPS`.
12. Restart the monitoring server, portal server, and Universal Agent.

Now the Universal Agent application should show up as version 00, so you can create workspaces and situations to be exported.

After the metafile reverts back to version 00, there may be queries hanging in the portal server database, which continued being incremented as new versions were being added. So it is very important to get rid of those queries; otherwise, you will see those attribute groups hanging under **Query Editor → Universal Data Provider → SubNodeName##.**

Note: Make changes in your test environments, before these metafiles are loaded to production, to minimize the effort needed to revert the versions of Universal Agents and all the cleanup effort involved.

8.3 Metafile Server for the Universal Agent

In an enterprise, the same metafile applications can be supported by several Tivoli Universal Agents at various locations. So, to ensure consistent definitions through the enterprise, and to reduce the complexity of managing and maintaining multiple Tivoli Universal Agents, store metafiles in a single location. It is also a good practice to regularly back up your metafiles because they represent an important part of your enterprise management system.

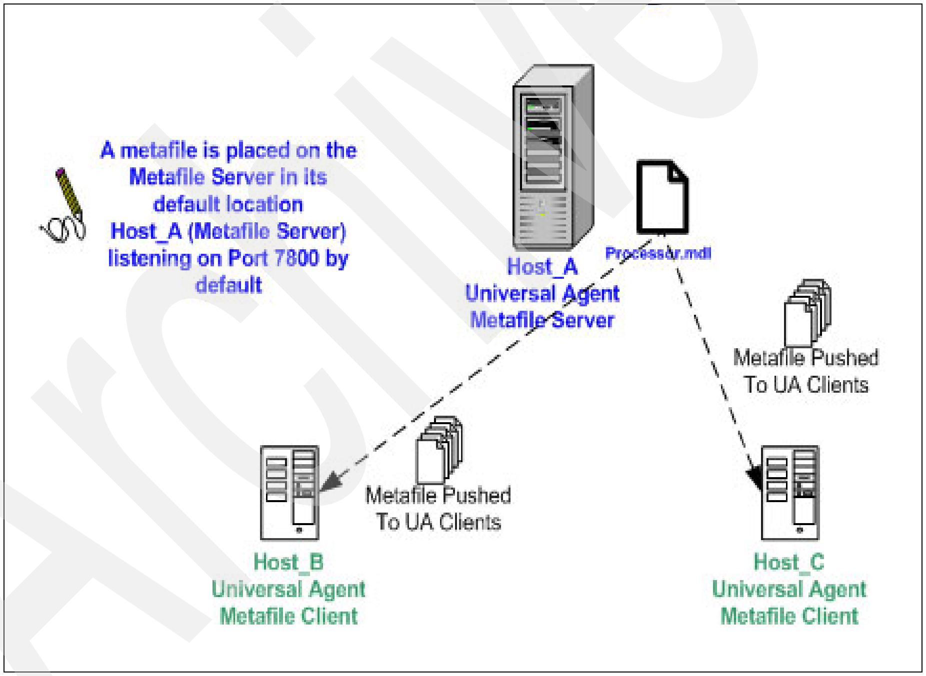


Figure 8-3 Metafile Server for the Universal Agent

The centralized metafile server facility enables you to designate one or more Tivoli Universal Agents as a metafile server. Client Universal Agents can retrieve the metafiles they require from the designated server at run time, rather than maintaining local copies.

Tip: Remember that each agent will open a socket to the Metafile Server, so firewalls must be taken into consideration. So, when using the Metafile Server in environments with firewalls, it is sometimes necessary to create multiple Metafile Servers (typically one for each network zone).

Designating a metafile server

You use the environment variable KUMP_META_SERVER to specify the name of the host of the Tivoli Universal Agent that you want to use as the server: KUMP_META_SERVER=hostname. The presence of the environment variable indicates to the Tivoli Universal Agent that it should use a centralized metafile server.

If this environment variable is not set, the Tivoli Universal Agent operates in stand-alone mode and looks for its required metafiles locally. If the host name specified by KUMP_META_SERVER cannot be resolved to a TCP/IP address, the metafile server feature is disabled and the data provider loads metafiles from the local metafile location.

Storing server metafiles

You must store the metafiles for the server in the Tivoli Universal Agent default working directory or in the location identified by the KUMP_META_PATH or KUM_WORK_PATH environment variables.

The Tivoli Universal Agent uses the value specified for the KUMP_INIT_CONFIG_PATH environment variable as the location of its configuration file. If no value is specified for KUMP_INIT_CONFIG_PATH, the Tivoli Universal Agent uses the default work directory, typically /um/work on UNIX computers and \TMAITM6\work on Windows computers, or the directory specified by KUM_WORK_PATH.

Note: Only the metafiles are transported over the port to the Universal Agents acting as clients. If the metafile is using the Script Data Provider and is pointing to a script local to that machine, make sure that script is present on the targeted Universal Agent.

Determining server and client roles on the same host

When a Tivoli Universal Agent starts up and discovers that KUMP_META_SERVER is set to the name of its own host, it makes the following determination:

- If it is the first Tivoli Universal Agent initialized on this system, it immediately takes the role of metafile server.

- If it is *not* the first Tivoli Universal Agent initialized on this system, it assumes that the metafile server has already started and it takes the role of metafile client, as does any Tivoli Universal Agent started elsewhere in the enterprise.

Synchronization of metafile server and client

If a Tivoli Universal Agent determines at startup that KUMP_META_SERVER is not set to the name of its own host, it assumes the role of client and immediately attempts to establish a socket connection to the designated metafile server. By default, the metafile server listens for connections on port 7800, which is a value that can be changed with the KUMP_META_SERVER_PORT environment variable.

If the connection is successful, the Tivoli Universal Agent sets up the required internal management structure and continues startup processing.

If the connection is unsuccessful, either because of network problems or because the Tivoli Universal Agent metafile server has not started, then the metafile client schedules periodic connection retries and completes initialization.

Communication between metafile client and server can be disrupted during normal operation for a variety of system or network reasons. If communication is disrupted, the client automatically reverts to standalone mode and schedules periodic retries until the connection to the server is re-established. Log messages are issued that clearly identify the status of the metafile service. You can monitor these messages in the DPLOG workspaces.

Activating metafiles with Take Action commands

You can activate metafiles dynamically using the Take Action Control Import and Control Refresh commands. Use the Take Action method if you do not want to restart the Tivoli Universal Agent after creating a new metafile and want to import it, or if after modifying an active metafile, you want to refresh the metafile. You must have the Tivoli Enterprise Portal client session active if you plan to use this method.

Perform the following steps to select the Tivoli Universal Agent Take Action commands from the Tivoli Enterprise Portal:

1. From any location in the Tivoli Enterprise Portal navigator tree under Universal Agent, select a leaf in the navigator tree.
2. Right-click the leaf and select **Take Action... -> Select...**. The Take Action window is displayed.
3. From the Take Action window, select an action from the Name: drop-down list. The Edit Argument Values window is displayed.

As an example, if you are going to import the processor.mdl file, choose **Control Import** from the Name: drop-down list. From the Edit Argument Values window, enter `processor.mdl` for the Value.

4. Type the name of the metafile that you want to activate in the Value field of the Edit Argument Values window.
5. Click **OK**.
6. From the Destination Systems section of the Take Action window, select a destination system for the action.

   Note: Always distribute the Take Action command to the destination system whose data provider type matches the data provider type of the metafile that you import. In this example, the metafile is using the Socket Data Provider, so you distribute the action to either the &localhostASFSdp:UAGENT00 or &localhostSOCKdp:UAGENT00 destination system, depending on whether you configured the ASFS DP or SOCK DP.
7. Click **OK**. The Action Status window is displayed and it indicates whether the action was successful.

Overriding the central metafile definition

If you need to override a data definition metafile while you are testing an application upgrade or editing the specifications, make the new application metafile available locally to the client Tivoli Universal Agent. A metafile client always checks for a local copy of the required metafile before attempting to download it from the metafile server.

9

Integrating data from external or third-party applications into Tivoli Data Warehouse

This chapter provides information about how to integrate data provided by custom or third-party data sources into the Tivoli Data Warehouse Version 2.1. It also introduces ways of presenting information from third-party warehouses and auxiliary data sources through the Tivoli Enterprise Portal (portal).

The following topics are addressed:

- Warehousing data using IBM Tivoli Monitoring V6.1 Universal Agent (script provider)
- Warehousing data using IBM Tivoli Monitoring V6.1 Universal Agent (ODBC provider)
- Tivoli Storage Manager Universal Agent in the Tivoli Enterprise Portal
- Viewing data in the Tivoli Enterprise Portal Server using an external ODBC data source

9.1 Warehousing data using IBM Tivoli Monitoring V6.1 Universal Agent (script provider)

In this section, we provide a practical example of how to configure the Tivoli Monitoring 6.1 Universal Agent. In doing so, we describe how data can be warehoused to the Tivoli Data Warehouse and viewed using the Tivoli Enterprise Portal client.

9.1.1 Configuring the Tivoli Universal Agent

The example we explore is a Universal Agent that is used to gather data disk metrics from an AIX system. To configure this agent, perform the following steps:

1. Install and configure Universal Agent. This is covered in detail in *IBM Tivoli Monitoring Installation and Setup Guide*, GC32-9407, and *IBM Tivoli Monitoring Universal Agent User's Guide*, SC32-9459. In our lab environment, we installed the agent on the AIX system (belfast). We installed the agent under (/opt/IBM/ITM/).
2. After you install the agent, create the mdl file for the agent. Create this file and place it in the (/opt/IBM/ITM/bin) directory. Example 9-1 shows the mdl file that we used.

Example 9-1 AIX disk metric .mdl example

```
/ITM61/bin/disk.mdl
//APPL DiskData
//Name DiskUsageStats S 1440 AddTimeStamp
//Source script /usr/bin/sh /ITM61/bin/disk.sh Interval=60
//Attributes
HostName (GetEnvValue = HOSTNAME)
Script (GetEnvValue = PROBE)
FileSystem      D 120
BlockSize       C 2147483647
SpaceAvailable  C 2147483647
Perc_Used       C 2147483647
Inodes_Used     C 2147483647
Perc_Inodes_Used    C 2147483647
MountPoint      D 120
```

3. Create a script that provides the required information to the Universal Agent. In our case, it was a simple shell script that was created under /opt/IBM/ITM/bin and is referenced in the .mdl example shown in Example 9-1 on page 272. Example 9-2 shows the script we used.

Example 9-2 AIX disk metrics query disk.sh script example

```
/usr/bin/df -k |grep -v Filesystem
```

4. After you create the .mdl file and data provider, register the .mdl file with the Universal Agent so that it knows the format of the data it will be receiving and the script that is used to provide this data. To do this, run the commands shown in Example 9-3.

Example 9-3 Commands used to import .mdl file on UNIX and Linux

```
export CANDLEHOME=/ITM61
cd $CANDLEHOME/bin

./um_console
KUMPS002I Enter console command <Application name or Metafile name or
file name>
import /ITM61/bin/disk.mdl
KUMPS001I Console input accepted.
KUMPS020I Import successfully completed for /ITM61/bin/disk.mdl
```

5. After you complete this, press Enter to exit.
6. Restart the Universal Agent.

9.1.2 Viewing data in the Tivoli Enterprise Portal

To view the data that is being collected by the newly configured Universal Agent, perform the following steps:

1. Log on to either your portal desktop client or your portal Web client.
2. After you log on, a new agent is displayed. You can view it by selecting **Enterprise**. Then select **UNIX Systems**, then **Hostname**. Finally, select **Universal Agent**. This is shown in Figure 9-1.

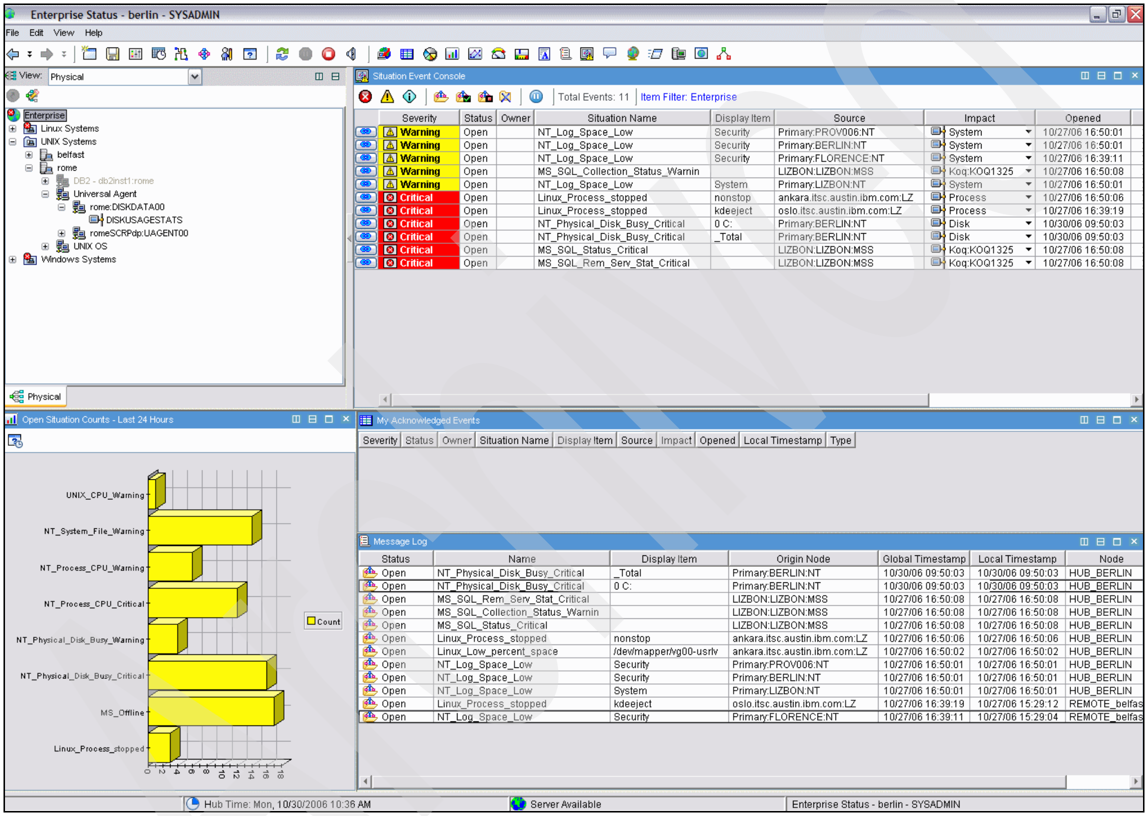


Figure 9-1 AIX disk Universal Agent in the portal

3. When you click this Universal Agent, it shows all the data that is being collected by the script, running under the Universal Agent; see Figure 9-2.

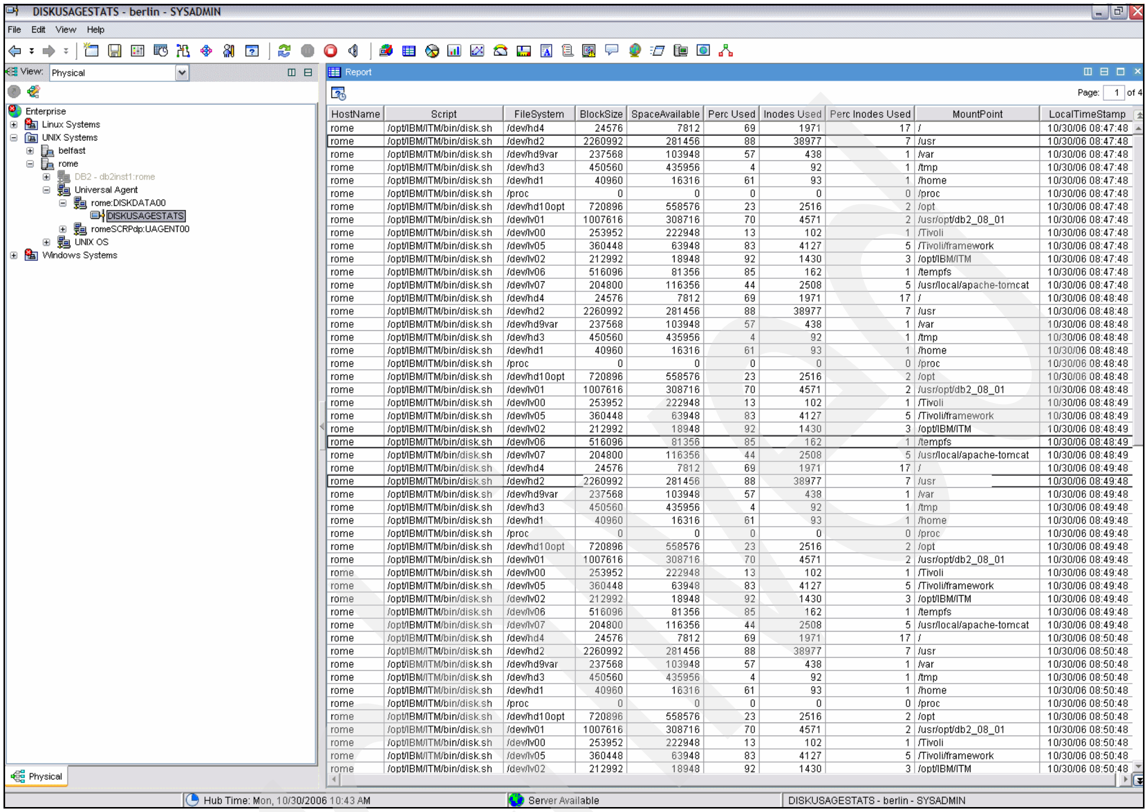


Figure 9-2 DISKDATA data view in portal

9.1.3 Warehousing the Universal Agent data

To warehouse the data that is being collected by the configured Universal Agent, perform the following steps:

1. In the tool bar, select the **History Configuration** button, as shown in Figure 9-3.

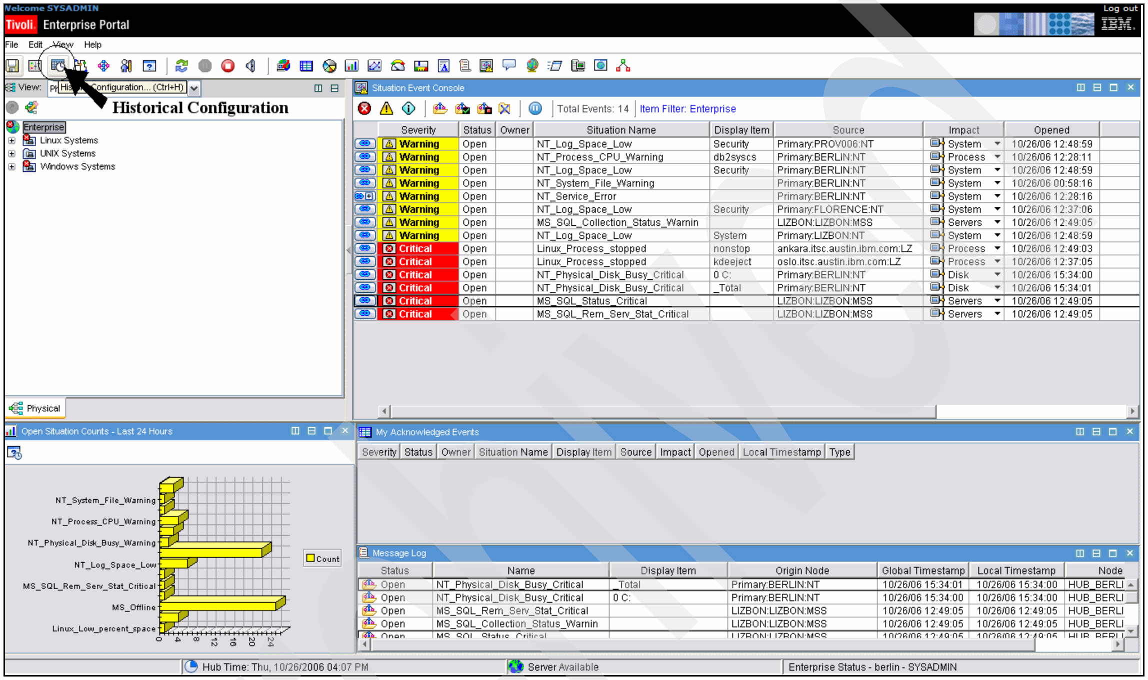


Figure 9-3 Portal Historical Configuration tab

2. The History Collection Configuration window opens. In the Select a product drop-down list, select the name (//APPL DiskData), which was given in the mdl file to the Universal Agent. This was DISKDATA, in our example; see Figure 9-4.

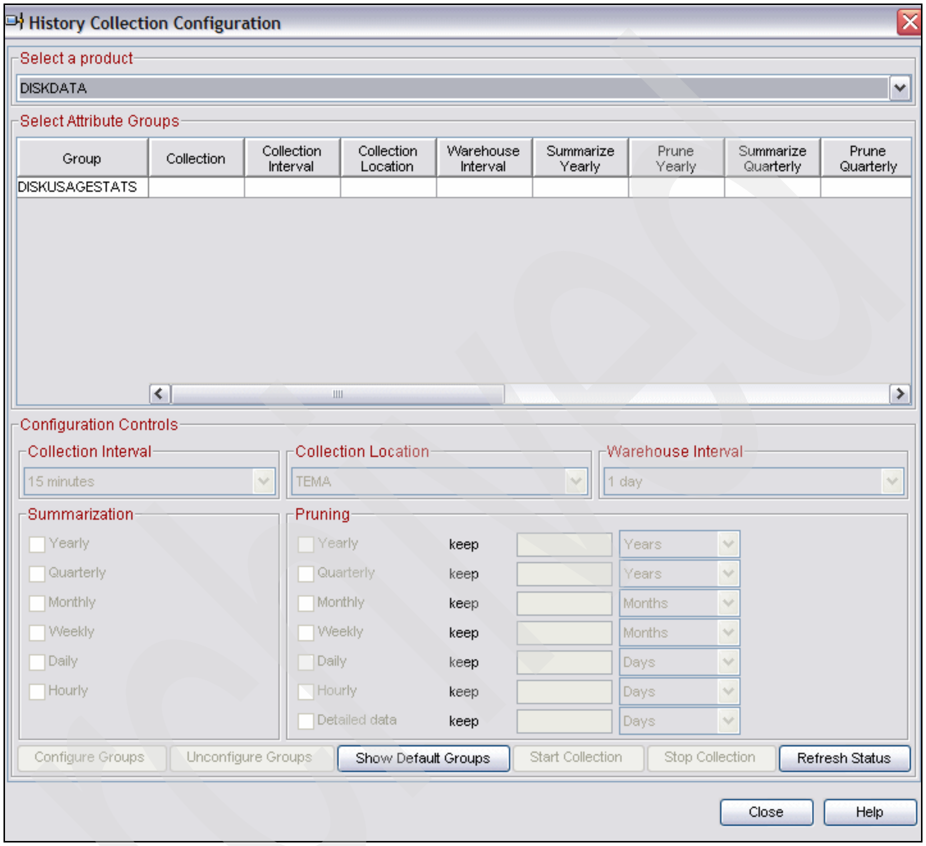


Figure 9-4 DISKDATA history collection configuration window

3. In the Select Attribute Groups list, click the group name. This enables the Configuration Controls options. You can specify the Collection Interval.

   Four options are available. The data for metrics presented under the DiskUsageStats attribute group can be collected every 5, 15, 30 or every 60 minutes, as shown in Figure 9-5 on page 278.

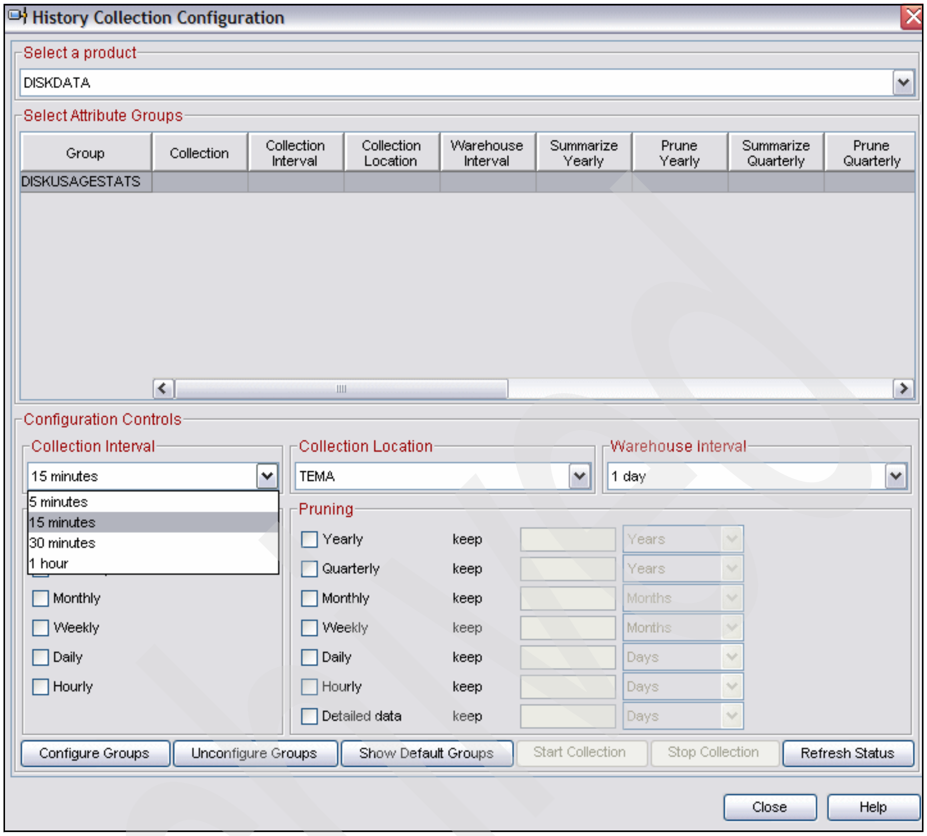


Figure 9-5 DISKDATA collection interval configuration window

4. You can select the collection location. Two options are available: Historical data can be collected at either the Tivoli Enterprise Monitoring Agent, or at the Tivoli Enterprise Monitoring Server (management server, hub or remote).

   We recommend that you specify to collect data at the Tivoli Enterprise Monitoring Agent location, as shown in Figure 9-6 on page 279.

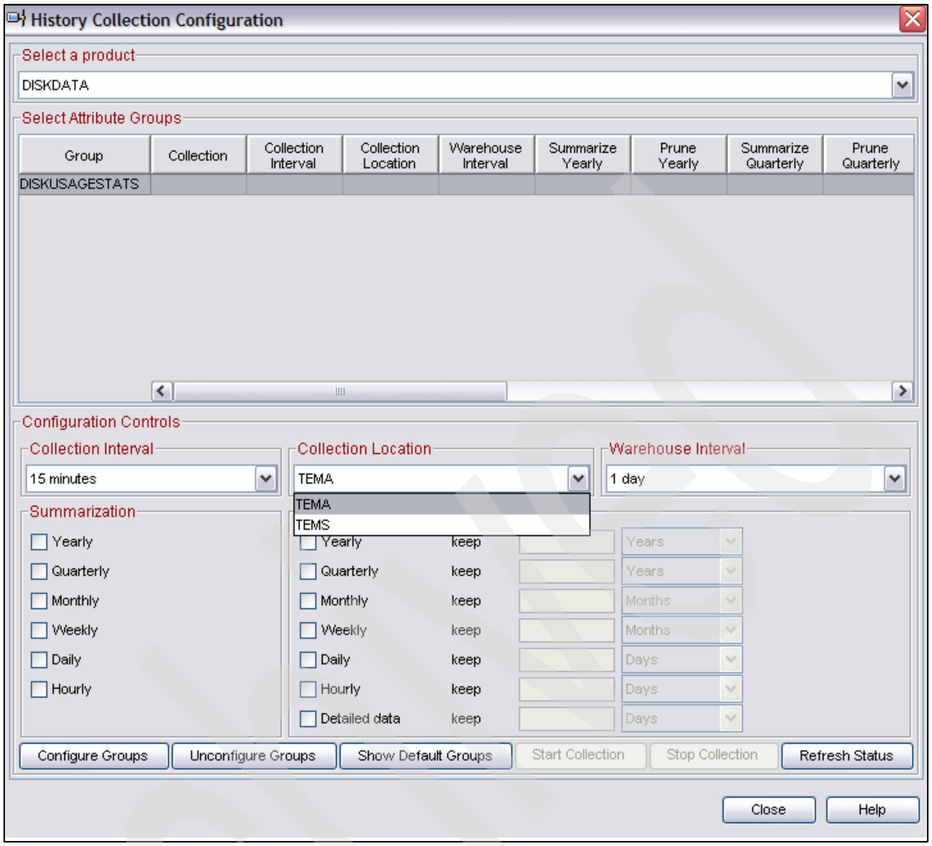


Figure 9-6 DISKDATA collection location configuration window

5. Select the warehouse interval. Two options are available: Historical data can be uploaded to Warehouse using the Warehouse Proxy agent ever hour, or every 24 hours, as shown in Figure 9-7.

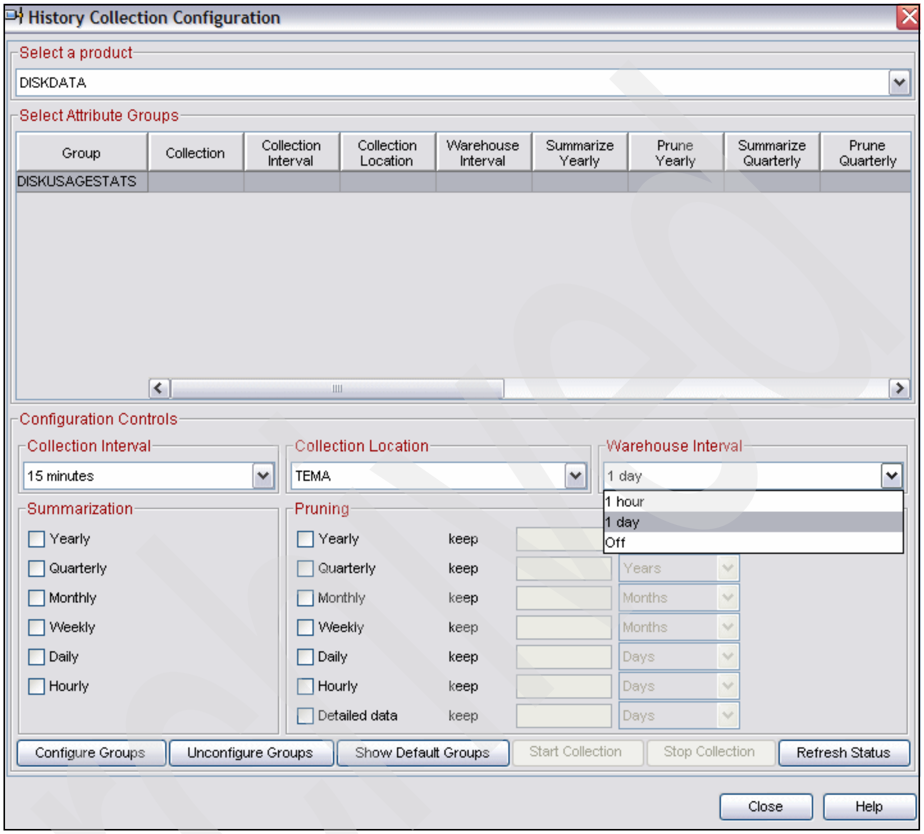


Figure 9-7 DISKDATA warehouse interval configuration window

6. After you select all the options under the configuration controls, select the **Configure Groups** button to save these changes. You see the saved options in the updated panel, as shown in Figure 9-8. You can see that the options are set to collect data every 5 minutes, which will be stored at the Tivoli Enterprise Monitoring Agent and written to the warehouse every hour.

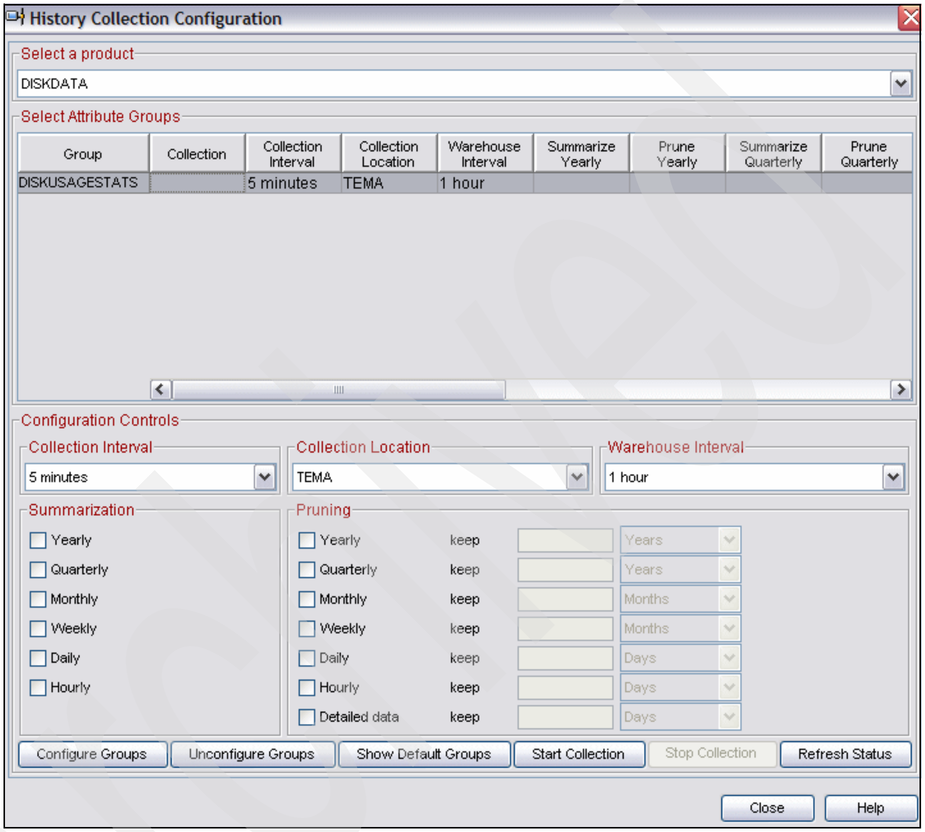


Figure 9-8 DISKDATA configuration window

> **Note:** Configuring the collection options for an attribute group does not start collection for that attribute group. You have to start the collection manually. We describe how to do this in more detail in the following steps.

7. To start the data collection, click the **Start Collection** button to enable historical collection for the DiskUsageStats Attribute Group.

   The Collection field in the Select Attribute Groups section shows the status of the agent as `Started`, as shown in Figure 9-9 on page 282 (that is, the agent has started collection).

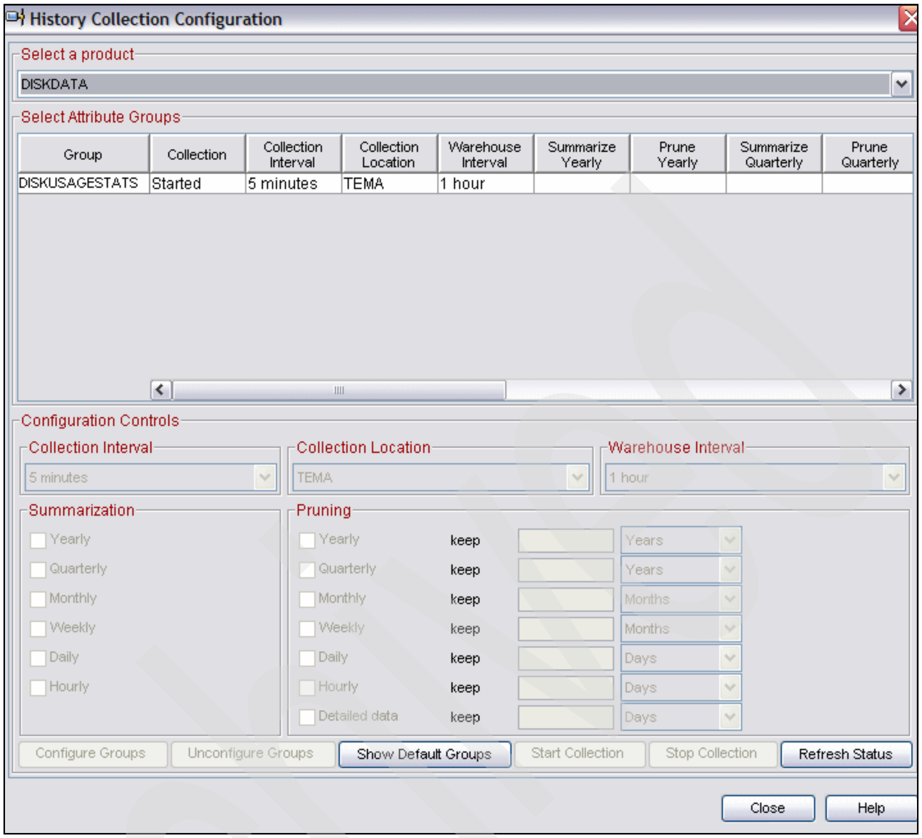


Figure 9-9 DISKDATA started status in configuration window

The history collection has now started. You can confirm this by checking the Universal Agents view, which displays a date/time icon, as shown in Figure 9-10.

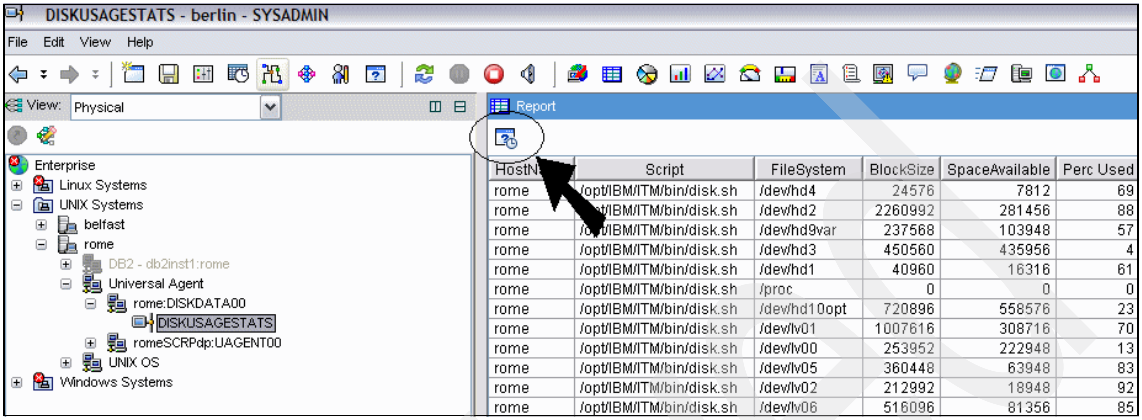


Figure 9-10 Portal DISKDATA date/time icon

To estimate the amount of data collected with the Universal Agent application, refer to 4.3.3, “Estimate the amount of historical data for Universal Agent applications” on page 96.

9.1.4 Creating graphical views for historical data

After you configure the historical data collection, you can create different views from this data, such as a bar chart. Figure 9-11 shows an example of a customized view.

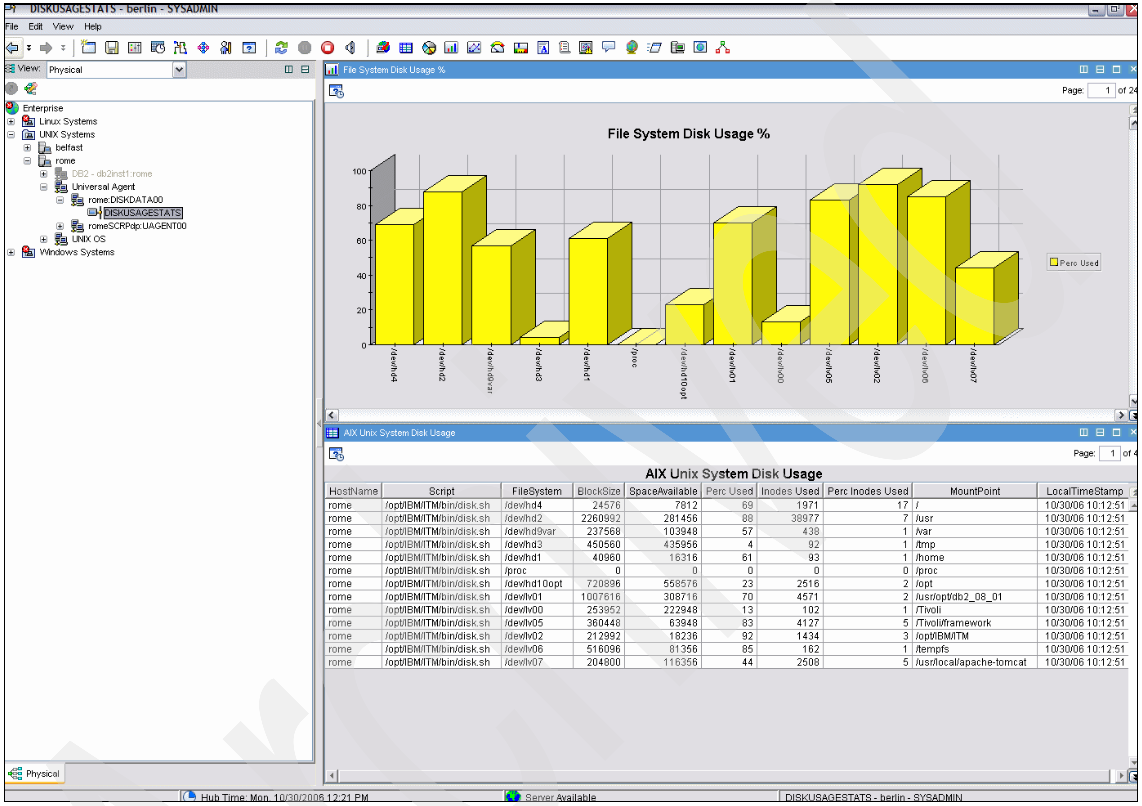


Figure 9-11 Customized portal workspaces example

To create a graphical view from table data format, perform the following steps:

1. From the list of icons in the menu, select **Bar Chart**, drag the icon, and drop it on the table view, as shown in Figure 9-12 on page 285.

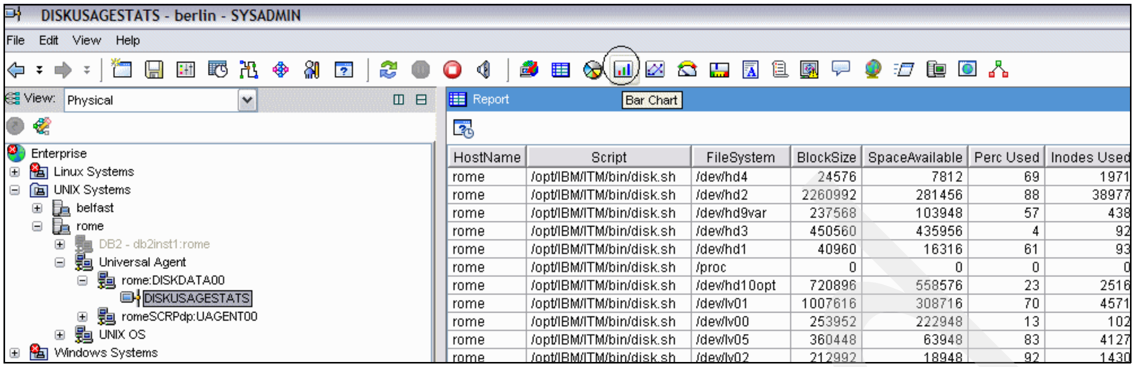


Figure 9-12 Bar chart selection icon

2. The Select attribute window opens. In the Attribute item field, select **Perc Used** metric and click **OK**, as shown in Figure 9-13.

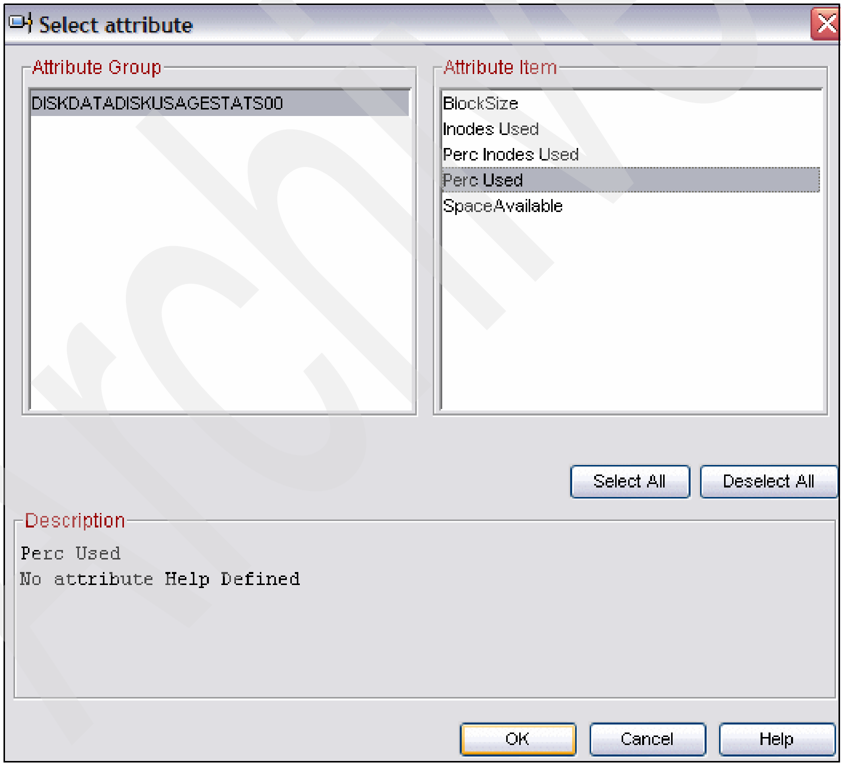


Figure 9-13 Bar chart attribute selection window

3. A view with the bar charts opens with the Perc Used data, as shown in Figure 9-14. You can customize this view. To do this, right-click the view and select **Properties**.

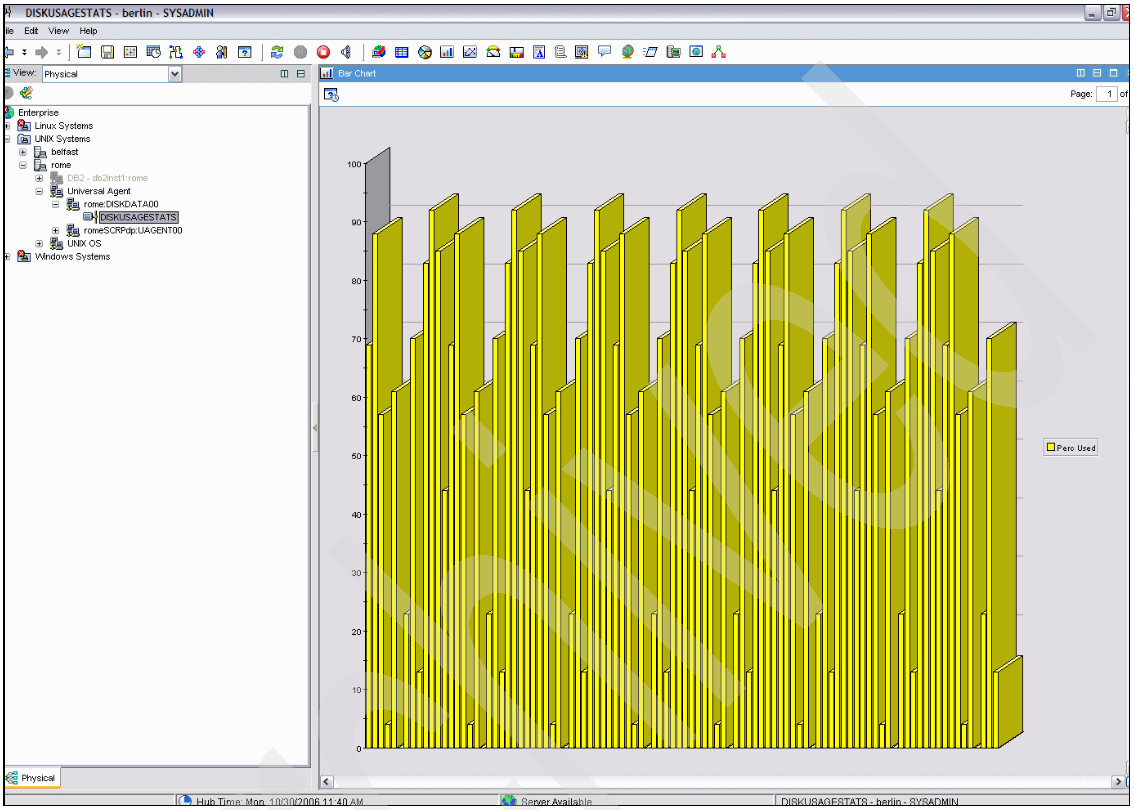


Figure 9-14 Bar chart data view

A window opens, as shown in Figure 9-15.

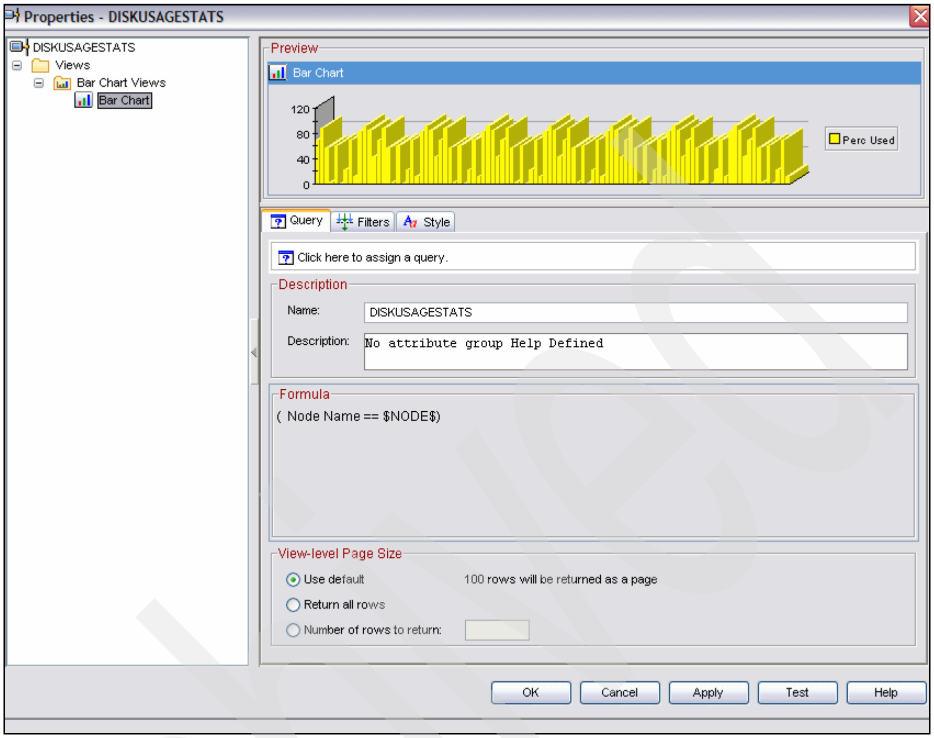


Figure 9-15 Bar chart properties customization window

4. Select the **Style** tab and enter the text `AIX UNIX System Disk Usage Data`. Select the **Show** check box, and then click **Apply** and **OK**, as shown in Figure 9-16.

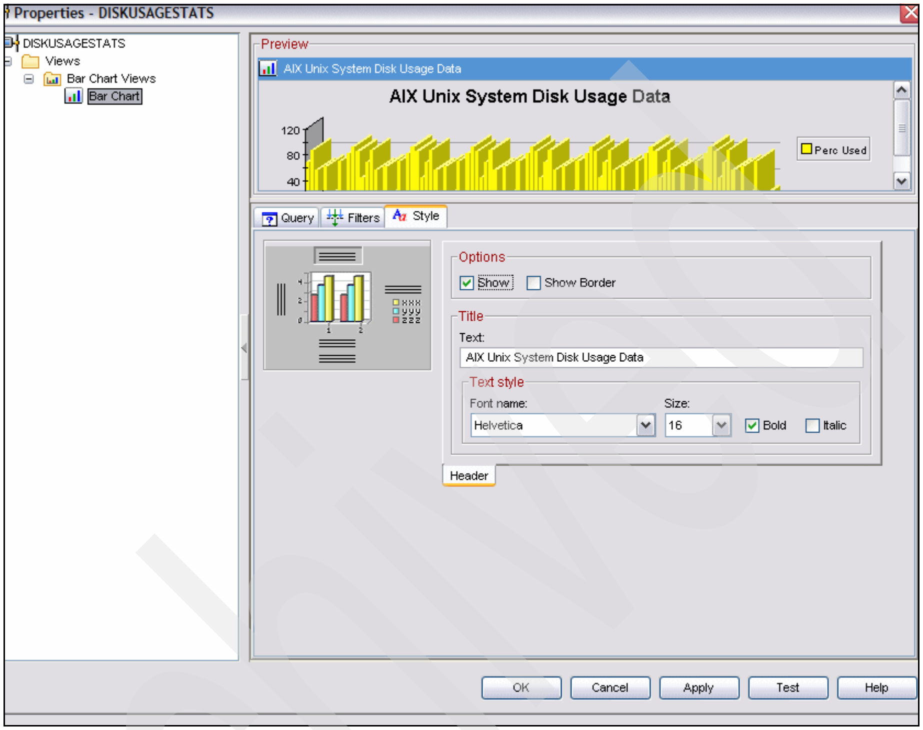


Figure 9-16 Bar chart style parameters properties window

5. The bar chart view opens showing the text that you entered, as shown in Figure 9-17. To view the historical data, click the **Date/Time icon**.

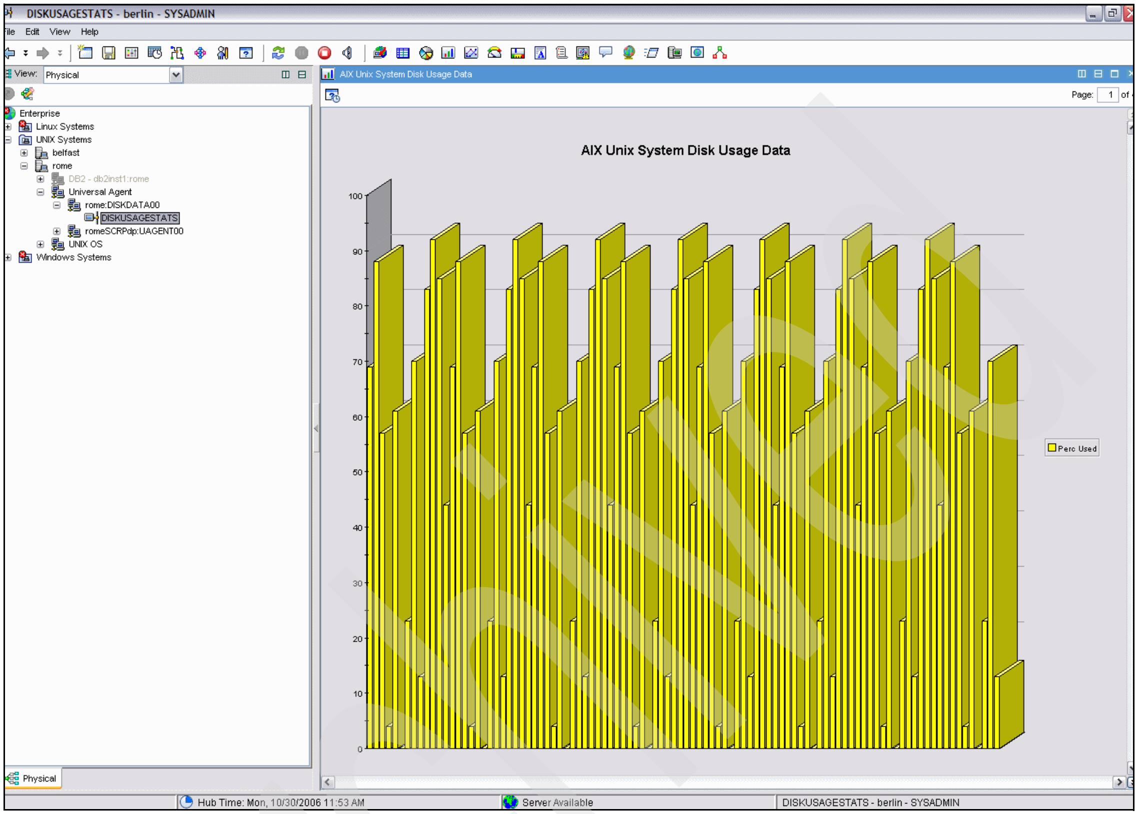


Figure 9-17 Stylized bar chart workspace

6. The Select the Time Span window opens. Select the **Last** option, and specify 2 to show data for the last two days, as shown in Figure 9-18 on

page 290. This makes a direct query in the warehouse database and gets the requested data.

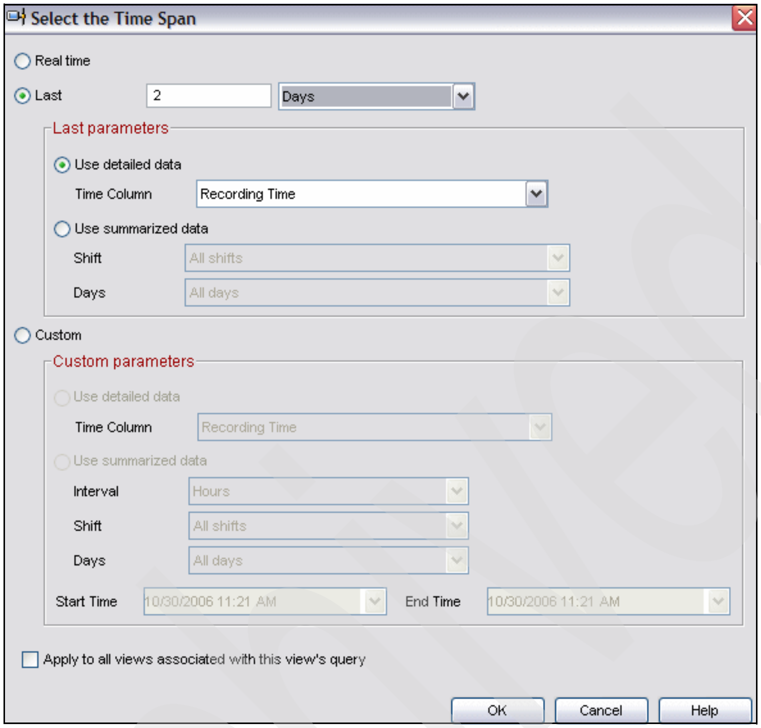


Figure 9-18 Historical time span view configuration window

7. The view now retrieves all the rows for the last two days, as shown in Figure 9-19. You can customize this by dragging and dropping the graphical icons to view the historical data.

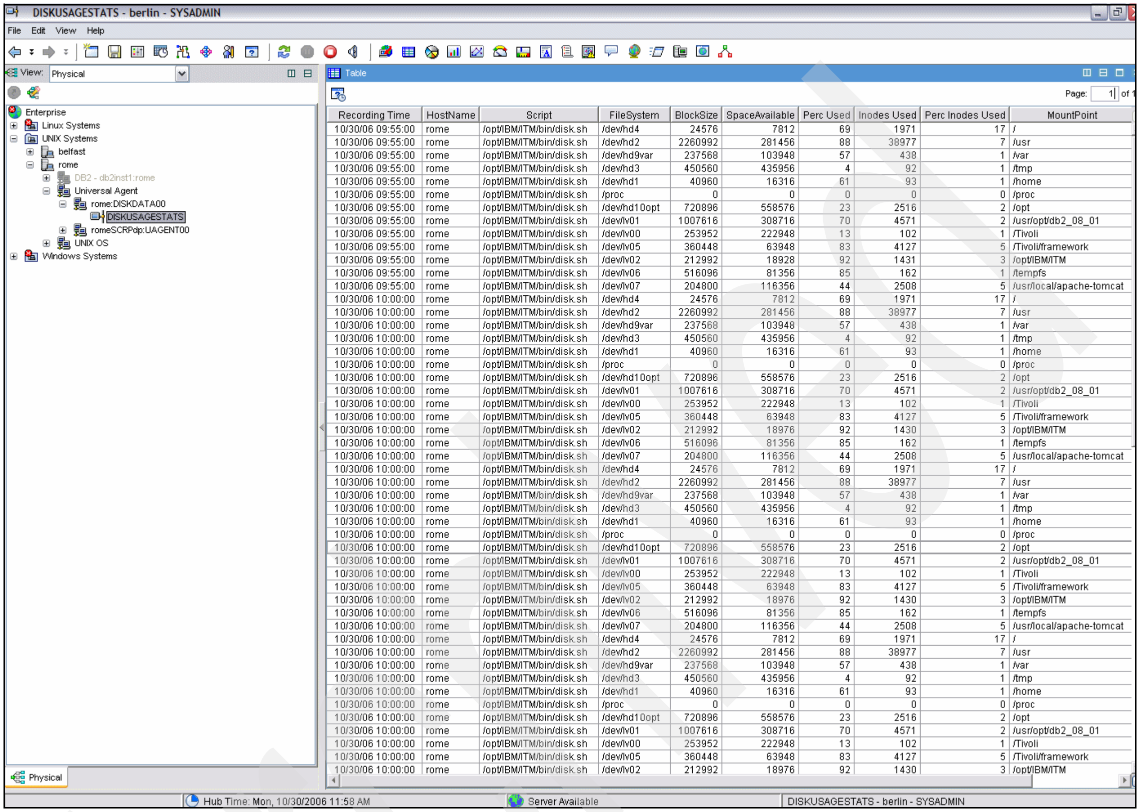


Figure 9-19 Two-day historical data view

This opens the plot graph shown in Figure 9-20.

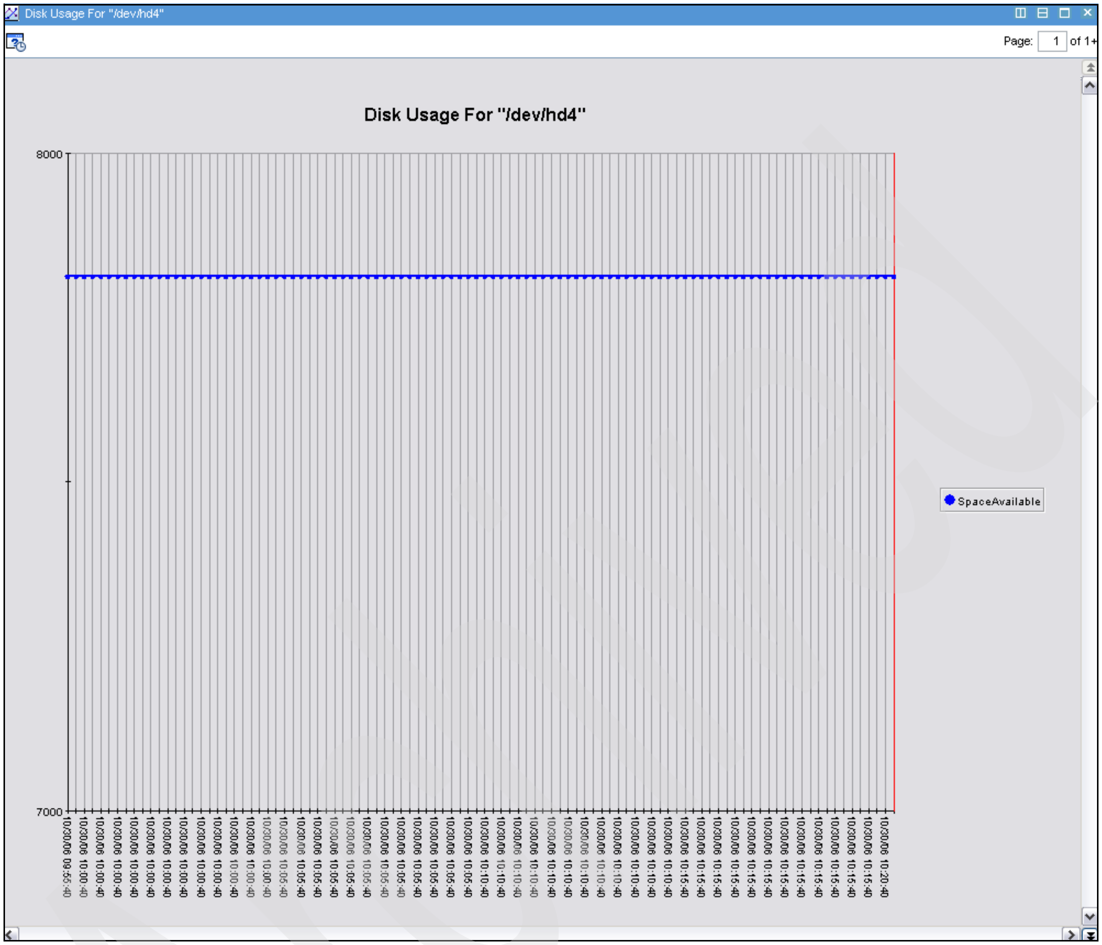


Figure 9-20 Historical data plot graph view

9.2 Warehousing data using IBM Tivoli Monitoring 6.1 Universal Agent (ODBC provider)

In this section, we provide another practical example of how to configure the Tivoli Monitoring 6.1 Universal Agent. In doing so, we describe how data can be warehoused to the Tivoli Data Warehouse and viewed using the Tivoli Enterprise Portal client.

This solution monitors Tivoli Storage Manager by providing reports similar to the *Daily Reports* provided by Tivoli Storage Manager. The solution uses the Tivoli Storage Manager ODBC data provider to extract useful metrics about the working condition of your Tivoli Storage Manager server. This provides you with useful data about the performance and availability characteristics of your Tivoli Storage Manager server including:

- Client schedules
- Administrative schedules
- Disk pool information
- Tape pool information
- Data movement (restore/backup)
- User and administrator information

You can use the collected information to perform trending analysis on your Tivoli Storage Manager server.

9.2.1 Configuring the Tivoli Universal Agent

The example that we provide is a Universal Agent that is used to gather data from a Tivoli Storage Manager database residing on a z/OS system. To configure this agent, perform the following steps:

1. Install and configure the Universal Agent. This is covered in detail in *IBM Tivoli Monitoring Installation and Setup Guide*, GC32-9407, and *IBM Tivoli Monitoring Universal Agent User's Guide*, SC32-9459. In our lab environment, we installed the agent on a Windows system (toronto) because of the dependency on ODBC. The ODBC data provider is only supported on a Windows environment. The agent was installed under (c:\IBM\ITM).
2. After you install the Universal Agent, configure it so that it has to use the ODBC data provider. To do this, edit the c:\ibm\itm\tmaitm6\kumenv file.

   Change the tag KUMA_STARTUP_DP=ASFS to read KUMA_STARTUP_DP=ODBC. After you do this, save the ENV file and restart the Universal Agent.

3. The ODBC data provider requires an ODBC data source. In our case, we installed and configured a data source using the Tivoli Storage Manager ODBC drivers located on the Tivoli Storage Manager client setup disks. For detailed information about how to install and configure a Tivoli Storage Manager ODBC connection, see:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp?topic=/com.ibm.itsmreadme.doc/WINDOWS/ODBC/README_odbc_enu.htm

4. After you install, configure, and test the ODBC data source, create the mdl file for the agent. The ODBC mdl file is heavily dependent on knowing the database schema of the database that you are querying, because direct SQL statements are used. The attribute groups are also mapped to table names, therefore it is essential to know the schema that you want to use this provider on. Create the file and place it in the (c:\ibm\itm\tmaitm6\metafiles) directory.

> **Important:** It is essential to modify the .mdl file and parameters to suit your environment. In our case, we used a system DSN named TSM ODBC as the data source, and the user name and password we used for querying the Tivoli Storage Manager server were both admin.

Note: It is possible to automatically build a metafile for the database that you are querying. To do this, issue the GENERATE command.

The GENERATE command automatically builds a complete and syntactically correct ODBC metafile when given a data source name as input. This command supports full generation of all tables that are defined to the specified data source. You can also limit the tables that are generated by selecting user tables, system tables, views, or some combination of the three, and specify a beginning string of characters to pattern match against any of the three table types.

The GENERATE command does not support metafile creation for stored procedures. You can start this command even when the IBM Tivoli Universal Agent is not running. GENERATE is only accessible on Windows operating systems and only through the kumpcon console interface. The syntax is as follows:

```
<ITMInstalldir>\tmaitm6\KUMPCON GENERATE dataSourceName
user=userid pswd=password
```

In this syntax:

- *<dataSourceName>* indicates the specific name of the configured data source that is used to create the ODBC metafile. This parameter is required. If the data source contains any embedded blanks, it must be surrounded with single quotation marks.
- *<userid>* is the user ID that connects to the ODBC data source. If no user and password combination is required for this particular data source, then you can omit the user= parameter from the GENERATE command.
- *<password>* is the password that is associated with the user ID connecting to the ODBC data source. If specified, the user and pswd values are inserted into every //SOURCE statement in the generated ODBC metafile.

In our case, we used the following command to generate the .mdl file:

```
c:\ibm\itm\tmaitm6\kumpcon GENERATE TSMODBC user=admin pswd=admin
```

5. After you create the .mdl file, register it with the Universal Agent so that it knows the format of the data that it will be receiving and the SQL statements that are used to provide this data. To do this, run the commands shown in Example 9-4.

Example 9-4 Commands used to import .mdl file on Windows

```
cd\ibm\itm\tmaitm6

kumpcon validate tsm.mdl
KUMPS001I Console input accepted.
KUMPV025I Processing input metafile
C:\IBM\ITM\TMAITM6\metafiles\tsm.mdl
KUMPV026I Processing record 0001 -> //APPL TSM
KUMPV149I Note: APPL names starting with letters N-Z are designated for
customer UA solutions.
KUMPV026I Processing record 0002 -> //NAME ACTLOG S 300  @Server
activity log
KUMPV026I Processing record 0003 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0004 -> //SQL Select * from ACTLOG
KUMPV026I Processing record 0005 -> //ATTRIBUTES
KUMPV026I Processing record 0006 -> DATE_TIME   D  28  @Date/Time
KUMPV026I Processing record 0007 -> MSGNO       N  8  @Message number
KUMPV026I Processing record 0008 -> SEVERITY    D  4  @Message severity
KUMPV026I Processing record 0009 -> MESSAGE     D  256  @Message
KUMPV026I Processing record 0010 -> ORIGINATOR  D  20  @Originator
KUMPV026I Processing record 0011 -> NODENAME    D  64  @Node Name
KUMPV026I Processing record 0012 -> OWNERNAME   D  64  @Owner Name
KUMPV026I Processing record 0013 -> SCHEDNAME   D  32  @Schedule Name
KUMPV026I Processing record 0014 -> DOMAINNAME  D  32  @Policy Domain
Name
KUMPV026I Processing record 0015 -> SESSID      N  8  @Sess Number
KUMPV026I Processing record 0016 -> SERVERNAME  D  64  @Server Name
KUMPV026I Processing record 0017 -> SESSION     N  8  @SESSION
KUMPV026I Processing record 0018 -> PROCESS     N  8  @PROCESS
KUMPV026I Processing record 0019 -> //NAME ADMINS K 300  @Server
administrators
KUMPV026I Processing record 0020 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0021 -> //SQL Select * from ADMINS
KUMPV026I Processing record 0022 -> //ATTRIBUTES
KUMPV026I Processing record 0023 -> ADMIN_NAME       D  64  KEY ATOMIC
@Administrator Name
KUMPV026I Processing record 0024 -> LASTACC_TIME      D  28  @Last
Access Date/Time
```

```
KUMPV026I Processing record 0025 -> PWSET_TIME        D  28  @Password
Set Date/Time
KUMPV026I Processing record 0026 -> CONTACT           D  128  @Contact
KUMPV026I Processing record 0027 -> LOCKED            D  12  @Locked?
KUMPV026I Processing record 0028 -> INVALID_PW_COUNT  C  999999
@Invalid Sign-on Count
KUMPV026I Processing record 0029 -> SYSTEM_PRIV       D  80  @System
Privilege
KUMPV026I Processing record 0030 -> POLICY_PRIV       D  100  @Policy
Privilege
KUMPV026I Processing record 0031 -> STORAGE_PRIV      D  100  @Storage
Privilege
KUMPV026I Processing record 0032 -> ANALYST_PRIV      D  80  @Analyst
Privilege
KUMPV026I Processing record 0033 -> OPERATOR_PRIV     D  80  @Operator
Privilege
KUMPV026I Processing record 0034 -> CLIENT_ACCESS     D  256  @Client
Access Privilege
KUMPV026I Processing record 0035 -> CLIENT_OWNER      D  256  @Client
Owner Privilege
KUMPV026I Processing record 0036 -> REG_TIME          D  28
@Registration Date/Time
KUMPV026I Processing record 0037 -> REG_ADMIN         D  64
@Registering Administrator
KUMPV026I Processing record 0038 -> PROFILE          D  256  @Managing
profile
KUMPV026I Processing record 0039 -> PASSEXP           N  8  @Password
Expiration Period
KUMPV026I Processing record 0040 -> //NAME ADMIN_SCHEDULES K 300
@Administrative command schedules
KUMPV026I Processing record 0041 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0042 -> //SQL Select * from ADMIN_SCHEDULES
KUMPV026I Processing record 0043 -> //ATTRIBUTES
KUMPV026I Processing record 0044 -> SCHEDULE_NAME  D  32  KEY ATOMIC
@Schedule Name
KUMPV026I Processing record 0045 -> DESCRIPTION    D  256  @Description
KUMPV026I Processing record 0046 -> COMMAND        D  256  @Command
KUMPV026I Processing record 0047 -> PRIORITY       C  999999  @Priority
KUMPV026I Processing record 0048 -> STARTDATE      D  12  @Start date
KUMPV026I Processing record 0049 -> STARTTIME      D  8  @Start time
KUMPV026I Processing record 0050 -> DURATION       N  8  @Duration
KUMPV026I Processing record 0051 -> DURUNITS       D  20  @Duration
units
KUMPV026I Processing record 0052 -> PERIOD         N  8  @Period
```

```
KUMPV026I Processing record 0053 -> PERUNITS       D  20  @Period units
KUMPV026I Processing record 0054 -> DAYOFWEEK      D  20  @Day of Week
KUMPV026I Processing record 0055 -> EXPIRATION     D  12  @Expiration
KUMPV026I Processing record 0056 -> ACTIVE         D  12  @Active?
KUMPV026I Processing record 0057 -> CHG_TIME       D  28  @Last Update
Date/Time
KUMPV026I Processing record 0058 -> CHG_ADMIN      D  32  @Last Update
by (administrator)
KUMPV026I Processing record 0059 -> PROFILE        D  256  @Managing
profile
KUMPV026I Processing record 0060 -> SCHED_STYLE    D  12  @Schedule
Style
KUMPV026I Processing record 0061 -> ENH_MONTH      D  52  @Month
KUMPV026I Processing record 0062 -> DAYOFMONTH     D  60  @Day of Month
KUMPV026I Processing record 0063 -> WEEKOFMONTH    D  52  @Week of
Month
KUMPV026I Processing record 0064 -> //NAME ARCHIVES S 300  @Client
archive files
KUMPV026I Processing record 0065 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0066 -> //SQL Select * from ARCHIVES
KUMPV026I Processing record 0067 -> //ATTRIBUTES
KUMPV026I Processing record 0068 -> NODE_NAME       D  64  @Node Name
KUMPV026I Processing record 0069 -> FILESPACE_NAME  D  256  @Filespace
Name
KUMPV026I Processing record 0070 -> FILESPACE_ID    N  28  @FSID
KUMPV026I Processing record 0071 -> TYPE            D  16  @Object type
KUMPV026I Processing record 0072 -> HL_NAME         D  256  @Client
high-level name
KUMPV026I Processing record 0073 -> LL_NAME         D  256  @Client
low-level name
KUMPV026I Processing record 0074 -> OBJECT_ID       N  20  @Server
object ID for the client object
KUMPV026I Processing record 0075 -> ARCHIVE_DATE    D  28  @Date/time
that the object was archived
KUMPV026I Processing record 0076 -> OWNER           D  64  @Client
object owner
KUMPV026I Processing record 0077 -> DESCRIPTION     D  256
@Description
KUMPV026I Processing record 0078 -> CLASS_NAME      D  32  @Mgmt Class
Name
KUMPV026I Processing record 0079 -> //NAME AR_COPYGROUPS S 300
@Management class archive copy groups
KUMPV026I Processing record 0080 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
```

```
KUMPV026I Processing record 0081 -> //SQL Select * from AR_COPYGROUPS
KUMPV026I Processing record 0082 -> //ATTRIBUTES
KUMPV026I Processing record 0083 -> DOMAIN_NAME     D  32  @Policy
Domain Name
KUMPV026I Processing record 0084 -> SET_NAME        D  32  @Policy Set
Name
KUMPV026I Processing record 0085 -> CLASS_NAME      D  32  @Mgmt Class
Name
KUMPV026I Processing record 0086 -> COPYGROUP_NAME  D  32  @Copy Group
Name
KUMPV026I Processing record 0087 -> RETVER          D  8  @Retain
Version
KUMPV026I Processing record 0088 -> SERIALIZATION   D  32  @Copy
Serialization
KUMPV026I Processing record 0089 -> DESTINATION     D  32  @Copy
Destination
KUMPV026I Processing record 0090 -> CHG_TIME       D  28  @Last Update
Date/Time
KUMPV026I Processing record 0091 -> CHG_ADMIN      D  32  @Last Update
by (administrator)
KUMPV026I Processing record 0092 -> PROFILE         D  256  @Managing
profile
KUMPV026I Processing record 0093 -> RETINIT         D  8  @Retention
Initiation
KUMPV026I Processing record 0094 -> RETMIN          N  8  @Retain
Minimum Days
KUMPV026I Processing record 0095 -> //NAME ASSOCIATIONS S 300  @Client
schedule associations
KUMPV026I Processing record 0096 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0097 -> //SQL Select * from ASSOCIATIONS
KUMPV026I Processing record 0098 -> //ATTRIBUTES
KUMPV026I Processing record 0099 -> DOMAIN_NAME    D  32  @Policy
Domain Name
KUMPV026I Processing record 0100 -> SCHEDULE_NAME  D  32  @Schedule
Name
KUMPV026I Processing record 0101 -> NODE_NAME      D  64  @Associated
Nodes
KUMPV026I Processing record 0102 -> CHG_TIME       D  28  @Last Update
Date/Time
KUMPV026I Processing record 0103 -> CHG_ADMIN      D  32  @Last Update
by (administrator)
KUMPV026I Processing record 0104 -> //NAME AUDITOCC K 300  @Server
audit occupancy results
```

```
KUMPV026I Processing record 0105 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0106 -> //SQL Select * from AUDITOCC
KUMPV026I Processing record 0107 -> //ATTRIBUTES
KUMPV026I Processing record 0108 -> NODE_NAME       D  64  KEY ATOMIC
@Node Name
KUMPV026I Processing record 0109 -> BACKUP_MB        N  8  @Backup
Storage Used (MB)
KUMPV026I Processing record 0110 -> BACKUP_COPY_MB   N  8  @Backup
Storage Used (MB)
KUMPV026I Processing record 0111 -> ARCHIVE_MB       N  8  @Archive
Storage Used (MB)
KUMPV026I Processing record 0112 -> ARCHIVE_COPY_MB  N  8  @Archive
Storage Used (MB)
KUMPV026I Processing record 0113 -> SPACEMG_MB       N  8
@Space-Managed Storage Used (MB)
KUMPV026I Processing record 0114 -> SPACEMG_COPY_MB  N  8
@Space-Managed Storage Used (MB)
KUMPV026I Processing record 0115 -> TOTAL_MB         N  8  @Total
Storage Used (MB)
KUMPV026I Processing record 0116 -> //NAME BACKUPS S 300  @Client
backup files
KUMPV026I Processing record 0117 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0118 -> //SQL Select * from BACKUPS
KUMPV026I Processing record 0119 -> //ATTRIBUTES
KUMPV026I Processing record 0120 -> NODE_NAME        D  64  @Node Name
KUMPV026I Processing record 0121 -> FILESPACE_NAME   D  256  @Filespace
Name
KUMPV026I Processing record 0122 -> FILESPACE_ID     N  28  @FSID
KUMPV026I Processing record 0123 -> STATE           D  16  @File state
(active, inactive)
KUMPV026I Processing record 0124 -> TYPE             D  16  @Object
type
KUMPV026I Processing record 0125 -> HL_NAME          D  256  @Client
high-level name
KUMPV026I Processing record 0126 -> LL_NAME          D  256  @Client
low-level name
KUMPV026I Processing record 0127 -> OBJECT_ID        N  20  @Server
object ID for the client object
KUMPV026I Processing record 0128 -> BACKUP_DATE      D  28  @Date/time
that the object was backed up
KUMPV026I Processing record 0129 -> DEACTIVATE_DATE  D  28  @Date/time
that the object was deactivated
```

```
KUMPV026I Processing record 0130 -> OWNER            D  64  @Client
object owner
KUMPV026I Processing record 0131 -> CLASS_NAME       D  32  @Mgmt Class
Name
KUMPV026I Processing record 0132 -> //NAME BACKUPSETS S 300  @Backup
Set
KUMPV026I Processing record 0133 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0134 -> //SQL Select * from BACKUPSETS
KUMPV026I Processing record 0135 -> //ATTRIBUTES
KUMPV026I Processing record 0136 -> NODE_NAME       D  64  @Node Name
KUMPV026I Processing record 0137 -> BACKUPSET_NAME  D  256  @Backup Set
Name
KUMPV026I Processing record 0138 -> OBJECT_ID       N  20  @Server
object ID for the client object
KUMPV026I Processing record 0139 -> DATE_TIME       D  28  @Date/Time
KUMPV026I Processing record 0140 -> RETENTION       D  8  @Retention
Period
KUMPV026I Processing record 0141 -> DESCRIPTION     D  256
@Description
KUMPV026I Processing record 0142 -> DEVCLASS        D  32  @Device
Class Name
KUMPV026I Processing record 0143 -> //NAME BU_COPYGROUPS S 300
@Management class backup copy groups
KUMPV026I Processing record 0144 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0145 -> //SQL Select * from BU_COPYGROUPS
KUMPV026I Processing record 0146 -> //ATTRIBUTES
KUMPV026I Processing record 0147 -> DOMAIN_NAME      D  32  @Policy
Domain Name
KUMPV026I Processing record 0148 -> SET_NAME         D  32  @Policy Set
Name
KUMPV026I Processing record 0149 -> CLASS_NAME       D  32  @Mgmt Class
Name
KUMPV026I Processing record 0150 -> COPYGROUP_NAME   D  32  @Copy Group
Name
KUMPV026I Processing record 0151 -> VEREXISTS        D  8  @Versions
Data Exists
KUMPV026I Processing record 0152 -> VERDELETED       D  8  @Versions
Data Deleted
KUMPV026I Processing record 0153 -> RETEXTRA         D  8  @Retain
Extra Versions
KUMPV026I Processing record 0154 -> RETONLY          D  8  @Retain Only
Version
KUMPV026I Processing record 0155 -> MODE             D  32  @Copy Mode
```

```
KUMPV026I Processing record 0156 -> SERIALIZATION    D  32  @Copy
Serialization
KUMPV026I Processing record 0157 -> FREQUENCY        C  999999  @Copy
Frequency
KUMPV026I Processing record 0158 -> DESTINATION      D  32  @Copy
Destination
KUMPV026I Processing record 0159 -> TOC_DESTINATION  D  32  @Table of
Contents (TOC) Destination
KUMPV026I Processing record 0160 -> CHG_TIME         D  28  @Last
Update Date/Time
KUMPV026I Processing record 0161 -> CHG_ADMIN        D  32  @Last
Update by (administrator)
KUMPV026I Processing record 0162 -> PROFILE          D  256  @Managing
profile
KUMPV026I Processing record 0163 -> //NAME CLIENTOPTS S 300  @Client
Options
KUMPV026I Processing record 0164 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0165 -> //SQL Select * from CLIENTOPTS
KUMPV026I Processing record 0166 -> //ATTRIBUTES
KUMPV026I Processing record 0167 -> OPTIONSET_NAME  D  68  @Optionset
KUMPV026I Processing record 0168 -> OPTION_NAME     D  68  @Option
KUMPV026I Processing record 0169 -> SEQNUMBER       N  8  @Sequence
number
KUMPV026I Processing record 0170 -> OPTION_VALUE    D  256  @Option
Value
KUMPV026I Processing record 0171 -> FORCE           D  4  @Use Option
Set Value (FORCE)
KUMPV026I Processing record 0172 -> OBSOLETE        D  12  @Obsolete
KUMPV026I Processing record 0173 -> WHEN_OBSOLETE   D  12  @When
Obsolete?
KUMPV026I Processing record 0174 -> //NAME CLIENT_SCHEDULES S 300
@Client schedules
KUMPV026I Processing record 0175 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0176 -> //SQL Select * from
CLIENT_SCHEDULES
KUMPV026I Processing record 0177 -> //ATTRIBUTES
KUMPV026I Processing record 0178 -> DOMAIN_NAME    D  32  @Policy
Domain Name
KUMPV026I Processing record 0179 -> SCHEDULE_NAME  D  32  @Schedule
Name
KUMPV026I Processing record 0180 -> DESCRIPTION    D  256  @Description
KUMPV026I Processing record 0181 -> ACTION         D  20  @Action
KUMPV026I Processing record 0182 -> OPTIONS        D  256  @Options
```

```
KUMPV026I Processing record 0183 -> OBJECTS        D  256  @Objects
KUMPV026I Processing record 0184 -> PRIORITY       C  999999  @Priority
KUMPV026I Processing record 0185 -> STARTDATE      D  12  @Start date
KUMPV026I Processing record 0186 -> STARTTIME      D  8  @Start time
KUMPV026I Processing record 0187 -> DURATION       N  8  @Duration
KUMPV026I Processing record 0188 -> DURUNITS       D  20  @Duration
units
KUMPV026I Processing record 0189 -> PERIOD         N  8  @Period
KUMPV026I Processing record 0190 -> PERUNITS       D  20  @Period units
KUMPV026I Processing record 0191 -> DAYOFWEEK      D  20  @Day of Week
KUMPV026I Processing record 0192 -> EXPIRATION     D  12  @Expiration
KUMPV026I Processing record 0193 -> CHG_TIME       D  28  @Last Update
Date/Time
KUMPV026I Processing record 0194 -> CHG_ADMIN      D  32  @Last Update
by (administrator)
KUMPV026I Processing record 0195 -> PROFILE        D  256  @Managing
profile
KUMPV026I Processing record 0196 -> SCHED_STYLE    D  12  @Schedule
Style
KUMPV026I Processing record 0197 -> ENH_MONTH      D  52  @Month
KUMPV026I Processing record 0198 -> DAYOFMONTH     D  60  @Day of Month
KUMPV026I Processing record 0199 -> WEEKOFMONTH    D  52  @Week of
Month
KUMPV026I Processing record 0200 -> //NAME CLOPTSETS K 300  @Client
Option Sets
KUMPV026I Processing record 0201 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0202 -> //SQL Select * from CLOPTSETS
KUMPV026I Processing record 0203 -> //ATTRIBUTES
KUMPV026I Processing record 0204 -> OPTIONSET_NAME  D  68  KEY ATOMIC
@Optionset
KUMPV026I Processing record 0205 -> DESCRIPTION     D  256
@Description
KUMPV026I Processing record 0206 -> LAST_UPDATE_BY  D  68  @Last Update
by (administrator)
KUMPV026I Processing record 0207 -> PROFILE         D  256  @Managing
profile
KUMPV026I Processing record 0208 -> //NAME COLLOCGROUP S 300
@Collocation groups
KUMPV026I Processing record 0209 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0210 -> //SQL Select * from COLLOCGROUP
KUMPV026I Processing record 0211 -> //ATTRIBUTES
KUMPV026I Processing record 0212 -> COLLOCGROUP_NAME  D  32
@Collocation Group Name
```

```
KUMPV026I Processing record 0213 -> DESCRIPTION       D  256
@Description
KUMPV026I Processing record 0214 -> CHG_TIME          D  28  @Last
Update Date/Time
KUMPV026I Processing record 0215 -> CHG_ADMIN         D  32  @Last
Update by (administrator)
KUMPV026I Processing record 0216 -> NODE_NAME         D  64  @Node Name
KUMPV026I Processing record 0217 -> //NAME CONTENTS S 300  @Storage
pool volume contents
KUMPV026I Processing record 0218 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0219 -> //SQL Select * from CONTENTS
KUMPV026I Processing record 0220 -> //ATTRIBUTES
KUMPV026I Processing record 0221 -> VOLUME_NAME        D  256  @Volume
Name
KUMPV026I Processing record 0222 -> NODE_NAME          D  64  @Node
Name
KUMPV026I Processing record 0223 -> TYPE               D  20  @Type
KUMPV026I Processing record 0224 -> FILESPACE_NAME     D  64
@Filespace Name
KUMPV026I Processing record 0225 -> FILE_NAME          D  256
@Client's Name for File
KUMPV026I Processing record 0226 -> AGGREGATED         D  20
@Aggregated?
KUMPV026I Processing record 0227 -> FILE_SIZE          N  20  @Stored
Size
KUMPV026I Processing record 0228 -> SEGMENT            D  20  @Segment
Number
KUMPV026I Processing record 0229 -> CACHED             D  20  @Cached
Copy?
KUMPV026I Processing record 0230 -> FILESPACE_ID       N  8  @FSID
KUMPV026I Processing record 0231 -> FILESPACE_HEXNAME  D  64
@Hexadecimal Filespace Name
KUMPV026I Processing record 0232 -> FILE_HEXNAME       D  256
@Hexadecimal Client's Name for File
KUMPV026I Processing record 0233 -> //NAME DATAMOVERS K 300  @Data
Movers
KUMPV026I Processing record 0234 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0235 -> //SQL Select * from DATAMOVERS
KUMPV026I Processing record 0236 -> //ATTRIBUTES
KUMPV026I Processing record 0237 -> MOVER_NAME      D  64  KEY ATOMIC
@Data Mover Name
KUMPV026I Processing record 0238 -> TYPE            D  16  @Data Mover
Type
```

```
KUMPV026I Processing record 0239 -> HL_ADDRESS      D  256  @IP Address
KUMPV026I Processing record 0240 -> LL_ADDRESS      D  256  @TCP/IP
Port Number
KUMPV026I Processing record 0241 -> USER_NAME       D  64  @User Name
KUMPV026I Processing record 0242 -> WWN             D  16  @WWN
KUMPV026I Processing record 0243 -> SERIAL          D  64  @Serial
Number
KUMPV026I Processing record 0244 -> COPYTHREADS     N  8  @Copy Threads
KUMPV026I Processing record 0245 -> DATA_FORMAT     D  32  @Storage
Pool Data Format
KUMPV026I Processing record 0246 -> ONLINE          D  40  @On-Line
KUMPV026I Processing record 0247 -> LAST_UPDATE_BY  D  64  @Last Update
by (administrator)
KUMPV026I Processing record 0248 -> LAST_UPDATE     D  28  @Last Update
Date/Time
KUMPV026I Processing record 0249 -> //NAME DB S 300  @Server database
information
KUMPV026I Processing record 0250 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0251 -> //SQL Select * from DB
KUMPV026I Processing record 0252 -> //ATTRIBUTES
KUMPV026I Processing record 0253 -> AVAIL_SPACE_MB     N  8  @Available
Space (MB)
KUMPV026I Processing record 0254 -> CAPACITY_MB        N  8  @Assigned
Capacity (MB)
KUMPV026I Processing record 0255 -> MAX_EXTENSION_MB   N  8  @Maximum
Extension (MB)
KUMPV026I Processing record 0256 -> MAX_REDUCTION_MB   N  12  @Maximum
Reduction (MB)
KUMPV026I Processing record 0257 -> PAGE_SIZE          C  999999  @Page
Size (bytes)
KUMPV026I Processing record 0258 -> USABLE_PAGES       N  8  @Total
Usable Pages
KUMPV026I Processing record 0259 -> USED_PAGES         N  8  @Used
Pages
KUMPV026I Processing record 0260 -> PCT_UTILIZED       N  4  @Pct Util
KUMPV026I Processing record 0261 -> MAX_PCT_UTILIZED   N  4  @Max. Pct
Util
KUMPV026I Processing record 0262 -> PHYSICAL_VOLUMES   N  8  @Physical
Volumes
KUMPV026I Processing record 0263 -> BUFF_POOL_PAGES    N  12  @Buffer
Pool Pages
KUMPV026I Processing record 0264 -> TOTAL_BUFFER_REQ   N  12  @Total
Buffer Requests
```

```
KUMPV026I Processing record 0265 -> CACHE_HIT_PCT      N  4  @Cache Hit
Pct.
KUMPV026I Processing record 0266 -> CACHE_WAIT_PCT     N  4  @Cache
Wait Pct.
KUMPV026I Processing record 0267 -> BACKUP_RUNNING     D  12  @Backup
in Progress?
KUMPV026I Processing record 0268 -> BACKUP_TYPE        D  20  @Type of
Backup In Progress
KUMPV026I Processing record 0269 -> NUM_BACKUP_INCR    C  999999
@Incrementals Since Last Full
KUMPV026I Processing record 0270 -> BACKUP_CHG_MB      N  4  @Changed
Since Last Backup (MB)
KUMPV026I Processing record 0271 -> BACKUP_CHG_PCT     N  4
@Percentage Changed
KUMPV026I Processing record 0272 -> LAST_BACKUP_DATE   D  28  @Last
Complete Backup Date/Time
KUMPV026I Processing record 0273 -> DB_REORG_EST       N  8  @Estimate
of Recoverable Space (MB)
KUMPV026I Processing record 0274 -> DB_REORG_EST_TIME  D  28  @Last
Estimate of Recoverable Space (MB)
KUMPV026I Processing record 0275 -> //NAME DBBACKUPTRIGGER S 300
@Database backup trigger information
KUMPV026I Processing record 0276 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0277 -> //SQL Select * from DBBACKUPTRIGGER
KUMPV026I Processing record 0278 -> //ATTRIBUTES
KUMPV026I Processing record 0279 -> DEVCLASS       D  32  @Full Device
Class
KUMPV026I Processing record 0280 -> INCRDEVCLASS   D  32  @Incremental
Device Class
KUMPV026I Processing record 0281 -> LOGFULLPCT     C  999999  @Log Full
Percentage
KUMPV026I Processing record 0282 -> NUMICREMENTAL  C  999999
@Incrementals Between Fulls
KUMPV026I Processing record 0283 -> CHG_TIME       D  28  @Last Update
Date/Time
KUMPV026I Processing record 0284 -> CHG_ADMIN      D  64  @Last Update
by (administrator)
KUMPV026I Processing record 0285 -> MININTERVAL    C  999999  @Minimum
Backup Interval (minutes)
KUMPV026I Processing record 0286 -> MINLOGFREE     C  999999  @Minimum
Log Percentage Freed
KUMPV026I Processing record 0287 -> //NAME DBSPACETRIGGER S 300
@Database space trigger information
```

```
KUMPV026I Processing record 0288 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0289 -> //SQL Select * from DBSPACETRIGGER
KUMPV026I Processing record 0290 -> //ATTRIBUTES
KUMPV026I Processing record 0291 -> FULLPCT           C  999999  @DB
Full Percentage
KUMPV026I Processing record 0292 -> EXPANSIONPCT      C  999999  @DB
Space Expansion Percentage
KUMPV026I Processing record 0293 -> EXPANSION_PREFIX  D  252  @DB
Expansion prefix
KUMPV026I Processing record 0294 -> MAXIMUM_DB_SIZE   N  8  @DB Maximum
Size (Megabytes)
KUMPV026I Processing record 0295 -> MIRROR_PREFIX_1   D  252  @Mirror
Prefix 1
KUMPV026I Processing record 0296 -> MIRROR_PREFIX_2   D  252  @Mirror
Prefix 2
KUMPV026I Processing record 0297 -> CHG_TIME          D  28  @Last
Update Date/Time
KUMPV026I Processing record 0298 -> CHG_ADMIN         D  64  @Last
Update by (administrator)
KUMPV026I Processing record 0299 -> //NAME DBVOLUMES S 300  @Database
volumes
KUMPV026I Processing record 0300 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0301 -> //SQL Select * from DBVOLUMES
KUMPV026I Processing record 0302 -> //ATTRIBUTES
KUMPV026I Processing record 0303 -> COPY1_NAME      D  256  @Volume
Name (Copy 1)
KUMPV026I Processing record 0304 -> COPY1_STATUS    D  20  @Copy Status
KUMPV026I Processing record 0305 -> COPY2_NAME      D  256  @Volume
Name (Copy 2)
KUMPV026I Processing record 0306 -> COPY2_STATUS    D  20  @Copy Status
KUMPV026I Processing record 0307 -> COPY3_NAME      D  256  @Volume
Name (Copy 3)
KUMPV026I Processing record 0308 -> COPY3_STATUS    D  20  @Copy Status
KUMPV026I Processing record 0309 -> AVAIL_SPACE_MB  N  8  @Available
Space (MB)
KUMPV026I Processing record 0310 -> ALLOC_SPACE_MB  N  8  @Allocated
Space (MB)
KUMPV026I Processing record 0311 -> FREE_SPACE_MB   N  8  @Free Space
(MB)
KUMPV026I Processing record 0312 -> //NAME DEVCLASSES K 300  @Device
Classes
KUMPV026I Processing record 0313 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
```

```
KUMPV026I Processing record 0314 -> //SQL Select * from DEVCLASSES
KUMPV026I Processing record 0315 -> //ATTRIBUTES
KUMPV026I Processing record 0316 -> DEVCLASS_NAME    D  32  KEY ATOMIC
@Device Class Name
KUMPV026I Processing record 0317 -> ACCESS_STRATEGY  D  12  @Device
Access Strategy
KUMPV026I Processing record 0318 -> STGPOOL_COUNT    N  8  @Storage
Pool Count
KUMPV026I Processing record 0319 -> DEVTYPE          D  16  @Device
Type
KUMPV026I Processing record 0320 -> FORMAT           D  16  @Format
KUMPV026I Processing record 0321 -> CAPACITY         D  40  @Est/Max
Capacity
KUMPV026I Processing record 0322 -> MOUNTLIMIT       D  12  @Mount
Limit
KUMPV026I Processing record 0323 -> MOUNTWAIT        N  8  @Mount Wait
(min)
KUMPV026I Processing record 0324 -> MOUNTRETENTION   N  8  @Mount
Retention (min)
KUMPV026I Processing record 0325 -> PREFIX           D  8  @Label
Prefix
KUMPV026I Processing record 0326 -> DRIVE            D  4  @Drive
Letter
KUMPV026I Processing record 0327 -> LIBRARY_NAME     D  32  @Library
KUMPV026I Processing record 0328 -> DIRECTORY        D  256  @Directory
KUMPV026I Processing record 0329 -> SERVERNAME       D  64  @Server
Name
KUMPV026I Processing record 0330 -> RETRYPERIOD      N  8  @Retry
Period
KUMPV026I Processing record 0331 -> RETRYINTERVAL    N  8  @Retry
Interval
KUMPV026I Processing record 0332 -> TWO_SIDED        D  4  @Twosided
KUMPV026I Processing record 0333 -> SHARED           D  4  @Shared
KUMPV026I Processing record 0334 -> HLADDRESS        D  256  @HLAddr
KUMPV026I Processing record 0335 -> MINCAPACITY      D  40  @Minimum
Capacity
KUMPV026I Processing record 0336 -> WORM             D  4  @WORM
KUMPV026I Processing record 0337 -> SCALECAPACITY    N  8  @Scaled
Capacity
KUMPV026I Processing record 0338 -> LAST_UPDATE_BY   D  64  @Last
Update by (administrator)
KUMPV026I Processing record 0339 -> LAST_UPDATE      D  28  @Last
Update Date/Time
KUMPV026I Processing record 0340 -> //NAME DISKS S 300  @Disks
```

```
KUMPV026I Processing record 0341 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0342 -> //SQL Select * from DISKS
KUMPV026I Processing record 0343 -> //ATTRIBUTES
KUMPV026I Processing record 0344 -> NODE_NAME       D  64  @Node Name
KUMPV026I Processing record 0345 -> DISK_NAME       D  64  @Disk Name
KUMPV026I Processing record 0346 -> WWN             D  16  @WWN
KUMPV026I Processing record 0347 -> SERIAL          D  64  @Serial
Number
KUMPV026I Processing record 0348 -> ONLINE          D  40  @On-Line
KUMPV026I Processing record 0349 -> LAST_UPDATE_BY  D  64  @Last Update
by (administrator)
KUMPV026I Processing record 0350 -> LAST_UPDATE     D  28  @Last Update
Date/Time
KUMPV026I Processing record 0351 -> //NAME DOMAINS K 300  @Policy
domains
KUMPV026I Processing record 0352 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0353 -> //SQL Select * from DOMAINS
KUMPV026I Processing record 0354 -> //ATTRIBUTES
KUMPV026I Processing record 0355 -> DOMAIN_NAME         D  32  KEY
ATOMIC  @Policy Domain Name
KUMPV026I Processing record 0356 -> SET_LAST_ACTIVATED  D  32
@Activated Policy Set
KUMPV026I Processing record 0357 -> ACTIVATE_DATE       D  28
@Activation Date/Time
KUMPV026I Processing record 0358 -> DEFMGMTCLASS        D  32
@Activated Default Mgmt Class
KUMPV026I Processing record 0359 -> NUM_NODES           N  8  @Number
of Registered Nodes
KUMPV026I Processing record 0360 -> BACKRETENTION       C  999999
@Backup Retention (Grace Period)
KUMPV026I Processing record 0361 -> ARCHRETENTION       C  999999
@Archive Retention (Grace Period)
KUMPV026I Processing record 0362 -> DESCRIPTION         D  256
@Description
KUMPV026I Processing record 0363 -> CHG_TIME            D  28  @Last
Update Date/Time
KUMPV026I Processing record 0364 -> CHG_ADMIN           D  32  @Last
Update Date/Time
KUMPV026I Processing record 0365 -> PROFILE             D  256
@Managing profile
KUMPV026I Processing record 0366 -> //NAME DRIVES S 300  @Drives
KUMPV026I Processing record 0367 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
```

```
KUMPV026I Processing record 0368 -> //SQL Select * from DRIVES
KUMPV026I Processing record 0369 -> //ATTRIBUTES
KUMPV026I Processing record 0370 -> LIBRARY_NAME    D  32  @Library
Name
KUMPV026I Processing record 0371 -> DRIVE_NAME      D  32  @Drive Name
KUMPV026I Processing record 0372 -> DEVICE_TYPE     D  16  @Device Type
KUMPV026I Processing record 0373 -> ONLINE          D  40  @On-Line
KUMPV026I Processing record 0374 -> READ_FORMATS    D  16  @Read
Formats
KUMPV026I Processing record 0375 -> WRITE_FORMATS   D  16  @Write
Formats
KUMPV026I Processing record 0376 -> ELEMENT         C  999999  @Element
KUMPV026I Processing record 0377 -> ACS_DRIVE_ID    D  16  @ACS DriveId
KUMPV026I Processing record 0378 -> DRIVE_STATE     D  40  @Drive State
KUMPV026I Processing record 0379 -> ALLOCATED_TO    D  64  @Allocated
to
KUMPV026I Processing record 0380 -> LAST_UPDATE_BY  D  64  @Last Update
by (administrator)
KUMPV026I Processing record 0381 -> LAST_UPDATE     D  28  @Last Update
Date/Time
KUMPV026I Processing record 0382 -> CLEAN_FREQ      D  12  @Cleaning
Frequency (Gigabytes/ASNEEDED/NONE)
KUMPV026I Processing record 0383 -> DRIVE_SERIAL    D  64  @Serial
Number
KUMPV026I Processing record 0384 -> VOLUME_NAME     D  256  @Volume
Name
KUMPV026I Processing record 0385 -> //NAME DRMCSTGPOOLS S 300  @Copy
storage pools managed by the disaster recovery manager
KUMPV026I Processing record 0386 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0387 -> //SQL Select * from DRMCSTGPOOLS
KUMPV026I Processing record 0388 -> //ATTRIBUTES
KUMPV026I Processing record 0389 -> STGPOOL_NAME  D  32  @Storage Pool
Name
KUMPV026I Processing record 0390 -> //NAME DRMEDIA S 300  @Physical
volumes managed by move drmedia
KUMPV026I Processing record 0391 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0392 -> //SQL Select * from DRMEDIA
KUMPV026I Processing record 0393 -> //ATTRIBUTES
KUMPV026I Processing record 0394 -> VOLUME_NAME   D  256  @Storage pool
volumes
KUMPV026I Processing record 0395 -> STATE         D  20  @State
KUMPV026I Processing record 0396 -> UPD_DATE      D  28  @Last Update
Date/Time
```

```
KUMPV026I Processing record 0397 -> LOCATION      D  256  @Location
KUMPV026I Processing record 0398 -> STGPOOL_NAME  D  32  @Storage Pool
Name
KUMPV026I Processing record 0399 -> LIB_NAME      D  32  @Automated
LibName
KUMPV026I Processing record 0400 -> VOLTYPE       D  12  @Volume Type
KUMPV026I Processing record 0401 -> //NAME DRMMACHINE S 300  @Disaster
recovery manager machine information
KUMPV026I Processing record 0402 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0403 -> //SQL Select * from DRMMACHINE
KUMPV026I Processing record 0404 -> //ATTRIBUTES
KUMPV026I Processing record 0405 -> MACHINE_NAME     D  64  @Machine
Name
KUMPV026I Processing record 0406 -> PRIORITY         C  999999
@Machine Priority
KUMPV026I Processing record 0407 -> BUILDING         D  16  @Building
KUMPV026I Processing record 0408 -> FLOOR            D  16  @Floor
KUMPV026I Processing record 0409 -> ROOM             D  16  @Room
KUMPV026I Processing record 0410 -> ADSM_SERVER      D  4  @Server?
KUMPV026I Processing record 0411 -> DESCRIPTION      D  256
@Description
KUMPV026I Processing record 0412 -> CHARACTERISTICS  D  4
@Characteristics?
KUMPV026I Processing record 0413 -> RECINSTRUCTIONS  D  4  @Recovery
Instructions?
KUMPV026I Processing record 0414 -> //NAME DRMMACHINECHARS S 300
@Disaster recovery manager machine characteristics
KUMPV026I Processing record 0415 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0416 -> //SQL Select * from DRMMACHINECHARS
KUMPV026I Processing record 0417 -> //ATTRIBUTES
KUMPV026I Processing record 0418 -> MACHINE_NAME     D  64  @Machine
Name
KUMPV026I Processing record 0419 -> CHARACTERISTICS  D  256
@Characteristics
KUMPV026I Processing record 0420 -> //NAME DRMMACHINENODE S 300
@Disaster recovery manager machine node associations
KUMPV026I Processing record 0421 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0422 -> //SQL Select * from DRMMACHINENODE
KUMPV026I Processing record 0423 -> //ATTRIBUTES
KUMPV026I Processing record 0424 -> MACHINE_NAME  D  64  @Machine Name
KUMPV026I Processing record 0425 -> NODE_NAME     D  64  @Node Name
```

```
KUMPV026I Processing record 0426 -> //NAME DRMMACHINERECINST S 300
@Disaster recovery manager machine recovery instructions
KUMPV026I Processing record 0427 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0428 -> //SQL Select * from
DRMMACHINERECINST
KUMPV026I Processing record 0429 -> //ATTRIBUTES
KUMPV026I Processing record 0430 -> MACHINE_NAME     D  64  @Machine
Name
KUMPV026I Processing record 0431 -> RECINSTRUCTIONS  D  256  @Recovery
Instructions
KUMPV026I Processing record 0432 -> //NAME DRMMACHINERECMEDIA S 300
@Disaster recovery manager machine recovery media associations
KUMPV026I Processing record 0433 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0434 -> //SQL Select * from
DRMMACHINERECMEDIA
KUMPV026I Processing record 0435 -> //ATTRIBUTES
KUMPV026I Processing record 0436 -> MACHINE_NAME   D  64  @Machine Name
KUMPV026I Processing record 0437 -> RECMEDIA_NAME  D  32  @Recovery
Media Name
KUMPV026I Processing record 0438 -> //NAME DRMPSTGPOOLS S 300  @Primary
storage pools managed by the disaster recovery manager
KUMPV026I Processing record 0439 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0440 -> //SQL Select * from DRMPSTGPOOLS
KUMPV026I Processing record 0441 -> //ATTRIBUTES
KUMPV026I Processing record 0442 -> STGPOOL_NAME  D  32  @Storage Pool
Name
KUMPV026I Processing record 0443 -> //NAME DRMRECOVERYMEDIA K 300
@Disaster recovery manager recovery media
KUMPV026I Processing record 0444 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0445 -> //SQL Select * from
DRMRECOVERYMEDIA
KUMPV026I Processing record 0446 -> //ATTRIBUTES
KUMPV026I Processing record 0447 -> RECMEDIA_NAME  D  32  KEY ATOMIC
@Recovery Media Name
KUMPV026I Processing record 0448 -> TYPE           D  8  @Type
KUMPV026I Processing record 0449 -> LOCATION       D  256  @Location
KUMPV026I Processing record 0450 -> DESCRIPTION    D  256  @Description
KUMPV026I Processing record 0451 -> PRODUCT        D  16  @Product
KUMPV026I Processing record 0452 -> PRODUCT_INFO   D  256  @Product
Information
```

```
KUMPV026I Processing record 0453 -> VOLUMES        D  256  @Volume
Names
KUMPV026I Processing record 0454 -> //NAME DRMSRPF S 300  @Recovery
plan files in source server
KUMPV026I Processing record 0455 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0456 -> //SQL Select * from DRMSRPF
KUMPV026I Processing record 0457 -> //ATTRIBUTES
KUMPV026I Processing record 0458 -> RPF_NAME        D  256  @Recovery
Plan File Name
KUMPV026I Processing record 0459 -> NODE_NAME       D  68  @Node Name
KUMPV026I Processing record 0460 -> DEVCLASS_NAME   D  32  @Device
Class Name
KUMPV026I Processing record 0461 -> TYPE            D  36  @Recovery
Plan File Type
KUMPV026I Processing record 0462 -> MGMTCLASS_NAME  D  32  @Mgmt Class
Name
KUMPV026I Processing record 0463 -> RPF_SIZE        N  12  @Recovery
Plan File Size
KUMPV026I Processing record 0464 -> RPF_DELETE      D  20  @Marked For
Deletion
KUMPV026I Processing record 0465 -> RPF_DELDATE     D  28  @Deletion
Date
KUMPV026I Processing record 0466 -> //NAME DRMSTANZA S 300  @Recovery
plan file stanza names
KUMPV026I Processing record 0467 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0468 -> //SQL Select * from DRMSTANZA
KUMPV026I Processing record 0469 -> //ATTRIBUTES
KUMPV026I Processing record 0470 -> STANZA_NAME  D  256  @Stanza Name
KUMPV026I Processing record 0471 -> //NAME DRMSTATUS S 300  @Disaster
recovery manager status information
KUMPV026I Processing record 0472 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0473 -> //SQL Select * from DRMSTATUS
KUMPV026I Processing record 0474 -> //ATTRIBUTES
KUMPV026I Processing record 0475 -> PLANPREFIX     D  200  @Recovery
Plan Prefix
KUMPV026I Processing record 0476 -> INSTRPREFIX    D  200  @Plan
Instructions Prefix
KUMPV026I Processing record 0477 -> PLANVPOSTFIX   D  4  @Replacement
Volume Postfix
KUMPV026I Processing record 0478 -> NONMOUNTNAME   D  256  @Not
Mountable Location Name
```

```
KUMPV026I Processing record 0479 -> COURIERNAME    D  256  @Courier
Name
KUMPV026I Processing record 0480 -> VAULTNAME      D  256  @Vault Site
Name
KUMPV026I Processing record 0481 -> DBBEXPIREDAYS  N  8  @DB Backup
Series Expiration Days
KUMPV026I Processing record 0482 -> CHECKLABEL     D  20  @Check Label?
KUMPV026I Processing record 0483 -> FILEPROCESS    D  20  @Process FILE
Device Type?
KUMPV026I Processing record 0484 -> CMDFILENAME    D  256  @Command
File Name
KUMPV026I Processing record 0485 -> RPFEXPIREDAYS  N  8  @Recovery Plan
File Expiration Days
KUMPV026I Processing record 0486 -> //NAME DRMTRPF S 300  @Recovery
plan files in target server
KUMPV026I Processing record 0487 -> //SOURCE ODBC TSMODBC user=vasfi
pswd=good4now
KUMPV026I Processing record 0488 -> //SQL Select * from DRMTRPF
KUMPV026I Processing record 0489 -> //ATTRIBUTES
KUMPV026I Processing record 0490 -> RPF_NAME        D  256  @Recovery
Plan File Name
KUMPV026I Processing record 0491 -> NODE_NAME       D  68  @Node Name
KUMPV026I Processing record 0492 -> DEVCLASS_NAME   D  32  @Device
Class Name
KUMPV026I Processing record 0493 -> TYPE            D  36  @Recovery
Plan File Type
KUMPV026I Processing record 0494 -> MGMTCLASS_NAME  D  32  @Mgmt Class
Name
KUMPV026I Processing record 0495 -> RPF_SIZE        N  12  @Recovery
Plan File Size
KUMPV026I Processing record 0496 -> RPF_DELETE      D  20  @Marked For
Deletion
KUMPV026I Processing record 0497 -> RPF_DELDATE     D  28  @Deletion
Date
KUMPV000I Validation completed successfully
KUMPV090I Application metafile validation report saved in file
C:\IBM\ITM\TMAITM6\metafiles\tivstm.rpt.

KUMPS065I Do you wish to Import or Refresh this metafile?
<Import/Refresh/Cancel>
import
KUMPS020I Import successfully completed for tivstm.mdl
```

6. After you do this, restart the Universal Agent.

9.2.2 Viewing the data in the Tivoli Enterprise Portal

To view the data that is collected by the newly configured Universal Agent, perform the following steps:

1. Log on to either your portal desktop client or your portal Web client.
2. After you log on, you can view the new agent that is added. First select **Enterprise**. Then select **Windows Systems**, then **Hostname**, and finally select **Universal Agent**; see Figure 9-21.

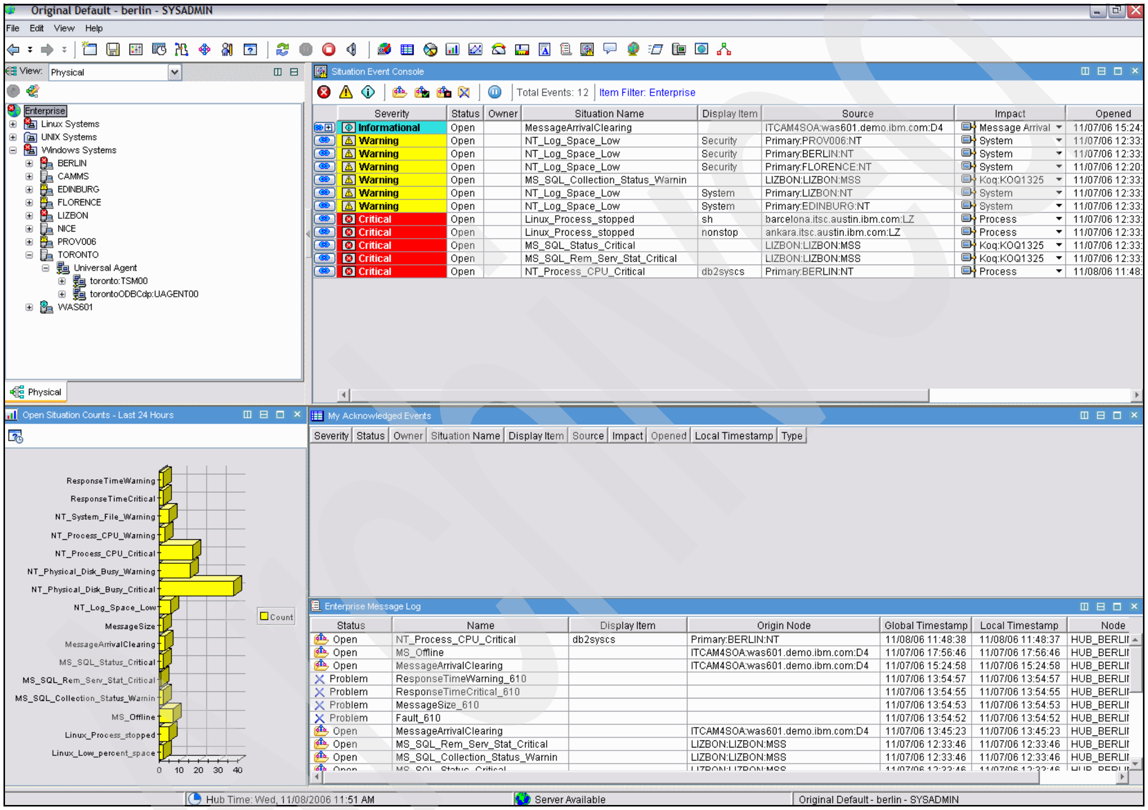


Figure 9-21 Viewing the data in portal

9.3 Tivoli Storage Manager Universal Agent in the Tivoli Enterprise Portal

To view all the data that is collected by the ODBC data provider running under the Universal Agent, click the Universal Agent, as shown in Figure 9-22.

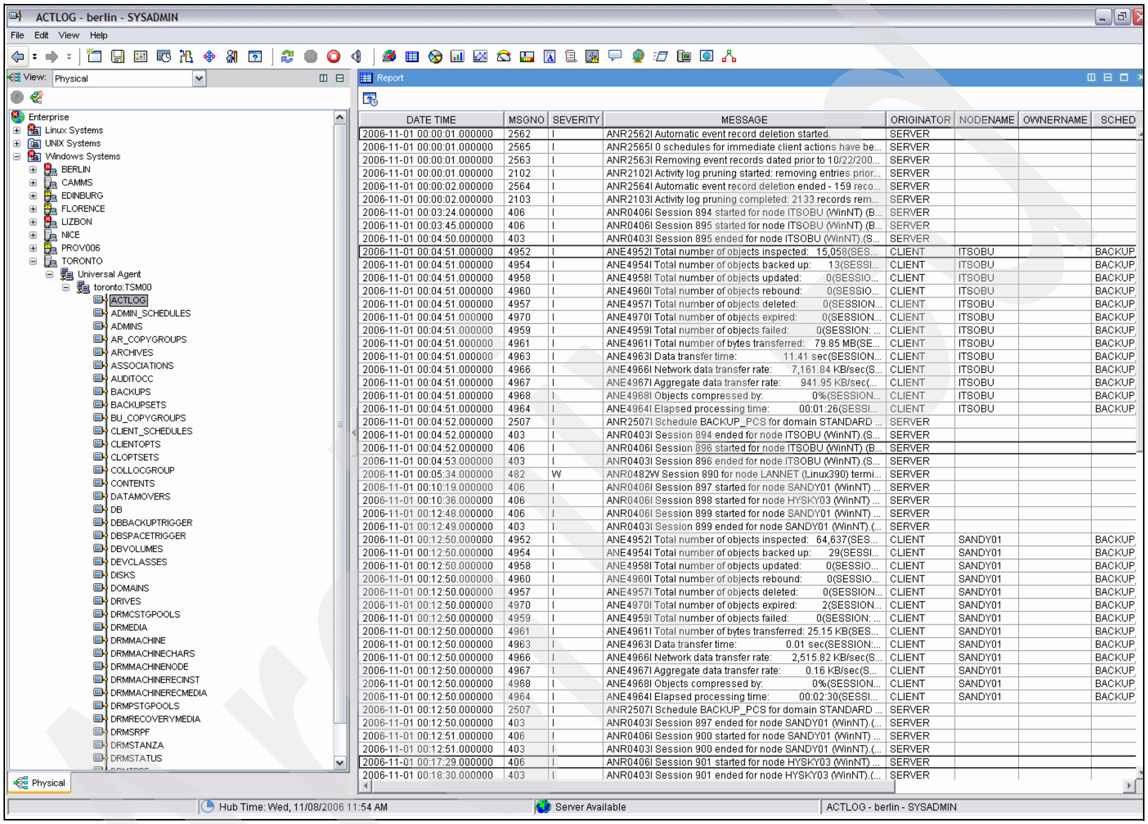


Figure 9-22 TSM00 data view in Tivoli Enterprise Portal

9.3.1 Warehousing the Universal Agent data

To warehouse the data that is being collected by the configured Universal Agent, perform the following steps:

1. In the tool bar, select the **History Configuration** button, as shown in Figure 9-23.

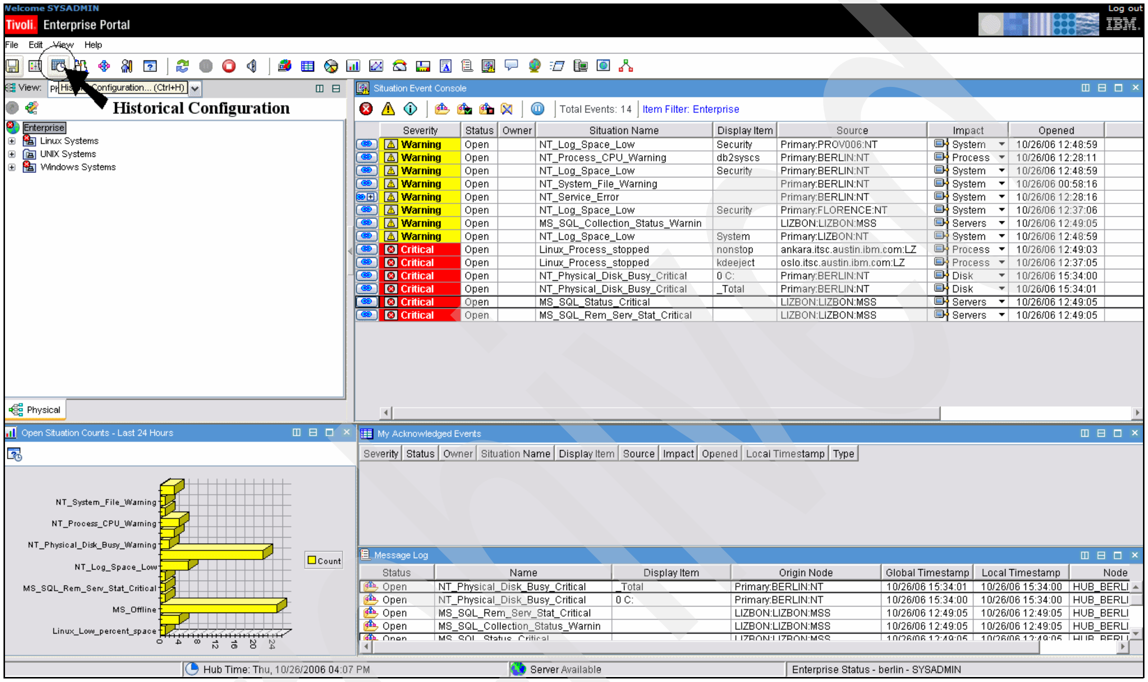


Figure 9-23 Portal Historical Configuration tab

2. The History Collection Configuration window opens. In the Select a product drop-down list, select the name, which is given in the mdl file to the Universal Agent (`//APPL TSM`). This was TSM, in our example; see Figure 9-24.

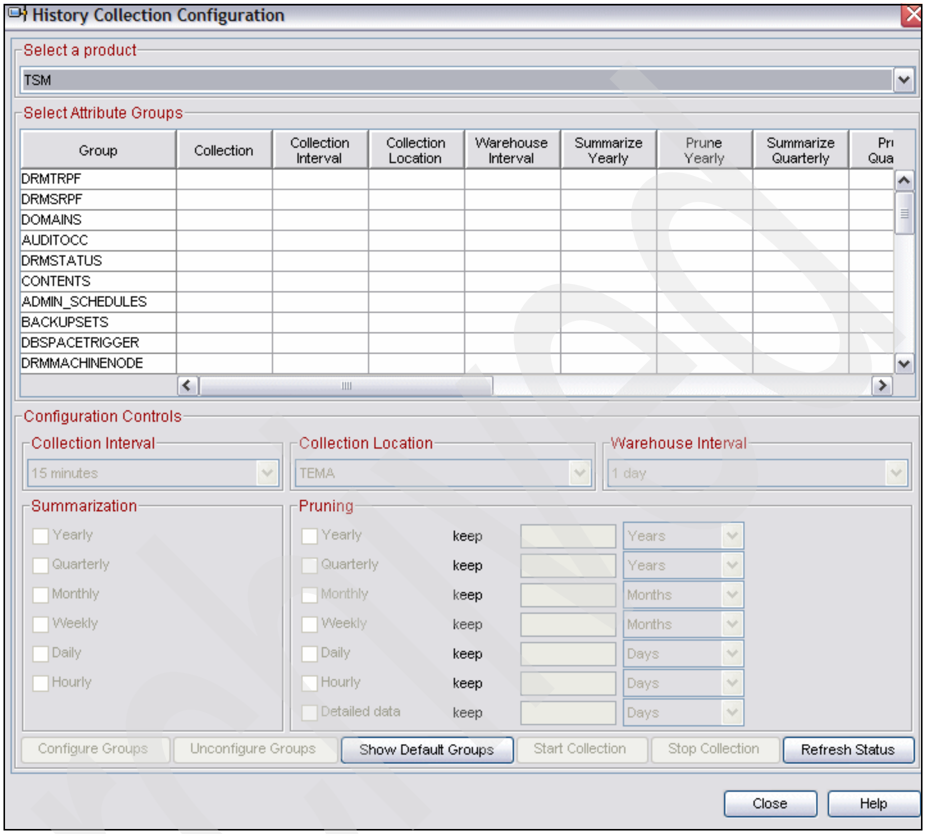


Figure 9-24 TSM history collection configuration window

3. In the Attribute Group list, click the group names. This enables the Configuration Control options. You can now specify the Collection Interval. Four options are available. Data for metrics presented under the Attribute Groups can be collected every 5, 15, 30, or 60 minutes, as shown in Figure 9-25.

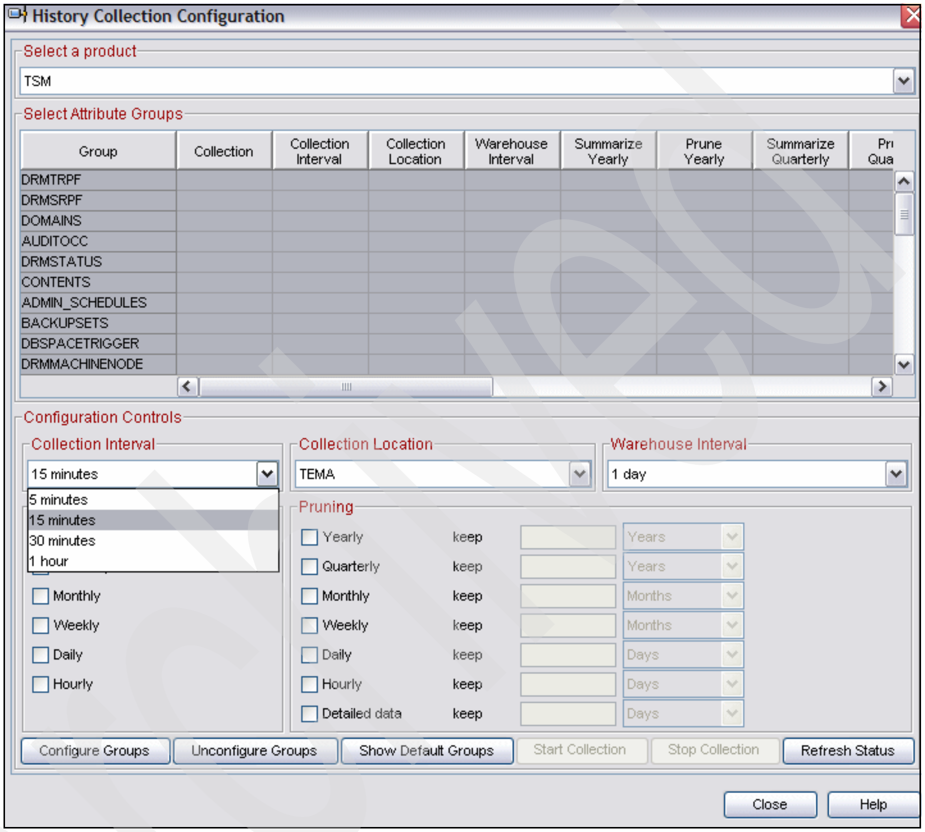


Figure 9-25 TSM collection interval configuration window

4. You can specify the collection location. Two options are available: Historical data can be collected at either the Tivoli Enterprise Monitoring Agent (agent), or at the Tivoli Enterprise Monitoring Server (management server, hub or remote). We recommend that you specify to collect data at the Tivoli Enterprise Monitoring Agent location, as shown in Figure 9-26.

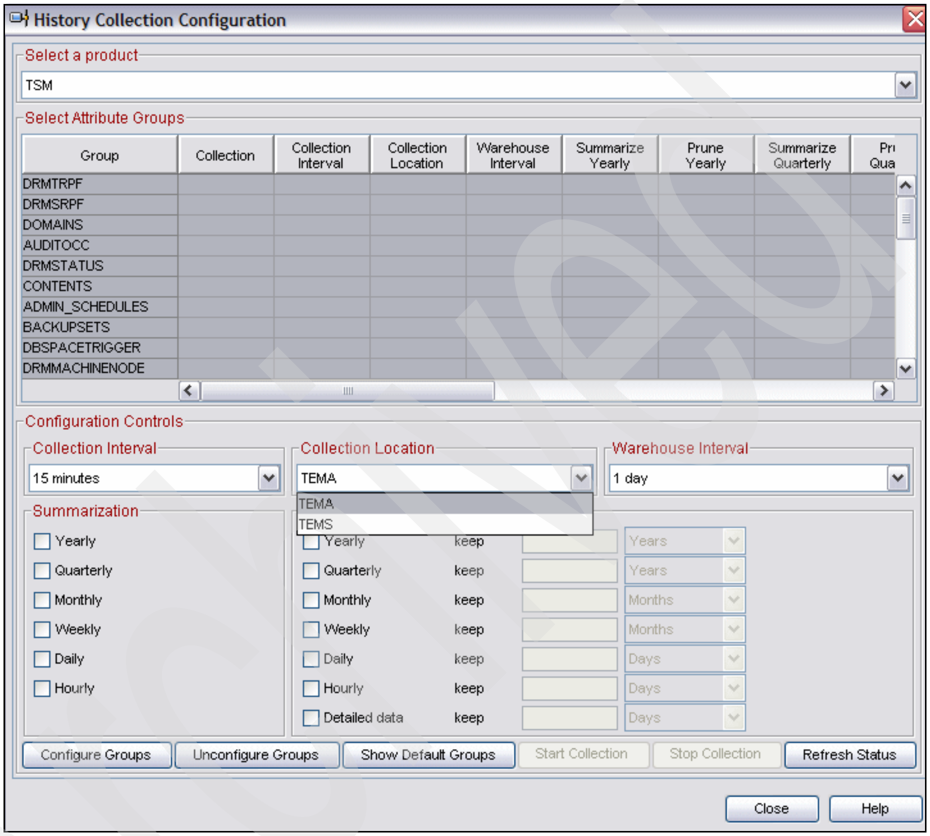


Figure 9-26 TSM collection location configuration window

5. Select the warehouse interval. Two options are available: Historical data can be uploaded to Warehouse using the Warehouse Proxy agent in every hour, or in every 24 hours; see Figure 9-27.

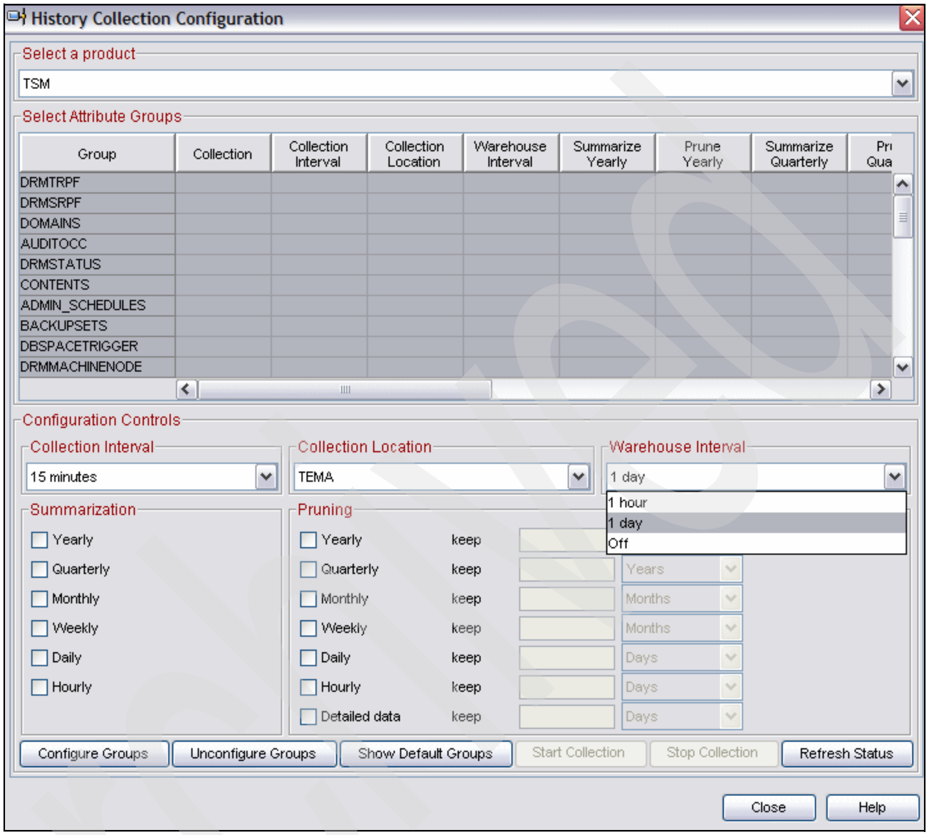


Figure 9-27 TSM warehouse interval configuration window

6. After you select all the options under the configuration controls, select the **Configure Groups** button to save these changes. You can see the saved options in the updated panel, as shown in Figure 9-28. You can see that the options are set to collect data every 15 minutes. This data is stored at the Tivoli Enterprise Monitoring Agent and written to the warehouse every hour.

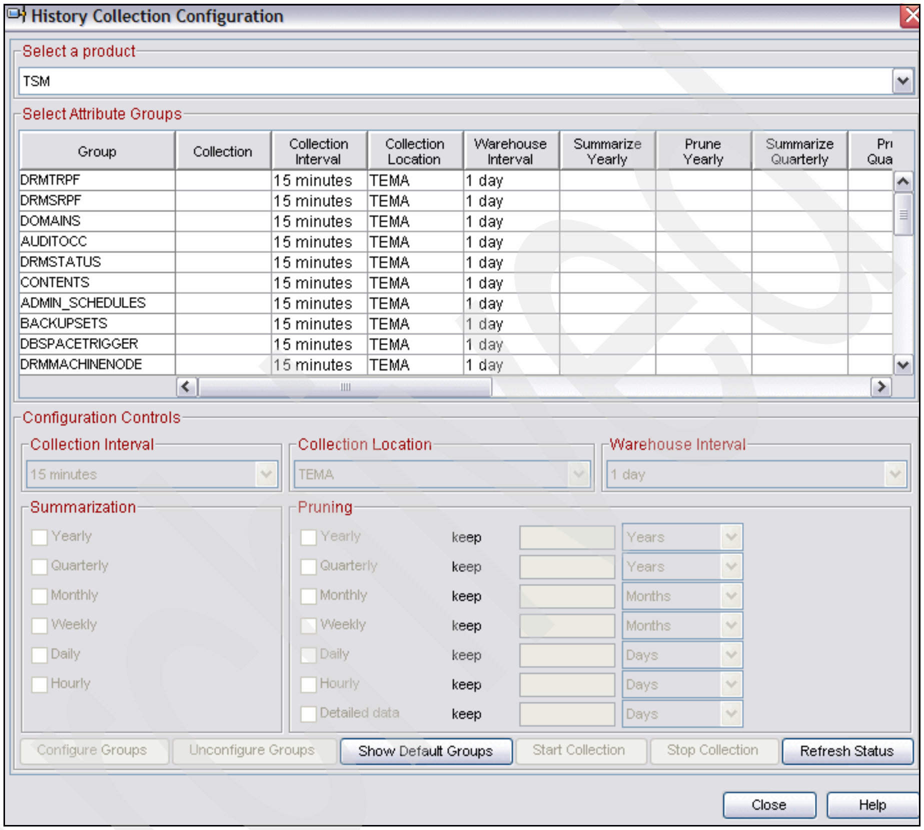


Figure 9-28 TSM configuration window

> **Note:** Configuring the collection options for a attribute group does not start collection for that attribute group. You have to start the collection manually. We describe how to do this in more detail in the following steps.

7. To start the data collection, select the **Start Collection** button to enable historical collection for the selected attribute groups.

   The Collection field in the Select Attribute Groups section shows the status of the agent as `Started`, as shown in Figure 9-29 on page 323 9 (that is, the agent has started collection).

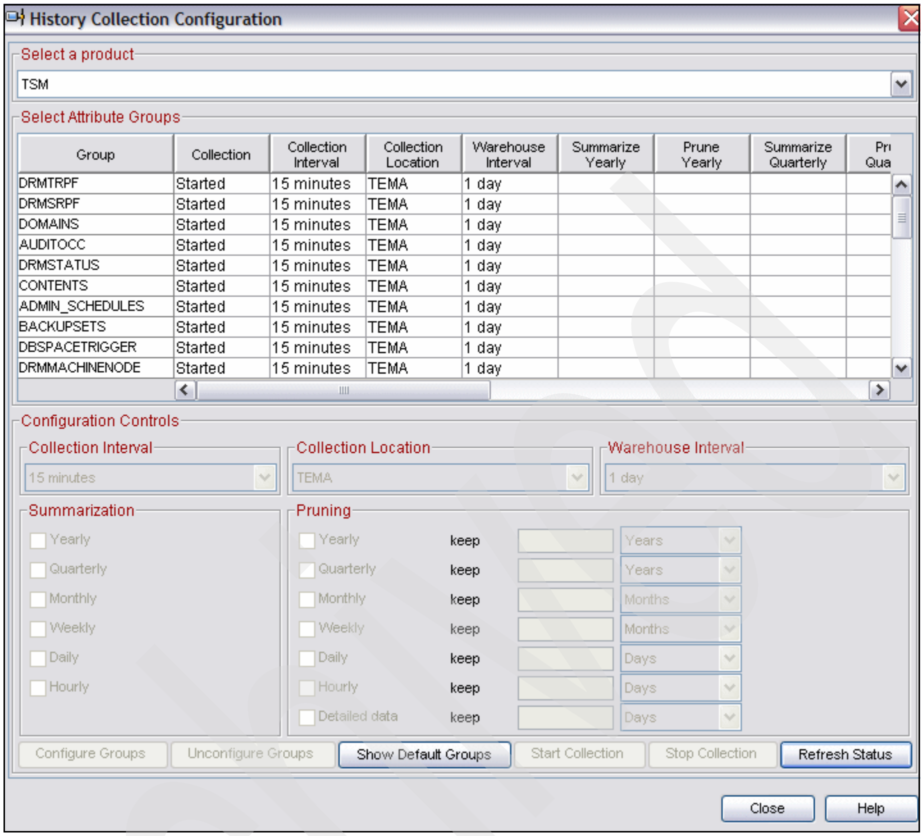


Figure 9-29 Tivoli Storage Manager started status in configuration window

The history collection has now started. You can confirm this by looking at the Universal Agents view, which shows a date/time icon, as shown in Figure 9-30.

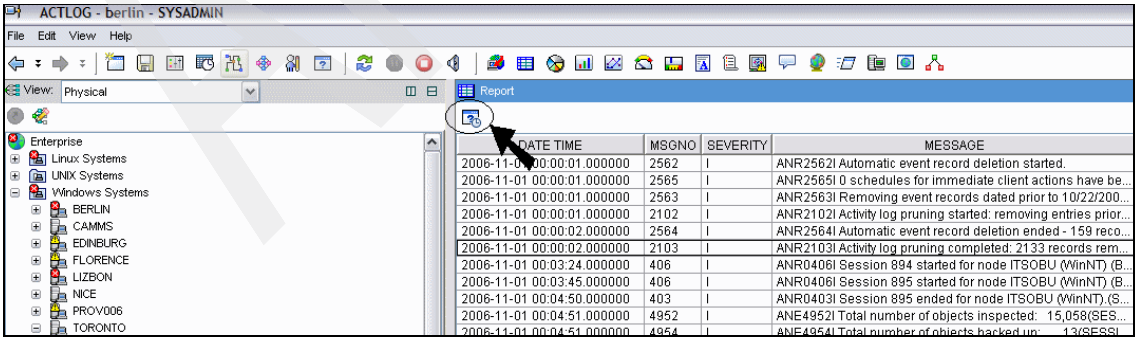


Figure 9-30 Tivoli Enterprise Portal Tivoli Storage Manager date/time icon

9.4 Viewing data in Tivoli Enterprise Portal Server using an external ODBC data source

You can add any ODBC data source to Tivoli Enterprise Portal Server and view the data in the portal desktop client. This data is only used for viewing purposes.

If you want to view data from an Oracle database, you can configure the portal server to add that data source by using the commands shown in Example 9-5.

Example 9-5 Adding a data source example

```
C:\IBM\ITM\BIN>tacmd login -u  sysadmin -s berlin
Password?
Validating user...
KUIC00007I: User sysadmin logged into server on https://berlin:3759.
C:\IBM\ITM\BIN>tacmd configureportalserver -s OracleDB -p DSN=OracleDB
UID=system PWD=password
 KUICRA031: Are you sure you want to create the OracleDB datasource in
the portal server configuration file C:\IBM\ITM\CNPS\kfwcma.ini and the
Windows Registry with the following properties?
DSN=OracleDB;UID=system;PWD=password
Enter Y for yes or N for no: Y
KUICCP013I: The OracleDB datasource in the portal server configuration
file C:\IBM\ITM\CNPS\kfwcma.ini has been created with the following
properties:
DSN=OracleDB;UID=system;PWD=password
KUICCP029I: The OracleDB datasource in the Windows Registry has been
created with the following properties:
DSN=OracleDB;UID=system;PWD=password
```

Note: After you add this data source, you must restart the Tivoli Enterprise Portal Server.

After you log in to the Tivoli Enterprise Portal Server, you can write a query to view data from your custom data source. To configure this, perform the following steps:

1. From the portal client, open the Query Editor. You can create a custom query by selecting **OracleDB data source** and assign this query (OracleSQL) under Tivoli Enterprise Monitoring Server category to a view. The Query and View are displayed, as shown in Figure 9-31 on page 325.

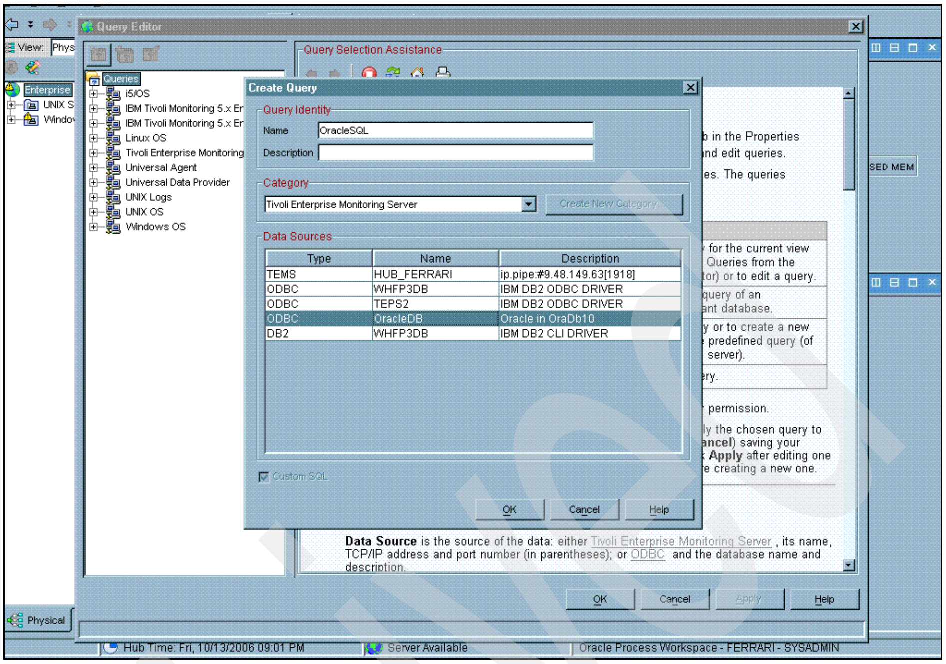


Figure 9-31 Creating a query from an ODBC data source

2. In the Custom SQL window, enter the following SQL (or an SQL of your choice):

```
SELECT PID, USERNAME, PROGRAM, PGA_USED_MEM, PGA_ALLOC_MEM,
PGA_FREEABLE_MEM, PGA_MAX_MEM FROM V$PROCESS;
```

   Click **Apply** and **OK**, as shown in Figure 9-32 on page 326.

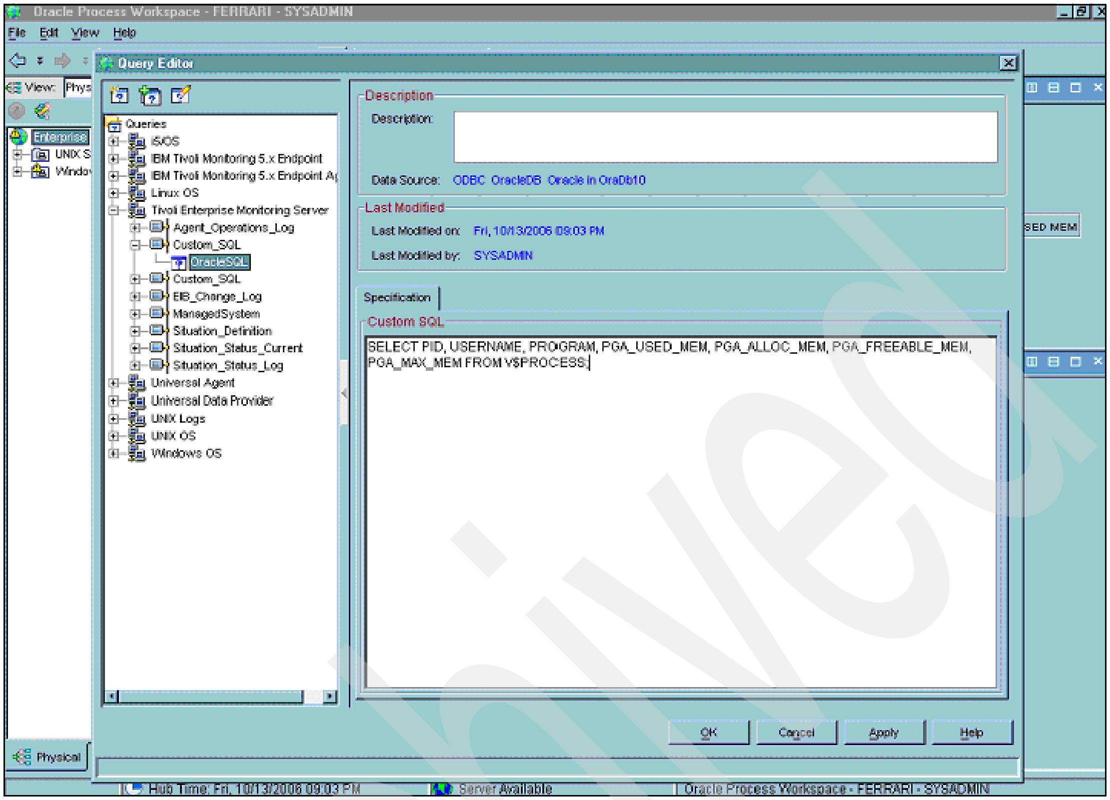


Figure 9-32 Custom SQL window

3. Select any workspace and you can apply the OracleSQL query created in the previous step.
   a. Select a workspace at Enterprise Level. Click **File**, then select **Save Workspace as**, and give it a name.
   b. Drag and drop a table icon on the empty view and click **Yes** to assign the query.

c. Select **Click here to assign a query** button, as shown in Figure 9-33.

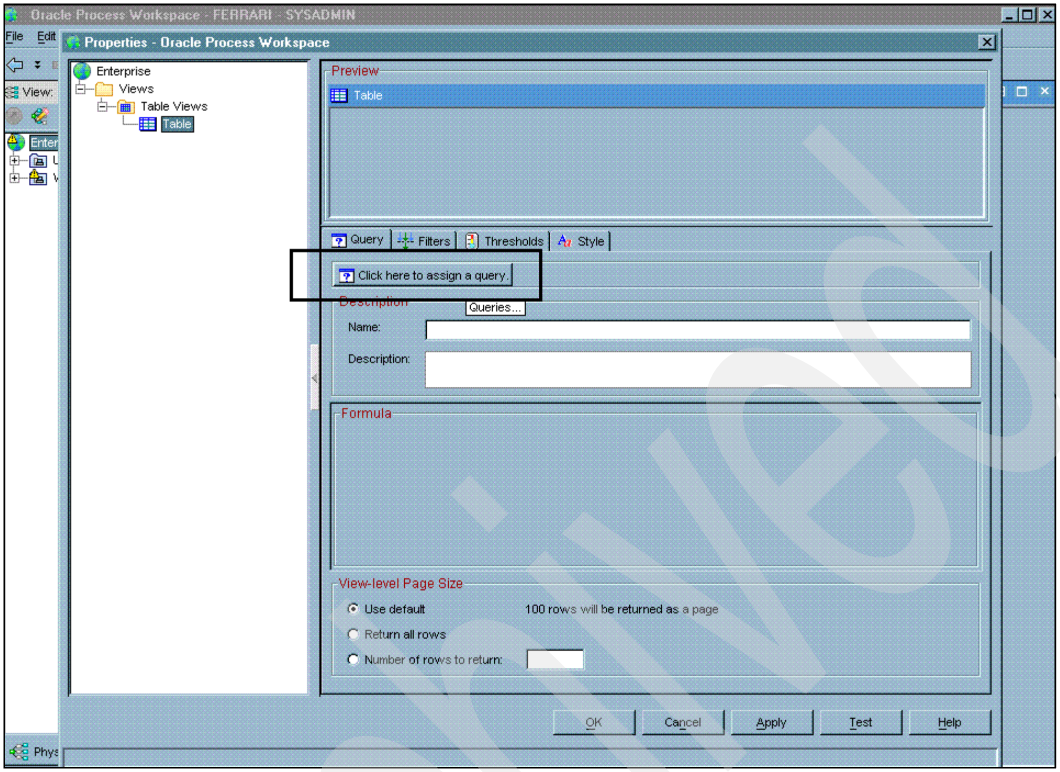


Figure 9-33 Assigning a custom query to a workspace

d. Select the query (**OracleSQL**), which was created in the previous step. First select **Queries**, then select **Tivoli Enterprise Monitoring**, then select **Server**, and then **Custom SQL**.

e. Click **OK**, **Apply**, and **OK** again, as shown in Figure 9-34.

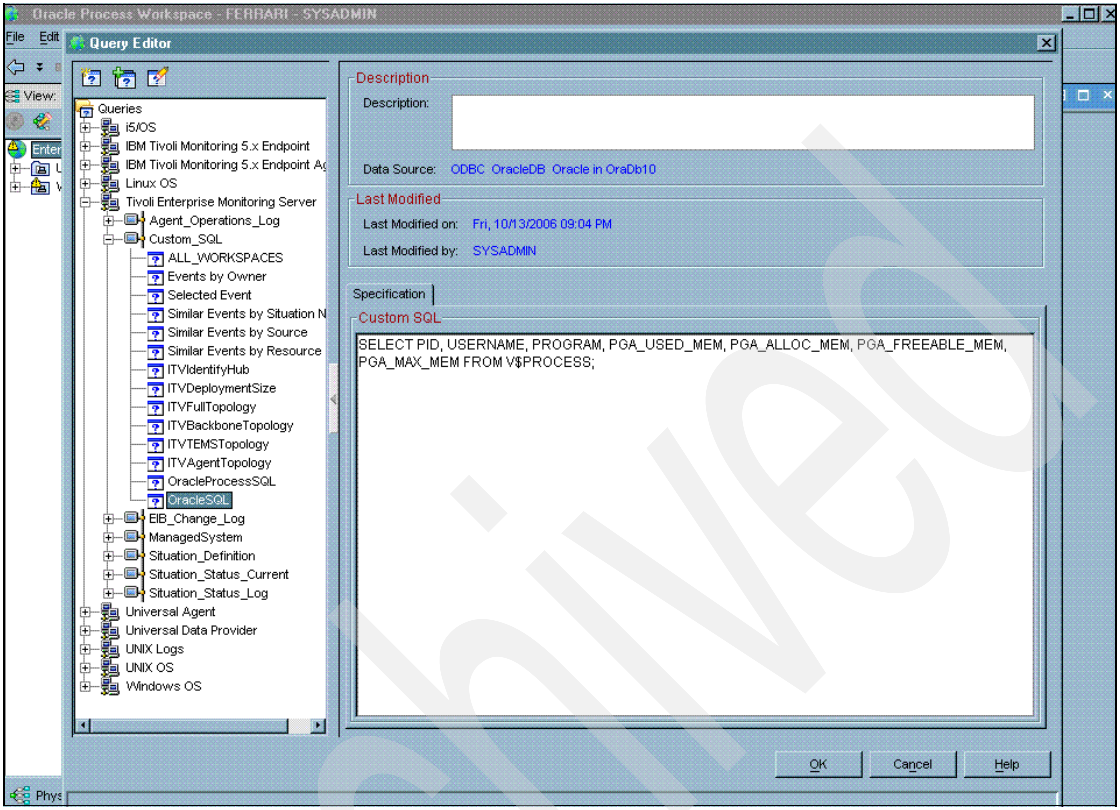
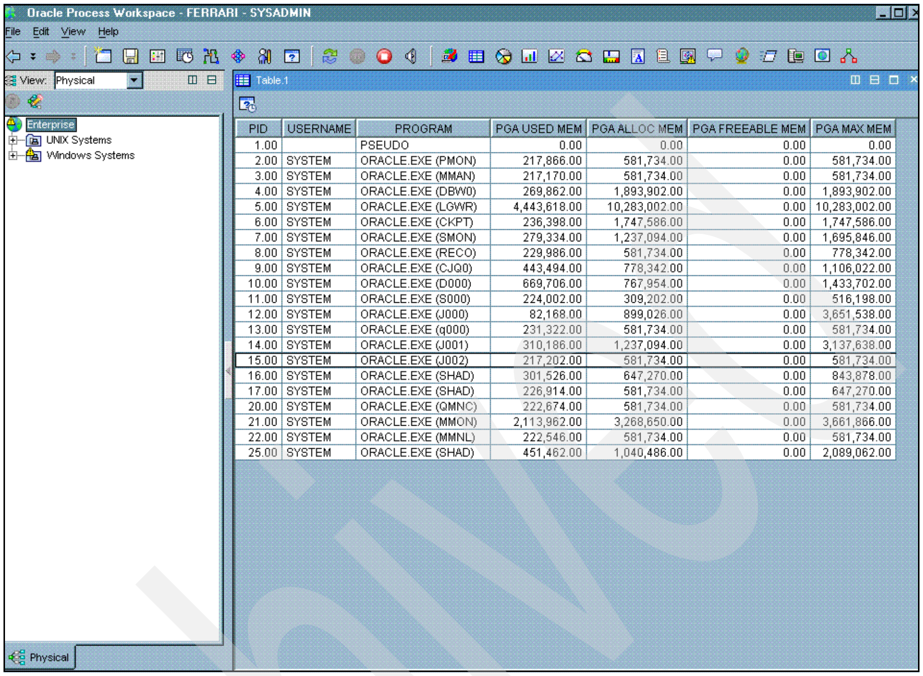


Figure 9-34 Query selection window

The view opens with the data retrieved using SQL query, as shown in Figure 9-35.



PID	USERNAME	PROGRAM	PGA USED MEM	PGA ALLOC MEM	PGA FREEABLE MEM	PGA MAX MEM
1.00		PSEUDO	0.00	0.00	0.00	0.00
2.00	SYSTEM	ORACLE.EXE (PMON)	217,866.00	581,734.00	0.00	581,734.00
3.00	SYSTEM	ORACLE.EXE (MMAN)	217,170.00	581,734.00	0.00	581,734.00
4.00	SYSTEM	ORACLE.EXE (DBW0)	269,862.00	1,893,902.00	0.00	1,893,902.00
5.00	SYSTEM	ORACLE.EXE (LGWR)	4,443,618.00	10,283,002.00	0.00	10,283,002.00
6.00	SYSTEM	ORACLE.EXE (CKPT)	236,398.00	1,747,586.00	0.00	1,747,586.00
7.00	SYSTEM	ORACLE.EXE (SMON)	279,334.00	1,237,094.00	0.00	1,695,846.00
8.00	SYSTEM	ORACLE.EXE (RECO)	229,986.00	581,734.00	0.00	778,342.00
9.00	SYSTEM	ORACLE.EXE (CJQ0)	443,494.00	778,342.00	0.00	1,106,022.00
10.00	SYSTEM	ORACLE.EXE (D000)	669,706.00	767,954.00	0.00	1,433,702.00
11.00	SYSTEM	ORACLE.EXE (S000)	224,002.00	309,202.00	0.00	516,198.00
12.00	SYSTEM	ORACLE.EXE (J000)	82,168.00	899,026.00	0.00	3,651,538.00
13.00	SYSTEM	ORACLE.EXE (q000)	231,322.00	581,734.00	0.00	581,734.00
14.00	SYSTEM	ORACLE.EXE (J001)	310,186.00	1,237,094.00	0.00	3,137,638.00
15.00	SYSTEM	ORACLE.EXE (J002)	217,202.00	581,734.00	0.00	581,734.00
16.00	SYSTEM	ORACLE.EXE (SHAD)	301,526.00	647,270.00	0.00	843,878.00
17.00	SYSTEM	ORACLE.EXE (SHAD)	226,914.00	581,734.00	0.00	647,270.00
20.00	SYSTEM	ORACLE.EXE (QMNC)	222,674.00	581,734.00	0.00	581,734.00
21.00	SYSTEM	ORACLE.EXE (MMON)	2,113,962.00	3,268,650.00	0.00	3,661,866.00
22.00	SYSTEM	ORACLE.EXE (MMNL)	222,546.00	581,734.00	0.00	581,734.00
25.00	SYSTEM	ORACLE.EXE (SHAD)	451,462.00	1,040,486.00	0.00	2,089,062.00

Figure 9-35 Custom SQL query workspace view example

You can also make bar charts for this data, as shown in Figure 9-36.

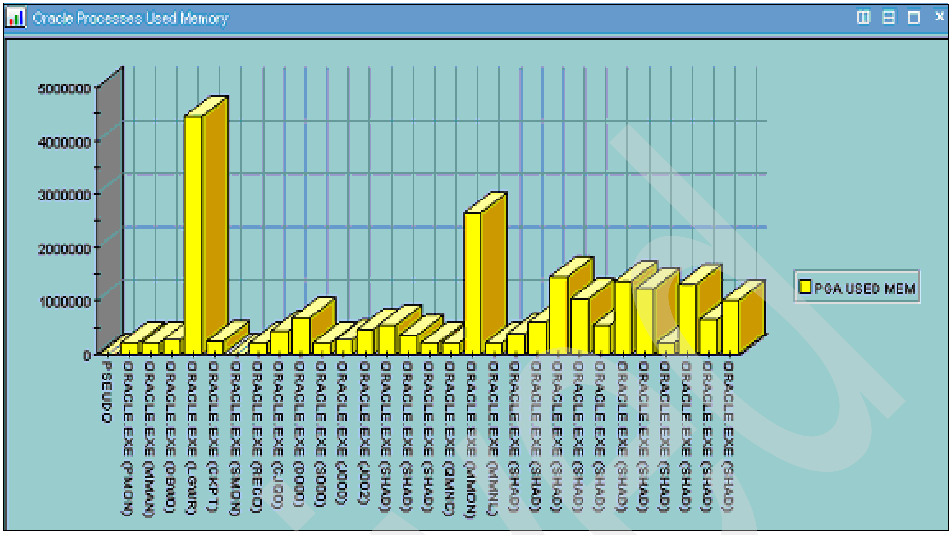


Figure 9-36 Custom data source bar graph view

Important: The data retrieved from a custom data source is for viewing and visualization purposes only. No history collection can be performed on the data queried using the custom data source. Use the Universal Agent to collect the data if you want to collect historical data for a source.

10

IBM Tivoli Monitoring resiliency and high availability

This chapter discusses the monitoring, resiliency and high availability of an IBM Tivoli Monitoring environment.

The following topics are addressed:

- Availability and resiliency of IBM Tivoli Monitoring components - general considerations
- Hub monitoring server Hot Standby option
- Clustering several IBM Tivoli Monitoring components
- Installation of IBM Tivoli Monitoring Agents within a cluster

10.1 Availability and resiliency of IBM Tivoli Monitoring components - general considerations

It is a common belief that if the monitoring environment fails, the business environment will still work. This is not always true.

Sometimes the reason for failure is the network. And when the network is not working, the business environment will most likely fail to work, too. Also, if you use your monitoring environment for automation, the monitoring environment will be a part of the business environment.

Your implementation for high availability depends on these issues, and it is often based on the size of your monitoring environment. Large scale environments typically need high availability solutions.

As you see in Figure 10-1, the IBM Tivoli Monitoring is composed of different components with different requirements. In a large scale environment, these components are mostly installed on separate systems.

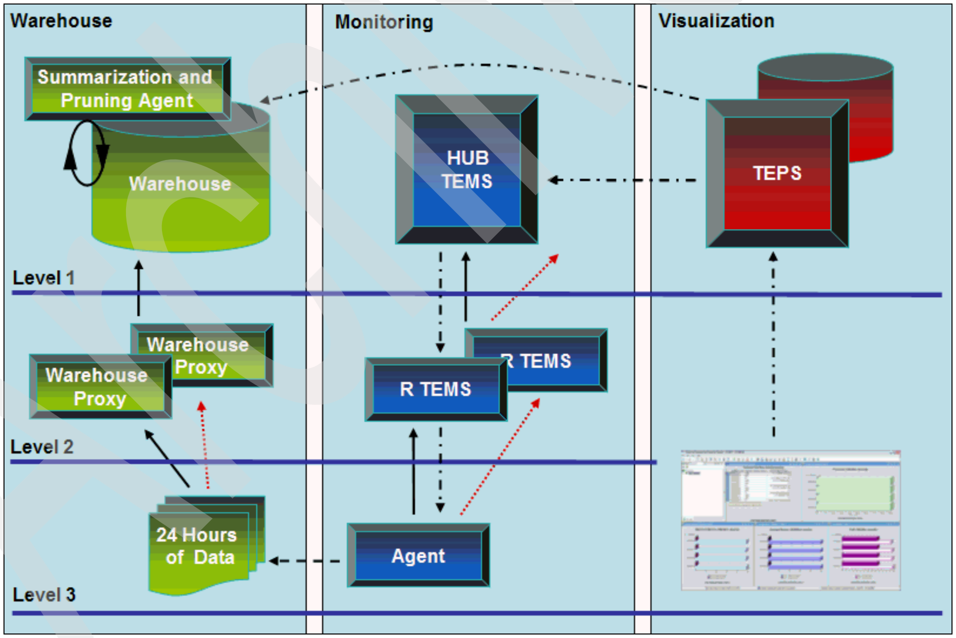


Figure 10-1 IBM Tivoli Monitoring component towers

Figure 10-1 on page 332 shows that there are three independent parts of the environment: the visualization environment; the monitoring environment; and the business environment.

If the visualization environment fails, the monitoring environment and the business environment should continue to work, but there is no chance to view it. That means you have no access to events, realtime and historical data.

In large scale environments, often an external event manager (like IBM Tivoli Enterprise Console or IBM Tivoli Netcool OMNIBus) is implemented. In this case, you can see the event data, and you know the monitoring environment works, without the visualization environment.

There are three levels for each environment:

- Level 1: The server level

  When the server level fails, the whole environment will not work.

- Level 2: The mid level

  When the mid level fails, the environment will not work for a number of units from level 3.

- Level 3: The unit level

  With a unit level failure, single unit will not work.

And finally, there are three tiers of availability:

- Tier 1: Monitoring of the components, with manual recovery.
- Tier 2: Resiliency, which means components have a backup or hot standby solution, and the switch occurs automatically in most cases.
- Tier 3: High availability, which means the component runs in a clustered solution that works automatically.

Notice that there is no unit level for historical data collection. Historical data collection is performed by the agents. Also, there is no mid level for visualization. Essentially the unit level for visualization, the portal, is not of interest in this consideration. You can start the portal in desktop or browser mode from different workplaces.

Ultimately, there are six different components to consider, but some of these components must be split into several subcomponents (like application and database).

10.1.1 Hub monitoring server availability

The hub monitoring server is the heart of the monitoring environment. It has to be treated differently than the remote monitoring server components. The hub monitoring server contains the internal monitoring database, called the Enterprise Information Base (EIB). The EIB contains the definitions of objects, such as the information about managed systems, situations, policies and definitions for historical data collection. It is the most critical component in the whole environment.

> **Important:** A remote monitoring server also has an EIB, but with different content. The hub EIB contains the *full* content for the *entire* environment. The remote monitoring server EIB only contain information for the agent that it takes care of.
>
> Also, the remote monitoring server EIB state (for example, situation status) is more volatile, in that it is rebuilt every time the remote monitoring server reconnects to a hub.

Although policies used for automation runs on the hub monitoring server independently, only a part of the situations run on the hub monitoring server. Situations regarding EIB objects and correlated situations across multiple managed systems run on the hub monitoring server. The situations on the agents, if deployed, run independent of the hub monitoring server. If the hub monitoring server fails, an agent cannot forward situation event notifications, but the situation *actions* will continue to run.

> **Tip:** If the agent is reporting to a remote monitoring server, the agent can still report situations to its remote monitoring server, even though the hub fails. This data will later be lost when the remote monitoring server reconnects to the hub and cleans its current situation status cache.

An agent can restart its own monitoring server via a restart action within the situation running on the agent. The notification might be lost, however, because the situations raised while the hub was down will be lost, unless they are still true at the agent when they are restarted as part of the reconnection behavior.

The Warehouse Proxy agent and the Summarization and Pruning agent will work independently, because the instructions are stored at the agent level.

Hub monitoring server monitoring

Basically, if you use a portal, you will see an error message box when the hub monitoring server is not available, as shown in Figure 10-2 on page 335.

Figure 10-2 Error message for lost connection from portal server to monitoring server within the portal

If you do not use the portal, the hub monitoring server service can be monitored by a situation to check on the Tivoli Enterprise Monitoring Server processes running on a agent on the same machine as the hub. You can use the Action option within the situation for an external notification via e-mail or pager, but the action has to be started from the agent, not from the monitoring server.

An agent will work 5 days without a monitoring server connection, if you use the default values for the variables CTIRA_MAX_RECONNECT_TRIES (720 times) and CTIRA_RECONNECT_WAIT (600 seconds).

If you use a Tivoli Enterprise Console or OMNIBus event manager, an event heartbeat can be implemented. You can use a simple situation for heartbeat running on monitoring server which creates an informational situation event notification when the server has the status of ONLINE (see “Remote monitoring server monitoring” on page 338).

The situation must *not* be associated with a navigator item, but the situation must forward the event notification via the event adapter to the event manager. Within the event server the incoming event can be dropped, but if there is no incoming event after a specified time period, an error event must be created within the event server via rule in the event management configuration. In every case, the hub monitoring server must be restarted manually.

10.1.2 Portal server availability

If the portal server is not operational, there will be no view of the environment. If you have an event manager, this aspect is somewhat less significant.

There are two components and two different considerations for the portal server availability:

- Portal server service and the database
- The database application (RDBMS)

The portal server service and the portal server database should run on the same system.

Portal server monitoring

The monitoring scenario for the portal server is simple. When there the connection between the monitoring server and portal server is lost, you will see a message box within the portal, as shown in Figure 10-3.

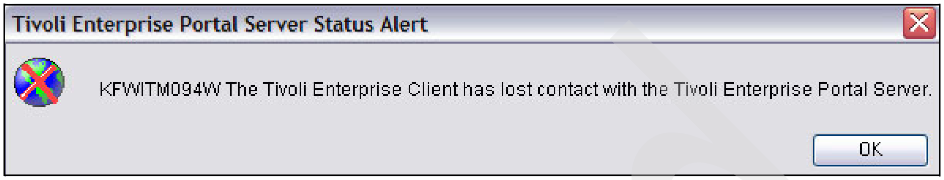


Figure 10-3 Error message for lost portal server connection within the portal

You can use a basic OS agent for monitoring the services for the portal server and the portal server database services within a situation again and restart these or send an external notification, as described for the hub monitoring server as well. If you have an event manager, you can forward the event notification, because the event adapter runs on the hub monitoring server.

For monitoring the portal server database and the RDBMS, you can use a solution based on the Universal Agent or the IBM Tivoli Monitoring Agent for Databases. Both vendor types are included in the agent package.

Using a portal server standby

For more resiliency of the portal server service, you can implement one or more view-only portal servers, as described in “Scenario - Deploying view-only portal server” on page 131. If the master portal server fails, you can monitor your environment, but you will not be able to customize the workspaces during the time of failure. This feature only ensures the availability of the service and of the database, because a replica of the database exists. In many cases, RDBMS availability is handled independent of the monitoring environment.

Portal server high availability

For highly critical environments there might be a requirement to have a high availability solution for the portal server, which means implementing the portal server within a cluster. This is discussed in 10.3.2, “Portal server cluster solution” on page 359, and is also a part of the solutions published on the OPAL Web site.

10.1.3 Availability of the Data Warehouse and Summarization and Pruning agent

If the warehouse database fails, you have no access to long-term historical data for the time of failure but you can still monitor the environment, so often this is not a showstopper.

Because the warehouse database does not work, no data will be lost. The short-term historical data will not be uploaded; instead, it will be stored on the local disks of the agents or monitoring server, depending on the configuration. You should monitor the database processing on the warehouse.

The Summarization and Pruning agent runs once a day. If it is not operational (because the database or the agent itself is not running), then you will not have summarized data from the last cycle. This task will be performed within the next cycle. Otherwise, you can initiate the summarization and pruning task manually again.

You can also consider using clustering for high availability of the Summarization and Pruning agent.

Monitoring the Warehouse

There are two parts involved in monitoring the Warehouse: the warehouse database called WAREHOUS, and the database application (RDBMS). For monitoring the warehouse database and the RDBMS, you can use a solution based on the Universal Agent or the Monitoring agent for databases. All needed vendor types for the warehouse database are included in the Monitoring agent for databases.

The Summarization and Pruning Agent is a service and can be monitored with the basic OS agent or with the heartbeat situation as shown in Figure 10-4 on page 338. The database services can also be monitored through the basic OS agent.

Warehouse resilience and high availability

The database application (RDBMS) can be a client or a server. For the RDBMS applications, a number of cluster solutions exist. The database can be integrated within an existing RDBMS Server. The warehouse database can be stored on multiple disks in a RAID environment.

For backing up the database, use a backup method on the database level that allows backing up *incrementally*. If you use the file level for backup, the whole database will be stored in each backup cycle.

10.1.4 Remote monitoring server availability

The remote monitoring server is an important part of the monitoring environment in large scale environments. It can be connected up to 1000 agents, depending on the version of IBM Tivoli Monitoring.

Remote monitoring server monitoring

Monitoring of the remote monitoring server can be done with a normal heartbeat monitoring situation running on the hub monitoring server. This can be done for either all components, or for only the monitoring server.

However, it is more desirable to run the situation for heartbeat for the remote monitoring server separately, because there is a different internal heartbeat interval per default (3 minutes) or as defined in the CTIRA_HEARTBEAT variable. So it makes sense to have a smaller sampling interval within a situation for remote monitoring server. But it makes no sense to have a sampling interval in the situation smaller than the one defined in the CTIRA_HEARTBEAT variable.

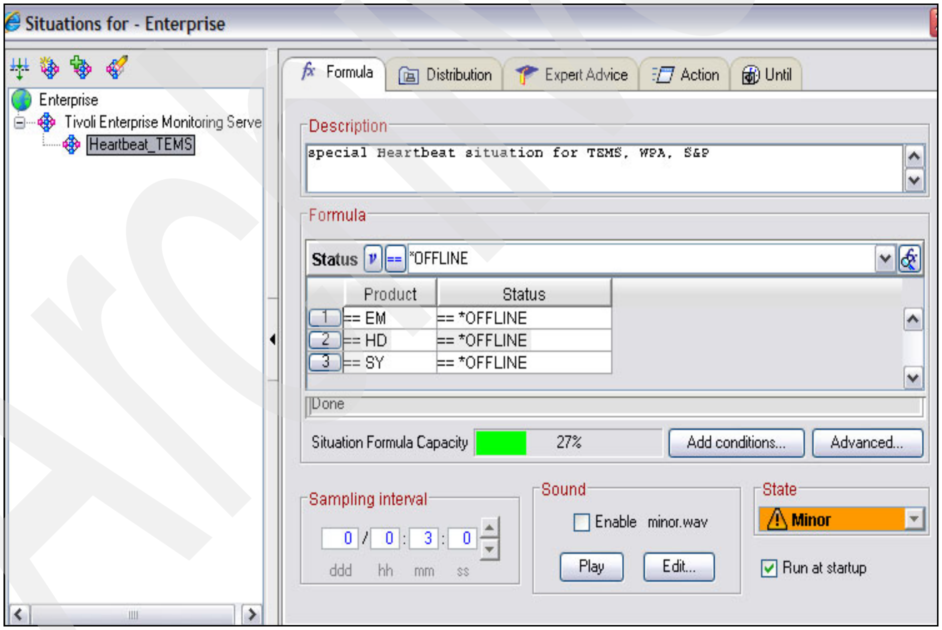


Figure 10-4 Example situation for heartbeat for monitoring server, Warehouse Proxy and Summarization & Pruning agent

Figure 10-4 shows a situation example for monitoring the remote monitoring server (product: EM). You must select the monitored application **Tivoli**

Enterprise Monitoring Server, the attribute group **ManagedSystem**, the attribute **Product** which contains the product code and **Status**, which can be compared with the value OFFLINE. The Sampling Interval is 3 minutes.

This example includes the monitoring for the Warehouse Proxy agents (product: HD) and the Summarization and Pruning agent (product: SY). The distribution must be done onto the hub monitoring server or the *HUB managed system list, because the hub monitoring server is the owner of this attribute group. The association should be done with the Enterprise navigator item.

You can monitor the monitoring server service on the system itself with a situation on the agent (using the restart or notification option within the Action tab of the situation). If the agent is connected to another monitoring server (for example, the hub monitoring server), then you can see the event notification from this situation on the portal.

Remote monitoring server resiliency

Remote monitoring server resiliency means, mostly, having one or more backup monitoring servers and using the secondary monitoring server support feature for the connected agents. The backup monitoring server can be an active remote monitoring server, or in some cases the hub monitoring server itself. Three considerations are important for using this feature:

- You should not exceed the recommended number of agents per remote monitoring server, after the takeover of (additional) agents.
- The necessary agent support files for the agents that will be taken over have to be installed, particularly for Universal Agents.
- The backup monitoring server must be reachable from the agents.

Use a heartbeat situation for monitoring the remote monitoring server (similar to the example given in Figure 10-4 on page 338).

10.1.5 Availability for Warehouse Proxy agents

When the Warehouse Proxy agent does not work, the short-term historical data cannot be uploaded into the warehouse database. You can use the short-term historical data for the last 24 hours (depending on the defined value in the KFW_REPORT_TERM_BREAK_POINT variable; refer to 5.5, “Tivoli Data Warehouse performance” on page 151 for a discussion of this parameter).

No data will be lost, if there is enough space on the local disk (agents or monitoring server). When a Warehouse Proxy agent is be connected, the data will be uploaded, so a cluster solution might not be needed for the Warehouse Proxy agent. The resiliency solution should be sufficient.

Warehouse Proxy agent monitoring

The monitoring of the Warehouse Proxy agent is important, because when the Warehouse Proxy agent fails, the data cannot be uploaded. So when an agents' disk is full, the system that the agent is installed usually will not continue to run and you might have a loss of historical data. You can use the heartbeat solution described in Figure 10-4 on page 338 to monitor the Warehouse Proxy agents.

Warehouse Proxy agent resiliency

This feature is described in 3.2.3, "Firewall Gateway Proxy" on page 45. You can implement multiple Warehouse Proxy agents in your environment. To do so, you must define the KHD_WAREHOUSE_TEMS_LIST variable on every Warehouse Proxy for assigning it to one or more monitoring server.

If the variable is not defined on one of the Warehouse Proxy agents, or the list is empty or the string *ANY exists in the list, then this Warehouse Proxy agent will work as the default.

If the Warehouse Proxy agent assigned to an agent via its monitoring server fails, the agent will use this default. But it is not supported to have a monitoring server in more than one Warehouse Proxy agents' list.

> **Note:** Warehouse Proxy clustering is a supported and certified configuration, so if you are looking at clustering the whole environment, you can implement this solution.

10.1.6 Availability of agents

Agents should be available at all times. When an agent is down, it cannot process situations and thus cannot collect any data; you will have a "hole" in the history of the data.

The importance of the agent depends on the importance of the system or application that the agent monitors. If the system can run in a cluster, then the agent must run with the clustered solution. If the agent monitors a non-critical system such as a file or print server, then the monitoring of the agent is not very important. Often more than one agent resides on any given system, such as an operating system, application agents, database agents, a log file agent, or a Universal Agent.

Monitoring of agents

For monitoring of the agents you can use a heartbeat situation as described in Figure 10-4 on page 338 according to product codes for your agent types. You can specify the agent's product codes, or you can exclude the product codes to filter the systems you monitor for heartbeat. You can also delete the product column completely to monitor the heartbeat of all systems.

However, it makes no sense to have a smaller sampling interval within the situation as defined in the CTIRA_HEARTBEAT variable for an agent. The default for the agents is 10 minutes. Every agent has its own heartbeat and you can change the heartbeat interval. This can be a performance issue. If you have different heartbeat intervals defined for your agents, it makes sense to have different situations with different sampling intervals for different agents or agent types.

High availability for agents

As previously discussed, when the server where your agents resides runs in a cluster, it is a rule of thumb that the application agents are installed in the logical system and the OS agents are installed on the physical nodes where the cluster resides. In the end it depends on the kind of the cluster configuration and the clustered application. Therefore a number of solutions, which are provided on the OPAL Web site, are discussed here in more detail in the following sections.

10.2 Hub monitoring server Hot Standby option

The Hot Standby feature, also referred to as the Fault Tolerant Option (FTO) provided by the Tivoli Monitoring platform, allows a standby hub monitoring server to be configured which can take over operations of the hub monitoring server in case it becomes unavailable. This mechanism specifically addresses the availability of the hub monitoring server, as opposed to the clustering solution (which can address a number of other Tivoli Monitoring components, as well).

> **Note:** For z/OS hub high availability considerations, you can refer to the "High-Availability HUB TEMS on z/OS" whiteaper at
> http://catalog.lotus.com/wps/poral/topal/details?cataog.label=1TW10TM61.

You must have IBM Tivoli Monitoring V6.1 Fix Pack 5 or above in order to use the Hot Standby option.

The following section is an excerpt from the white paper “IBM Tivoli Monitoring Resiliency and High Availability” which is published on the OPAL Web site:

http://www-306.ibm.com/software/tivoli/opal/?NavCode=1TW10TM4H

The paper provides a through description of the Hot Standby feature, including implementation tips and best practices.

10.2.1 Configuration and operation

The Hot Standby feature is deployed in a managed environment by installing a redundant hub monitoring server as the standby hub, and configuring various components appropriately.

Configuration

Figure 10-5 shows the configuration with the Hot Standby feature implemented. The Tivoli Data Warehouse components are omitted for the sake of simplicity. Significant points about the configuration, and additional terminology introduced for this feature, are described in the following sections.

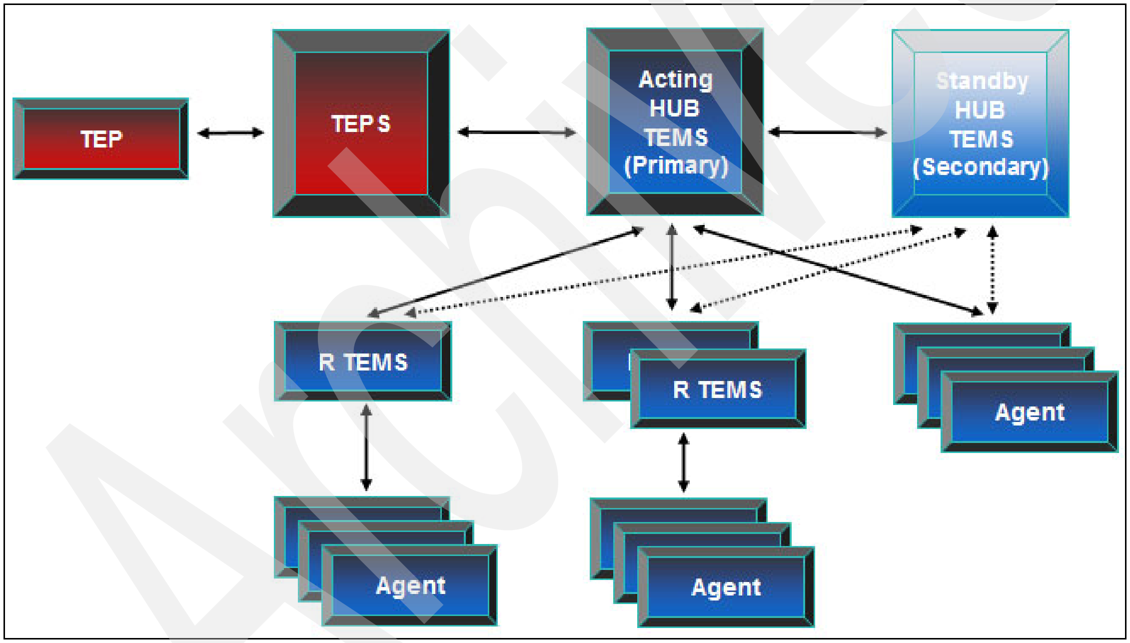


Figure 10-5 Tivoli Monitoring configuration with Hot Standby

Hub Tivoli Enterprise Monitoring Server

In a Hot Standby environment, there are two hub monitoring servers. Each hub monitoring server is installed and configured as a hub. Additionally, the configuration of each hub monitoring server designates the other hub monitoring server as the Hot Standby hub monitoring server.

In a given Tivoli monitoring environment thus configured, one of the two hub monitoring servers is actually serving as “the hub”” and will be referred to as the ‘acting hub”. The other hub monitoring server is in the standby mode, and is therefore referred to as the “standby hub”.

When the two hub monitoring servers are running, they continuously synchronize their Enterprise Information Base (EIB) data, and therefore the standby hub is capable of taking over the role of the acting hub in case it becomes unavailable. The EIB contains various definition objects such as situations and policies, information about managed systems, and the distribution or assignment of situations and policies to managed systems.

In general, the two hub monitoring servers are symmetrical, but for reasons that will become clear shortly, one hub monitoring server is designated as the “primary hub monitoring server”, and the other as the “secondary hub monitoring server”. Although it is not necessary, users can also designate the hub monitoring server that will be the secondary hub most of the time to be the primary hub, and the other hub as the secondary hub.

It should be noted at this point that the terms *acting* and *standby* refer to an operational state which can change over a period of time. The terms *primary* and *secondary* refer to configuration, which is relatively permanent.

Remote Tivoli Enterprise Monitoring Server

All the remote monitoring servers in the environment need to be configured to operate in the Hot Standby environment. For each remote monitoring server, the designated primary hub monitoring server is identified as the primary monitoring server, and the secondary hub monitoring server is provided as the Hot Standby hub in each configuration. It is required that all the remote monitoring servers be configured consistently with respect to the primary and standby hubs.

In Figure 10-5 on page 342, the connections from the remote monitoring server to the primary hub monitoring server are depicted with solid arrows. The relationship with the standby hub is depicted with dashed arrows.

Monitoring agents

For monitoring agents that report directly to the hub, as well as for the Warehouse Proxy agent, and the Warehouse Summarization and Pruning agent, the primary hub monitoring server is configured as the monitoring server to connect to, and the secondary hub monitoring server is configured as the optional secondary monitoring server.

In Figure 10-5 on page 342, the connections from the agents to the primary hub monitoring server are depicted with solid arrows. The relationship with the standby hub is depicted with dashed arrows.

Tivoli Enterprise Portal Server

At present, the portal server cannot be configured to fail over to a standby hub. When the standby hub takes over as the acting hub, the portal server needs to be reconfigured to point to the new hub monitoring server. Tivoli Enterprise Portal clients connected to the portal server do not have to be reconfigured, because they automatically reconnect to the portal server when it is brought up after the reconfiguration.

10.2.2 Operation

The Hot Standby operation is best illustrated by describing a scenario. In this scenario, all the monitoring components are started in order. In general, this happens only when the product is initially installed. Subsequently, the components can be started and stopped independently. In the scenario, HUB_PRIMARY and HUB_SECONDARY are the names given to the primary and the secondary hub monitoring server respectively. Actual names can be different.

Starting up the components

1. Initially, the primary hub monitoring server is started.
   a. When the primary hub monitoring server is started, it emits a message similar to the following in its operations log to indicate that it is configured for Hot Standby:
   ```
   04/03/07 09:38:54 KQM0001 FTO started at 04/03/07 09:38:54.
   ```
   b. Then it attempts to connect to its standby hub (the secondary hub monitoring server). Because the standby hub is not available, it takes on the role of the acting hub.
2. The secondary hub monitoring server is started.
   a. When the secondary hub monitoring server is started, it emits a message in its own log indicating that it is enabled for Hot Standby:
   ```
   04/03/07 09:47:04 KQM0001 FTO started at 04/03/07 09:47:04.
   ```
   b. Then it attempts to connect to the primary hub monitoring server. The connection succeeds. Sensing that the primary hub monitoring server started up earlier (and is therefore the acting hub), it takes on the role of the standby hub, indicated with the following messages:
   ```
   04/03/07 09:47:18 KQM0003 FTO connected to IP.PIPE:#9.52.104.155
   at 04/03/07 09:47:18.
   04/03/07 09:47:33 KQM0009 FTO promoted HUB_PRIMARY as the acting
   HUB.
   ```

c. The primary hub monitoring server also succeeds in connecting with the secondary, and emits the following messages:

```
04/03/07 09:47:50 KQM0003 FTO connected to IP.PIPE:#9.52.105.33
at 04/03/07 09:47:50.
04/03/07 09:47:58 KQM0009 FTO promoted HUB_PRIMARY as the acting
HUB.
```

d. The standby hub monitoring server queries the acting hub monitoring server to see if there were any updates to the EIB data since it last communicated with it. It replicates all such updates.

e. Subsequently, the two hub monitoring servers monitor connections with each other periodically to ensure that the other hub monitoring server is running and that there is no change in status.

f. The standby hub monitoring server also monitors the acting hub monitoring server periodically for further updates to the EIB, and replicates the updates in its own EIB. These updates are driven by change event notifications. It is thus ready to take over the role of acting hub when required. Common examples of such updates are the creation, update or deletion of a situation or workflow policy, and the assignment of a situation or a workflow policy to a managed system or a managed system list. Note that event history is not replicated.

3. Remote monitoring server and other agents are started.

   a. When these components are started up, they first attempt to connect to the primary hub monitoring server in their configuration. As it happens in this scenario, the primary hub monitoring server is also the current acting hub. The connection attempt is therefore successful, and these components start reporting to the primary hub monitoring server.

4. Portal server is connected to the primary hub monitoring server.

   a. The portal server is configured to connect to the primary hub monitoring server. One or more portal clients are connected to the portal server for monitoring purposes.

Failover

The acting hub can become unavailable for a number of reasons. The machine on which it is running may need to be shut down or may have crashed, or it could be experiencing networking problems.

When the standby hub discovers that the acting hub is unavailable, it takes over the role of the acting hub, and emits the following messages:

```
04/03/07 10:46:40 KQM0004 FTO detected lost parent connection at
04/03/07 10:46:40.
```

```
04/03/07 10:46:40 KQM0009 FTO promoted HUB_SECONDARY as the
acting HUB.
```

As the remote monitoring server and agents connected to the previously acting hub discover that the hub monitoring server is no longer available, they switch and reconnect with the new acting hub. Because these components are in various states of processing and communicating with the hub monitoring server, their discovery and reconnection with the new hub is not synchronized.

The portal server needs to be reconfigured to point to the new acting hub monitoring server, and restarted. When the portal server is shut down for reconfiguration, all portal clients connected to the portal server will display a message saying they lost contact with the Tivoli Enterprise portal server. When the portal server is restarted, the portal clients will automatically reconnect with the portal server, and display a message indicating that the portal server is available again.

The processing that takes place on reconnection is similar to what happens on reconnection in a normal environment without Hot Standby. In particular, situations and policies are restarted. As a result:

1. Pure events that occurred before the failover will not be visible. Subsequent pure events will be reported when they occur.
2. Sampled situations will be reevaluated, and will be reported again if they are still true.
3. If the monitoring server is configured to send events to the Tivoli Enterprise Console, a Master Reset Event will be sent to it when the failover occurs. Events resulting from situation reevaluation will be re-sent to it.

The new acting hub will retain its role even after the other hub monitoring server is brought up. The other hub monitoring server will now be the standby hub. When it comes up, it will check the new acting hub's EIB for updates, and perform updates to its own EIB if necessary. The two hubs will also start monitoring each other's availability.

All remote monitoring servers and agents will now report to the acting hub. There is no mechanism available to switch them back to the standby while the acting hub is still running, so the only way to switch them to the standby is to shut down the acting hub.

If an agent or remote monitoring server experiences a transient communication problem communicating with the acting hub and switches over to the standby, the standby instructs it to go back and retry the connection with the acting hub, because the standby knows that the hub is still available. Figure 10-6 depicts the new configuration.

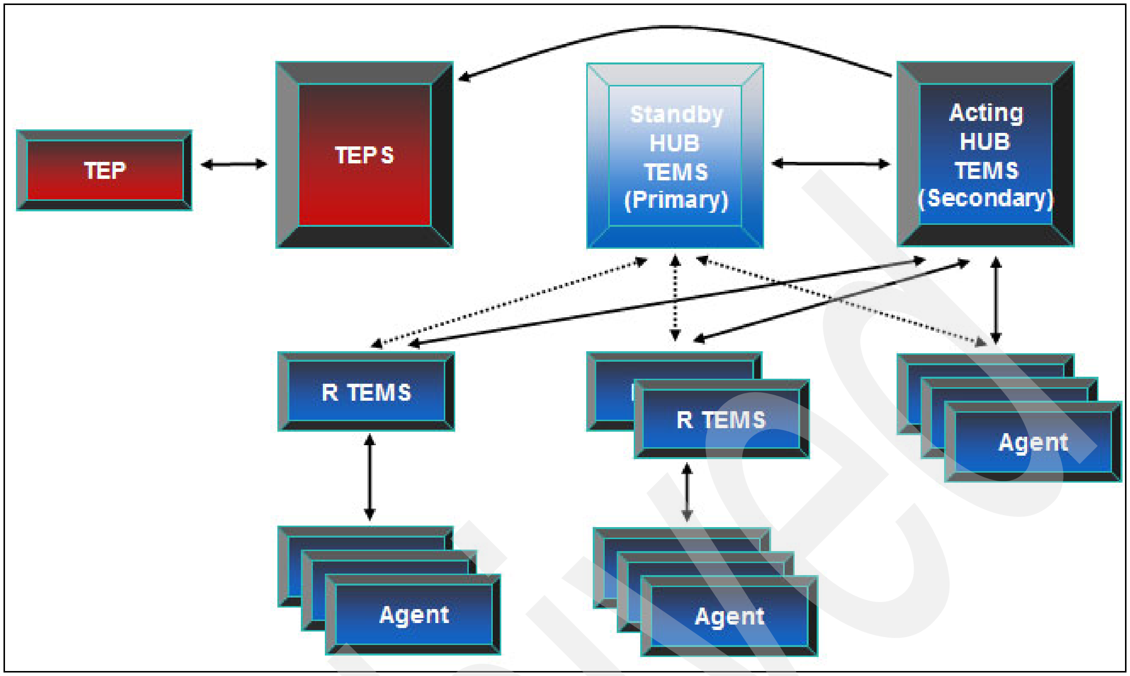


Figure 10-6 Configuration after failover

The system can continue to operate with this configuration until there is a need to shut down the acting hub, or if the machine on which it is running becomes unavailable. Each time the acting hub becomes unavailable, the scenario described in this section is repeated.

10.2.3 Tips and best practices for using Hot Standby

Some tips and best practices for using Hot Standby are presented in this section.

Configuration

1. Identify the primary and secondary hub monitoring servers to implement Hot Standby. The Tivoli Monitoring software on both machines is required to be at the same release and maintenance level. Application support for all the required applications should be installed at both hub monitoring servers. Hot Standby is not supported at present for hub monitoring server on the z/OS platform.
2. Ensure that the primary and secondary hub monitoring servers refer to each other as the standby hub in their configuration. On the Windows platform, for

example, the configuration would look like this for the primary hub monitoring server (Figure 10-7, Figure 10-8 on page 348 and Figure 10-9 on page 349).

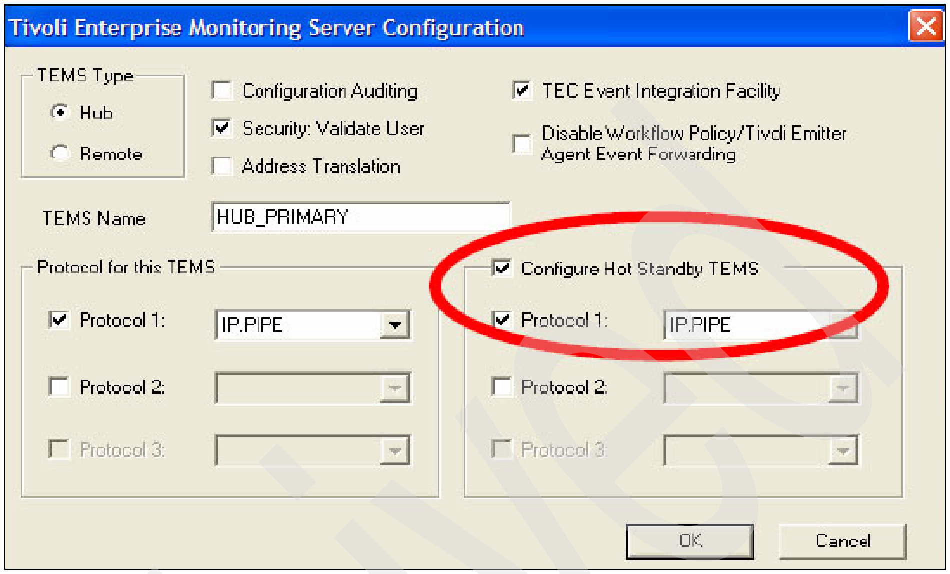


Figure 10-7 Primary hub monitoring server configuration - 1

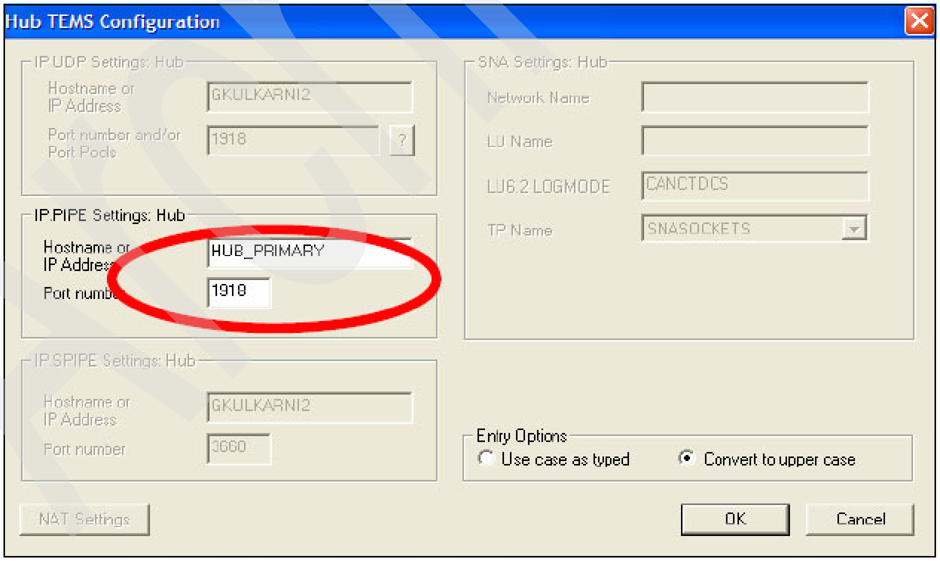


Figure 10-8 Secondary hub monitoring server configuration - 2

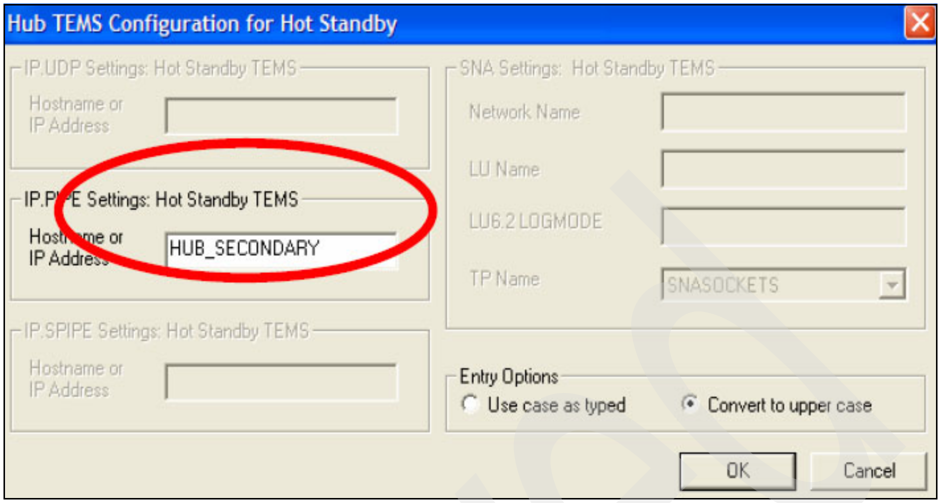


Figure 10-9 Primary hub monitoring server configuration - 3

The configuration for the secondary hub monitoring server should look like this (Figure 10-10 on page 349, Figure 10-11 on page 350 and Figure 10-12 on page 350).

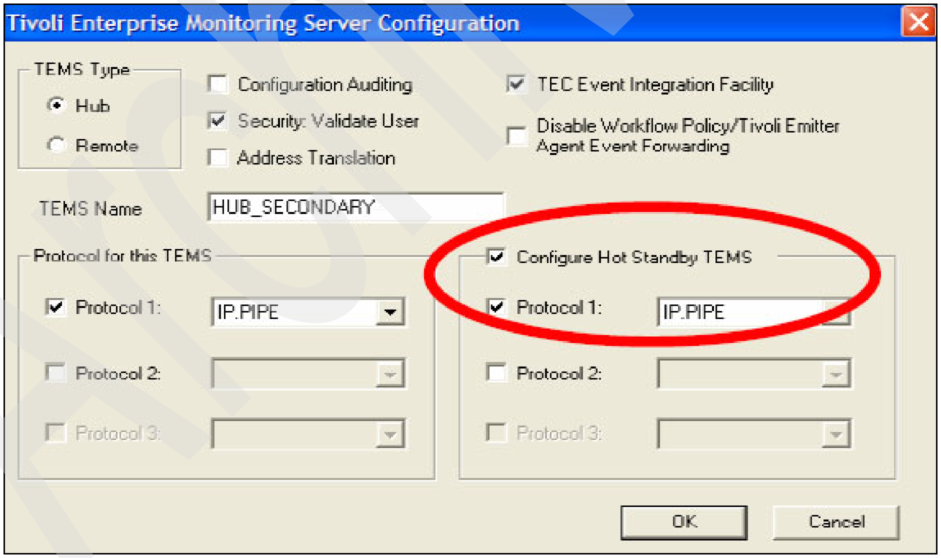


Figure 10-10 Secondary hub monitoring server configuration - 1

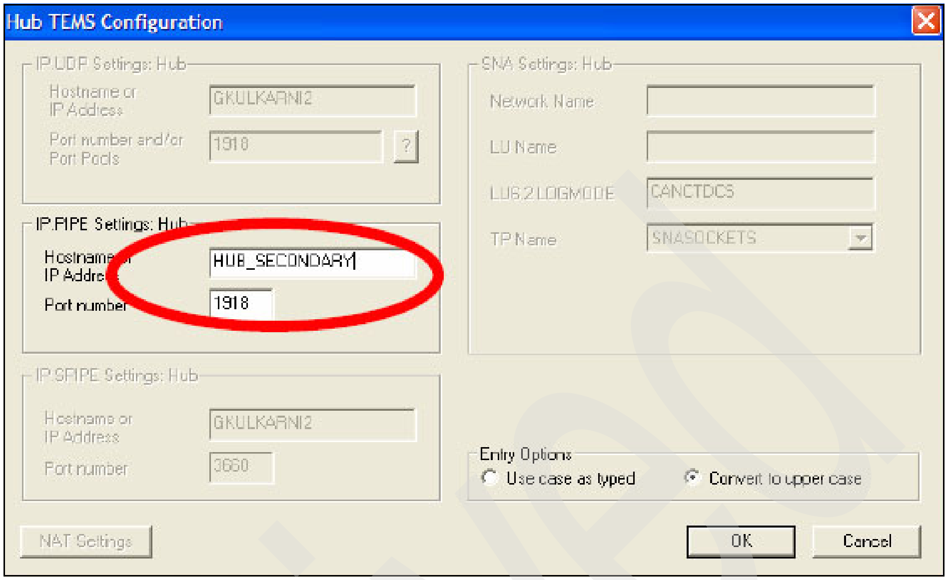


Figure 10-11 Secondary hub monitoring server configuration - 2

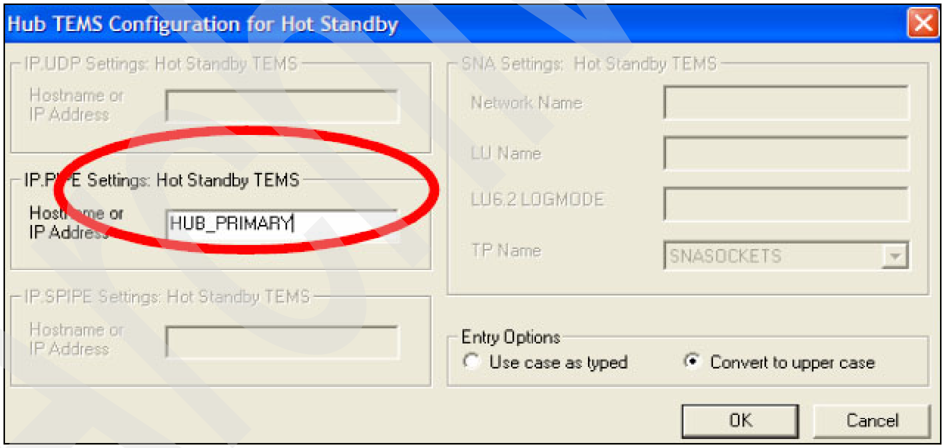


Figure 10-12 Secondary hub monitoring server configuration - 3

Further, review the operations log for each hub monitoring server after it is started to ensure that FTO messages are emitted.

3. Ensure that all remote monitoring servers are configured correctly and consistently, with the primary hub monitoring server as the monitoring server they connect to, and the secondary hub monitoring server as the standby hub (Figure 10-13, Figure 10-14 on page 351 and Figure 10-15 on page 352).

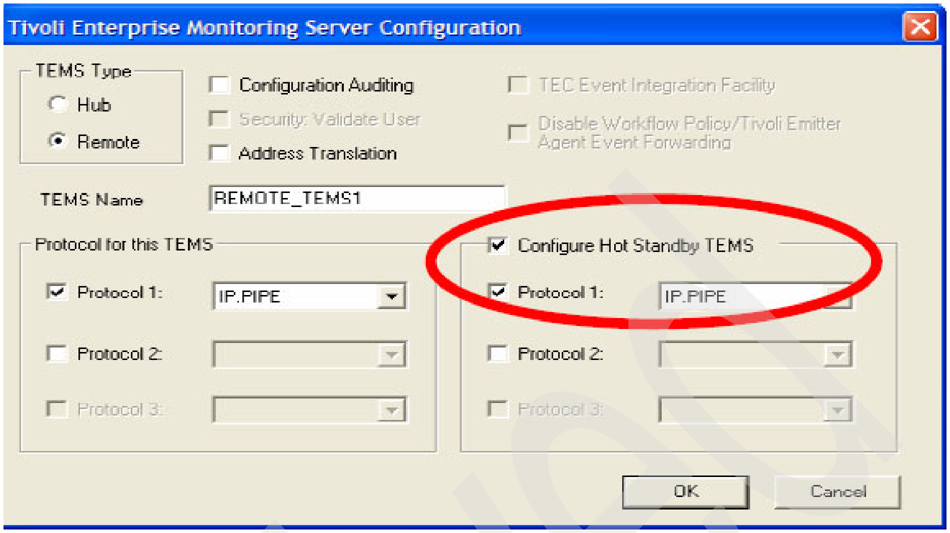


Figure 10-13 Remote monitoring server configuration – 1

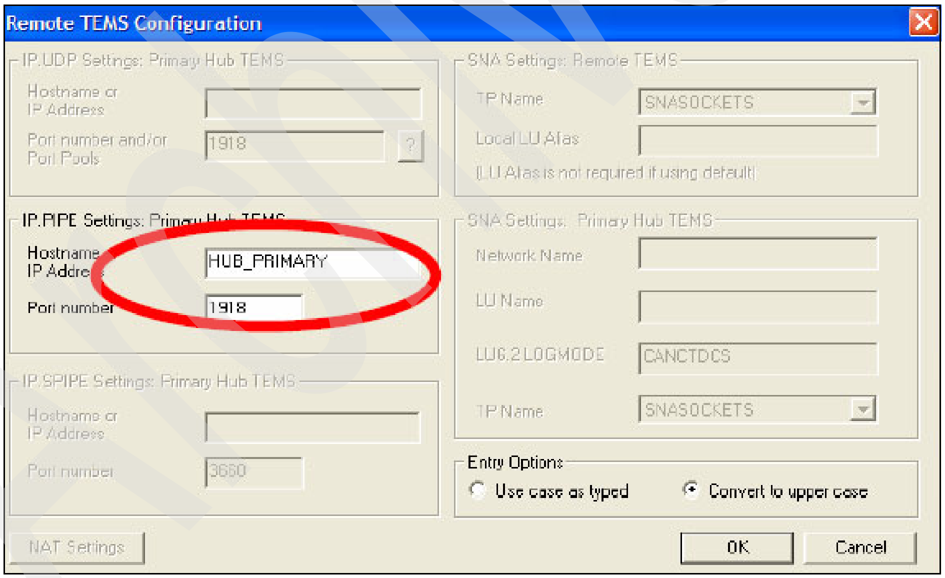


Figure 10-14 Remote monitoring server configuration - 2

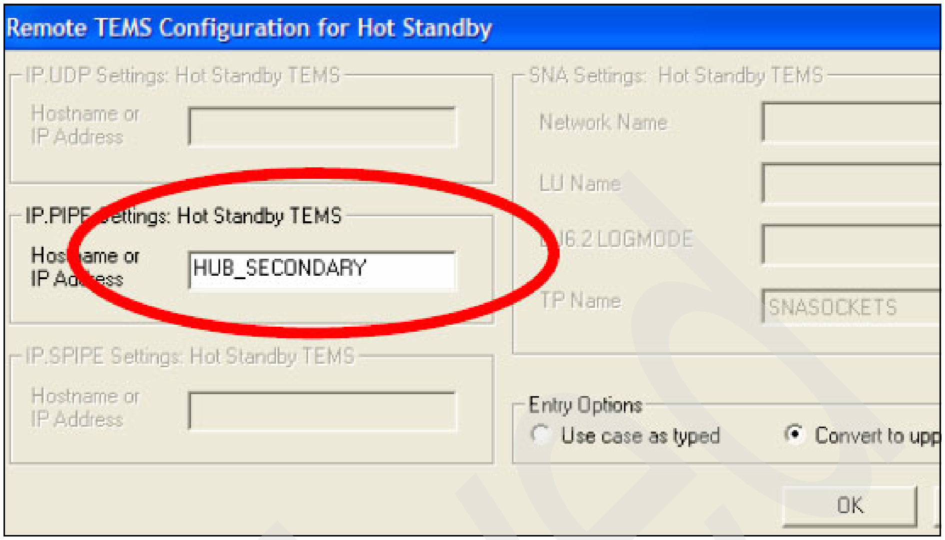


Figure 10-15 Remote monitoring server configuration - 3

4. Agents that connect directly to the hub monitoring server should also be configured in a similar manner. By default, agents send heartbeats to the *monitoring server* every 10 minutes; this means that an agent can take up to 10 minutes to detect a lost connection with the monitoring server. This environment was tested with the default of 10 minutes for the heartbeat. Changing this setting should only be made after consultation with certified Tivoli Service or Tivoli Monitoring Architecture personnel.
5. In general, agents should be configured to report to a remote monitoring server rather than to hub monitoring server where possible.

   For agents connected to the hub monitoring server, historical data should be collected at the agent rather than at the monitoring server. This way, the data will be available at the agent at all times, even when the acting hub role is transferred from the current hub to the standby hub.

Operation

1. The hub monitoring servers emit informational messages regarding communication between the acting and the standby hubs, replication, and failover. These messages have message id KQMnnnn, and can be reviewed periodically to ascertain the progress and status of the Hot Standby operation.
2. Remote monitoring servers and agents also provide messages when they connect or reconnect with or switch to a hub monitoring server. These messages are useful for understanding the status of Hot Standby.

Here is an example of a remote monitoring server message:

```
(46143C7C.0004-6:ko4ib.cpp,9805,"IBInterface::selectHub") selected
CMS <HUB_SECONDARY> as the HUB
```

Here is an example of an agent message:

```
1070404231321000KRAREG001 CMS lookup failed.
1070405085121000KRAREG000 Connecting to CMS HUB_SECONDARY.
```

3. The first time the secondary hub monitoring server is started in the standby mode, it can take several minutes to initialize in a large environment. This happens only at start up, and the time taken depends on the number of updates. For instance, if a standby hub is introduced in a large operational environment with 3000 managed systems and 200 situations, this synchronization can take in the neighborhood of 20 minutes the first time the standby hub is brought up. Subsequent restarts will be much quicker, as the standby hub continues to synchronize its data with the acting hub while it is running.
4. At any given time, if both the hub monitoring servers are running, the hub monitoring server that started up first is the acting hub, and the second hub monitoring server is the standby hub.

   In the unlikely event that the two hub monitoring servers are started at exactly the same time, the hub monitoring server with the name higher in the alphabetical sort order will take the role of the acting hub.

   If there is a communication problem between the two hub monitoring servers, then it is possible that each hub assumes the role of the acting hub. If some of the remote monitoring servers have communication problems with the acting hub as well, they may fail over to the standby, thus resulting is a split environment. The configuration will revert to normal as soon as the two hub monitoring servers start communicating again. When the two hubs establish communication again, the hub that started first will be accorded the status of the acting hub.

   This would be the same hub that was the acting hub before they lost communication. In order to minimize the possibility of occurrence of this situation, the two hub monitoring servers should be placed on machines with a reliable network connection between them.
5. Periodic backups of the portal server and hub monitoring servers should be taken.

Considerations for the Hot Standby feature

This section summarizes the important points regarding the feature you should know about before implementing Hot Standby:

- A single portal server which is reconfigured to point to the acting hub monitoring server should be used, rather than having two portal server instances pointing to each hub monitoring server. This means that after each failover, the portal server needs to be reconfigured to point to the new acting hub. This is because the portal server database is not replicated.
- The standby hub replicates definition data only. Configuration files and settings, catalog and attribute files, and software depots used for remote deployment are not replicated. All changes made to the configuration subsequent to the initial installation should be made at both hub monitoring servers. Thus, if support for a new agent was installed at one hub, it would need to be installed at the other hub as well.
- Situation history is not replicated.
- Universal Agent metadata, as well as Catalog and Attribute files generated for the Universal Agent, are not replicated between the hub monitoring servers. For Universal Agents connected to the hub, these files should be copied to the standby hub monitoring server when the files are generated. For Universal Agents connected to remote monitoring servers, the files need to be copied to both hub monitoring servers.
- The OMEGAMON for WBI version 1.1, and the OMEGAMON for Messaging version 6.0 products use a configuration database at the hub monitoring server. This database is not replicated by the standby hub.

  If these products are being used for MQ Configuration, then:

  – Only the primary hub monitoring server should be used for performing MQ Configuration functions.
  – The secondary hub monitoring server should be used for standby purposes only. In case of a failover from primary to secondary, the secondary hub monitoring server should be shut down after the primary becomes available again, and restarted only after the primary takes over.
  – MQ configuration functions should not be performed while the secondary hub monitoring server is the acting hub.
- When the acting hub is shut down, it should be restarted only after all remote monitoring servers and agents connected to the hub have successfully failed over to the standby.
- If the standby hub is to be promoted as the acting hub, the currently acting hub needs to be shut down. There is no automatic failback.
- When the secondary hub is the acting hub, the standby hub should be temporarily shut down when remote servers are being recycled.
- The processing that takes place when a remote monitoring server fails over to the standby hub is similar to what happens on reconnection in a normal

environment without Hot Standby. In particular, situations and policies are restarted. As a result:

– Failover is *not* instantaneous.
– Pure events that occurred before the failover will *not* be visible. Subsequent pure events will be reported when they occur.
– Sampled situations will be reevaluated, and will be reported again if they are still true.
– If the *monitoring server* is configured to send events to the Tivoli Enterprise Console, then a Master Reset Event will be sent to Tivoli Enterprise Console when the failover occurs. Events resulting from situation reevaluation will be re-sent to the Tivoli Enterprise Console.

Tip: When using the Hot Standby feature, it a best practice to configure your environment so that only remote monitoring servers connect to the hub monitoring server, not the agents. In this case, the time for takeover will be decreased because the reconnecting time for the remote monitoring server is typically smaller than for the agents, and switchback is easier and faster.

Due to these Hot Standby considerations, using a high availability solution such as Tivoli Systems Automation might be more effective in large scale environments. This means implementing the hub monitoring server within a cluster. A number of solutions are provided on the OPAL Web site, and we also discuss them in the following section.

10.3 Clustering several IBM Tivoli Monitoring components

Cluster or clustered environments are units consisting of two or more systems (nodes) with a number of sharing resources for access from all nodes and a piece of cluster software for occurring the monitoring and take-over scenarios. Cluster solutions are tailored for operating systems where an application runs that needs high availability.

Some applications are not ready for running in a cluster, because they need certain properties of the hardware where they run. IBM Tivoli Monitoring components, however, are cluster-enabled.

The cluster solution must deliver four resources that are needed for the IBM Tivoli Monitoring component. There are:

- Unique IP address

- Unique hostname
- A shared disk
- The possibility to activate services of the operating system in every node of the cluster

Typically within the cluster each node has two network interfaces: one for public, and one for the internal heartbeat function of the cluster. The unique IP address and the unique hostname are mostly configured as virtual and assigned to the active node where the application runs. Furthermore, it is good practice to place the shared disk in a mirror mode.

Important: For Windows, IBM Tivoli Monitoring components cannot be installed in multiple installation directories as you do for UNIX or Linux. Therefore, each IBM Tivoli Monitoring component must be used within the same cluster shared disk and defined in the same cluster resource group. If you have installed a Windows OS agent on the same system as another clustered IBM Tivoli Monitoring component, the Windows OS agent must run within the shared resource group and will be moved with them.

Note: As of IBM Tivoli Monitoring V6.1, including Fix Pack FP4, FP5, FP6, the following considerations are true for GSKit and Java:

- For Windows:

  GSKit and Java have to be installed into the *same* install directory. Both can be installed into user-defined install directory (for example, \shareddisk\itm).

- For UNIX/Linux:

  GSKit cannot be installed into a user-defined install directory. Instead, it goes to the default directory (/usr/local/ibm/gsk7).

  Java *can* be installed into a user-defined IBM Tivoli Monitoring install directory (for example, /shareddisk/itm).

10.3.1 Hub monitoring server cluster solution

The hub monitoring server is the most critical component, and is therefore the first candidate for running within a cluster. Figure 10-16 on page 357 shows the implementation for a hub monitoring server within a cluster.

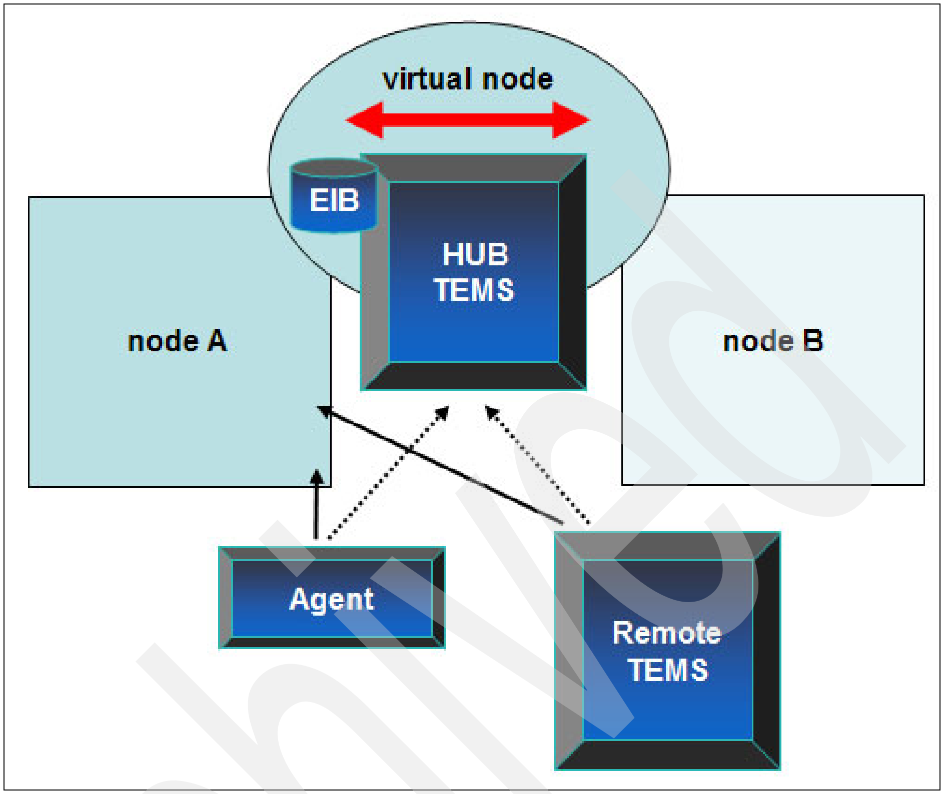


Figure 10-16 Monitoring server cluster

Using the cluster solution compared to the Hot Standby solution offers the following advantages:

- Using the same Enterprise Information Base (EIB, no synchronizing)
- No IP address change
  - The remote monitoring server and direct connected agents must not use the secondary TEMS support, so there is no loss of time for the switching period of the cluster.
  - No portal server switching is necessary.
- Less effort for maintenance within the Monitoring environment, for:
  - Maintenance of TEMS support files
  - Configuration of secondary TEMS support
  - Configuration and synchronization of a second portal server

Figure 10-16 on page 357 shows the implementation for a hub monitoring server within a cluster. Following are the basic steps for implementing a hub monitoring server within a cluster:

1. Be sure that the DNS is working for the virtual IP address.
2. Create the shared disk (or file system), which you can use for IBM Tivoli Monitoring installation path (CANDLEHOME):
   - For Windows: /IBM/ITM
   - For UNIX and Linux: /opt/IBM/ITM
3. Install the hub monitoring server on the active node of the cluster (use the shared disk).
4. For UNIX/Linux systems, install the Global Security Kit (gskit) and the Java Run Time Environment (JRE) on both nodes to the same path.

> **Tip:** For Windows, you do not need to install those on both nodes, only install them into shared disk.

5. Configure the hub monitoring server and install all necessary support files.
   - Use the virtual hostname for the monitoring server name.
6. Configure the cluster software for starting the monitoring server service.
   - For the basic installation on the node, the startup for the service must be set to manual (no automatic restart).
7. Test the cluster, before and after a switch:
   - The same Managed Objects should included in the EIB.
   - The portal server should work without reconfiguration.

Steps 2 and 6 vary for different cluster solutions. Several cluster solutions have been tested and described for a hub monitoring server on the OPAL Web site. Use these solutions for configuring your own environment. The existing solutions are listed in Table 9-1. The OPAL Web site is located at:

http://catalog.lotus.com/wps/portal/tm

Table 10-1 Cluster solutions for the hub monitoring server on the OPAL Web site

Operating system	Cluster	Heading on the OPAL Web site
Windows	Microsoft Cluster Server	Clustering IBM Tivoli Monitoring Components in a Microsoft Cluster Server Environment (current link: http://www.ibm.com/software/tivoli/opal?NavCode=1TW10TM4E)

Operating system	Cluster	Heading on the OPAL Web site
AIX	HACMP™	Clustering IBM Tivoli Monitoring Components in a HACMP Environment (current link: http://www-306.ibm.com/software/tivoli/opal/?NavCode=1TW10TM4G)
Linux (SLES 10)	Tivoli System Automation for Multiplatforms	Clustering IBM Tivoli Monitoring Components in a IBM Tivoli System Automation for Multiplatforms Environment (current link: http://www-306.ibm.com/software/tivoli/opal/?NavCode=1TW10TM4F)
z/OS (MVS)	1	Creating a Moveable HUB TEMS on z/OS (current link: http://www-306.ibm.com/software/tivoli/opal/?NavCode=1TW10TM1W)

1 This is not really a cluster solution. The white paper describes how it will allow the hub monitoring server to be relocated to any suitable LPAR in the environment with no changes, and with minimal disruption to all connecting components.

Notice the versions used for IBM Tivoli Monitoring components, operating systems, and cluster software that were tested in the scenarios. Implementation might be different, for example, for a different version of the cluster software.

Note: IBM Tivoli Monitoring V6.2 will ship with a copy of IBM Tivoli System Automation for Multiplatforms high availability software, to be used for the high availability of IBM Tivoli Monitoring V6.2 components.

10.3.2 Portal server cluster solution

For critical environments, visual monitoring of the environment is essential. This can be a reason for installing the portal server within a cluster. For the installation within a cluster, there are three considerations:

- Database application (RDBMS)
- Database
- Portal server application

Figure 10-17 on page 360 shows the implementation for a portal server within a cluster.

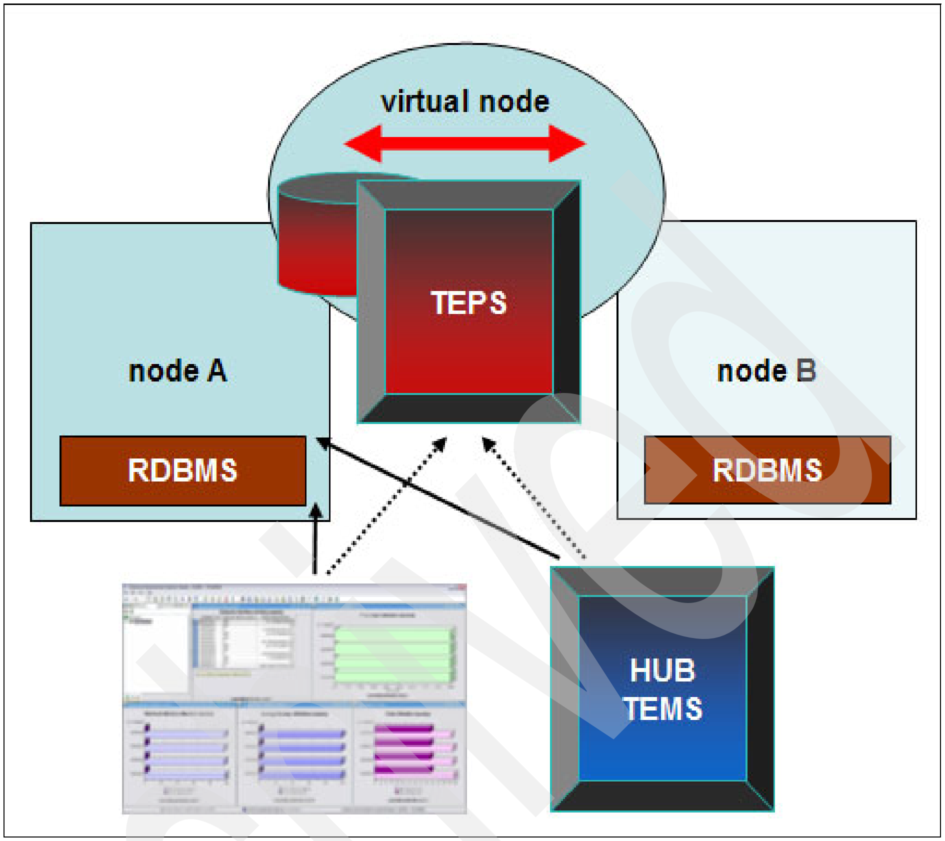


Figure 10-17 Portal server cluster

The advantages of using the portal server within a cluster compared to using a read-only portal server are:

- The same IP address and hostname are used for connecting components.
- The cluster solution allows write access to your workspaces.

One key point is, the database application used must be enabled for working within a cluster. DB2 is such a database, and so is MS-SQL (for Windows). The steps for implementing a portal server within a cluster are:

1. Be sure that the DNS is working for the virtual IP address.
2. Create the shared disk (or file system), which you can use for IBM Tivoli Monitoring installation path (CANDLEHOME) and the portal server database:
   - For Windows: /IBM/ITM
   - For UNIX and Linux: /opt/IBM/ITM

3. Install the RDBMS on both nodes (not on the shared disk).
   - Set up database users and groups for both nodes, and be sure they are exactly the same.
4. Create a portal server database on the active node of the cluster (use the shared disk).
   - Catalog this database on the second (inactive) node.
5. Install the portal server on the active node of the cluster (use the shared disk).
6. For UNIX/Linux systems, install the Global Security Kit (gskit) and the Java Run Time Environment (JRE) on both nodes to the same path.

> **Tip:** For Windows, you do not need to install those on both nodes, only install them into the shared disk.

7. Configure the portal server and install all necessary support files.
   - Use the virtual hostname and IP address for the portal server.
8. Configure the cluster software for starting the portal server and database services.
   - For the basic installation on the node, the startup for the services must be set to manual (no automatic restart).
9. Test the cluster, before and after the switch:
   - A portal can connect without reconfiguration.
   - The portal server and hub monitoring server connection works.

Steps 2 and 8 vary for different cluster solutions. Several cluster solutions have been tested and described for a portal server with DB2 within the OPAL Web site, listed in Table 10-1 on page 358 in rows 1 - 3. Use these solutions to configure your own environment.

> **Note:** For Windows, the IBM Tivoli Monitoring installer cannot create the sysadmin Administrator ID as an Active Directory Domain User. So the default user ID for configuring the portal “sysadmin” must be created manually as a local user on the second node with the same settings as on first node of the cluster.

10.3.3 Warehouse cluster solution

For some customer environments, it is important to have continuous access to long-term historical data. For such environments, the warehouse database can be installed within a cluster. If the Summarization and Pruning agent resides on the same system as the database, it makes sense to put it within the cluster resource, as well. If the agent does not work, however, the data will not be summarized or pruned for the last day (unless the server fails within the scheduled time when the agent works).

It is very similar to the installation of the portal server database within a cluster. There are the RDBMS and the database. The RDBMS must be enabled for a cluster, as the case for DB2, MS-SQL, and Oracle.

The general procedure for installation within a cluster is:

1. Be sure that the DNS is working for the virtual IP address.
2. Create the shared disk (or file system) that you can use for the WAREHOUS database.
3. Install the RDBMS on both nodes (*not* on the shared disk).
   - Set up database users and groups for both nodes; be sure they are exactly the same.
4. Create a WAREHOUS database on the active node of the cluster (use the shared disk).
   - Catalog this database on the second (inactive) node.
5. Install the Warehouse Proxy agent, GSKit and Java components.
6. Configure the cluster software for starting the database services.
   - For the basic installation on a node, the startup for the services must set to manual (no automatic restart).
7. Test the cluster, before and after a switch:
   - A portal can access to any kind of summarized data.
   - Check the log file of a Warehouse Proxy agent, make sure that the upload of short-term historical data works.

If you want to install the Summarization and Pruning agent within the cluster as well, you need the following additional steps:

8. Install the Summarization and Pruning agent on the active node of the cluster (use the shared disk).
9. For UNIX/Linux systems, install the Global Security Kit (gskit) and the Java Run Time Environment (JRE) on both nodes to the same path.

Tip: For Windows, you do not need to install those on both nodes, only install them into the shared disk.

10. Configure the Summarization and Pruning agent.
   – Use the virtual hostname and IP address for the portal server.
11. Configure the cluster software for starting the Summarization and Pruning service.
   – For the basic installation, the startup for the service must set to manual (no automatic restart).
12. Test the cluster, before and after a switch:
   – A portal can access summarized data built within the last scheduled interval of the Summarization and Pruning agent.
   – A portal has no more access of detailed data older than the pruning definition defined for a destined attribute group.

Steps 2, 5, and 10 vary for different cluster solutions. Several cluster solutions have been tested and described for a portal server with DB2 within the OPAL Web site, listed in Table 10-1 on page 358, rows 1-3. Use these solutions to configure your own environment.

10.4 Installing IBM Tivoli Monitoring Agents within a cluster

The reason for installing IBM Tivoli Monitoring Agents within a cluster is to have an agent follow a monitored application installed in a cluster. The agent has to go with the application in order to perform its monitoring job. Two things must be unique for different types of agents:

- The OS agent has to monitor the node and the operating system.
- The application agents have to monitor the application.

It is the intention of a cluster that the application resides on the shared resource, so the application agent should also be installed on the shared resource to move with the application. Placement of the OS agent should be done carefully; some of the components of the operating system are included in the node, although others are moved with the shared resource. This is different for Windows compared to UNIX or Linux clusters, because of the Windows OS limitations.

> **Note:** If you install multiple OS agents on the same system, there is no additional impact on your licences. That is, licensing will be determined by the number of processors, not by the number of agents.

10.4.1 Agents within a Windows cluster

Important: There are two limitations for Windows:

- IBM Tivoli Monitoring components on Windows operating systems are registered as a service. There can be only *one* instance of each service on the computer.
- IBM Tivoli Monitoring components *cannot* be installed in multiple installation directories. On Windows, IBM Tivoli Monitoring 6.1 requires agents to be installed in the same directory path as the OS agent, therefore each node in a cluster must have all agents installed (on the nodes system disk) that are required to support the cluster applications that can "possibly" run on that cluster node.

Install agents within the shared disk

As a consequence of these limitations, if you have one or more application agents on your system, the installation directory of IBM Tivoli Monitoring should be on the shared disk, because the application agents will be installed within the shared disk. So, the Windows NT® OS agent must be installed on the shared disk as well, and will be moved together with the resource group. With the OS agent on the shared disk (including the heartbeat situation), you have no chance to monitor the inactive node to see whether it is ready for take over.

You can use a solution to monitor the nodes with the Universal Agent remote monitoring, based on the POST, AVIS, or SNMP data provider. For this solution, the Universal Agent must *not* be installed within the cluster. Figure 10-18 on page 365 shows such an example implementation within a Windows cluster.

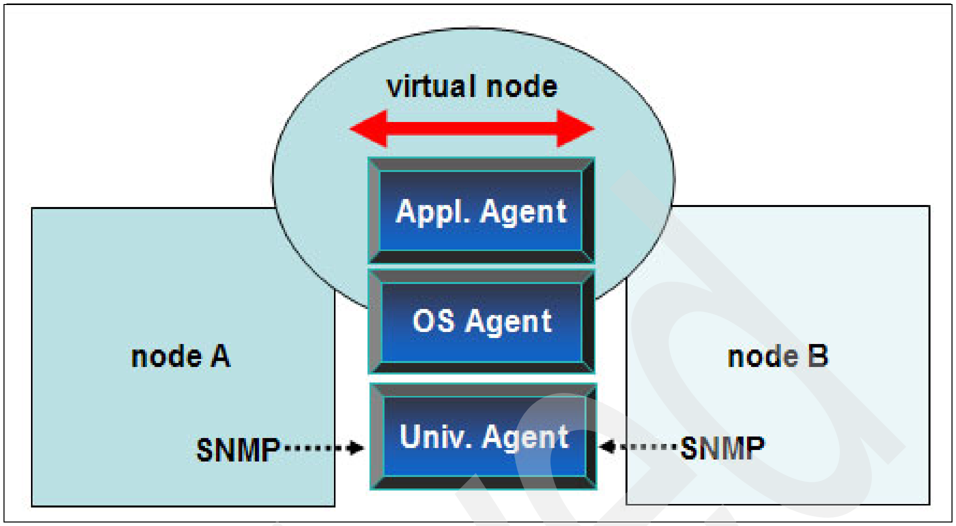


Figure 10-18 Windows cluster with agents within the shared disk

The installation procedure for agents on Windows within a cluster is as follows:

1. Install all agents on the active node of the cluster (use the shared disk).
2. Configure the agents.
3. Configure the cluster software for starting the services, one per agent.
   – For the basic installation, the startup for the services must set to manual (no automatic restart).
4. Test the cluster, before and after a switch:
   – Check if the agent within the portal is active.

A number of solutions exist on the OPAL Web site for monitoring the cluster based hardware with the Universal Agent. You can use these solutions to monitor the health of your active and inactive nodes of the cluster; refer to row 1 in Table 10-2 on page 367 for such solution.

Install agents outside the shared disk

Another solution is to define the IBM Tivoli Monitoring directory *outside* of the shared disk. In this case, you must install the Windows OS agent and the application agents on *every* node of the cluster. When the application moves, monitoring will be done from the other node.

For this solution, it is important to set the virtual hostname manually for the application agents using by the variable CTIRA_HOSTNAME. For the application agents, the start option on the nodes must be set manually and the automated start must be integrated in the cluster resource. So, the cluster will start the application agents on the active node and you will see the virtual hostname on the portal.

The Windows OS agents are installed with the physical nodes, running and monitoring both the physical nodes and the shared resource. You can group these agents within a logical Navigator item on the portal for visualization. Figure 10-19 shows such an example implementation within a Windows cluster.

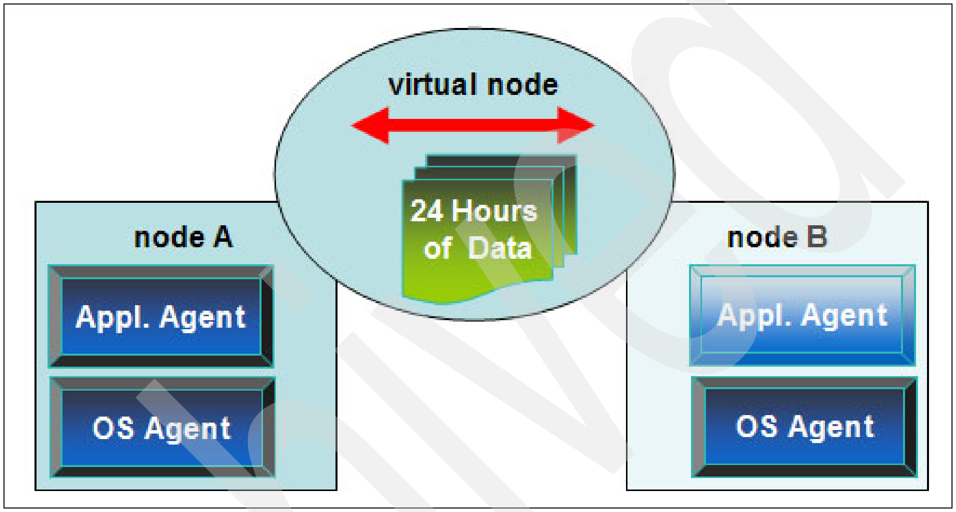


Figure 10-19 Windows cluster with agents on physical nodes

> **Note:** When deciding on the value CTIRA_HOSTNAME, keep in mind that the managed system name is comprised of three parts by using the variables CTIRA_HOSTNAME, CTIRA_SUBSYSTEM_ID, and CTIRA_NODETYPE. It is limited to 31 characters.

For a continuous collection of short-term historical data collected by the application agents from both nodes, you can configure the agents to write this data into the same directory within the shared disk. Use the CTIRA_HIST_DIR variable to set an alternate directory for data collection for both agents or use the monitoring server as the collection location. All configuration settings are included in the agent's configuration files, where ITMinstall_dir is the directory where the product is installed and pc is the product code of the agent:

```
[ITMinstall_dir]\TMAITM6\K[pc]ENV
```

The solution is described in detail and provided on the OPAL Web site, listed in row 2 of Table 10-2.

Table 10-2 Monitoring solutions for cluster, based on the Universal Agent

Operating system	Cluster	Heading on the OPAL Web site
Windows, Linux, UNIX	Compaq	Compaq Hardware Monitoring Solution using the IBM Tivoli Monitoring V6.1 Universal Agent (current link: `http://www-306.ibm.com/software/tivoli/opal/?NavCode=1TW10TM09`)
Windows	Microsoft Cluster Server	Document on Installing IBM Tivoli Monitoring V6.1 Agents in a Microsoft Cluster Server Environment (current link: `http://www-306.ibm.com/software/tivoli/opal/?NavCode=1TW10TM2H`)

10.4.2 Agents within UNIX or Linux clusters

On UNIX or Linux systems, you can have multiple OS agents on one system, so you can install one OS agent and the application agents within the shared file system of the cluster and one OS agent on each node. It makes sense to place the IBM Tivoli Monitoring default directory within the shared file system and install the OS agents on the nodes on different directories, so you can use the default directory for the application agents.

The installation procedure for the agents within the shared resource of the cluster is similar to the Windows OS agents (see 10.4.1, “Agents within a Windows cluster” on page 364). It is important to set the start option to manual for the moved agents and integrate the automatic restart option within the cluster software as usual.

The installation procedure for agents on the nodes is the default installation procedure. If you have installed OS agents in the shared resource with the default directory, you must select an alternative directory within the start procedure. The managed system name of these agents should be the hostname of the virtual cluster node, so that the managed system name is always the same, no matter on which physical node they're running. Figure 10-20 on page 368 shows such an example implementation within a UNIX or Linux cluster.

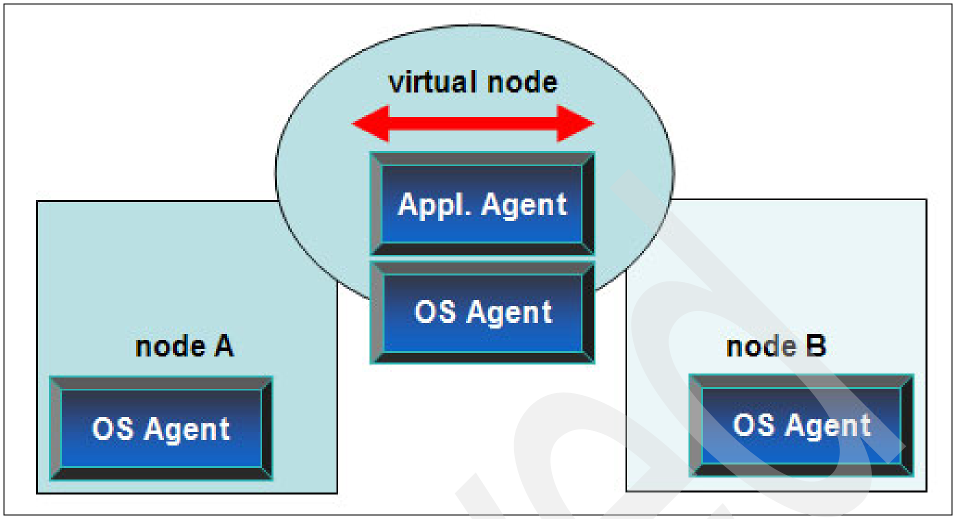


Figure 10-20 UNIX or Linux cluster with agents on virtual and physical nodes

In addition to the heartbeat monitoring, it might be necessary to monitor the cluster with the situations configured for the OS agent as well. In this case, you can implement a solution based on the Universal Agent for monitoring the cluster. You can find such a solution on the OPAL Web site; also refer to Table 10-3.

Table 10-3 Monitoring solutions for cluster based on the Universal Agent

Operating system	Cluster	Heading on the OPAL Web site
Solaris	Sun Cluster, Veritas Cluster	Sun Cluster and Veritas Cluster Monitoring Solution (current link: `http://www-306.ibm.com/software/tivoli/opal/?NavCode=1TW10TM3T`)
AIX	HACMP	IBM High Availability Cluster Multi-Processing (HACMP) for AIX Monitoring Solution using the ITM 6.1 Universal Agent (current link: `http://www-306.ibm.com/software/tivoli/opal/?NavCode=1TW10TM1Z`)

11

Reporting enhancements

In this chapter we first discuss some of the IBM Tivoli Monitoring enhancements in the reporting area. Then we introduce the Tivoli Common Reporting Initiative, which is an initiative to provide a set of reporting services that can be used across the Tivoli portfolio consistently. Finally we talk about the BIRT reporting solution and how this can integrate into the forthcoming Common Reporting Initiative's infrastructure.

The following topics are addressed:

- New reporting features in IBM Tivoli Monitoring V6.2
- Tivoli Common Reporting Initiative
- Case Study 1 - IBM Tivoli Monitoring V6.2 and BIRT integration scenario
- Case Study 2 - IBM Tivoli Monitoring V6.1 and BIRT integration scenario
- BIRT tools
- OPAL solutions

11.1 New reporting features in IBM Tivoli Monitoring V6.2

IBM Tivoli Monitoring V6.2 has planned improvements in the area of reporting. This section describes several of these new functions, as well as the Tivoli initiative for shared reporting.

- Multi-row data plot
- Combined historical and real-time data
- Shared reporting
- Shared reporting ecosystem
- Samples of Tivoli reports

11.1.1 Multi-row data plot

You can now plot multiple processes or other data sets on multiple rows, rather than points on a line. So multiple CPUs will plot as multiple lines rather than multiple dots on a line. This function works for historical data, real-time data, or historical combined with real-time. See examples in Figure 11-1 and Figure 11-2 on page 371.

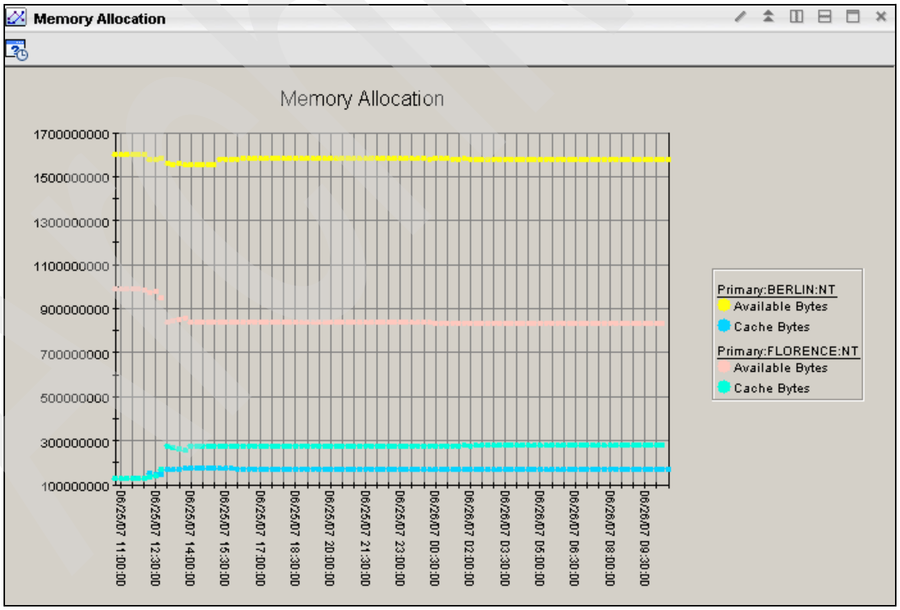


Figure 11-1 Memory allocation multi-row chart

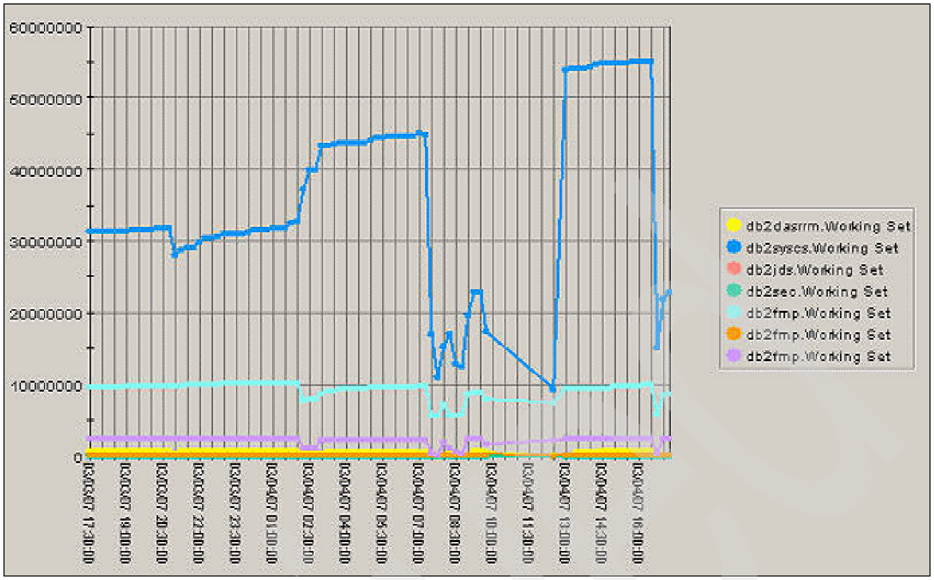


Figure 11-2 Multi-row chart

This new function is configured in the style properties of the Line Chart. Go to **Properties** → **Style** → **Select Plots Points** and choose **Attribute(s) across multiple rows**. See details on Figure 11-3 on page 372.

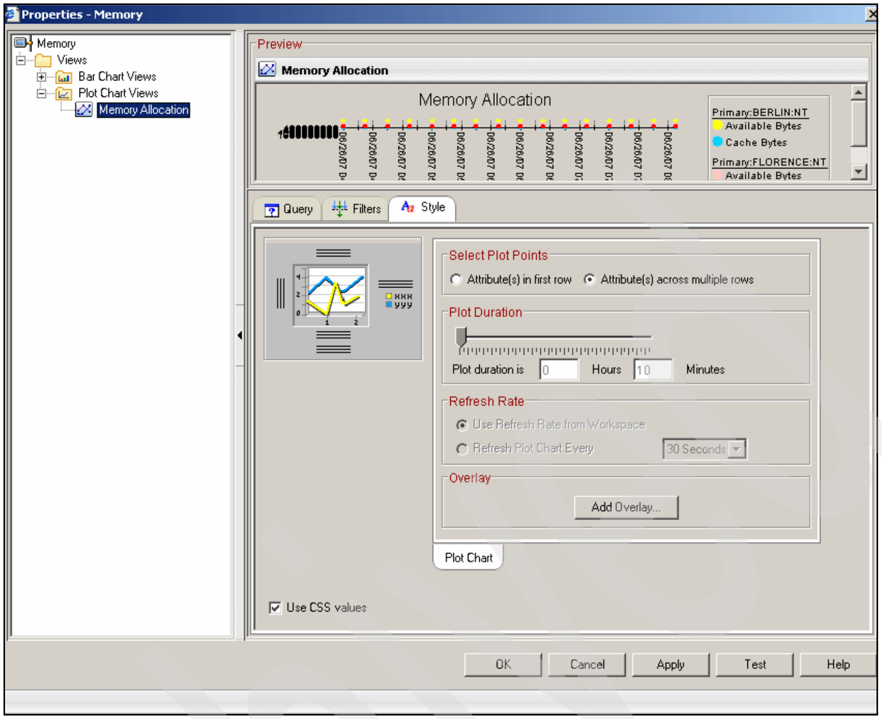


Figure 11-3 Style properties

11.1.2 Combined historical and real-time data

Now you can combine real-time and historical data in the same plot chart with IBM Tivoli Monitoring V6.2, so when a plot chart is initiated, it will not be empty. This function can be used to see trends from past hours or days through the present-to-present contiguous view of the tracked metrics. A plot chart can be updated every 10 seconds. The red line indicates the current time, as shown in Figure 11-4 on page 373 and Figure 11-5 on page 374.

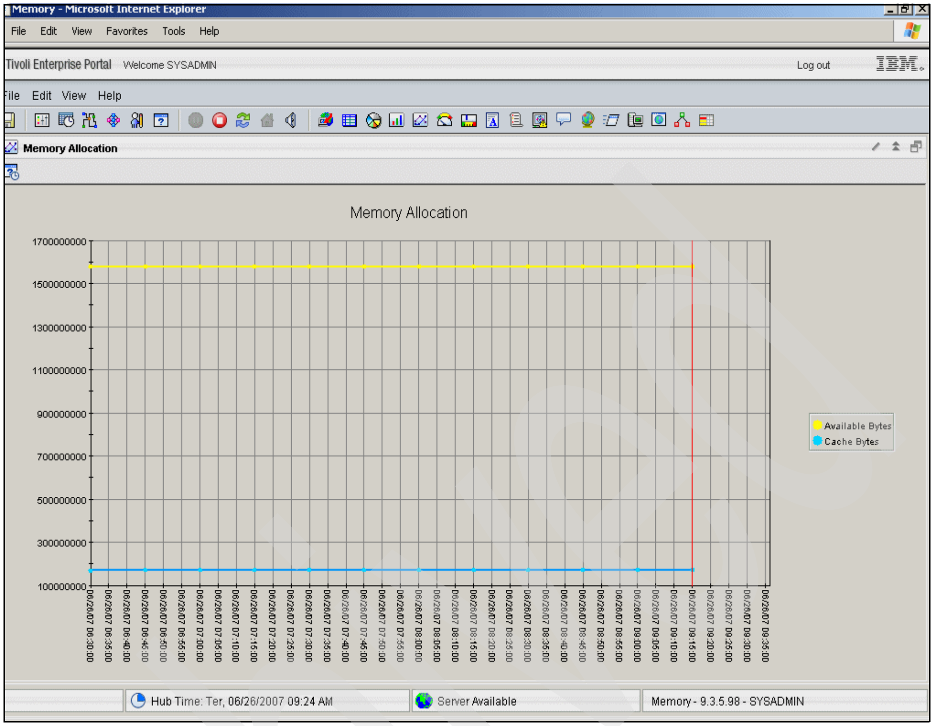


Figure 11-4 Plot chart with historical time and real time

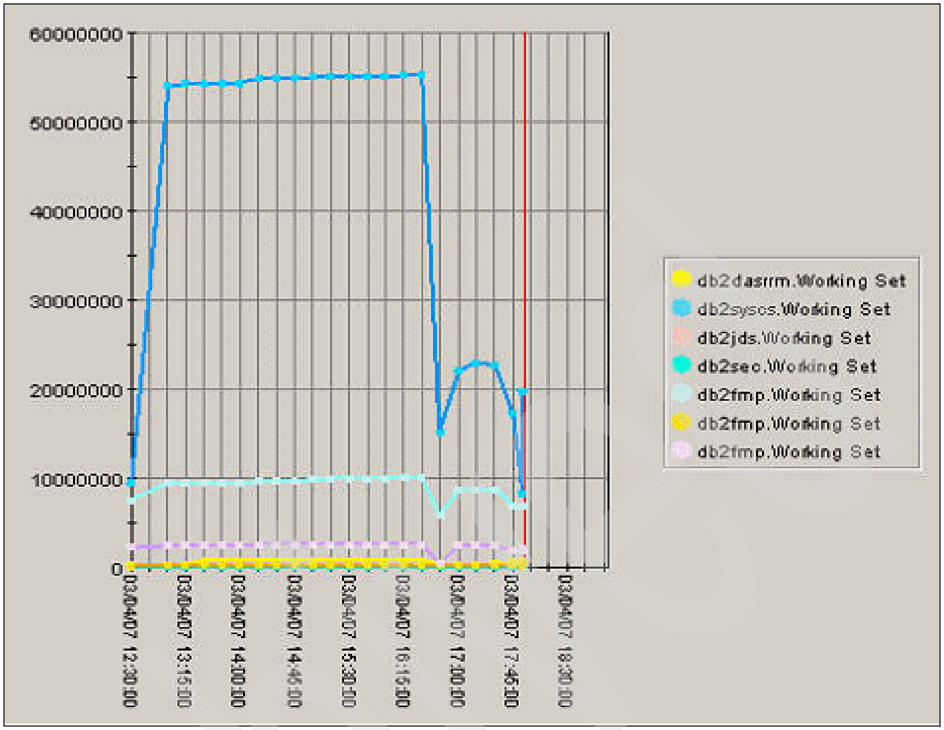


Figure 11-5 Plot chart with historical and real time

This function is accessed though the time span button. Click **Time span** in the plot chart and select **Real time plus Last.** Specify the amount of past hours you want to display. View Figure 11-6 on page 375.

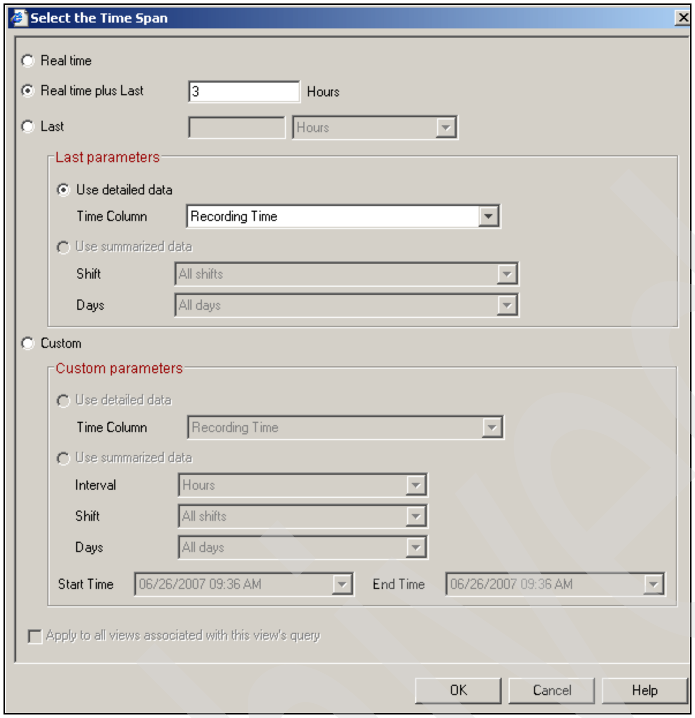


Figure 11-6 Time span

11.2 Tivoli Common Reporting Initiative

Tivoli Common Reporting is a new initiative from IBM designed to provide a set of reporting services that can be used across the Tivoli portfolio consistently. This initiative will provide three services:

- A flexible infrastructure made up of loosely coupled services that can be used by individual products or in combined solutions for cross-product reporting
- A set of out-of-the-box reports that represent known best practices for the products or solutions they are delivered with, as well as a mechanism (OPAL) to share reports as experience of community grows
- Tooling that helps the customer or service provider modify or extend the reports to meet their needs instead of having to submit product requirements against development to make the changes

11.2.1 Shared reporting ecosystem

The reporting ecosystem that will be provided to support this initiative is illustrated in the diagram shown in Figure 11-7.

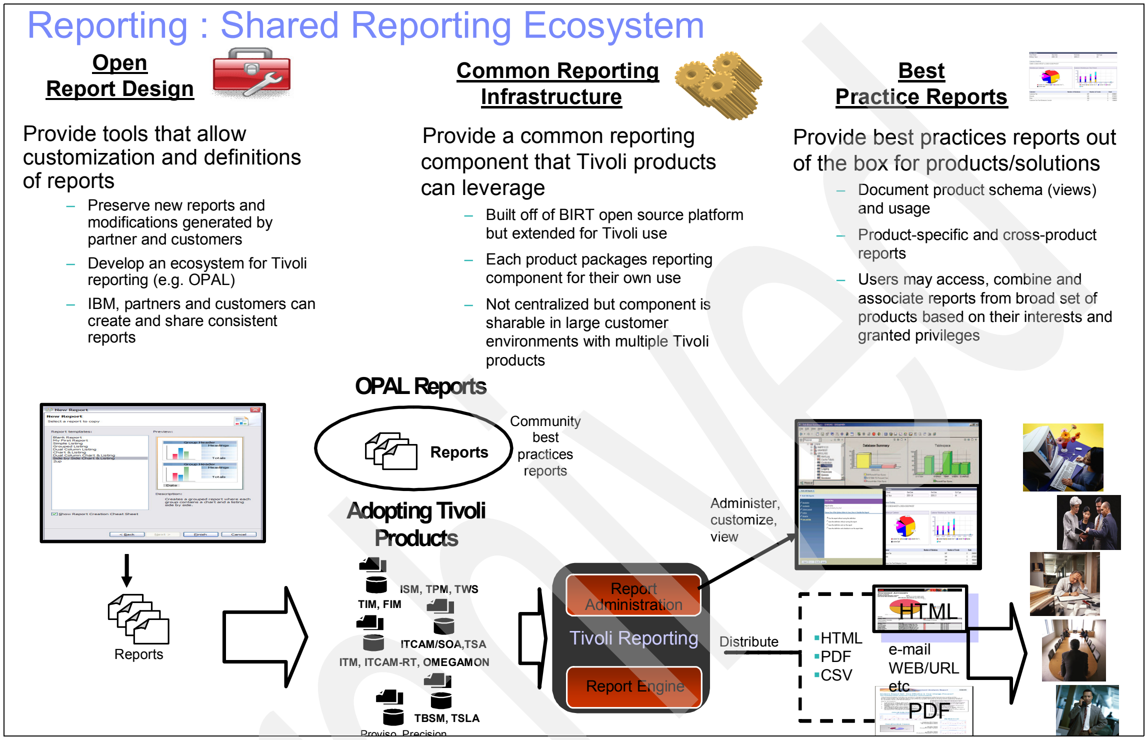


Figure 11-7 Reporting ecosystem

11.2.2 Where Common Reporting fits

The overview illustrating how Tivoli Common Reporting fits into the infrastructure is shown in Figure 11-8 on page 377. Note that BIRT, with the Tivoli extensions, is an important value add function to the Common Reporting Infrastructure, part of the Tivoli Common Reporting Initiative.

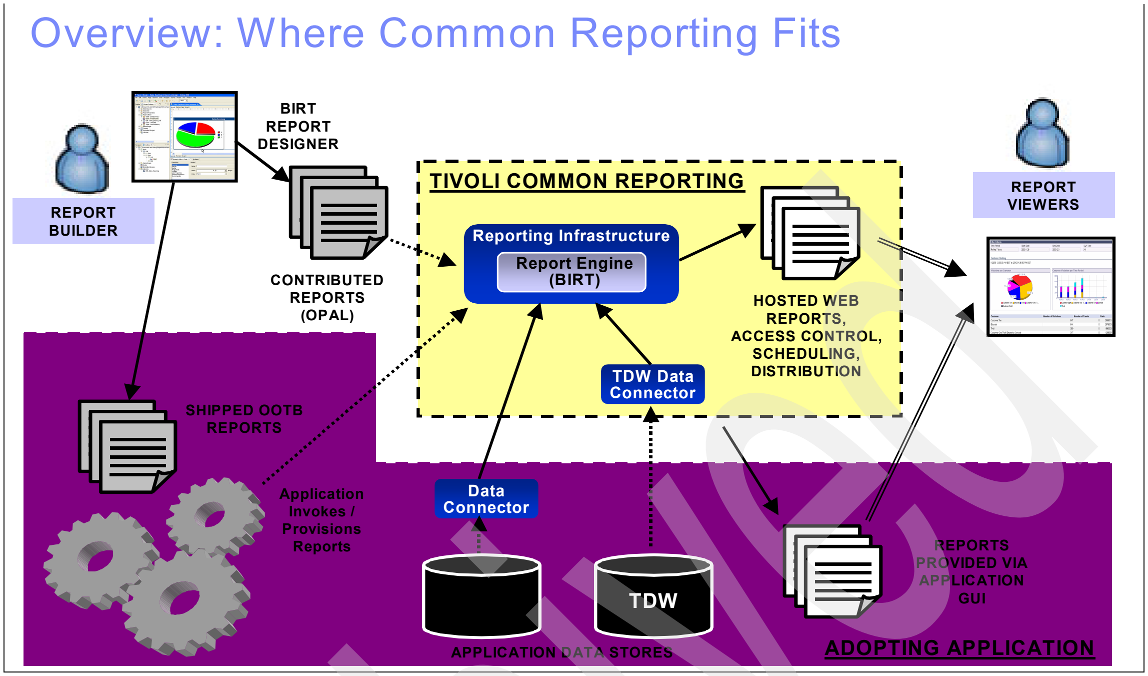


Figure 11-8 Overview: Where the common reporting fits

11.2.3 Samples of Tivoli reports

The following are examples the sample reports that are planned to be available on OPAL. These are referred as Contributed Reports in Figure 11-8.

Important: This is not a finalized list and is subject to change. It is provided here to give you an idea of the types of reports that you should expect to see on the OPAL Web site in the near future. The final list will be determined based on client feedback.

- Top UNIX Hosts by CPU Usage
- Top Windows Hosts by CPU Usage
- Top Windows Hosts by Disk Usage
- Top Windows Processes by Resource Usage
- Windows System Overview
- Top 5 Worst Performing Clients
- Top 5 Worst Performing Applications

- Application Uptime and Downtime
- Application Performance Over Time
- Response Time & Availability for Selected Transactions
- Overall Transactions Over Time
- Line Charts for Sub-transaction Response Time
- Top Requests & Methods Used
- Top Slowest Requests & Methods
- Top CPU Intensive Requests and Methods
- Top SQL Intensive Requests and Methods
- Top Tables Used
- Top SQL Used

11.2.4 Business Intelligence and Reporting Tools

Business Intelligence and Reporting Tools (BIRT) is an Eclipse-based open source reporting system for Web applications, especially those based on Java and J2EE™. BIRT has two main components: a report designer based on Eclipse, and a runtime component that you can add to you application server. BIRT also offers a charting engine that lets you add charts to your own application.

BIRT reports consist of four main parts: data, data transforms, business logic and presentation. These are explained in further detail here.

- Data

  Databases, Web services, Java objects all can supply data to your BIRT report. BIRT provides JDBC, XML, Web Services, and Flat File support, as well as support for using code to get at other sources of data. BIRT's use of the Open Data Access (ODA) framework allows anyone to build new UI and runtime support for any kind of tabular data. Furthermore, a single report can include data from any number of data sources. BIRT also supplies a feature that allows disparate data sources to be combined using inner and outer joins.

- Data Transforms

  Reports present data sorted, summarized, filtered and grouped to fit the user's needs. Although databases can do some of this work, BIRT must do it for “simple” data sources such as flat files or Java objects. BIRT allows sophisticated operations such as grouping on sums, percentages of overall totals and more.

- Business Logic

  Real-world data is seldom structured exactly as you would like for a report. Many reports require business-specific logic to convert raw data into information useful for the user. If the logic is just for the report, you can script it using BIRT's JavaScript™ support. If your application already contains the logic, you can call into your existing Java code.

- Presentation

  After the data is ready, you have a wide range of options for presenting it to the user, such as tables, charts, text and more. A single data set can appear in multiple ways, and a single report can present data from multiple data sets.

Figure 11-9 provides an overview of the BIRT Project and the various components involved.

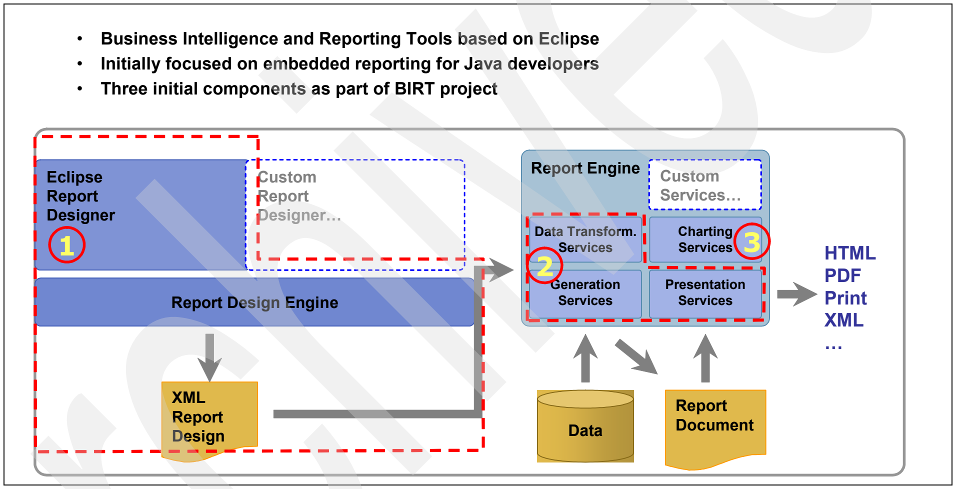


Figure 11-9 BIRT Project

11.2.5 BIRT terminology

In this section we explain basic BIRT terminology:

- Data Explorer

  Data Explorer organizes your data sources (connections) and data sets (queries). The data set editor allows you to test your data set to ensure the report receives the correct data. This view also is used to design report parameters.

- Layout View

  The Layout View is a WYSIWYG editor that provides drag & drop creation of the presentation portion of your report.

- Palette

  The Palette contains the standard BIRT report elements such as labels, tables, and charts. It is used in conjunction with the Layout View to design reports.

- Property Editor

  The Property Editor presents the most commonly used properties in a convenient format that makes editing quick and easy. BIRT also integrates with the standard Eclipse property view to provide a detailed listing of all properties for an item.

- Report Preview

  Using Report Preview, you can test your report at any time with real data. The preview is a window directly within Eclipse.

- Script Editor

  Scripting adds business logic to reports during data access, during report generation, or during viewing. The code editor provides standard Eclipse features for editing your scripts: syntax coloring, auto-complete, and more.

- Outline

  BIRT reports are organized as a tree structure with the overall report as the root, and separate categories for styles, report content, data sources, data sets, report parameters and more. The Outline view provides a compact overview of your entire report structure.

- Cheat Sheets

  Learning a new tool is always a challenge, but Eclipse offers an innovative solution: Cheat Sheets. These are brief bits of documentation that walk you through new tasks.

- Library Explorer

  BIRT allows the reuse of report objects, such as tables, data sources and styles. Objects created for reuse are stored in a library file. BIRT supplies the Library Explorer view to allow you to browse the contents of report libraries.

- Chart Builder

  Adding Charts to BIRT designs is expedited with the Chart Builder. Chart creation is separated into three phases: Select Chart Type, Select Data, and Format Chart.

- Expression® Builder

  BIRT expressions are basically simple scripts that return a value. Expressions are used for assigning data values to report elements, building image locations, hyperlinks, parameter default values and many other places. Expressions are constructed within BIRT using the Expression Builder.

For more information, refer to the following Web site:

http://www.eclipse.org/birt/phoenix/

11.2.6 How BIRT interacts with IBM Tivoli Monitoring

Reporting can be implemented to display historical information from the data warehouse that collects historical information from the IBM Tivoli Monitoring server, directly from the Operation Management Product (OMP) management database or from any other OMP data source to which the OMP provides a data connector. The first two cases are illustrated Figure 11-10.

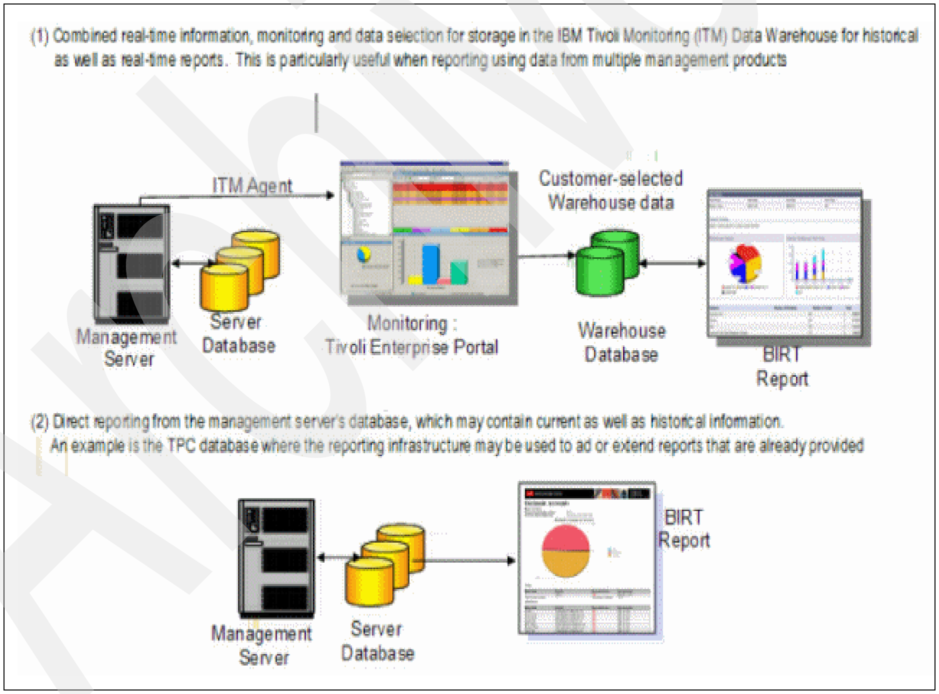


Figure 11-10 IBM Tivoli Monitoring interaction with BIRT

> **Note:** Operational Management Product refers to any Tivoli product that can potentially use common reporting components.

As mentioned in 11.2, “Tivoli Common Reporting Initiative” on page 375, you can use BIRT to develop reports that integrates easily into the forthcoming Common Reporting Initiative's infrastructure, both product-specific as well as more thoroughly integrated cross-product best practice reports.

In the following sections we guide you through the steps to implementation-based reports in your IBM Tivoli Monitoring V6.2 and IBM Tivoli Monitoring V6.1 environments.

11.3 Case Study 1 - IBM Tivoli Monitoring V6.2 and BIRT integration scenario

We have implemented an IBM Tivoli Monitoring V6.2 environment to show you some of the new reporting features available with IBM Tivoli Monitoring V6.2. These new features are discussed in 11.1, “New reporting features in IBM Tivoli Monitoring V6.2” on page 370. Now we will use the same environment to create reports with BIRT.

> **Note:** Because we are using IBM Tivoli Monitoring V6.2 agents, some of the attributes that you will see in the reporting examples in this chapter might be different or not found if you use the IBM Tivoli Monitoring V6.1 version of the monitoring agents.

11.3.1 Our environment

Our IBM Tivoli Monitoring V6.2 environment is illustrated in Figure 11-11 on page 383. We installed the following components:

- Hub monitoring server: Windows 2003
- Portal Server: Windows 2003
- Remote monitoring server: Linux (Red Hat 4.0)
- Remote monitoring server: Windows 2003
- Warehouse Proxy and Summarization and Prunning: Windows 2003

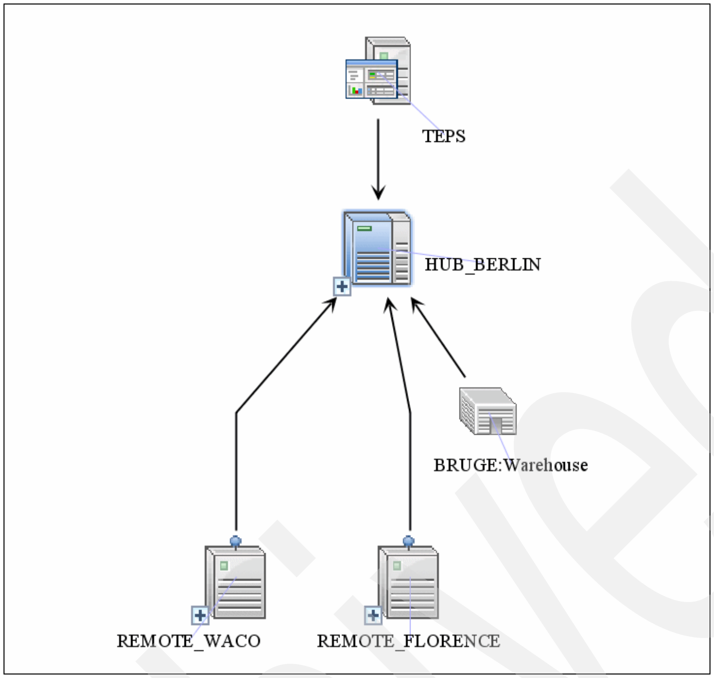


Figure 11-11 IBM Tivoli Monitoring V6.2 environment for reporting

Before you begin to create reports, ensue sure that:

- IBM Tivoli Monitoring V6.x is installed and operational.
- Data Warehouse database is created and populated.
- Monitoring agents are active and collecting data.

11.3.2 Installation

For implementing the reporting solution the following products are installed:

Java

We used the Java Runtime Environment 1.4.2_08.

BIRT

We used the following BIRT files: birt-report-designer-all-in-one-2.1.2.zip and itext-1.3.jar. They can be downloaded from the following Web site:

http://www.eclipse.org/birt/phoenix/

Eclipse installation is simple. Simply unzip the file and make sure **eclipse.exe** is working.

JDBC

We needed the following DB2 JDBC plug-ins in order to connect to the warehouse database: db2jcc.jar and db2jcc_license_cu.jar.

Those files can be downloaded from the following Web site:

http://www14.software.ibm.com/webapp/download/preconfig.jsp?id=2004-09-20+10%3A09%3A21.003415R&cat=database&fam=&s=c&S_TACT=104CBW71&S_CMP=

Alternatively, you can copy those from IBM Tivoli Monitoring server. The files are located in the default installation directory: x:\Program Files\IBM\SQLLIB\java.

11.3.3 Configuration

For configuration, you need to consider the data source and the itext file.

Data source

Copy the files db2jcc.jar and db2jcc_license_cu.jar to the directory: x:\eclipse\plugins\org.eclipse.birt.report.data.oda.jdbc_2.1.1.v20070205-1728\drvers

Itext file

The itext-1.3.jar file needs to be placed in the directory: x:\eclipse\plugins\com.lowagie.itext_1.3.0.v20070205-1728.

11.3.4 Report creation

To help you familiarize with the report creation process, we work on three different types of reports with the historical data collected from Windows, Linux, and DB2 agents. The reports created are:

- Windows Memory Usage
- DB2 Buffer Pool I/O Trends
- Linux Memory Usage

Common steps

First we need to create a new project.

1. Go to **File -> New -> Project**, choose **BIRT -> Report Project -> Next**.

2. Type the Project Name and click **Finish.**
3. A window will prompt asking to switch to a report design perspective, so click **Yes.**
4. For the report, click with the right button at your **Project -> New -> Report** as shown in Figure 11-12.

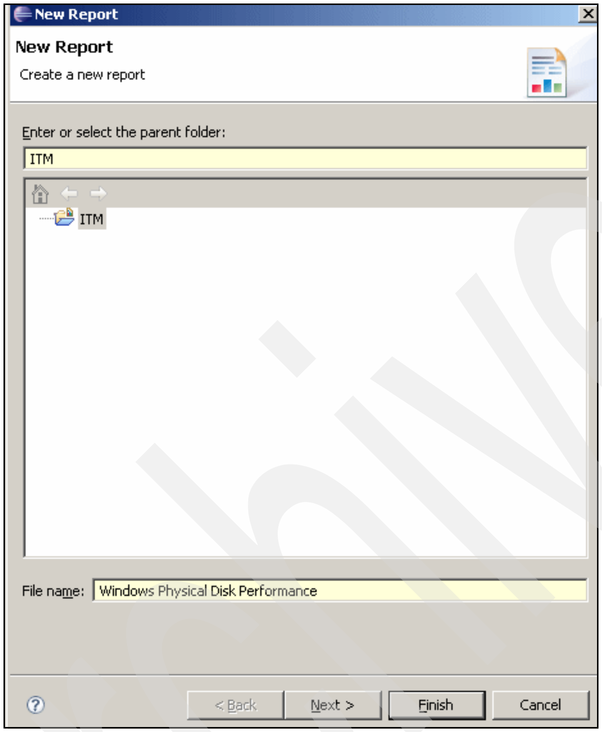


Figure 11-12 New report

This is an important step. We need specify the location of the warehouse database and to do that we first need to create a data source.

5. Go to data explorer right click on **Data Source -> New Data Source**.
6. Select **Create from a data source type in the following list ->** and choose **JDBC Data Source**, as shown Figure 11-13 on page 386.

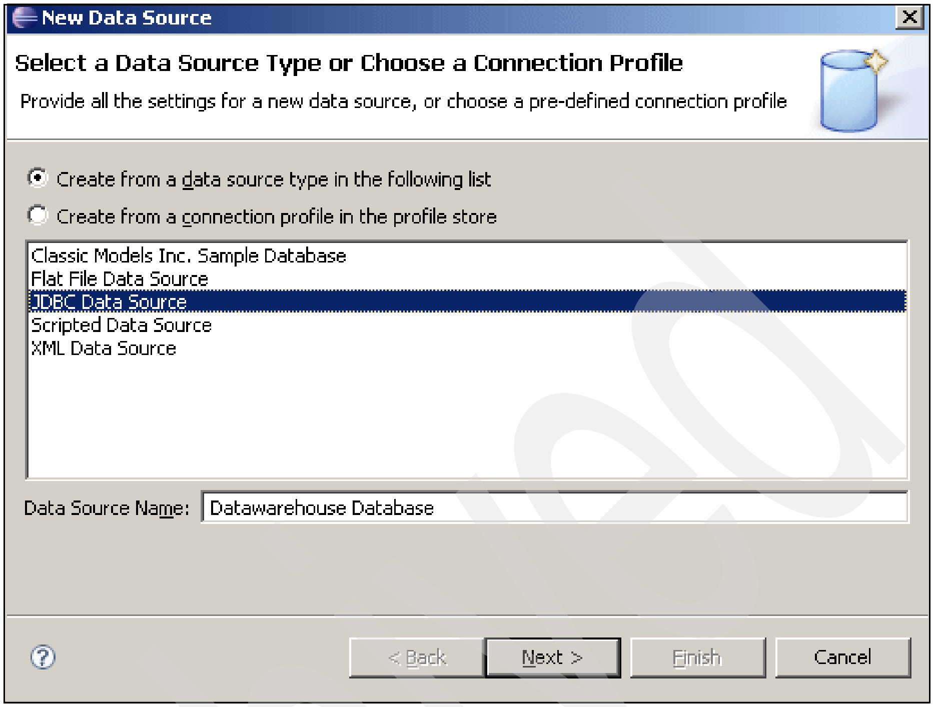


Figure 11-13 New data source

The next screen, displayed in Figure 11-14 on page 387, shows the connection information to the warehouse database.

- Drive class: com.ibm.db2.jcc.DB2Driver (v2.7)
- Database URL: jdbc:db2://9.3.5.99:50000/warehous (where Port and IP reflects to your environment.
- User name: ITMUSER
- password: ITMUSER password
- JNDI URL: can be left empty.

7. Test the connection. If it is successful, click **Finish**.

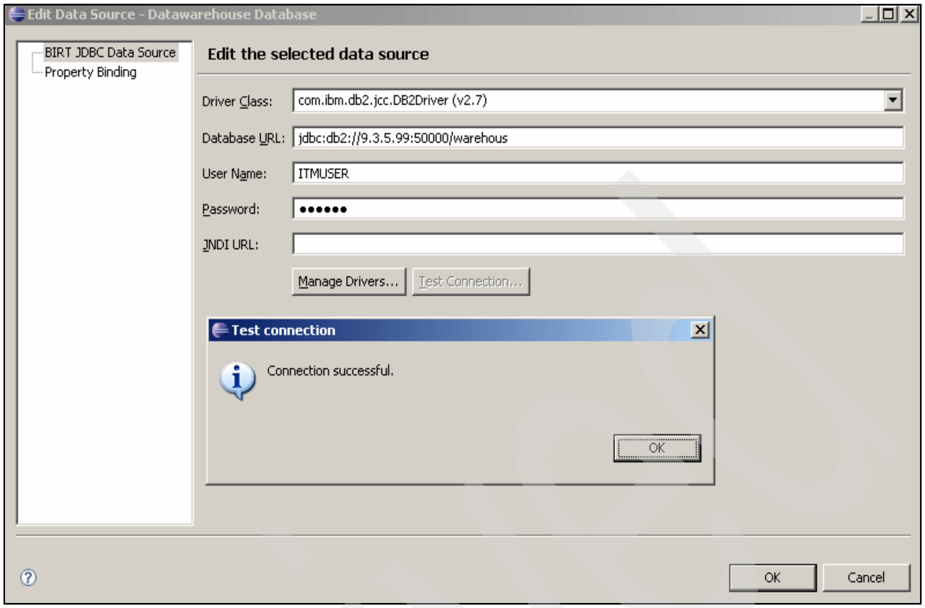


Figure 11-14 Data source information

With the common part finished, we now work on each report separately.

Windows Memory Usage

We need to create a data set, so that we display data from the data source in a report. You can create as many data sets as you want for the same data source.

1. In the Data Explorer, right-click **Data Sets** -> **New Data Set** -> **Data Set** and provide the following information:
   - Name: Windows Memory Usage
   - Data Source: Data source name that you created to connect to data warehouse
   - Data Set Type: SQL Select Query

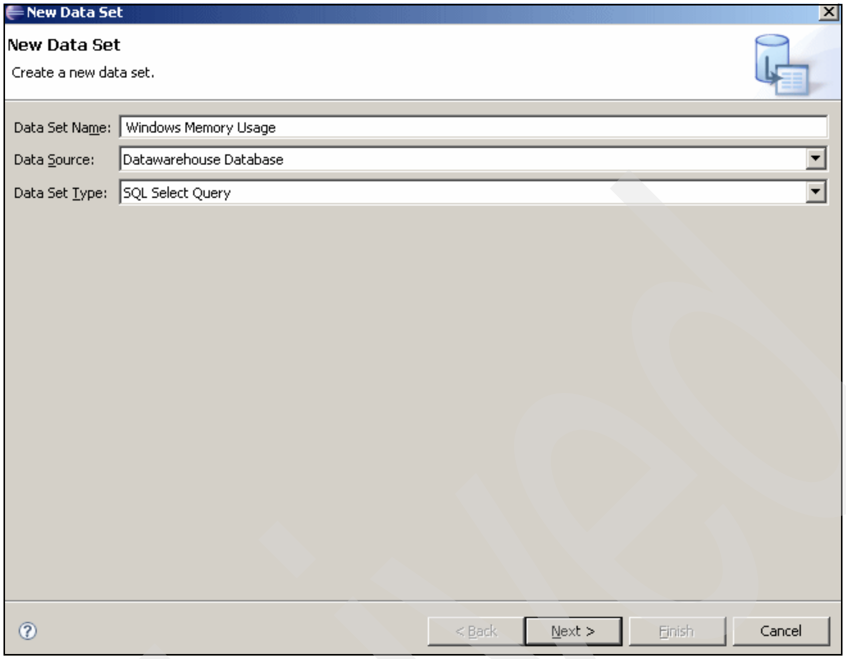


Figure 11-15 New data set

2. When you reach the screen shown in Figure 11-15, click **Next.**
3. The following screen is the Query Builder (shown in Figure 11-16 on page 389), where we define the sql query. The sql query for this report will use the ITMUSER tables that are displayed on "Available Items". On the right side of the window, type the sql statement.

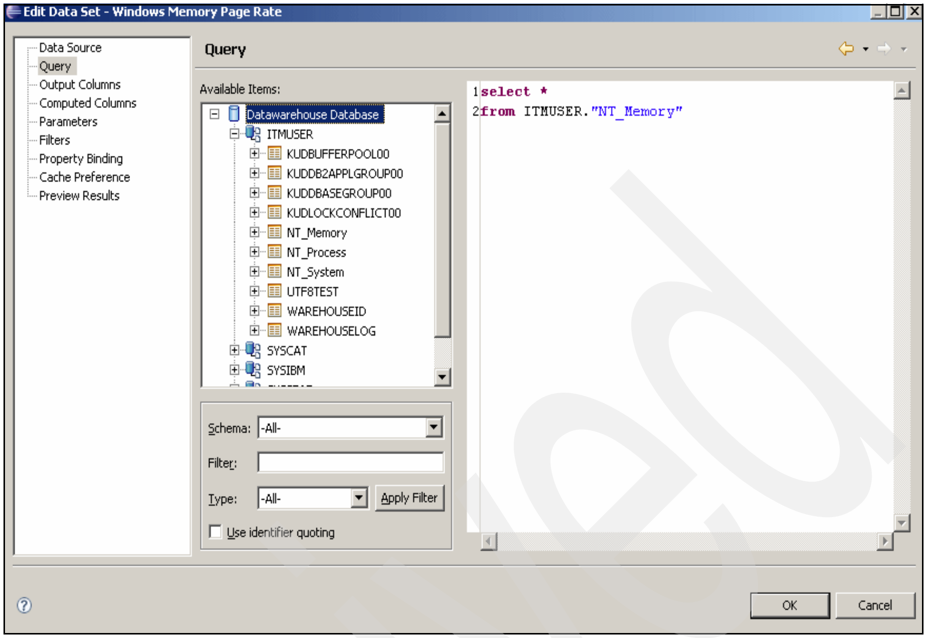


Figure 11-16 Query Editor

> **Important:** Keep in mind that all Tivoli Data Warehouse V2.1 table names are created with quotation marks for the table name. As an example, if yearly, quarterly, monthly, weekly, daily, and hourly summarization are turned on for the NT_Memory attribute group, then the following tables are created in the Tivoli Data Warehouse:
>
> | | |
> |---|---|
> | **"NT_Memory"** | The detailed historical table for NT_Memory |
> | **"NT_Memory_H"** | The summarized hourly historical table for NT_Memory |
> | **"NT_Memory_D"** | The summarized daily historical table for NT_Memory |
> | **"NT_Memory_W"** | The summarized weekly historical table for NT_Memory |
> | **"NT_Memory_M"** | The summarized monthly historical table for NT_Memory |
> | **"NT_Memory_Q"** | The summarized quarterly historical table for NT_Memory |
> | **"NT_Memory_Y"** | The summarized yearly historical table for NT_Memory |
>
> When referencing historical data in the Tivoli Data Warehouse database, you *must* use quotation marks to ensure correct access to that data, as shown in Figure 11-16 on page 389.

4. For this report, we worked with the following attributes:

- Cache Usage Percentage
- Available Usage Percentage
- Memory Usage Percentage
- Server Name
- Timestamp

For the complete sql statement, see Figure 11-17 on page 391.

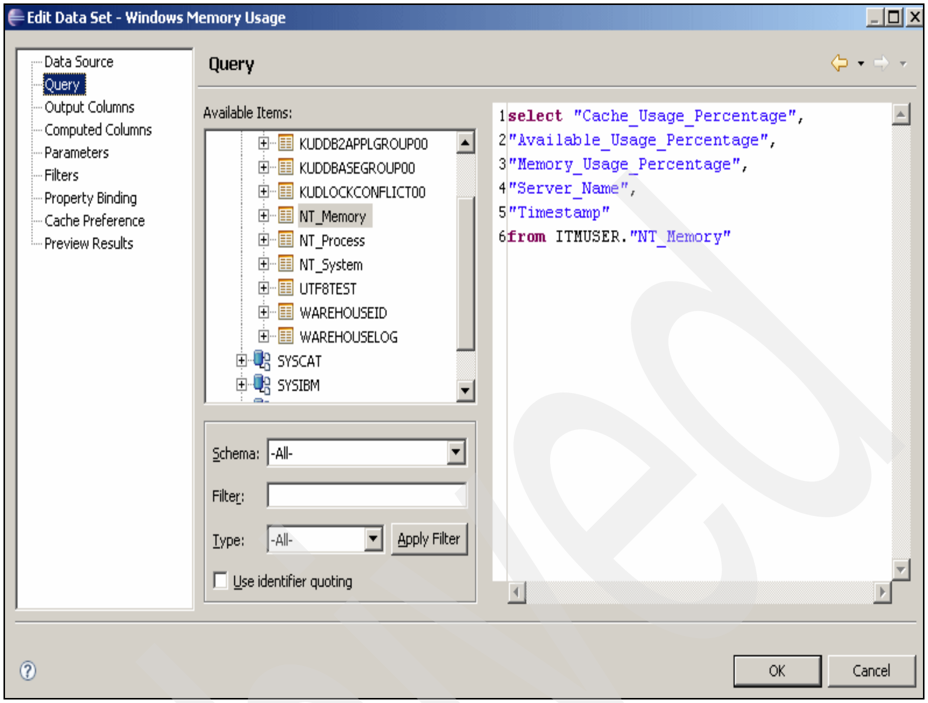


Figure 11-17 Complete query

5. The result of this query can be seen in the Preview Results screen; see Figure 11-18 on page 392.

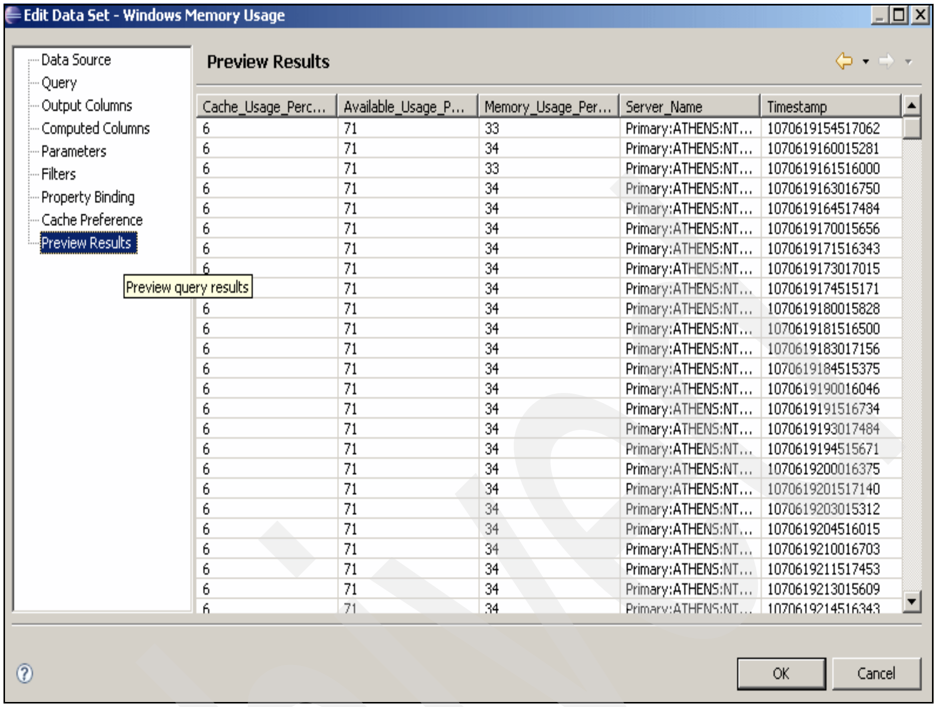


Figure 11-18 Preview Results

In order to have a specific report, you need to filter the data. In our case, the Server Name attribute is filtered to display only the server Athens. This is called "filtering at the source."

6. From the Data Set window, select **Filters**.
7. Define Filters by selecting **New** and entering the expression; see the example in Figure 11-19 on page 393.

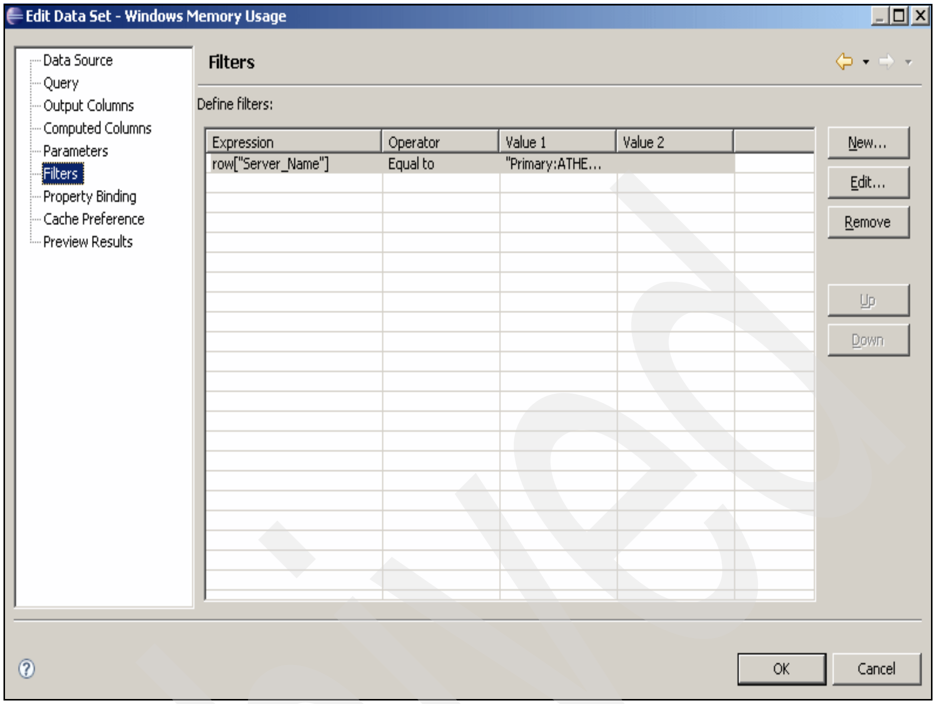


Figure 11-19 Filter

The last step is to define the report layout. You can drag and drop the dataset Windows Memory Usage to the layout panel. You will see a table report, as shown in Figure 11-20 on page 394.

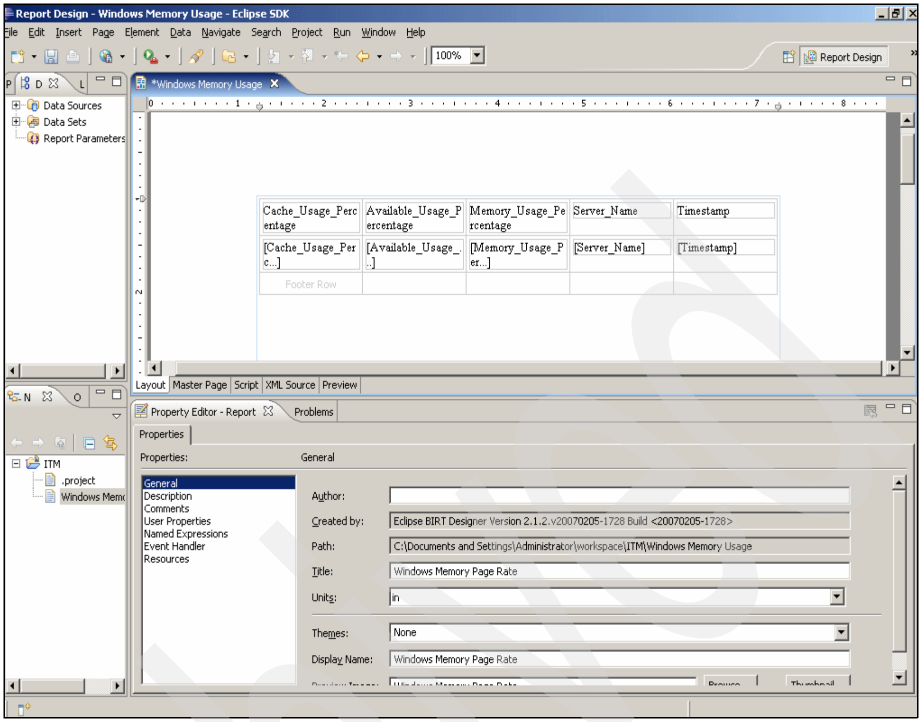


Figure 11-20 Table layout

The BIRT report interface tool allows you to create different reports with several types of charts, so you have to decide what is the best way to display the information.

8. For the Memory Usage report, we used the Bar Chart. To insert this chart, right-click **Layout -> Insert -> Chart -> Bar Chart -> Next**.
9. In the Select Data, check the option: **Use Data Set** and choose the data set that we have created.
10. The attribute Timestamp that is collected by IBM Tivoli Monitoring V6.x is not amenable for reporting. It needs to be reformatted for better readability.

Important: The IBM Tivoli Monitoring 16-digit timestamp is in the following format: CYYMMDDHHMMSSNNN, where:

- C=1 means the 2000 century, C=0 means 1900 century
- YYMMD year/month/day
- HHMMSS hour/minute/second
- NNN is sequence information, used to order multiple events that are recognized at the same time. 000 means one event. If there were three events recognized at once, you would see 001/002/999.

You need to convert this to a more readable format in the reports. You have three choices, as explained here:

- The preferred way is to use the built-in DateForm function. For example, if you are using DB2, you can locate this function in the DB2 Control Center GUI on the WAREHOUS database. To see the actual create statement (which shows what the function is doing) you can run the following query against the syscat.functions table:

```
“select body from syscat.functions where funcname='DATEFORM'"
```

You can use the DateForm function in your SQL query. For example:

select DateForm("Timestamp",'H'), "ServerName", "Available_Bytes" from ITMUSER."NT_Memory"

You can replace 'H' with 'D' to achieve a daily format.

- You can create your own RDBMS. See 11.4.2, “Report creation” on page 402 for an example of this via a DB2 function. Similar functions can be written in Oracle and MS-SQL, as well.
- If you do not use one of the previous two solutions, then you have to use the Function button (fx) in the panel Select Data and input the following expression:

```
var year = 20+row["Timestamp"].substring(1,3);
var month = row["Timestamp"].substring(3,5);
var day = row["Timestamp"].substring(5,7);
var hour = row["Timestamp"].substring(7,9);
var minute = row["Timestamp"].substring(9,11);
new Date (year, month-1, day, hour, minute);
```

11. Enter the attributes for Value (y) Series. Because we are working with three attributes in this report, there will be three series (see Figure 11-21 on page 396).

```
Series 1: row["Cache_Usage_Percentage"]
Series 2: row["Available_Usage_Percentage"]
Series 3: row["Memory_Usage_Percentage"]
```

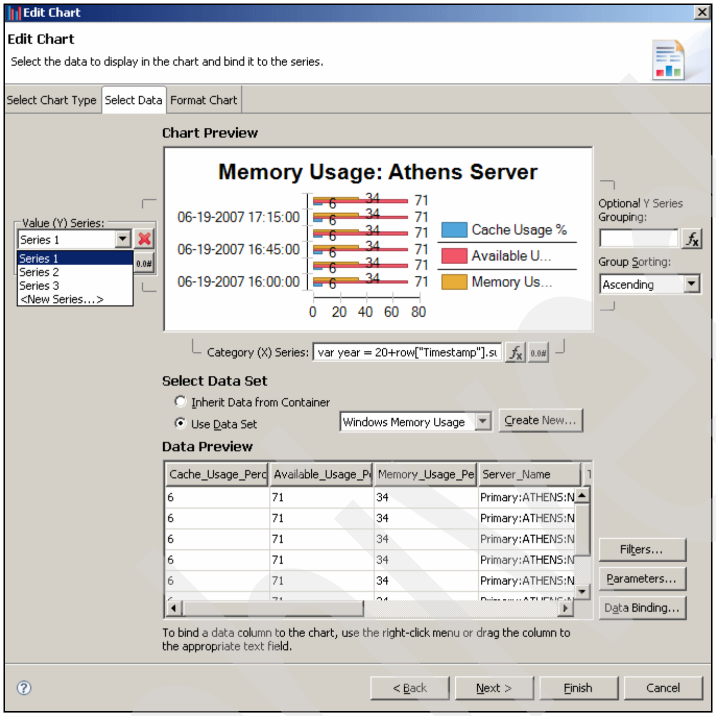


Figure 11-21 Select data

During the design of the report, we decide to use Filter again in order to have a better display of the data.

12. Click **Next**; the following panel displayed is Format Chart, where you define the style of your report such as colors, legend, and what will be visible or not visible.

13. Figure 11-22 on page 397 shows the preview of a Bar Chart evaluating Memory Usage from the Athens server.

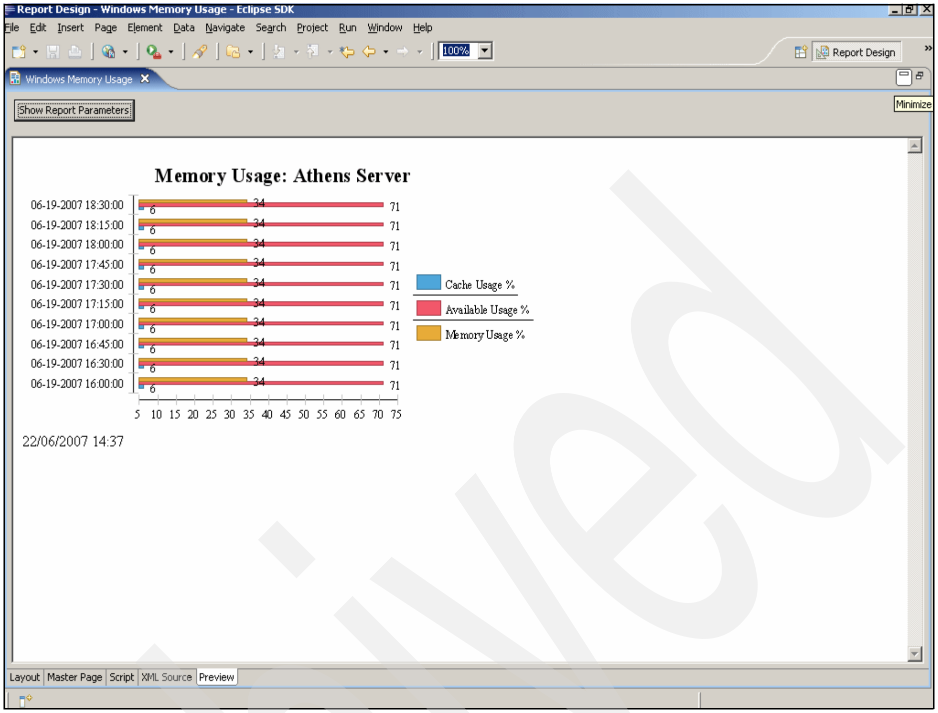


Figure 11-22 Preview report

BIRT allows you to see this report in three different types: Web Viewer, HTML, and PDF.

14. Figure 11-23 on page 398 shows you the report as a Web page.

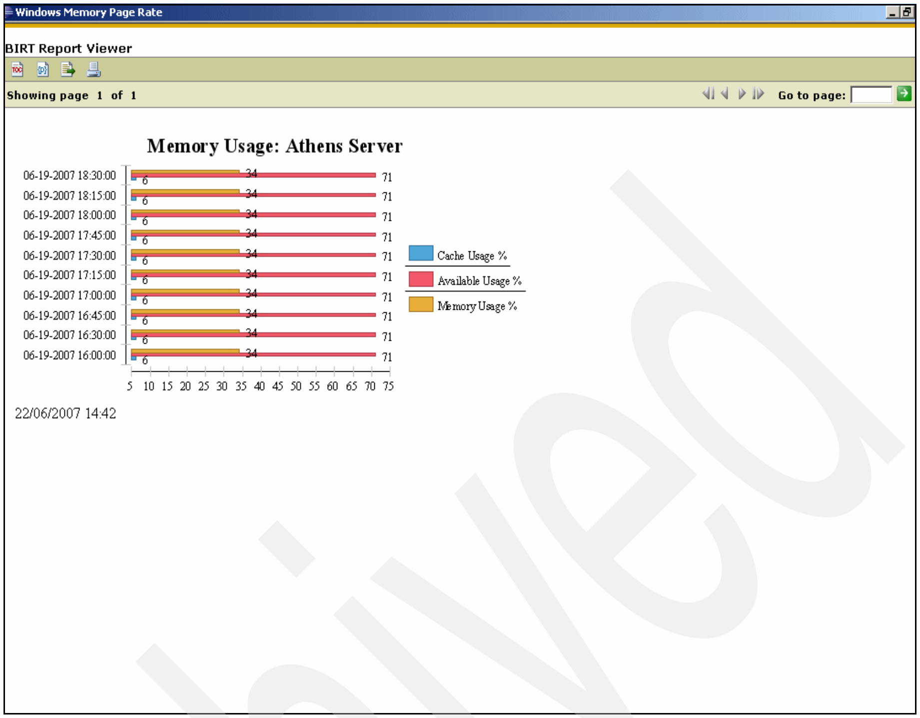


Figure 11-23 Web page report

DB2 Buffer Pool I/O Trends

Now we explain how to create a DB2 Buffer Pool I/O Trends report. This time we use the following attributes:

- pool read time
- pool write time
- db name
- writetime

We also want to filter the writetime attribute to display data for a smaller period of time. Figure 11-24 on page 399 illustrates this task.

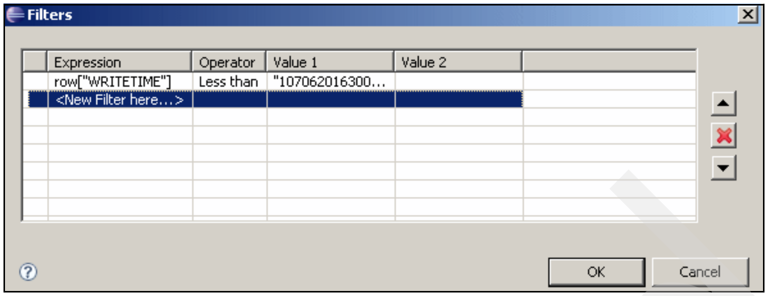
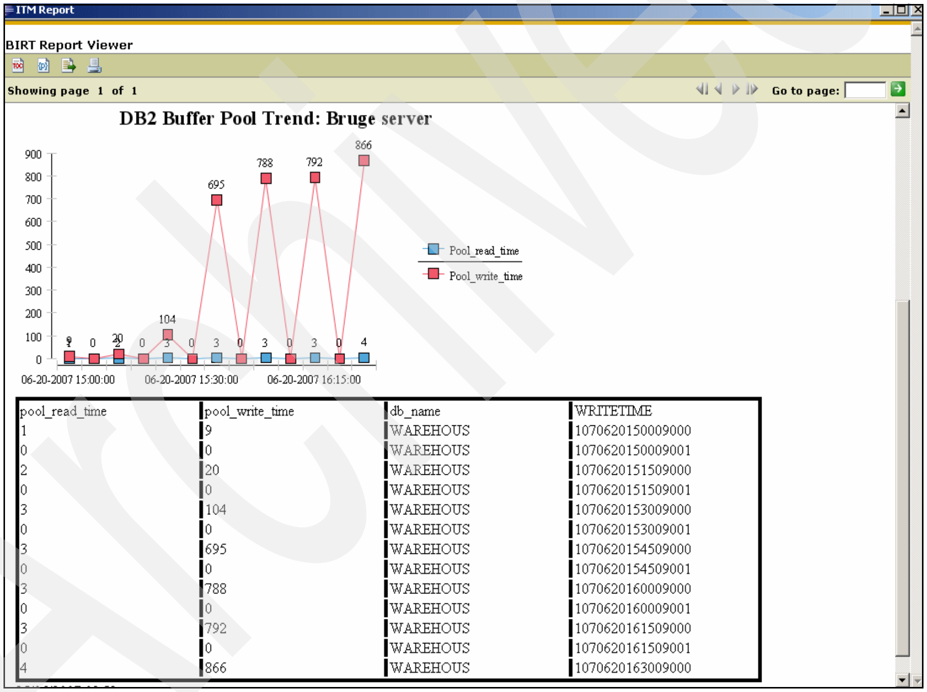


Figure 11-24 DB2 filter

A table is included in addition to the layout. The final version of the design report is displayed in Figure 11-25.



pool_read_time	pool_write_time	db_name	WRITETIME
1	9	WAREHOUS	1070620150009000
0	0	WAREHOUS	1070620150009001
2	20	WAREHOUS	1070620151509000
0	0	WAREHOUS	1070620151509001
3	104	WAREHOUS	1070620153009000
0	0	WAREHOUS	1070620153009001
3	695	WAREHOUS	1070620154509000
0	0	WAREHOUS	1070620154509001
3	788	WAREHOUS	1070620160009000
0	0	WAREHOUS	1070620160009001
3	792	WAREHOUS	1070620161509000
0	0	WAREHOUS	1070620161509001
4	866	WAREHOUS	1070620163009000

Figure 11-25 DB2 Web Viewer

Linux Disk Space

Finally, we will create the Linux Disk Space report.

1. As usual, we first create a new dataset, called Linux Disk Space.
2. For this report we used the Attributes Space Used, Space Available, Disk Name and Writetime from Linux_Disk table. The same filter, conversion and

design process was used. The final result it can be seen in Figure 11-26 and Figure 11-27 on page 401.

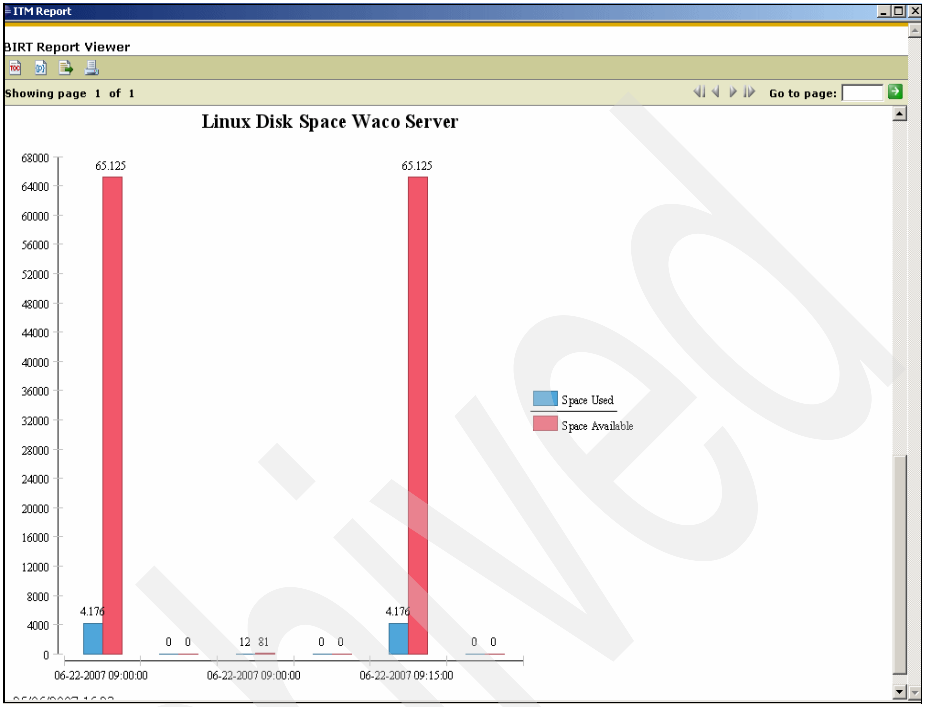


Figure 11-26 Linux report

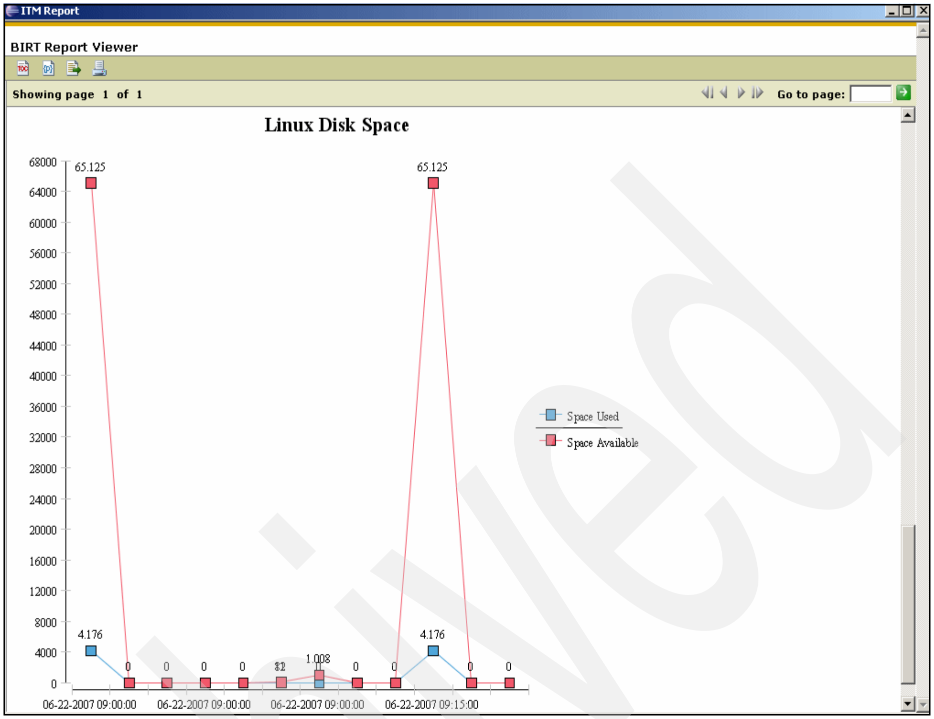


Figure 11-27 Linux Plot Chart

11.4 Case Study 2 - IBM Tivoli Monitoring V6.1 and BIRT integration scenario

In this case study we show how integrate BIRT with IBM Tivoli Monitoring V6.1. As part of the scenario, we illustrate how to use DB2 functions to simplify reporting, how to use parameters in report creation, and how to publish the reports created on a Web site.

11.4.1 Our environment

Our test environment consists of three machines, as shown in Figure 11-28.

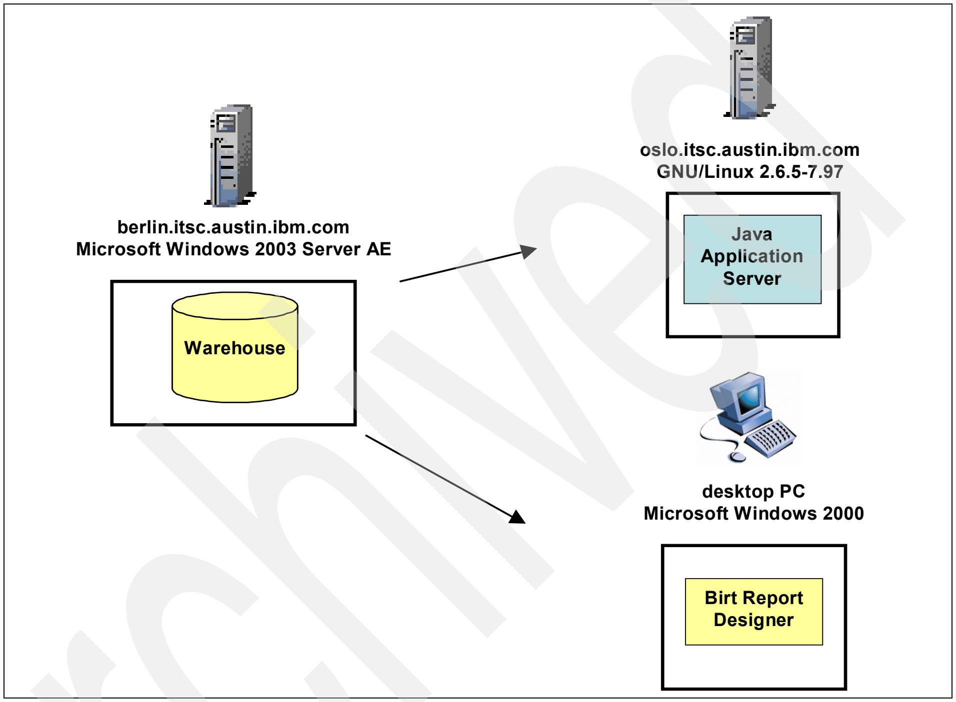


Figure 11-28 Our test environment

The warehouse database resides on an IBM DB2 Universal Database™ 8.1.2, and the chosen Application Server is an Apache Tomcat 5.5.20 with Java Developer Kit 1.4. See the Apache Tomcat Web site for specific installation instructions:

`http://tomcat.apache.org/index.html`

11.4.2 Report creation

In our case, we create the “CPU Usage per Host” report. First we develop a specific DB2 function to translate the format of the Timestamp field of IBM Tivoli Monitoring tables to a more human-readable format. This is shown in Example 11-1 on page 403.

Note: You can also use the DateForm function that is discussed in 11.3, "Case Study 1 - IBM Tivoli Monitoring V6.2 and BIRT integration scenario" on page 382. The following example simply shows an alternative way of performing this conversion.

Example 11-1 Defining DB2 function TWH_TIMESTAMP_FMT

```
-- This function translate the Timestamp field of ITM tables
-- in an human-readable format. E.g
-- From : 1061003161020000
-- To   : 20061003:1610:20000
--
CREATE FUNCTION TDW_TIMESTAMP_FMT (PARAM VARCHAR(16))
RETURNS VARCHAR(20)
RETURN '20'     || substr(PARAM,2,6) || ':'
                || substr(PARAM,8,4) || ':'
                || substr(PARAM,12,5);
```

To create a report, you first need to create a report project.

1. From the File menu, select **New** -> **Project** to launch the New Project Wizard.
2. Open the Business Intelligence and Reporting Tools folder, select **Report Project**, and click **Next**.
3. Name the project `Sample` and click **Finish**.
4. In the window that prompts you to switch to the Report Design perspective, click **Yes**.
5. To create a report, right-click the **Sample** project in the Navigator, select **New**, and then click **Report**. Name the report `CPU_Usage.rptdesign`. When you expand your Navigator, it looks like Figure 11-29.

Figure 11-29 Creating a new project report

Creating a new data source

Create a data source that tells BIRT how to connect to the database. To do so, perform the following steps:

1. In the Data Explorer, right-click **Data Sources** and click **New Data Source**.
2. Click **JDBC Data Source** and name it `Sample Data Source`. Click **Next**, as shown in Figure 11-30.

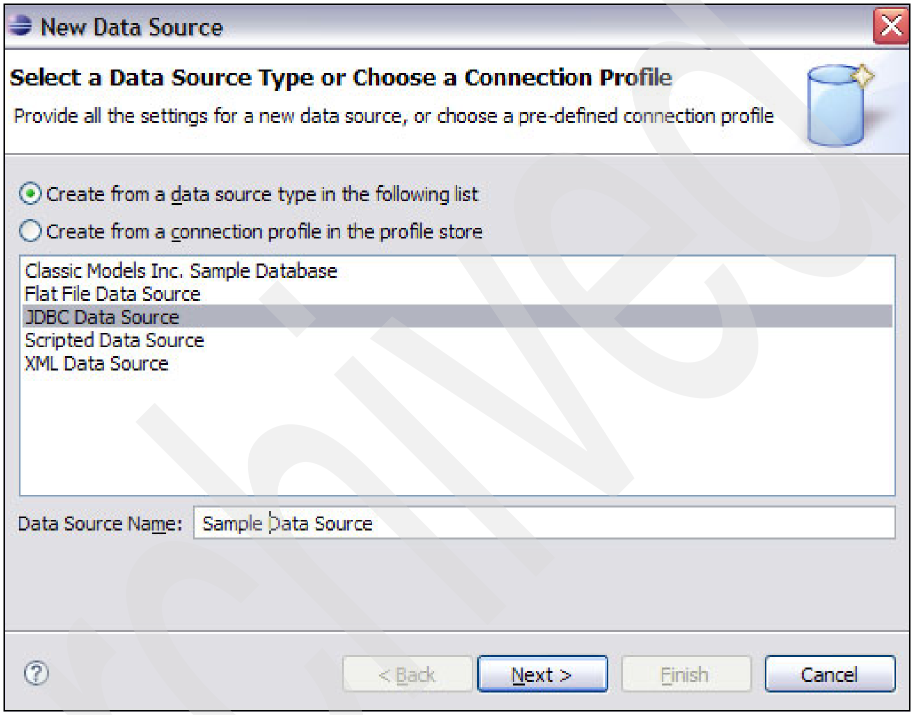


Figure 11-30 Creating a new data source

3. In the resulting page, shown in Figure 11-31 on page 405, specify the connection string for the data source:
   a. Select the driver class **com.ibm.db2.jcc.DB2Driver (v2.9)**.
   b. Type the database URL in the format `jdbc:db2://`*`hostname`*`:`*`port`*`/`*`dbname`* (in our environment, it is jdbc:db2://berlin.itsc.austin.ibm.com:50000/warehous).
   c. Type your RDBMS user ID and password and test the connection.
   d. When the test is successful, click **Finish**.

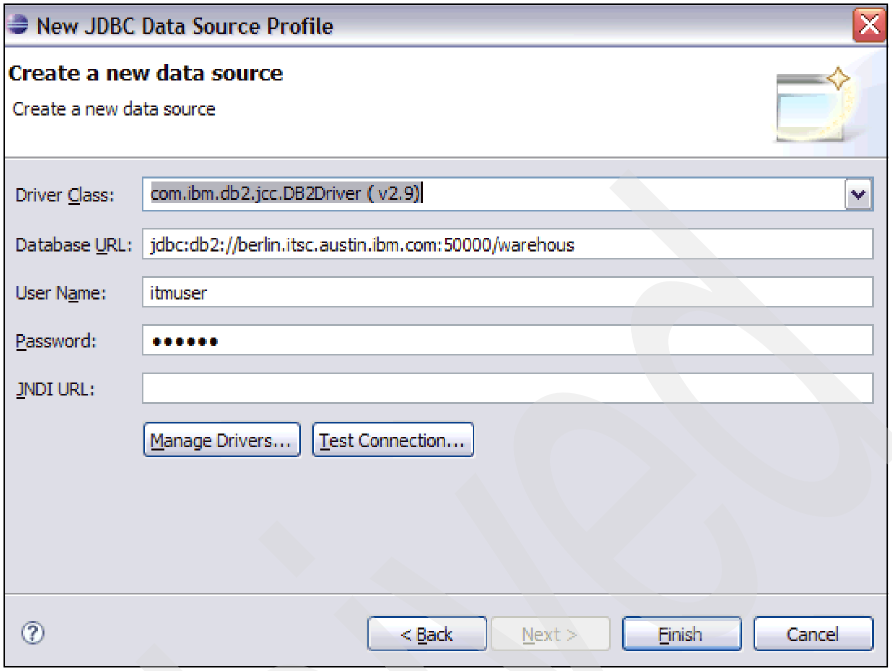


Figure 11-31 Defining a new data source

Creating a new data set

To display data from the data source on the report, you must create data sets for it. A *data set* represents the data that is available from the database to be put on the report. You can have many data sets for one data source. If your reports are talking to several databases, each one will be a data source, and each query that you want displayed on the report will be a data set for that data source.

To create a data set, perform the following steps:

1. In the **Data Explorer**, right-click **Data Sets** and click **New Data Set**.
2. The New Data Set window opens, as shown in Figure 11-32 on page 406.
   a. Give a name to the new data set (for example, `Linux_CPU`).
   b. In the Data Source field, select the newly created data source.
   c. In the Data Set Type field, select **SQL Select Query**.
   d. Click **Next**.

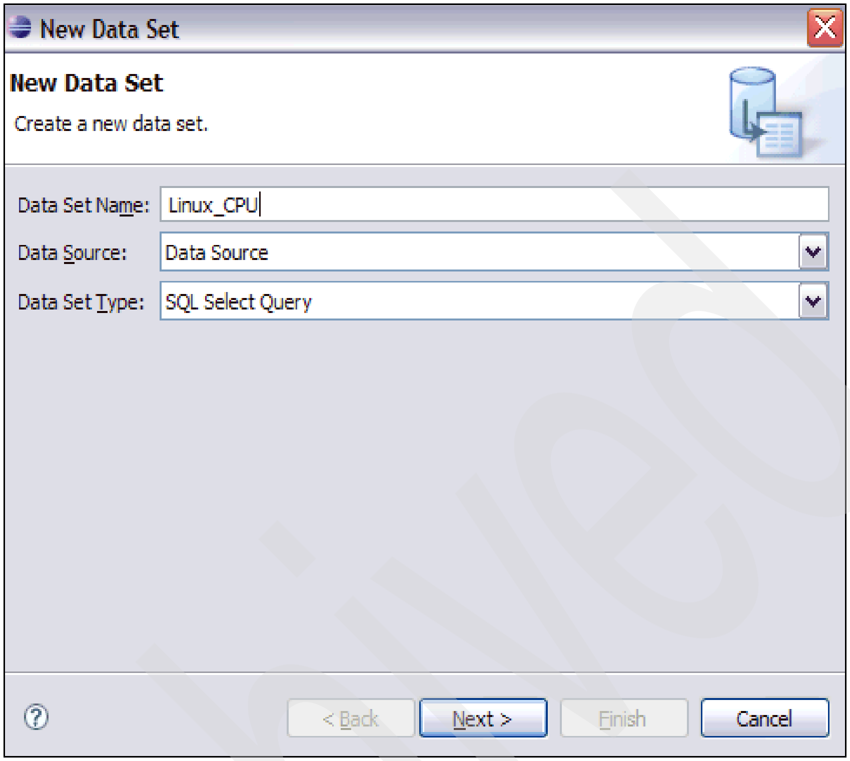


Figure 11-32 Creating a new data set

3. Define an SQL query in the window shown in Figure 11-33 on page 407. In the right pane of the window, enter the SQL statements or you can compose it by selecting the required tables and fields from the left pane.

In this example, the user-defined db2 function TWH_TIMESTAMP_FMT translates the Timestamp field to a human-readable format. For specific SQL statements that are used to define the functions, see 11.4, “Case Study 2 - IBM Tivoli Monitoring V6.1 and BIRT integration scenario” on page 401.

After you define the function and the query, click **Finish**.

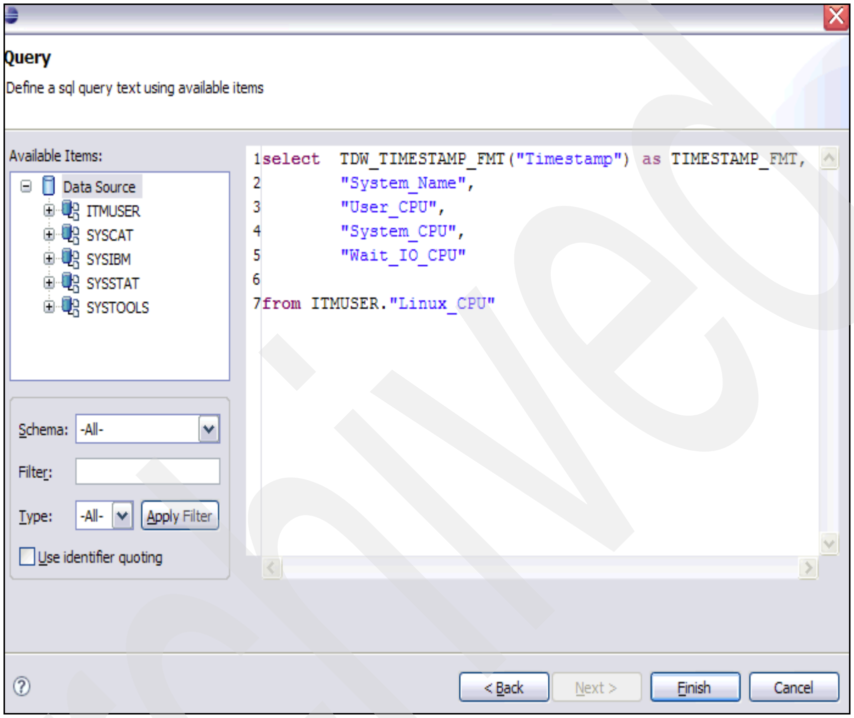


Figure 11-33 Defining a data set query

4. In the left pane of the window, click **Preview Results**. You see the output of the query, as shown in Figure 11-34. Now you are ready to put the dynamic content into the report.

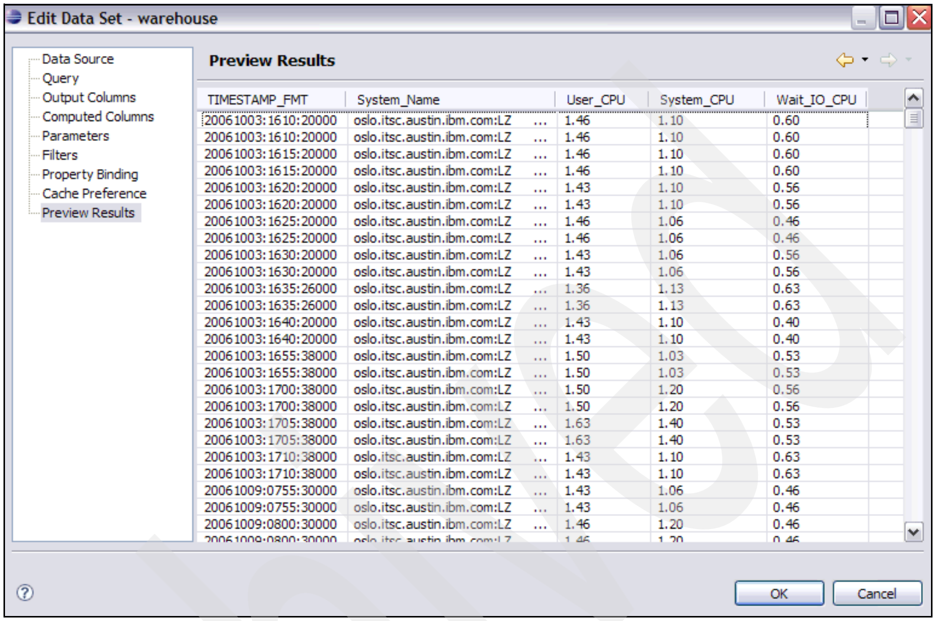


Figure 11-34 Previewing results

Creating the new report parameters

You can use BIRT report parameters to interactively filter the report's data to gain deeper insights. To do this, perform the following steps:

1. In the Data Explorer pane, right-click **Report Parameters** and select **New Parameter**.
2. The New Parameter window opens, as shown in Figure 11-35 on page 409. In this window, you can specify the following basic properties:
   – Name:

     Type a name for the parameter. If you do not specify a value for Prompt text, the report user sees the Name value as the prompt.

     It is good practice to supply an easily-understood name for the parameter in Prompt text and use the suffix *Par* for the value that you select for Name to help distinguish report parameters from other parameter types, such as data set parameters.

– Type:

Select a data type for the parameter. The data type that you select does not have to match the data type of the field in the data source. The data type that you select for the report parameter determines the formatting options that are available, if you choose to provide a default value or a list of values for the report parameter.

– Date:

Select either Date Time or String. Specifying the string data type provides more flexibility for the user to specify dates in a variety of formats.

3. Click **OK**. The parameter is displayed under Report Parameters in the Data Explorer. If necessary, repeat these steps to create additional report parameters.

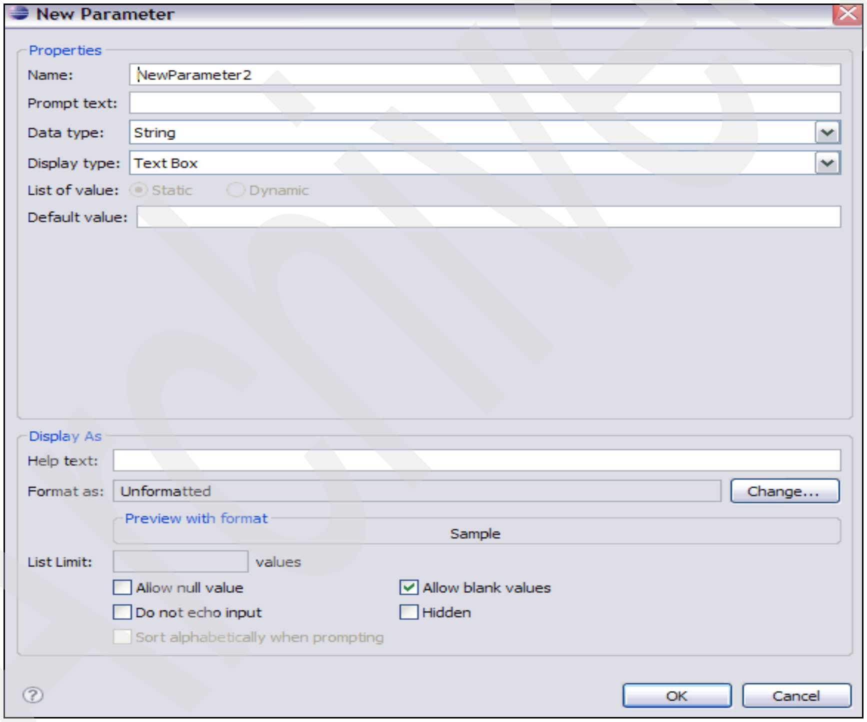


Figure 11-35 Defining a new parameter

In our example, we have to configure the capability of filter the selected data between two specific timestamps and for a particular system name. To do that, we have to define three new parameters:

– SystemNamePar (preferred system name)
– TimeStamp1Par (timestamp lower limit value)

– Timestamp2Par (timestamp upper limit value)

Figure 11-36, Figure 11-37 on page 411, and Figure 11-38 on page 412 show these parameters in the definition boxes.

Figure 11-36 shows the SystemNamePar parameter definition.

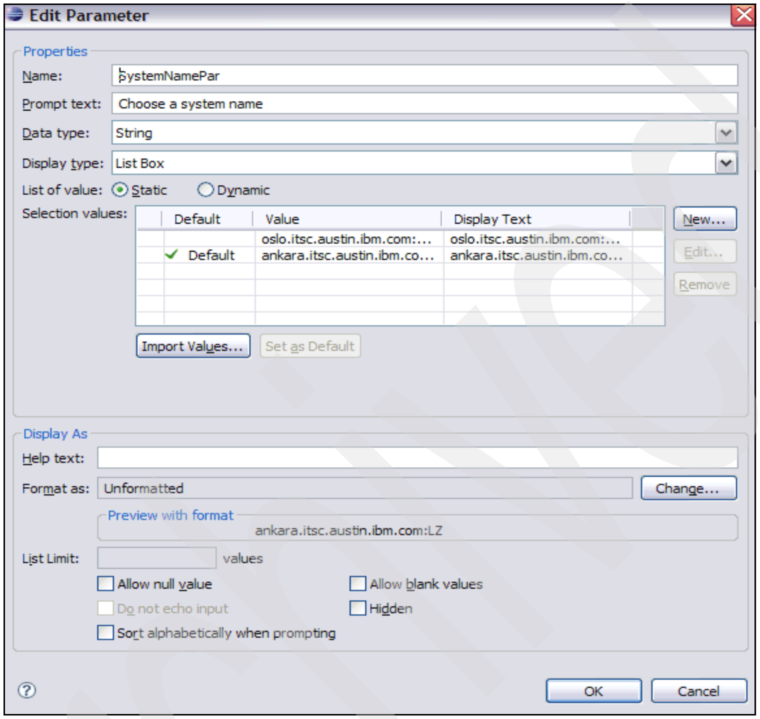


Figure 11-36 Defining the parameter SystemNamePar

Figure 11-37 shows the Timestamp1Par parameter definition.

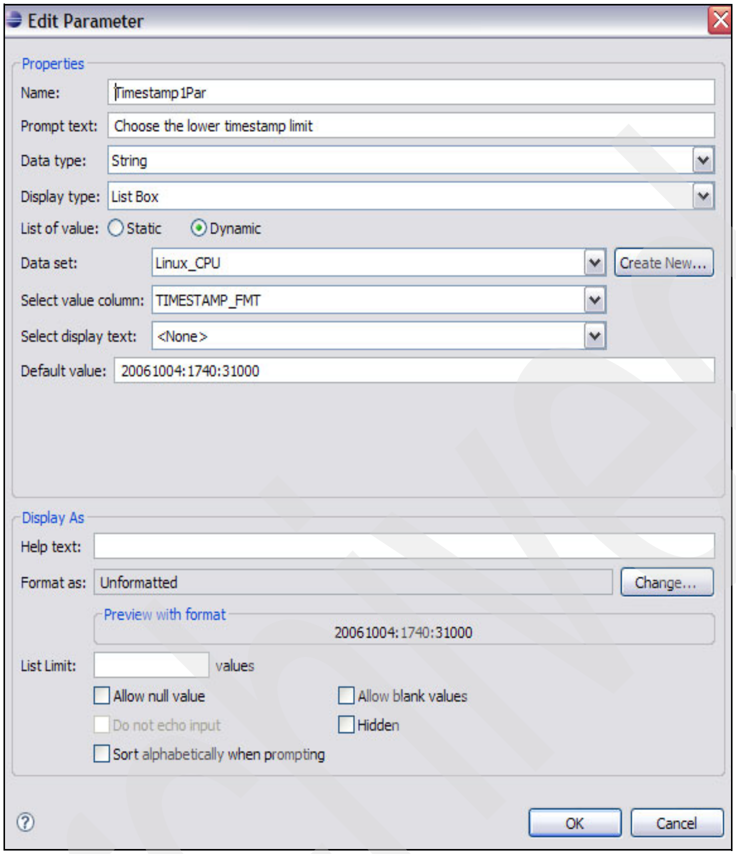


Figure 11-37 Defining the parameter Timestamp1Par

Figure 11-38 shows the Timestamp2Par parameter definition.

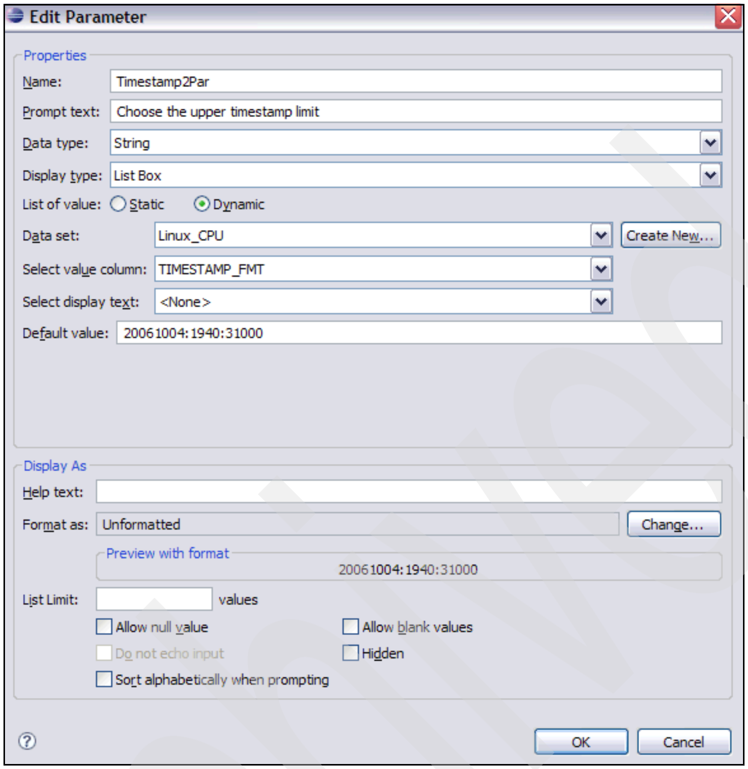


Figure 11-38 Defining the parameter Timestamp2Par

Defining the filtering

To define filtering, perform the following steps:

1. In the Data Explorer pane, double-click the **Linux_CPU** data set.
2. The Edit Data Set window opens. In the left pane, click **Filters** and define the filtering criteria based on the newly defined parameters, as shown in Figure 11-39.

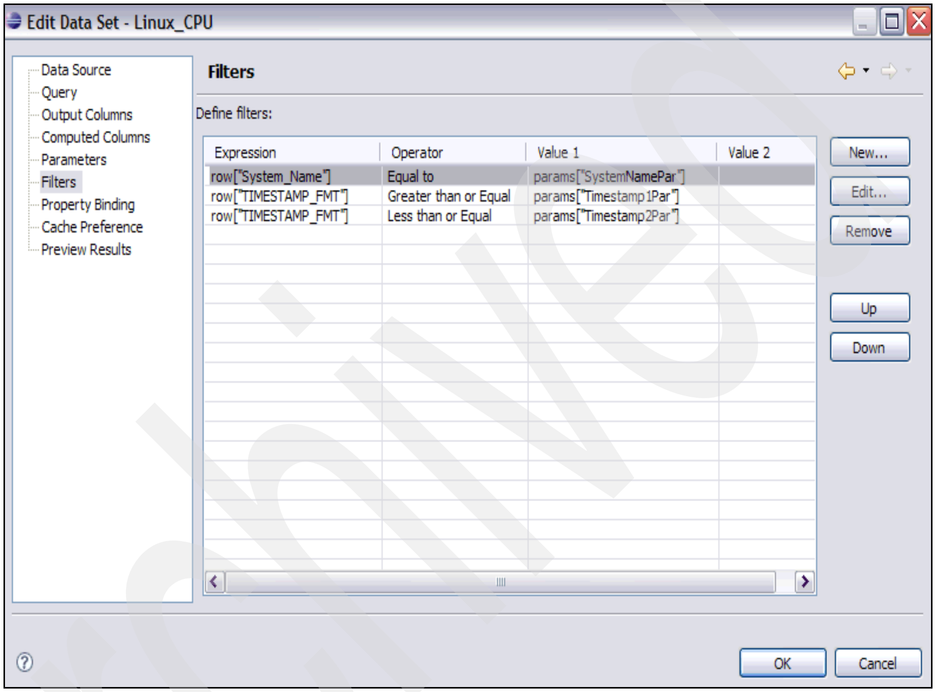


Figure 11-39 Defining filtering criteria

3. This filtering definition produces a runtime dialog box at report execution, as shown in Figure 11-40. The default values are shown. You can confirm these default values or choose different ones using the text box or by using the dynamic list box selections.

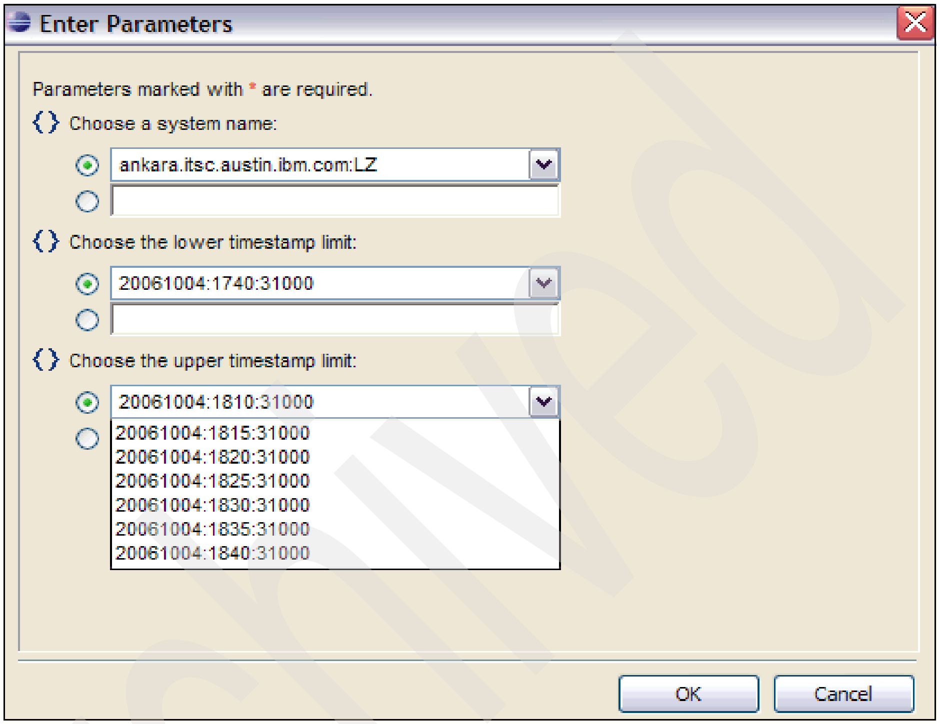


Figure 11-40 Parameter dialog box

Defining the report layout

By dragging-and-dropping the Linux_CPU data set from the Data Explorer pane to the CPU_Usage.rptdesign pane, you can obtain a basic table layout, as shown in Figure 11-41. Click the **Preview** tab to display your query.

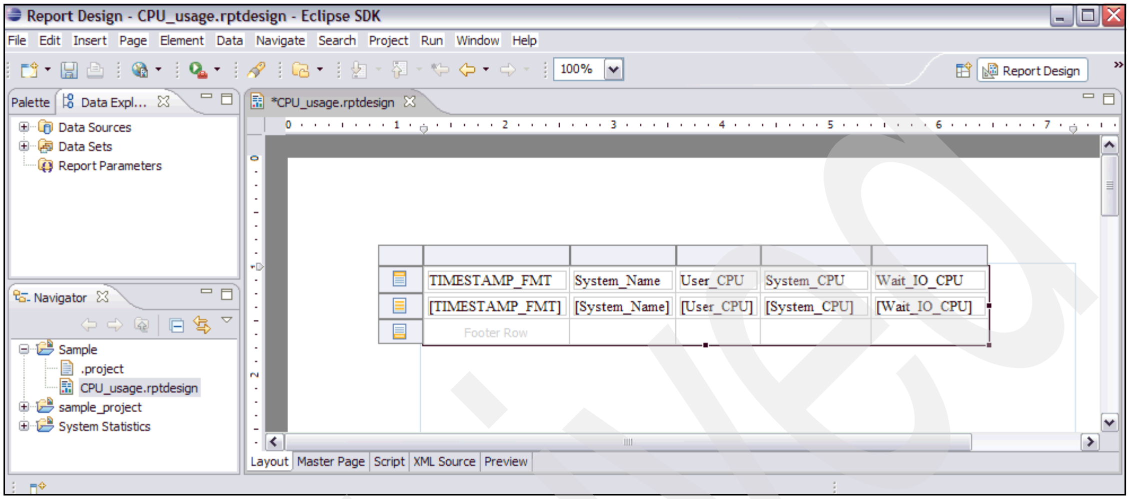


Figure 11-41 Defining a simple table layout

You see the output shown in Figure 11-42.

TIMESTAMP_FMT	System_Name	User_CPU	System_CPU	Wait_IO_CPU
20061004:1740:31000	ankara.itsc.austin.ibm.com:LZ	,66	,53	,59
20061004:1740:31000	ankara.itsc.austin.ibm.com:LZ	,66	,53	,59
20061004:1745:31000	ankara.itsc.austin.ibm.com:LZ	,63	,63	,66
20061004:1745:31000	ankara.itsc.austin.ibm.com:LZ	,63	,63	,66
20061004:1750:31000	ankara.itsc.austin.ibm.com:LZ	,66	,53	,56
20061004:1750:31000	ankara.itsc.austin.ibm.com:LZ	,66	,53	,56
20061004:1755:31000	ankara.itsc.austin.ibm.com:LZ	,66	,53	,7
20061004:1755:31000	ankara.itsc.austin.ibm.com:LZ	,66	,53	,7
20061004:1800:31000	ankara.itsc.austin.ibm.com:LZ	,7	,46	,63
20061004:1800:31000	ankara.itsc.austin.ibm.com:LZ	,7	,46	,63
20061004:1805:31000	ankara.itsc.austin.ibm.com:LZ	,7	,46	,66

Figure 11-42 Previewing query result

The BIRT Reporting Interface offers various capabilities of report and chart design and configuration. Figure 11-43 shows an example of a bar chart report representation of the extracted data.

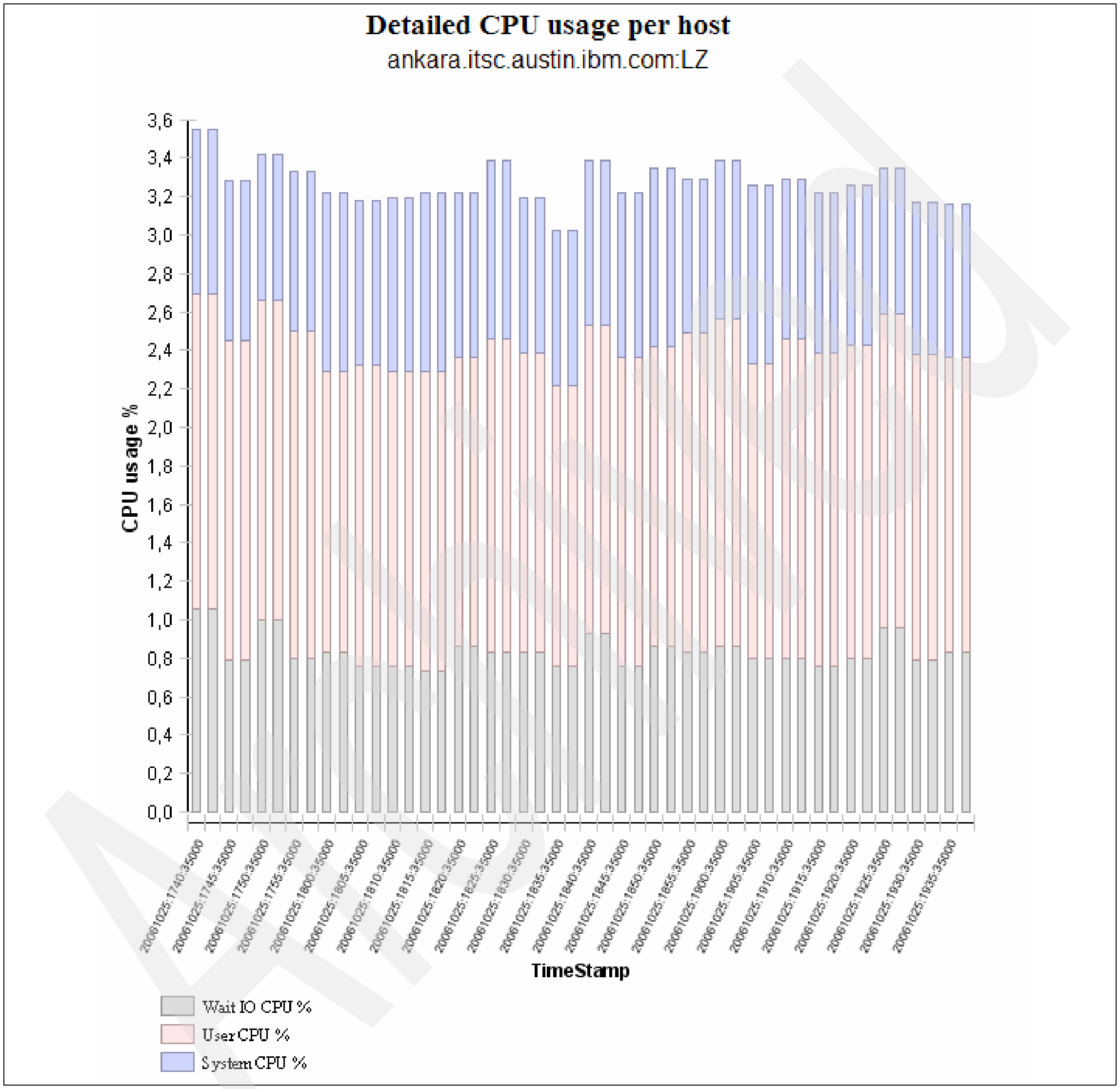


Figure 11-43 Bar chart representation

Publishing the results

You can publish the generated report (.rptdesign file) on your application server. Rptdesign files are located by default in your BIRT Report Designer machine under the directory C:\Documents and Settings\Administrator\workspace.

To publish the report to the application server, copy the .rptdesign file from the desktop where it was designed, to the report directory of the application server. By default, this is the directory where BIRT is installed.

You can view the report by accessing the following link using a Web browser:

`http://applicationserver:8080/birt/frameset?__report=reportname`

Scheduling the report

You can schedule your report creation at a specific point in time and produce output that can be placed in a specific directory. This enables users to download reports and view them when they are offline. To do this, you can use the **wget** command and schedule it in crontab.

In the following example, we used the **wget** command to produce a report output in HTML format:

```
wget

http://applicationserver:8080/birt/run?__report=reportname&__format=for
mat -O filename
```

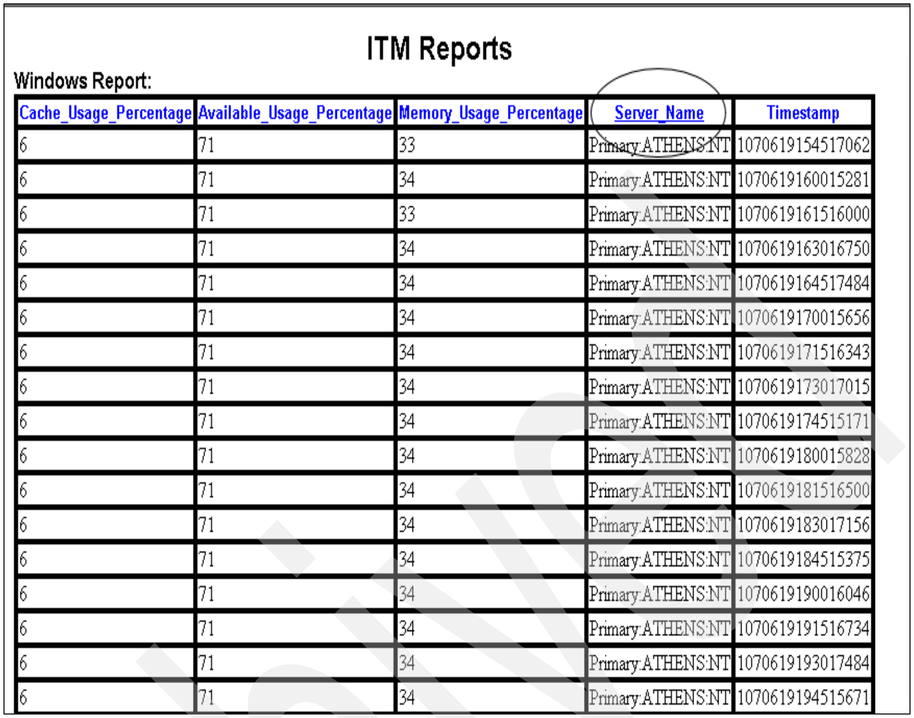
11.5 BIRT tools

There are two BIRT tools that you might want to use during creation of reports. These are:

- Hyperlink
- Combination chart

Hyperlink

BIRT allows you to use hyperlinks between the reports. It is a handy function that can help you go from one report to another. As an example, we create a initial report with a table displaying data from all Windows server of our environment. If customers want to see detailed information about this server, they can use the hyperlink placed at the server name to move to a report chart; Figure 11-27 on page 401 shows a link example.



ITM Reports

Windows Report:

Cache_Usage_Percentage	Available_Usage_Percentage	Memory_Usage_Percentage	Server_Name	Timestamp
6	71	33	Primary:ATHENS:NT	1070619154517062
6	71	34	Primary:ATHENS:NT	1070619160015281
6	71	33	Primary:ATHENS:NT	1070619161516000
6	71	34	Primary:ATHENS:NT	1070619163016750
6	71	34	Primary:ATHENS:NT	1070619164517484
6	71	34	Primary:ATHENS:NT	1070619170015656
6	71	34	Primary:ATHENS:NT	1070619171516343
6	71	34	Primary:ATHENS:NT	1070619173017015
6	71	34	Primary:ATHENS:NT	1070619174515171
6	71	34	Primary:ATHENS:NT	1070619180015828
6	71	34	Primary:ATHENS:NT	1070619181516500
6	71	34	Primary:ATHENS:NT	1070619183017156
6	71	34	Primary:ATHENS:NT	1070619184515375
6	71	34	Primary:ATHENS:NT	1070619190016046
6	71	34	Primary:ATHENS:NT	1070619191516734
6	71	34	Primary:ATHENS:NT	1070619193017484
6	71	34	Primary:ATHENS:NT	1070619194515671

Figure 11-44 Link example

Figure 11-44 shows the link result.

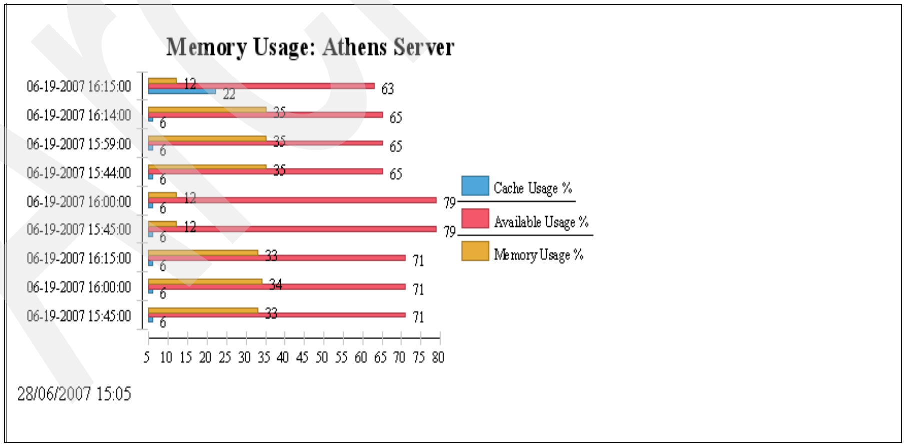


Figure 11-45 Link result

Combination chart

Combination chart is a type of report used to compare two different types of data, so you have multiple Y axes.

The bars within the chart represents the Y axis. The bar chart format with a Line series represents the secondary Y axis and has its own scale.

Figure 11-46 demonstrates an example of a combination chart that can be created within BIRT.

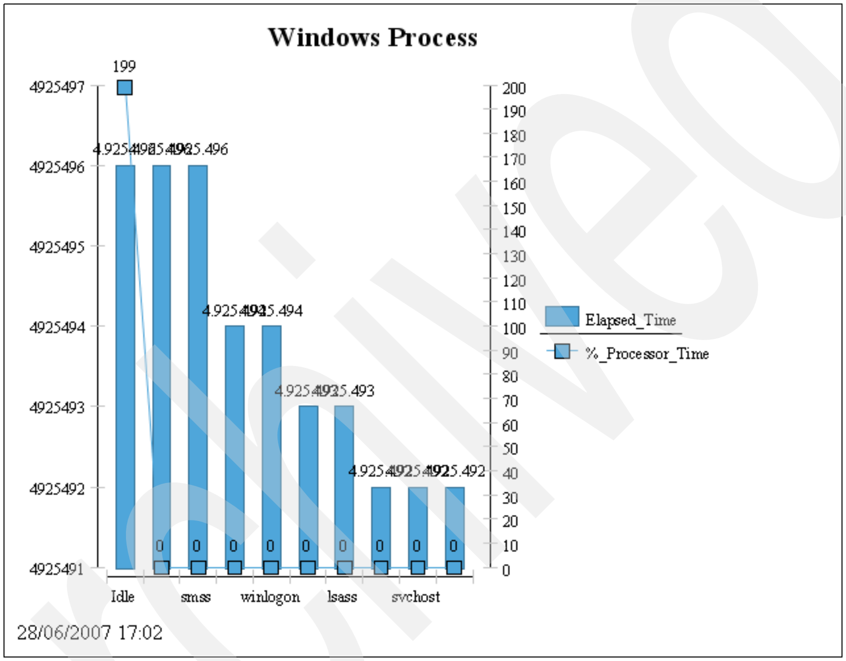


Figure 11-46 Combination chart example

11.6 OPAL solutions

The IBM Tivoli Open Process Automation Library (OPAL) Web site is a catalog of solutions provided by IBM and IBM Business Partners.

IBM Tivoli Open Process Automation Library

There are several reporting solutions that you can use with the Tivoli Data Warehouse. These solutions are listed in the IBM Tivoli Open Process Automation Library (OPAL) Web site.

The IBM Tivoli Open Process Automation Library is a worldwide online catalog accessible at:

http://catalog.lotus.com/wps/portal/topal/

This Web site provides a central location for hundreds of technically validated, ready-to-use Tivoli product extensions and services provided by IBM and IBM Business Partners. OPAL contains downloadable product extensions such as:

- Automation packages
- Integration adapters
- Documentation
- Plug-in tool kits
- Technical services

The following are some of the OPAL Solutions related with reporting:

- Web Services/SOAP Real-Time and Scheduled Reporting Tools for IBM Tivoli Monitoring V6.1

  This Solution includes a few tools that use Web Services/SOAP to perform both real-time and scheduled reporting on the IBM Tivoli Monitoring V6.1 environment. There are different tools to report on the health of the IBM Tivoli Monitoring environment, and to perform real-time analysis and warehouse reporting.

- Warehouse reporting using BIRT

  Two papers have been published on OPAL, providing instructions and examples for creating Warehouse Reports using BIRT. For details about reporting Solutions, visit the OPAL Web site.

12

IBM Change and Configuration Management Database integration

This chapter discusses intergration between IBM Tivoli Monitoring and the IBM Change and Configuration Management Database (CCMD) tools that are available. The chapter focuses on how to use Tivoli Application Dependency Discovery Manager (TADDM) to identify systems that are not currently monitored by IBM Tivoli Monitoring, and how to automatically deploy agents to those machines.

The chapter also covers using TADDM to identify underutilized servers with few dependencies within an IBM Tivoli Monitoring environment for the purpose of server consolidation. The following topics are addressed:

- IBM CCMDB and TADDM fundamentals
- Using TADDM and IBM Tivoli Monitoring to discover and deploy to unmanaged systems
- Using TADDM and IBM Tivoli Monitoring to facilitate server consolidation

12.1 IBM Change and Configuration Management Database and Tivoli Application Dependency Discovery Manager fundamentals

IBM Change and Configuration Management Database (CCMDB) is an integrated productivity tool and database that helps you manage, audit, and coordinate change and configuration management within an enterprise by helping you understand what resources are available.

This chapter discusses using a database component of CCMDB called Tivoli Application Dependency Discovery Manager (TADDM). TADDM provides the ability for users to determine a topological view of systems and their applications with current configurations.

Unlike IBM Tivoli Monitoring, TADDM is an agent-free application. It creates a Configuration Management Database by polling resources across an enterprise according to a defined scope. There is no need to deploy additional resources on each machine in the environment.

After the database of the environment has been created, you are able to query it for the following purposes:

- To understand the dependencies between resources.
- To identify systems that are incorrectly configured, missing maintenance, or that are unmonitored by your IBM Tivoli Monitoring infrastructure
- To aid planning for application changes to minimize or eliminate unplanned disruptions

TADDM performs a complete discovery and stores information:

- On the applications, including their dependencies
- On the servers that are hosting the applications
- On the network (to Layer 2)

TADDM uses sensors to discover this information. Discovery Library Adapters (DLA) are also used to integrate other applications (such as IBM Tivoli Monitoring) with CCMDB/TADDM.

Figure 12-1 on page 423 shows the TADDM GUI, available sensors, and discovery library adapters.

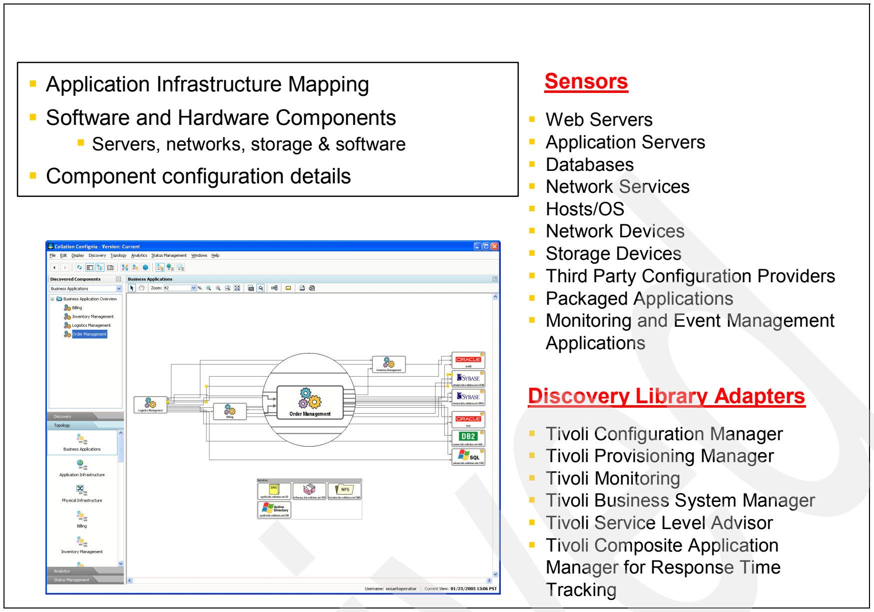


Figure 12-1 CCMDB – Discovery, application mapping and topology

> **Note:** In this chapter we will be using TADDM version 5.1.1 Fix Pack 3 in our examples, in conjunction with IBM Tivoli Monitoring V6.1 Fix Pack 5.

12.2 Using Tivoli Application Dependency Discovery Manager and IBM Tivoli Monitoring to discover and deploy to unmanaged systems

This section describes a scenario which utilizes the system discovery capabilities of TADDM to locate systems that are currently unmanaged by IBM Tivoli Monitoring. After these unmanaged systems have been identified, corrective action is taken to ensure that the appropriate monitoring agents are deployed using the IBM Tivoli Monitoring remote deployment capabilities.

This scenario is particularly appropriate for users with large numbers of physical systems (for example, Linux or Windows machines), where it can be difficult to ensure that all systems are part of the managed environment and are up-to-date with the latest maintenance levels.

12.2.1 Scenario background

The scenario environment contains numerous physical systems based on several operating systems (Windows, Linux, UNIX). The aim is ensure that all systems are known to be monitored within the IBM Tivoli Monitoring topology.

Each physical machine box should have its appropriate operating system agent installed and running. There may also be other monitoring agents (for example, Universal Agent) installed on these systems, and the goal is to be aware of these agents too. Figure 12-2 shows the scenario environment and the various physical systems to be monitored.

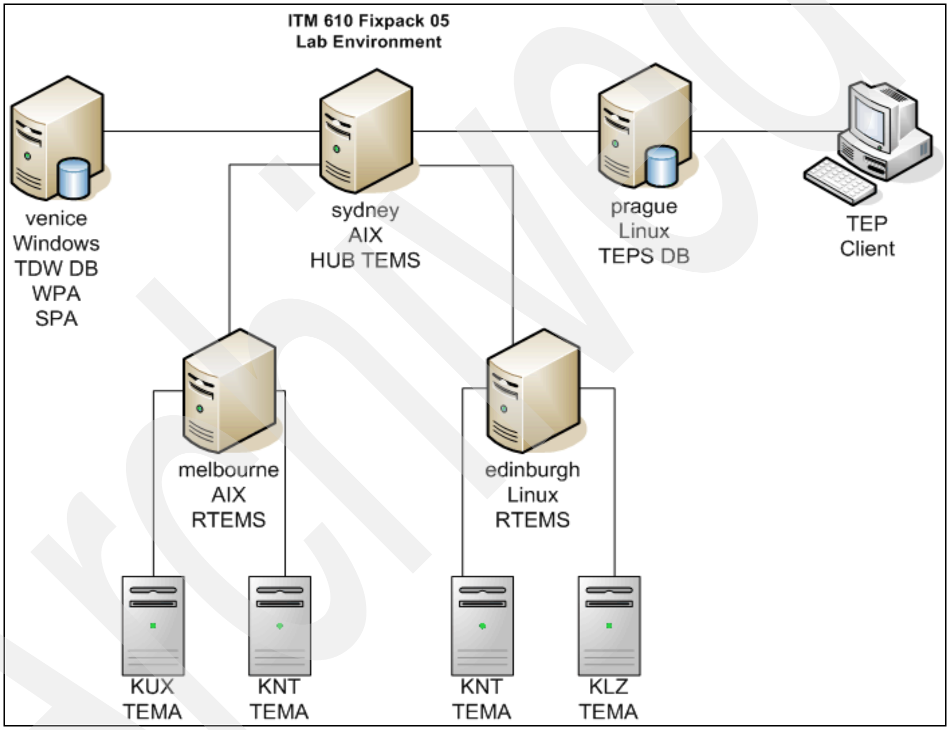


Figure 12-2 ITSO IBM Tivoli Monitoring environment

Figure 12-3 on page 425 shows the overview of this scenario.

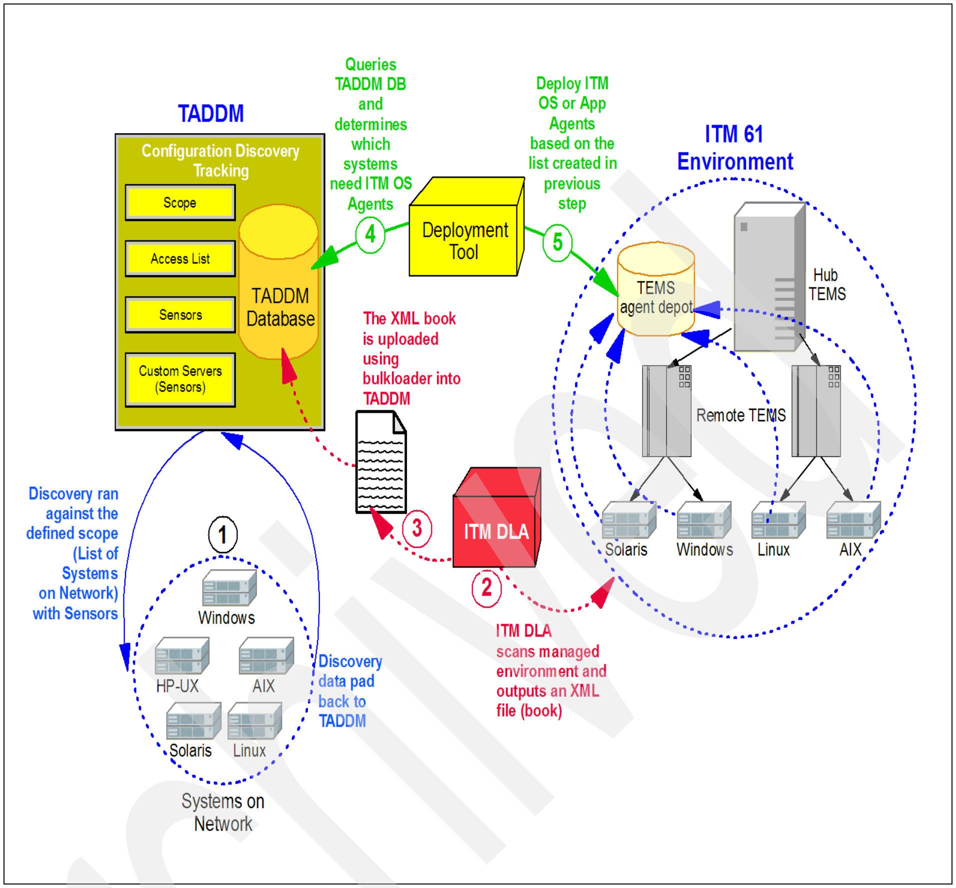


Figure 12-3 Scenario overview of identifying unmanaged systems

Note that, scaled up in a customer environment, there may be more than 1000+ physical systems to manage. If older systems are removed to be replaced with new systems over time, it becomes very difficult to ensure that all systems have been correctly installed with the appropriate IBM Tivoli Monitoring components. Furthermore, it is easy for some systems to be missing from the monitoring environment.

12.2.2 Using TADDM to discover all systems on the network

To establish a comphrensive list of all physical systems within the network, in this scenario the *discovery* process within TADDM was used. Depending on the size of the environment, the discovery process can take some time to complete.

Therefore, it is important to define the criteria as precisely as possible so that all the physical systems you are interested in are found—but also so that the process does not waste time searching on criteria that is irrelevant to your environment.

In the scenario, we defined the following criteria for the environment under the Discover tab in TADDM:

- Under **Scope**, we defined a range of IP addresses that we knew all our physical systems will be connected on. No system outside of this range (9.3.5.0 to 9.3.5.255) needed to be monitored as part of our IBM Tivoli Monitoring environment, so we did not want to waste time checking to see whether these systems had IBM Tivoli Monitoring OS agents installed.

  > **Note:** If your environment covers several ranges of IP addresses, this can be accommodated in TADDM by adding additional ranges.

- Under **Access List**, we defined all the username and password combinations that will be required to allow TADDM to connect to the physical machines and probe the system for active processes. There may be several username and password combinations to add for different operating systems.

  The usernames you define here must exist on the target machines and have appropriate access to find IBM Tivoli Monitoring processes; otherwise, the discovery will fail. To improve efficiency, in the scenario we also limited the usernames to the scope range defined previously.

- Under **Custom Server**, we created a search criteria for the discovery process to use in order to establish if the processes we were interested in were running on these machines.

  In our example, these were the IBM Tivoli Monitoring OS agents for Windows (knt), UNIX (kux), and Linux (klz). Figure 12-4 on page 427 shows the settings we chose for the scenario.

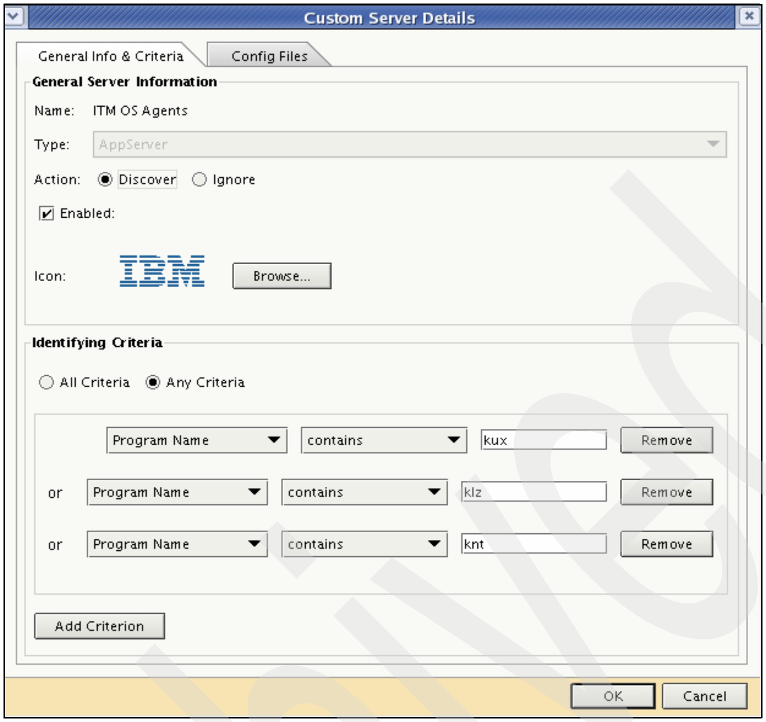


Figure 12-4 TADDM discovery customer server details

The settings chosen are listed in Table 12-1.

Table 12-1 Customer Server property settings

Custom Server property	**Value**
Name	IBM Tivoli Monitoring OS Agents
Type	AppServer
Action	Discover
Enabled	Yes
Identifying Criteria	Any Criteria
Criteria	Program Name contains kux OR Program Name contains klz OR Program Name contains knt

The criteria settings should be altered to fit the product code for the agents you want to discover in your environment.

> **Note:** Ensure that you choose the correct case for any agents you may be searching for on case-sensitive Linux or UNIX environments. Typically this should be lower case (for example, kux rather than KUX).

Once these settings have been defined, the discover process can begin. TADDM will build a topological view of the defined environment.

12.2.3 Scan IBM Tivoli Monitoring environment using Discovery Library Adapters

The next step is to scan the known IBM Tivoli Monitoring environment using a tool known as the IBM Tivoli Monitoring Discovery Library Adapters (DLA). A DLA is a program that extracts data from a source application (in this scenario, our IBM Tivoli Monitoring environment), and generates an XML file specifying details about the managed environment. We then upload the generated XML file to TADDM for comparison with the discovered environment we generated previously.

The DLA gathers information from the hub monitoring server by querying for all managed systems and mapping them according to the agent product code and managed system name format. For agents that use IP, IP.PIPE or IP.SPIPE as their communication protocol, the DLA can discover the IP address where the agent is running plus the operating system of the machine hosting the agent regardless of whether an OS monitoring agent is running on that machine or not.

The DLA also queries the portal server for information regarding logical groupings within the environment. For this reason, both the portal server and all monitoring server instances must be *online* when running the DLA tool. Monitoring agents do not have to be online since the information is derived from the monitoring server and portal server.

The IBM Tivoli Monitoring DLA tool is an executable installed under the portal server folder. The executable is known by the following names:

- Windows:

  KfwTmsDla.exe

- UNIX and Linux:

  KfwTmsDla

Run the executable from the command line. The tool has the following arguments available.

```
KfwTmsDla [-?] [-b] [-d] [-l] [-o orgname] [-s] [-x outputfile]
```

where:

-? Display the syntax help information.

-b Opens a browser with which to view the output of the adapter.

-d Creates a diagnostic file during the discovery process. You can use this file for debugging.

-l Indicates to discover logical views.

-o orgname Sets the organization global name.

-s Use HTTPS URL instead of HTTP URL.

-x outputfile Set an alternative name for the XML output file.

We ran it against our environment with the following arguments:

```
KfwTmsDla -l -o ITSO -x prague.xml
```

> **Tip:** On UNIX and Linux, you may need to run the following steps to ensure the PATH environment variable is set up correctly to run the script:
>
> ```
> cd /opt/IBM/ITM/config
> .  ./cq.config
> cd /opt/IBM/ITM/<OS Specific Directory>/cq/bin
> .  ./pathsetup.sh
> ```
>
> where <OS Specific Directory> varies according to the operating system you are using.

By default, the DLA generates the XML output file in the \tmsdla subdirectory on the portal server. The name of this file follows the standard Discovery Library file name format. Example 12-1 shows a sample of the generated XML output.

Example 12-1 Snapshot™ of XML data generated by KfwTmsDwl tool

```
<cdm:sys.ComputerSystem id="sydney.itsc.austin.ibm.com-ComputerSystem"
sourceToken="ip_address=9.3.5.33">
   <cdm:Name>sydney.itsc.austin.ibm.com</cdm:Name>
   <cdm:Signature>9.3.5.33</cdm:Signature>
   <cdm:Type>ComputerSystem</cdm:Type>
   <cdm:Fqdn>sydney.itsc.austin.ibm.com</cdm:Fqdn>
</cdm:sys.ComputerSystem>
<cdm:sys.unix.Unix id="sydney.itsc.austin.ibm.com-Unix"
sourceToken="managed_system_name=sydney.itsc.austin.ibm.com:KUX&amp;obj
ect_id=p@sydney.itsc.austin.ibm.com:KUX">
   <cdm:ManagedSystemName>sydney.itsc.austin.ibm.com:KUX
   </cdm:ManagedSystemName>
```

```
   <cdm:Name>sydney.itsc.austin.ibm.com</cdm:Name>
</cdm:sys.unix.Unix>
```

> **Important:** The IBM Tivoli Monitoring DLA tool was first provided with IBM Tivoli Monitoring V6.1 Fix Pack 3. It is installed as a portal server component and *must* be run on the machine where the portal server has been installed.

12.2.4 Upload generated XML file to TADDM

Following the generation of the XML book describing the IBM Tivoli Monitoring managed environment, it must be uploaded to the TADDM database so that it can be compared to the environment discovered by TADDM.

The file is uploaded using the bulk loader program provided by TADDM. Follow these steps to perform this task:

1. Copy the generated XML to the machine running TADDM. The directory where the XML file is placed must not be the same as the working directory of the bulk loader, because this will cause errors.
2. Log on to TADDM as the Administrator user.
3. Run the bulk loader tool (ldfxidml), specifying the fully-qualified path to the XML file. For example, in our scenario we specified:

   ```
   ./ldfxidml.sh -f /tmp/prague.xml
   ```

   Example 12-2 shows the output on screen of a successful execution of the script.

Example 12-2 Command line output from running bulk loader

```
[cdtadmin@CDT511 bin]$ ./ldfxidml.sh -f /tmp/prague.xml
Bulk Load Program starting.
Bulk Load Program running.
Bulk Load Program running.
Bulk Load Program succeeded. Return code is: 0
0
Bulk Load Program ending.
```

> **Note:** ldfxidml was introduced in TADDM Version 5.1.1 Fix Pack 2, and is very similar to the **`loadidml`** command. However, it performs processing of the DLA book specific to IBM Tivoli Monitoring.

4. Check the results file for problems during the bulkload process. Look for a file with an extension of `.results`, and having the same name as the input XML

file. For example, if the input file was test.xml, the name of the results file will be test.results.

Important entries in this file are marked with `SUCCESS` or `FAILURE` tags. `Percentage successful` messages are also recorded if statistics are enabled.

> **Note:** Entries tagged `FAILURE` are for individual objects, and do not necessarily indicate a failure of the entire file.

For more details about the bulk loader tool provided by TADDM, consult *IBM Tivoli Configuration Discovery and Tracking User's Guide*, found at the following URL:

http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.ccmdb.doc/ccmdb_welcome.htm

12.2.5 Query the TADDM database

After the details were successfully loaded into TADDM, we could create a report about which systems are currently unmanaged by our IBM Tivoli Monitoring infrastructure.

We utilized a tool developed for this purpose that simplifies this process by querying the TADDM database for the details about the systems discovered, and then parses the information to produce a report. The report contained a list of systems in three categories:

- All systems discovered by TADDM, as performed in 12.2.2, "Using TADDM to discover all systems on the network" on page 425
- All systems with IBM Tivoli Monitoring OS agents installed (managed systems)
- All systems with no IBM Tivoli Monitoring OS agents installed (unmanaged systems)

With this report, we were able to see what action needed to be performed in order to ensure that all machines in the defined environment were managed by an IBM Tivoli Monitoring OS agent.

To download the tool, go to the IBM Tivoli OPAL (Open Process Automation Library) Web site. Search under "ITM TADDM unmanaged systems".

The tool is provided as a zip or tar file, depending on your operating system. The file should be uncompressed on the machine running the hub monitoring server.

After it is uncompressed, the following subdirectories will be created:

/discovered This contains the generated reports after the tool is run.

/etc This contains the files log4j.xml, policy.all and collation.properties. The collation.properties file needs to be copied from the TADDAM machine located under $COLLATION_HOME/dist/etc.

/external This contains a Java runtime for use by the tool. The Java files are not provided in the downloaded image. Copy the JRE as supplied by IBM Tivoli Monitoring into this folder.

/lib This contains source files as used by the tool. These files are *not* provided in the downloaded image. Copy the following files from subdirectories under the TADDM install:

api-client.jar, api-dep.jar and platform-model.jar from dist/lib

xalan.jar from dist/sdk/lib

commons-jxpath-1.1.jar from dist/deploy-tomcat/cdm/WEB-INF/lib

/log This contains log files produced by the tool when making calls to TADDM.

/results This contains output from running the scripts.

/scripts This contains the script files used.

> **Note:** In order to keep the download size smaller, several required files are *not* provided within the compressed file. Therefore, you need to copy the appropriate files as listed here.

The scripts must be edited prior to use in order to set the variables as appropriate for your environment. These variables will be used by the script to log on to the TADDM database and any other machines. We edited the script itm.sh on our AIX machine (sydney) to the settings shown in Example 12-3.

Example 12-3 Variables defined within itm.sh

```
export TOOL_HOME=/tmp/taddm_tool
SCRIPTS=$TOOL_HOME/scripts
export CANDLEHOME=/opt/IBM/ITM
TADDMID=administrator
TADDMPASS=password
TADDMHOST=CDT511
HUBUSERID=sysadmin
```

```
HUBPASS=password
HUBTEMS=sydney.itsc.austin.ibm.com
WINID=Administrator
WINPASS=password
UNIXID=root
UNIXPASS=password
UNIXDIR=/opt/IBM/ITMOSAgent
WINDIR=C:/IBM/ITMOSAgent
OUTPUTDIR=$TOOL_HOME/results
OUTPUTDIR_A=$TOOL_HOME/discovered
```

The script is run from the command line. It has the following syntax and arguments:

```
itm.sh [-c inputfile | -d | -v]
```

where:

-c inputfile This invokes the script for the purpose of deploying OS agents on unmanaged machines using the IBM Tivoli Monitoring-provided **tacmd createnode** command.

The supplied input file argument lists all systems for which the script should attempt to create nodes on. The script keeps track of which systems were deployed to successfully and which failed. Refer to 12.2.6, “Deploy new IBM Tivoli Monitoring OS agents” on page 435 for more information about using this option.

-d This invokes the script for the purpose of querying the TADDM database to produce the reports indicating which systems have the appropriate IBM Tivoli Monitoring agents installed.

-v This invokes the script to verify whether or not the agent connected to the managed environment successfully.

We ran the following command against our environment to determine which systems were correctly part of our managed IBM Tivoli Monitoring environment and which are not:

```
itm.sh -d
```

The following reports were produced:

Example 12-4 Sample output from itm.sh -d

```
######################
Starting Tool
```

```
Thu Jul 12 14:37:56 CDT 2007
######################
Gathering Stats On Discovered Objects By TADDM
----------------------------------
Systems Discovered By TADDM - (17)
----------------------------------

---------------------------------
Systems With ITM OS Agents - (10)
---------------------------------
chicago.itsc.austin.ibm.com
istanbul.itsc.austin.ibm.com
melbourne.itsc.austin.ibm.com
newyork.itsc.austin.ibm.com
nice.itsc.austin.ibm.com
oslo.itsc.austin.ibm.com
paris.itsc.austin.ibm.com
prague.itsc.austin.ibm.com
sydney.itsc.austin.ibm.com
venice.itsc.austin.ibm.com

------------------------------------
Systems WithOut ITM OS Agents - (7)
------------------------------------
ankara.itsc.austin.ibm.com
belgrade.itsc.austin.ibm.com
elpaso.itsc.austin.ibbm.com
oslo.itsc.austin.ibm.com
paris.itsc.austin.ibm.com
rp3410.itsc.austin.ibm.com
waco.itsc.austin.ibm.com
######################
Exiting Tool
Thu Jul 12 14:46:12 CDT 2007
######################
```

Tip: On AIX, you may experience problems with Java heap allocation. If you do, you will see the following message when running the script:

```
[ Unable to mmap Java heap of requested size, perhaps the maxdata
value is too large - see Java README.HTML for more information. ]
```

In our environment, we corrected this by editing the script scripts/api.sh and reducing the value specified by the -Xmx parameter, as shown:

```
-Xmx1024M
```

This reduced the maximum heap size that the JVM would allocate.

As can be seen, the machines we deployed as part the managed environment used throughout this book were identified as having IBM Tivoli Monitoring OS agents deployed on them. However, several other machines were identified as not having agents deployed, and we wanted to correct this situation.

12.2.6 Deploy new IBM Tivoli Monitoring OS agents

Following a successful execution of the script to determine which systems were part of our managed IBM Tivoli Monitoring environment and more importantly, which were not, we re-ran the script with the `-c` argument to deploy IBM Tivoli Monitoring OS agents to the machines in the latter category.

The script requires an input file containing a list of systems with no IBM Tivoli Monitoring OS agents on them. The file is then parsed by the script, which makes successive calls to the IBM Tivoli Monitoring command **`tacmd createnode`** in order to deploy the appropriate agent. The input file is generated by the previous call of the script and, assuming default settings, it can be found at /discovered/ITMAgentRequired.

As with the previous section, consider editing the script so that **`tacmd`** uses arguments needed for your environment. The default settings are shown in Example 12-5.

Example 12-5 tacmd command within itm.sh script

```
#Invoking CreateNode
$CANDLEHOME/bin/tacmd createnode -h $ep -u $user -w $passwd -d $dir
2>&1 | tee > $OUTPUTDIR/deploy_result-$ep.txt
```

For our scenario, we ran the following command to execute this script:

```
itm.sh -c /tmp/taddm_tool/discovered/ITMAgentRequired
```

Example 12-6 shows the output from running this command with an input file specifying three machines that we wanted to deploy IBM Tivoli Monitoring OS agents to.

Example 12-6 Sample output from itm.sh -c

```
######################
Starting Tool
Mon Jul 16 12:08:24 CDT 2007
######################

--------------------------------------------
Deploying ITM OS Agents on Following Systems
--------------------------------------------

 Password?
Validating user...

KUIC00007I: User sysadmin logged into server on
https://sydney.itsc.austin.ibm.com:37048.
SystemName: oslo.itsc.austin.ibm.com
/opt/IBM/ITM/bin/tacmd createnode -h oslo.itsc.austin.ibm.com -u root
-w itso05 -d /opt/IBM/ITMOSAgent
Deploy Of oslo.itsc.austin.ibm.com was SUCCESSFUL !
KUICCN001I Initializing required services...
KUICCN039I Attempting to connect to host oslo.itsc.austin.ibm.com ...
KUICCN050I Distributing file 51 of 51 (112.8 MB / 113 MB)...
KUICCN002I Beginning the installation and configuration process...

KUICCN065I The node creation operation was a success.
-------------------------------
SystemName: paris.itsc.austin.ibm.com
/opt/IBM/ITM/bin/tacmd createnode -h paris.itsc.austin.ibm.com -u root
-w itso05 -d /opt/IBM/ITMOSAgent
Deploy Of paris.itsc.austin.ibm.com was SUCCESSFUL !
KUICCN001I Initializing required services...
KUICCN039I Attempting to connect to host paris.itsc.austin.ibm.com ...
KUICCN050I Distributing file 54 of 54 (131.4 MB / 131.5 MB)...
KUICCN002I Beginning the installation and configuration process...

KUICCN065I The node creation operation was a success.
-------------------------------
SystemName: prague.itsc.austin.ibm.com
/opt/IBM/ITM/bin/tacmd createnode -h prague.itsc.austin.ibm.com -u root
-w itso05 -d /opt/IBM/ITMOSAgent
```

```
Deploy Of prague.itsc.austin.ibm.com was SUCCESSFUL !
KUICCN001I Initializing required services...
KUICCN039I Attempting to connect to host prague.itsc.austin.ibm.com ...
KUICCN050I Distributing file 51 of 51 (112.8 MB / 113 MB)...
KUICCN002I Beginning the installation and configuration process...

KUICCN065I The node creation operation was a success.
###############################

################ Results #################
Successful Deployment:    3
UnSuccessful Deployment:  0

--------- SuccessfulAgents ---------
oslo.itsc.austin.ibm.com
paris.itsc.austin.ibm.com
prague.itsc.austin.ibm.com

--------- FailedAgents ---------

################ ******* ################
######################
Exiting Tool
Mon Jul 16 12:47:36 CDT 2007
######################
```

Users can rerun the tool with the `-c` option again with the same input file, because this file is updated by the executing scripts so that successfully deployed systems are removed from the missing agents report. You may also want to rerun the script with the `-d` option again so that the known environment and reports are fully up-to-date regarding managed system status.

12.2.7 Using the TADDM reporting feature to assess coverage

As an alternative to the scripted scenario described here, you can use the built-in reporting feature of TADDM to display the monitoring coverage as a generated report displaying clearly which systems currently are monitored and which are not.

TADDM Version 5.1 introduced the Monitoring Coverage Report tool designed specifically for IBM Tivoli Monitoring users to aid the process of assessing monitoring coverage.

The process for using this tool is similar to the process described previously. However, instead of running the scripts described in 12.2.5, “Query the TADDM database” on page 431, this time we ran the TADDM reporting tool to generate our output.

Follow these steps to use the TADDM reporting tool to generate output:

1. Run through instructions given in 12.2.2, “Using TADDM to discover all systems on the network” on page 425 through 12.2.4, “Upload generated XML file to TADDM” on page 430, if you have not already done so.
2. Using a Web browser, connect to the TADDM host machine and launch the Domain Manager.
3. After logging in, select **Monitoring Coverage Report** under the Reports tab in the left-side navigation pane as shown in Figure 12-5. (Note that the report may take several minutes to create, depending on the size of your environment.)

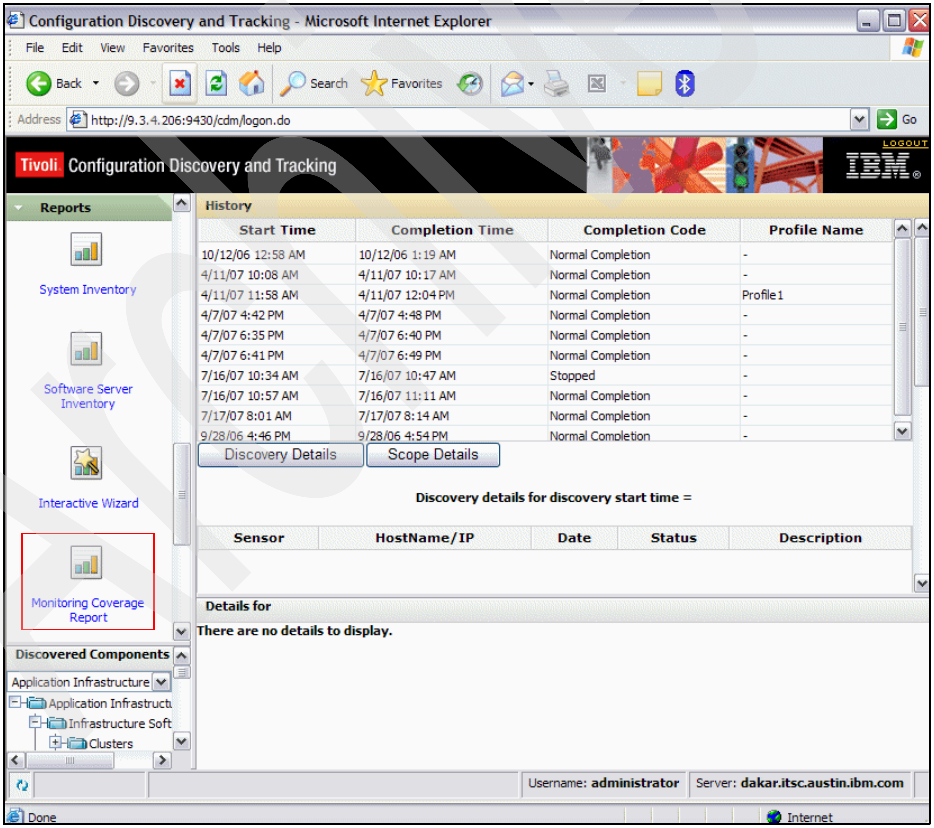


Figure 12-5 TADDM Domain Manager

The generated report contains the following sections:

- Table of contents

  These are a series of links that will take you to the monitoring coverage summary of the environment and to each Management Software System section created for this environment report. There will be a separate section for each portal server from which data has be acquired.

- Management Software System (MSS) reports

  An MSS is created for each portal server instance you have uploaded data to TADDM about, that is, each portal server that has had the KfwTmsDla script run against it. Each MSS provides details about every managed system known to that monitoring server and monitoring agents available on each machine, as well as the version number and affinity strings for those agents.

  The following information is given for every MSS:

Product Name	IBM Tivoli Monitoring Services
Product Version	The version of IBM Tivoli Monitoring
MSS Name	The name by which TADDM uniquely identifies this MSS
GUID	A generated globally unique ID of the MSS
Access Info	A URL pointing to where the user interfaces residing on the portal server can be accessed

  In each MSS in the Monitoring Coverage Report, TADDM will display a list of the IBM Tivoli Monitoring agents deployed in the environment. The following information is provided:

Product Code	The two-letter IBM Tivoli Monitoring product code for the agent (for example, NT for Windows, LZ for Linux)
Managed System Name	The IBM Tivoli Monitoring Managed System name for the agent
Affinity	The affinity string code corresponding to the product code.
Software Version	The version of the agent bundle installed.
Source Token	A URL to launch the browser-based portal console to the workspace for the agent

- Coverage summary section

  This is an easy-to-read view of databases and operating systems which are known to TADDM. Alongside each system is a color-coded indicator of the monitoring status.

– Systems where TADDM could not determine a monitoring relationship are indicated in red with the label `Unmonitored`.

– For systems where a monitoring relationship could be determined, the type of monitoring agent is displayed.

– For further details about the listed operating systems and databases, click the URL in the left column to display a detailed report for panel.

Figure 12-6 shows an example of a generated monitoring coverage report. From this a user can make a decision about the appropriate actions to take, such as employing the scripts listed in 12.2.6, “Deploy new IBM Tivoli Monitoring OS agents” on page 435 to deploy missing agents.

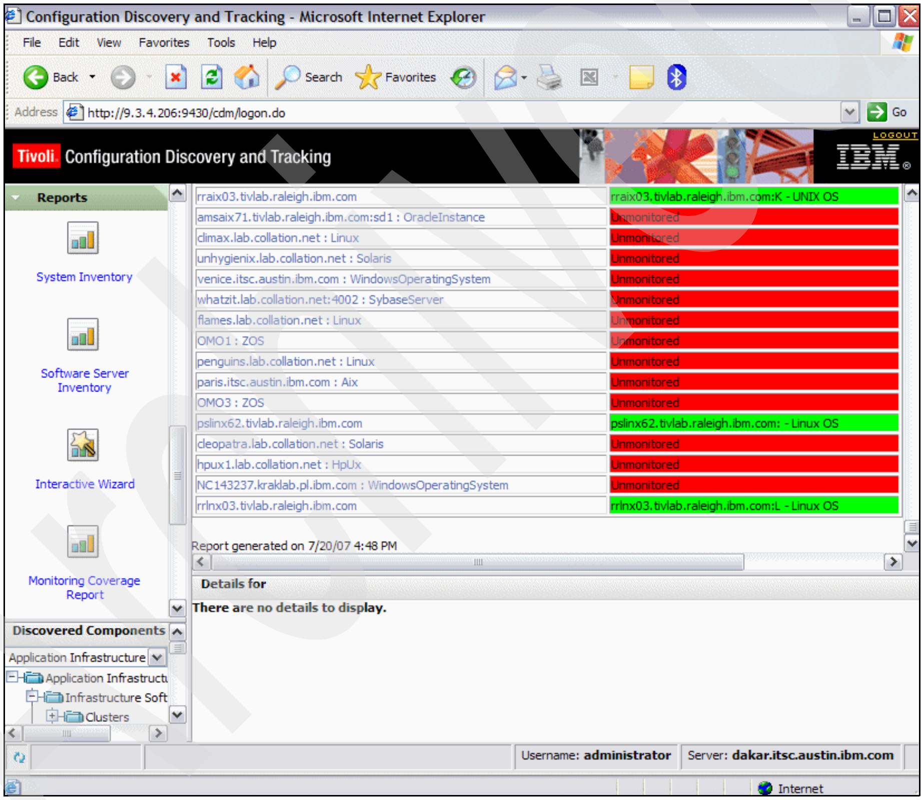


Figure 12-6 Monitoring coverage report for IBM Tivoli Monitoring environment

12.3 Using Tivoli Application Dependency Discovery Manager and IBM Tivoli Monitoring to facilitate server consolidation

This section describe a scenario that uses the system discovery capabilities of TADDM combined with a script to identify the dependencies between systems installed in the managed environment. In addition, the tool described identifies underutilized servers.

After server utilization levels and dependencies are identified, then judgments can be made based on reliable evidence for server consolidation and more efficient use of available resources.

This scenario can be of use to customers with a large number of resources available that they may wish to rationalize after a period of change. The consequences of making changes need to be fully understood, because you will want to shut down servers with no or very few dependencies. Without understanding the dependencies, incorrect choices could be made on server consolidation that could affect other business services.

12.3.1 Scenario background

In most enterprises, system administrators are faced with the challenge of using available resources as efficiently as possible. This may require server consolidation, relocation and rationalization of a large environment where the consequences of change may not be not fully understood. Shutting down servers may affect several other business services that have a dependency on the modified system.

The scenario environment presented here is similar to that described in 12.2.1, "Scenario background" on page 424. It includes a number of physical systems based on several operating systems (Windows, Linux, UNIX). The aim is to show the utilization of the current server and dependencies within this environment. Figure 12-7 on page 442 shows the overview of this scenario.

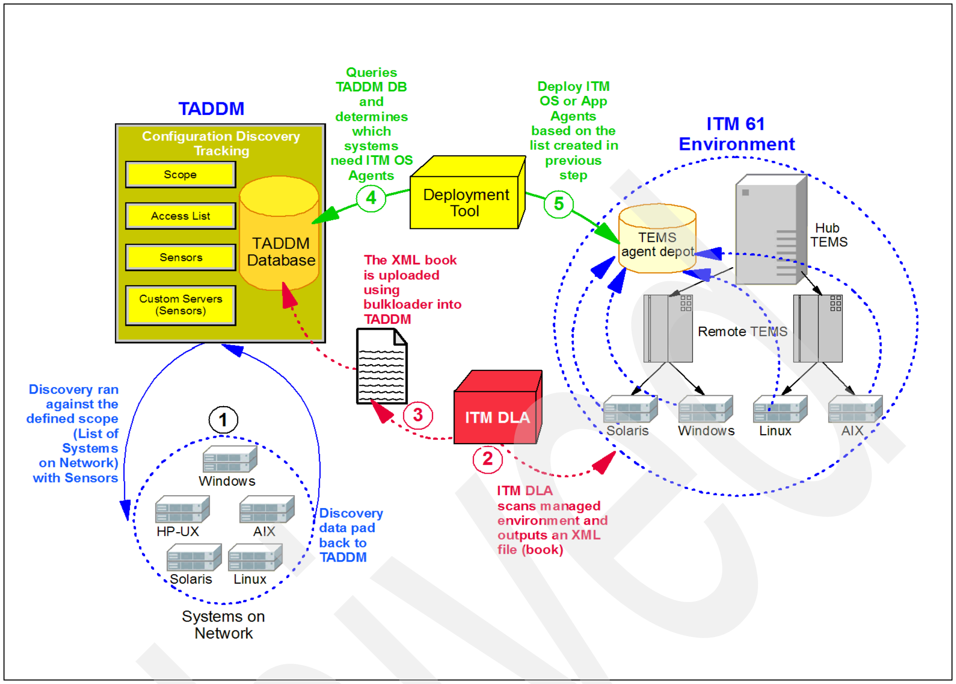


Figure 12-7 Scenario overview of IBM Tivoli Monitoring and TADDM integration

In a much larger customer environment, the means of identifying dependency and utilization rate becomes much harder to do. By using TADDM and some scripts we were able to create reports from which system administrators can take appropriate action to provide server consolidation with minimal negative impact to the overall environment.

12.3.2 Use TADDM to discover all systems on the network

As in 12.2, “Using Tivoli Application Dependency Discovery Manager and IBM Tivoli Monitoring to discover and deploy to unmanaged systems” on page 423, we used the discovery process within TADDM to establish a comprehensive list of the physical systems within the environment. This will form the basis of the dependency analysis between various systems.

Use the same scope and criteria as specified in 12.2.2, “Using TADDM to discover all systems on the network” on page 425.

12.3.3 Collect historical data for CPU utilization

To make an assessment on the utilization rate of each managed system, we need to make use of the CPU utilizations as recorded by the OS monitoring agent. This requires us to choose these attribute groups for historical collection and, if desired, summarization.

Depending on the operating systems being monitored, choose the appropriate attribute group for that particular agent.

- Windows

  We configured the attribute NT_Processor for this task because it contains the attribute % Processor Time.

- UNIX

  The SMP_CPU attribute group contains appropriate attributes for CPU utilization.

- Linux

  The CPU_Averages attribute group contains appropriate attributes for CPU utilization analysis.

- z/OS

  OMEGAMON XE on z/OS provides an attribute group System_CPU_Utilization for this task.

The sampling intervals should be chosen as appropriate for your environment. Although the task of evaluating CPU utilization will be assessed over a long time period (perhaps quaterly or annually), you will want to be aware of changes in usage throughout the day. Because if no sample is taken during a moment of peak usage, then a false impression may be created of the system's importance to your environment.

We recommend that you take regular samples throughout the day and that you use the Summarization and Pruning agent to aggregate the data across a longer time period to establish a realistic assessment. In our scenario, we configured the Summerization and Pruning agent to aggregate data from our Windows OS agent over a quarterly period.

12.3.4 Identify dependencies and utilization of systems

When the TADDM discovery process completed, we used a script developed for the purpose of producing a report detailing all the dependencies and utilization of the managed systems.

In total, the tool performs three tasks:

1. It makes a connection to TADDM to retrieve an XML report showing the dependency details for each system.
2. It retrieves the appropriate data for CPU utilization from the Tivoli Data Warehouse through a query.
3. It parses the information collected in the previous steps to produce the report showing underutilized systems along with their dependencies by matching the Tivoli Data Warehouse with that from TADDM.

To download the tool, go to the IBM Tivoli Open Process Automation Library (OPAL) Web site. Search under "ITM TADDM dependencies for under utilized systems".

The tool is provided as a zip or tar file, depending on your operating system. The file should be uncompressed on the machine running the hub monitoring server. After uncompression, the following subdirectories will be created:

/etc This contains the files log4j.xml, policy.all and collation.properties. The collation.properties file needs to be copied from the TADDAM machine located under $COLLATION_HOME/dist/etc.

/external This contains a Java runtime for use by the tool. Note that the Java files are *not* provided in the downloaded image. You should copy the JRE as supplied by IBM Tivoli Monitoring into this folder.

/lib This contains source files as used by the tool. Note that these files are *not* provided in the downloaded image. You should copy the following files from subdirectories under the TADDM install:

api-client.jar, api-dep.jar and platform-model.jar from dist/lib

xalan.jar from dist/sdk/lib

commons-jxpath-1.1.jar from dist/deploy-tomcat/cdm/WEB-INF/lib

/log This contains the log files produced by the tool when making calls to TADDM.

/results This contains output from running the scripts.

/scripts This contains the script files used.

The scripts for this tool are written in Perl, so Perl should be installed on the system. Specifically, you need to install the Perl modules (*.pm) for NSLookup, DNC, and IP.

> **Note:** In order to keep the download size smaller, several required files are not provided within the compressed file, so you need to copy the appropriate files as detailed.

The scripts must be edited prior to use in order to set the variables as appropriate for your environment. These variables will be used by the script to log on to the TADDM database and the TDW. In our scenario, we edited the script **run.pl** to the settings shown in Example 12-7.

Example 12-7 Variables defined in run.pl

```
$db2_datasource="WAREHOUS";
$db2_user="ITMUSER";
$db2_passwd="password";
$taddmhost="CDT511";
$taddmid="administrator";
$taddmpass="password";

$ENV{'TOOL_HOME'} = '/tmp/taddm_tool';
```

The default behavior of the script is to query the TDW database for entries in the table “NT_Processor_Q” (quarterly) that have a value of less than 10%. Modify the $sql variable to suit your environment and utilization level. The variable is defined as an SQL statement. Modify the line shown in Example 12-8.

Example 12-8 SQL statement in run.pl

```
$sql="select distinct \"Server_Name\" from \"NT_Processor_Q\" where
\"AVG_%_Processor_Time\" < 10";
```

The script is run from the command line. It has no arguments to supply at runtime:

```
perl run.pl
```

13

Troubleshooting

This chapter introduces the troubleshooting components of the IBM Tivoli Monitoring V6.1 product set.

The following topics are addressed:

- Log files directory
- General terms, extensions, and product code
- Log file names and their usages
- Identifying the latest log file
- How to enable the tracing
- Using DIGUP to retrieve UNIX log files
- Using KINLOGS to format log file time stamps
- Common configuration problems

13.1 Log files directory

In this topic we introduce the log files directory, where the log file is created for each component for the following platforms:

- Windows
- UNIX and Linux
- OS/400®
- z/OS

13.1.1 Windows

The Windows directory for logs:

- OS Agent: x:\IBM\ITM\TMAITM6\logs
- Monitoring server, portal server: x:\IBM\ITM\logs
- Portal: x:\IBM\ITM\CNP\LOGS

13.1.2 UNIX and Linux

The UNIX and Linux directory for logs:

- OS Agent

13.1.3 OS/400

OS/400 directory for logs:

- OS Agent: The log is found in the QAUTOTEMP library.

13.1.4 z/OS

Follow this process to obtain the log from a z/OS monitoring server. Note that this process requires an interactive session with the z/OS system.

- The name of the monitoring server process is CANSDSST. To view the current log file, put a question mark (?) in the prefix area to the left of the process name and click **Return**.
- The log file for the z/OS monitoring server is called RKLVLOG. To read it, enter `S` in the prefix area and press **Return**.

13.2 General terms, extensions, and product code

This section explains the terminology used in the file names and the extension. This will give you a better understanding of the log file names.

13.2.1 Log terms

The following list summarizes the log terms:

Time stamp	An 8-character hexadecimal time stamp that represents the time at which the program started.
program	Agent, collector and server process names and Take Action command names.
nn	The rolling log suffix, starts with 01 then 02 and 03.
nnnnn	The process id number (pid).
product	The two-digit product code of an agent or server.

When a trace log having a .log extension wraps (or reaches the maximum size), it is given a .lg1 extension and a new .log file is created. Similarly, a trace log having a .out extension that wraps is given a .ou1 extension, and a trace log having a .LG0 extension that wraps is given a .LG1 extension.

> **Note:** For a complete list of product codes, refer to Appendix A, “Product codes” on page 471.

13.3 Log file names and their usages

This section describes the log file name for each monitoring agent available in IBM Tivoli Monitoring V6.1, as well as for components S&P, Warehouse Agent, portal server, and monitoring server. A brief explanation about their usage is also given.

MS SQL agent (Product Code: oq) log files

- hostname_oq_program_timestamp-nn.log

RAS1 logs containing the activity of the monitoring agent or program. Other programs are koqerr, koqsql, and KOQCOLL.

- hostname_OQ_Instance_col.log.

MS SQL agent collector trace log showing communication with the MS SQL Instance, SQL commands executed, and the data returned from the Instance.

- hostname_OQ_OracleInstance_agt.log

MS SQL agent trace log.

- Instance:hostname:MSS.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server for the MS SQL agent.

Oracle agent (Product Code: or) log files

- hostname_or_program_timestamp-nn.log

RAS1 log containing traces of the activity of the monitoring agent. Other programs are ko9coll and ko9sql. For the ko9coll and ko9sql programs on Windows, the or is replaced as follows: Oracle 8i - O8, Oracle 9 - O9 and Oracle 10g - OR.

- hostname_or_OracleInstance_col.log

Oracle agent collector trace log. On UNIX there is a hostname_or_OracleInstnace_col.out file that contains stderr and stdout messages from the collector process. On Windows, the or is replaced as follows: Oracle 8i - O8, Oracle 9 - O9, and Oracle 10g - OR.

- hostname_or_OracleInstnace_agt.log

Oracle agent trace log. On UNIX there is a hostname_or_OracleInstance_agt.out file that contains stderr and stdout messages from the agent process.

- OracleInstance:hostname:ORA.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server for Oracle agent.

Sybase agent (Product Code: oy) log files

- hostname_oy_program_timestamp-nn.log

RAS1 log containing traces of the activity of the monitoring agent.

- hostname_oy_adaptiveservername_col.log

Sybase agent collector trace log. On UNIX there is a hostname_oy_adaptiveservername_col.out file that contains stderr and stdout messages from the collector process.

- hostname_oy_adaptiveservername_agt.log

Sybase monitoring agent trace log. On UNIX there is a hostname_oy_adaptiveservername_agt.out file that contains stderr and stdout messages from the agent process.

- Adaptiveservername:hostname:SYB.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server for Sybase agent.

DB2 agent (Product Code: ud) log files

- hostname_ud_DB2Instance.log

Log file created at the start of the DB2 agent. Typically, this log is empty.

- hostname_ud_program_timestamp-nn.log

RAS1 trace log of the activity of the monitoring agent including the metrics collected from the DB2 Instance being monitored. The log from the kuddb2 program are the main logs of the DB2 agent.

- DB2Instance:hostname:KUD.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

Microsoft Exchange agent (Product Code: ex) log files

- hostname_ex_program_timestamp-nn.log

Shows the startup and shutdown of the Microsoft Exchange agent. RAS1 log containing traces of the activity of the monitoring agent.

- hostname_EX.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

ADO (Active Directory Option) agent (Product Code: 3z) log files

- hostname_3z_program_timestamp-nn.log

Shows the startup and shutdown of the Active Directory agent. RAS1 log containing traces of the activity of the monitoring agent.

- hostname_3Z.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

Windows OS agent (Product Code: nt) log files

- hostname_nt_program_timestamp-nn.log

Shows the startup and shutdown of Windows OS agent. RAS1 log containing traces of the activity of the monitoring agent.

- Primary_hostname_NT.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

UNIX OS Agent (Product Code: ux) log files

- hostname_ux_timestamp.log

Shows the startup and shutdown of the UNIX OS agent.

- hostname_ux_timestamp.pidnnnnn

Lists the process id (pid) of the kuxagent process. The kuxagent process will start all the other UNIX OS agent processes.

- hostname_ux_program_timestamp-nn.log

RAS1 log containing traces of the activity of the monitoring agent. Common program names of the monitoring processes are kux_vmstat, stat_daemon, ifstat and kuxagent.

- hostname:KUX.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

UNIX log agent (Product Code: ul) log files

- hostname_ul_timestamp.log

Shows the startup and shutdown of the log agent.

- hostname_ul_timestamp.pidnnnnn

Lists the process id (pid) of the kulagent process.

- hostname_ul_program_timestamp-nn.log

RAS1 log containing traces of the activity of the monitoring agent.

- hostname:KUL.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

Linux OS agent (Product Code: lz) log files

- hostname_lz_timestamp.log

Shows the startup and shutdown of the Linux OS agent.

- hostname_lz_timestamp.pidnnnnn

Lists the process id (pid) of the klzagent process.

- hostname_lz_program_timestamp-nn.log

RAS1 log containing traces of the activity of the monitoring agent.

- hostname:LZ.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

I5/OS (OS/400) agent (Product Code: 4a) log files

- KA4AGENTNN

By default this log is set to wrap and will create the three trace logs KA4AGENT01, KA4AGENT02, and KA4AGENT03.

- DSPOMALOG

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

Universal Agent (Product Code: um) log files

- hostname_um_timestamp.log

Shows the startup and shutdown of the agent.

- hostname_um_timestamp.pidnnnnn

Lists the process id (pid) of the universal agent process.

- hostname_um_program_timestamp-nn.log

RAS1 log containing traces of the activity of the monitoring agent.

- hostname:UA.LG0

Shows status of Take Action commands, situations that were started, and connection to monitoring server.

Summarization and Pruning agent (Product Code: sy) logfiles

The Summarization and Pruning agent log files are listed here:

- hostname_sy_timestamp-nn.log

RAS1 log contains the traces of the Summarization and Prunning Agent.

- hostname_sy_java_timestamp-nn.log

Trace of the java process used by the Summarization and Prunning Agent.

Warehouse Proxy agent (Product Code: hd) log files

- hostname_hd_timestamp-nn.log

RAS1 log contains the traces of the Warehouse Proxy agent.

- hostname_Warehouse.LG0

Shows the startup and shutdown of the Warehouse Proxy agent.

Tivoli Portal Server (Product Code: cq) log files

- hostname_cq_timestamp.log

Shows the startup and shutdown of the Tivoli Enterprise Portal Server process.

- hostname_cq_timestamp.pidnnnnn

Lists the process id (pid) of the Tivoli Enterprise Portal Server on UNIX and Linux.

- hostname_cq_timestamp-nn.log

RAS1 log containing traces of the activity of the portal server process.

Tivoli Enterprise Monitoring Server (Product Code: ms) log files

- hostname_ms_timestamp.log

Shows the startup and shutdown of the main Tivoli Enterprise Monitoring Server process.

- hostname_ms_timestamp.pidnnnnn

On UNIX and Linux lists the process id (pid) of the Tivoli Enterprise Monitoring Server.

- hostname_ms_timestamp-nn.log

RAS1 log containing traces of the activity of the Tivoli Enterprise Monitoring Server process.

- hostname_cms_timestamp.log

Shows the startup and shutdown of the Tivoli Enterprise Monitoring Server cms process.

13.4 Identifying the latest log file

Because of the log-rolling mechanism, it can be difficult to determine which is the most recent log file. This section explains how to identify the latest log file generated.

The most recent log file can be determined by consulting the log file information file, using the following format:

<hostname>_<code>.inv

Example 13-1 shows a list of log files for a portal server on Linux.

Example 13-1 List of log files

```
-rw-rw-rw-   1 root     root    8297575 Oct 29 04:30
giskard_cq_43615bfc-01.log
-rw-r-----   1 root     root    8268895 Oct 29 12:28
giskard_cq_43615bfc-02.log
-rw-r-----   1 root     root    8286766 Oct 29 12:51
giskard_cq_43615bfc-03.log
-rw-r-----   1 root     root    8298932 Oct 29 13:29
giskard_cq_43615bfc-04.log
-rw-r-----   1 root     root    3982541 Oct 29 13:45
giskard_cq_43615bfc-05.log
-rw-rw-rw-   1 root     root    2212051 Oct 29 13:45
giskard_cq_1130454011.log
-rw-rw-rw-   1 root     root        376 Oct 30 18:06 giskard_cq.inv
-rw-rw-rw-   1 root     root     106654 Nov  3 12:13
giskard_cq_43655ff0-01.log

In the TEPS log information file, the top entry is the one currently
being written to:
giskard:/data/RTMBuild/logs # more giskard_cq.inv
/data/RTMBuild/logs/giskard_cq_43655ff0-01.log
/data/RTMBuild/logs/giskard_cq_43615bfc-05.log
/data/RTMBuild/logs/giskard_cq_43615bfc-04.log
/data/RTMBuild/logs/giskard_cq_43615bfc-03.log
/data/RTMBuild/logs/giskard_cq_43615bfc-02.log
/data/RTMBuild/logs/giskard_cq_43615bfc-01.log
```

13.5 How to enable tracing

When investigating a problem with a particular component, there are two ways to turn up the tracing level for diagnosis:

- Manage Tivoli Enterprise Monitoring Services
- IBM Tivoli Monitoring Service Console

Manage Tivoli Enterprise Monitoring Services

To use the Manage Tivoli Enterprise Monitoring Services user interface to increase the trace level, follow these instructions.

Go to **Start** → **Programs** → **IBM Tivoli Monitoring** → **Manage Tivoli Enterprise Monitoring Services** → right-click in the **component** → **Advanced** → **Edit trace parms**. See Figure 13-1.

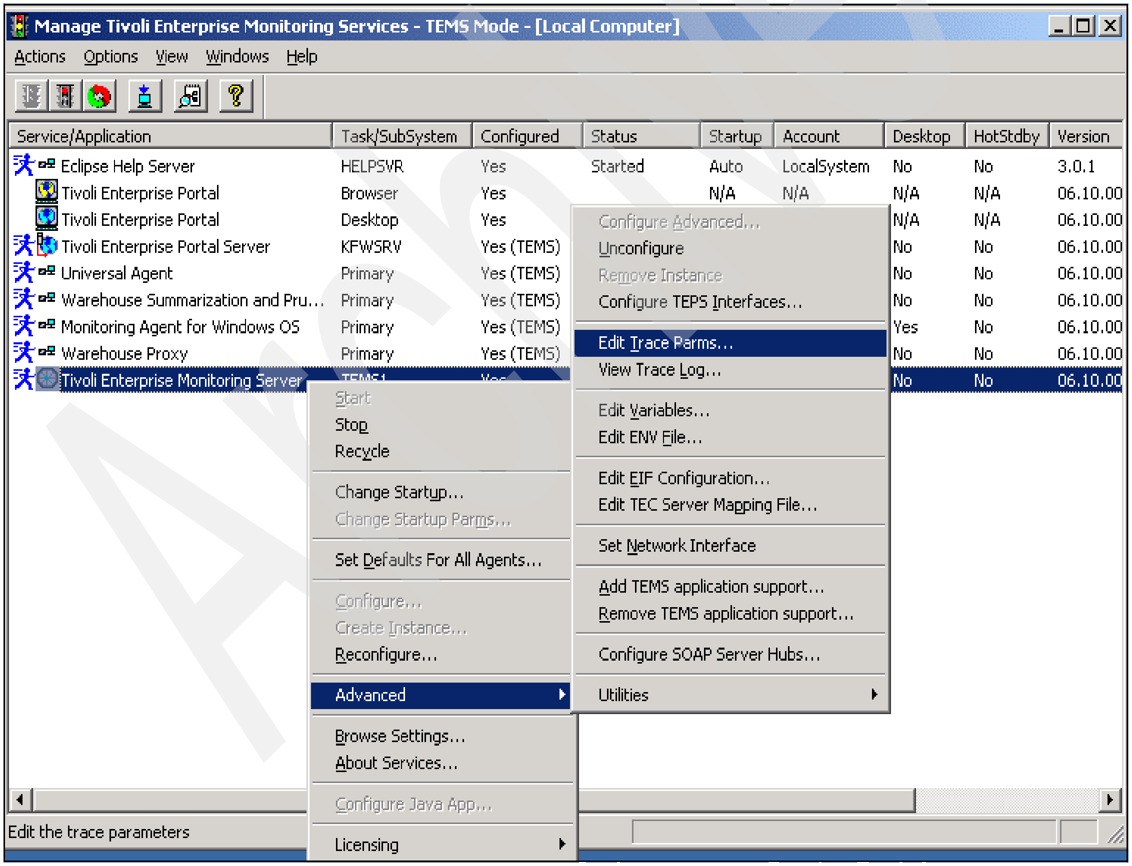


Figure 13-1 Manage Tivoli Enterprise Monitoring Services User Interface

Figure 13-2 shows the Tivoli Enterprise Monitoring Server: Trace Parameters panel. You can use this panel to specify the number and size of log files, and to select a trace level from the pull-down list.

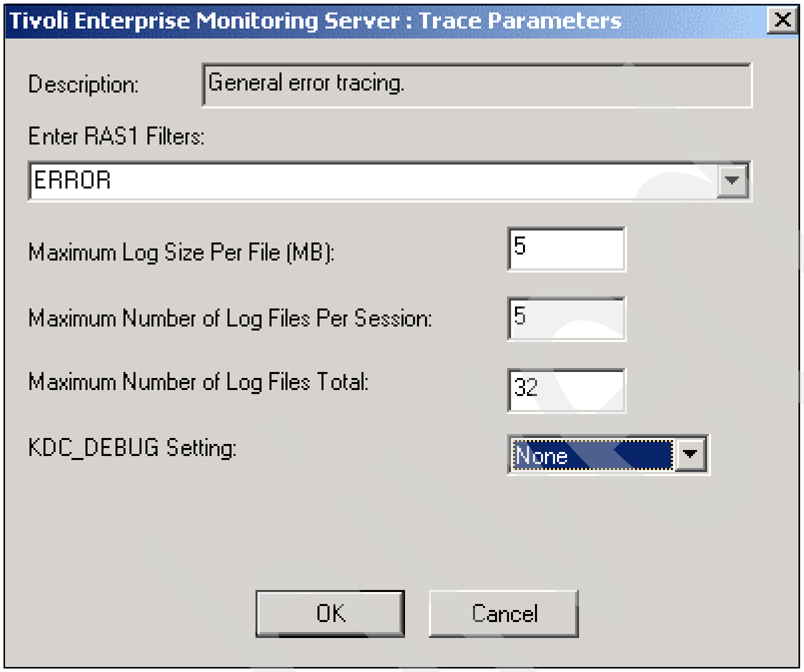


Figure 13-2 Edit trace parms

> **Note:** The tracing changes set here do not take effect until the component is restarted.

IBM Tivoli Monitoring Service Console

To use the IBM Tivoli Monitoring Service Console to increase the trace levels, use this information.

Each product installed on the system has a corresponding Service Console:

- SY = Summarization & Pruning
- UM = Universal Agent
- HD = Warehouse Proxy Agent
- NT = Windows OS Agent
- CNP = TEPS
- CMS = monitoring server

You can access the available service consoles for a given machine by pointing a Web browser at the following URL:

```
http://<hostname>:1920
```

Figure 13-3 shows the link to access the service console.

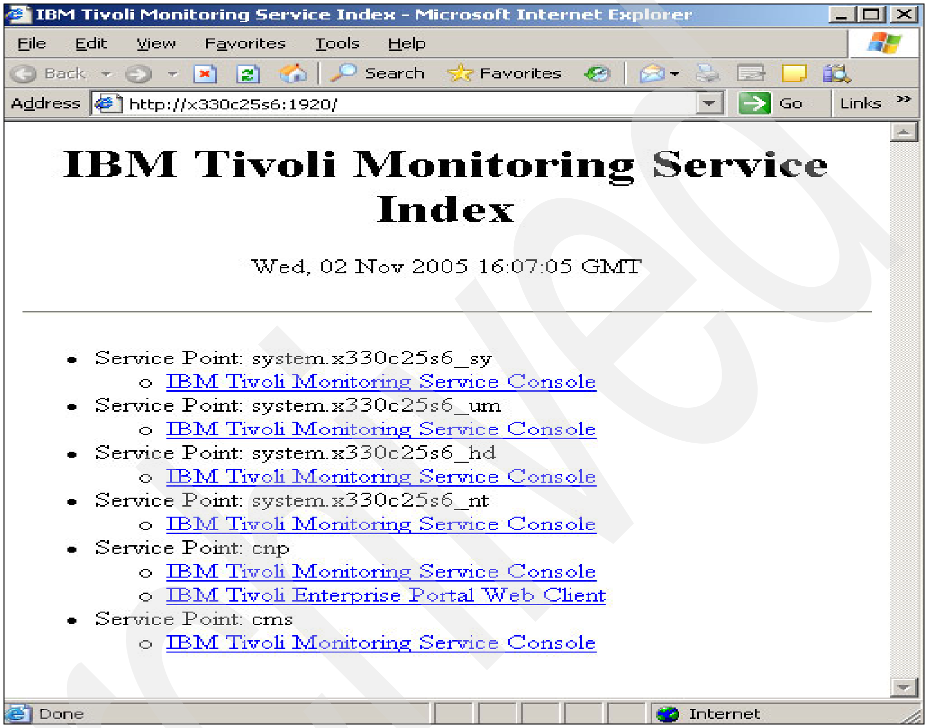


Figure 13-3 IBM Tivoli Monitoring Service Console

To log on to the service console for monitoring server, click one of the service console hyperlinks to bring up an authentication panel. This will utilize the operating system access and authentication credentials of the host system. See Figure 13-4 on page 459.

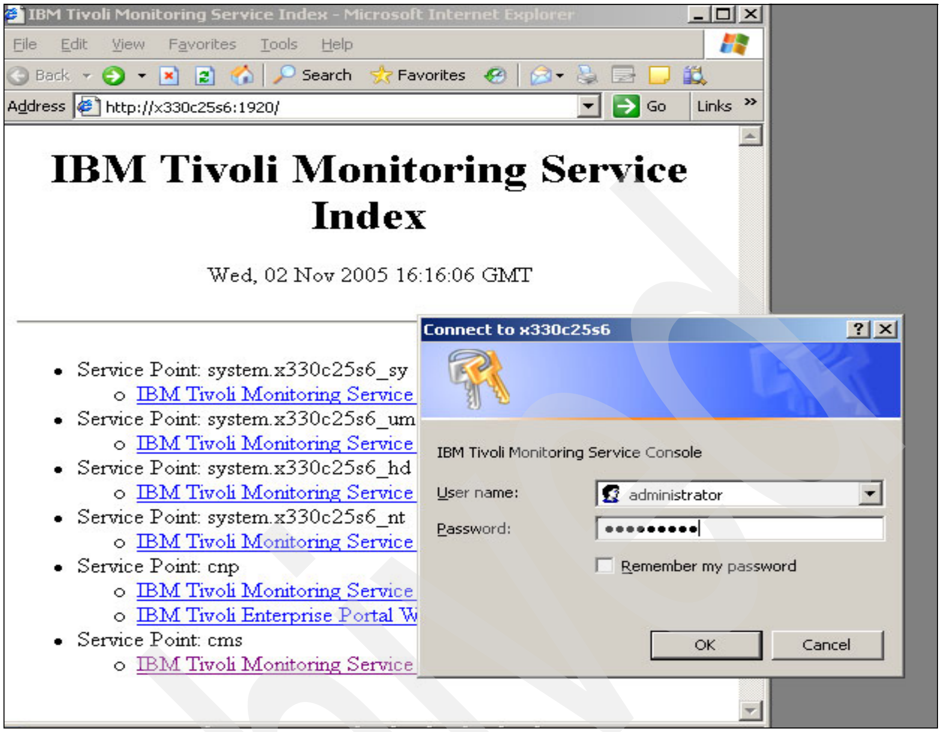


Figure 13-4 Logging on service console

Figure 13-5 on page 460 shows what the service console looks like after logging on. You can enter trace parameters in the text box at the bottom of the screen.

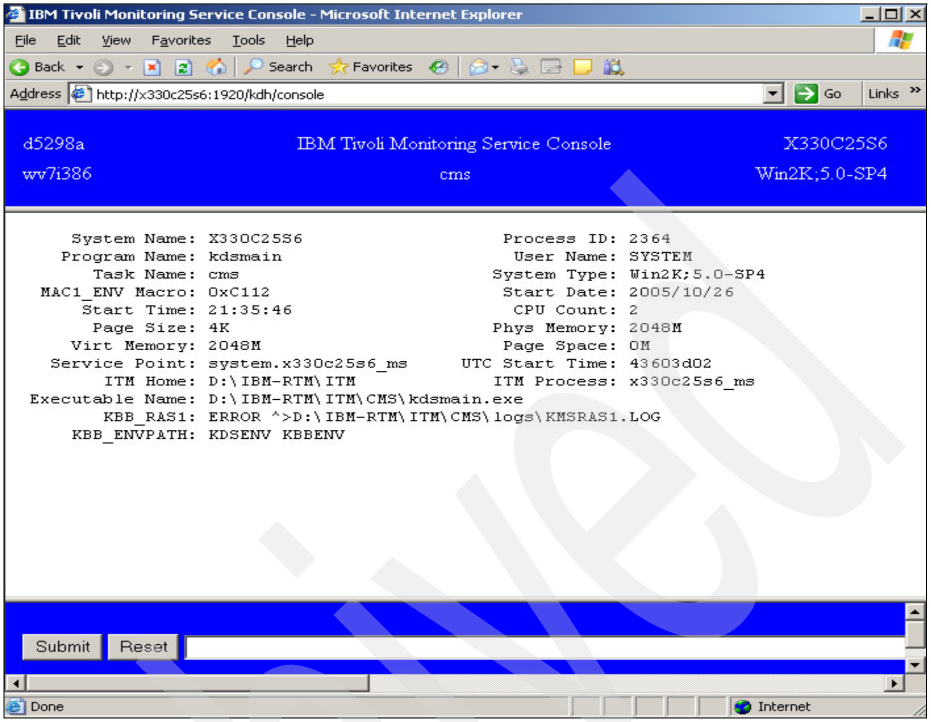


Figure 13-5 After logging

Figure 13-6 on page 461 is an example of a modified trace setting.

Note: The advantage of setting tracing in this manner is that it takes effect dynamically.

Figure 13-6 Modified trace level

The service console displays a confirmation of the revised trace setting, as shown in Figure 13-7 on page 462.

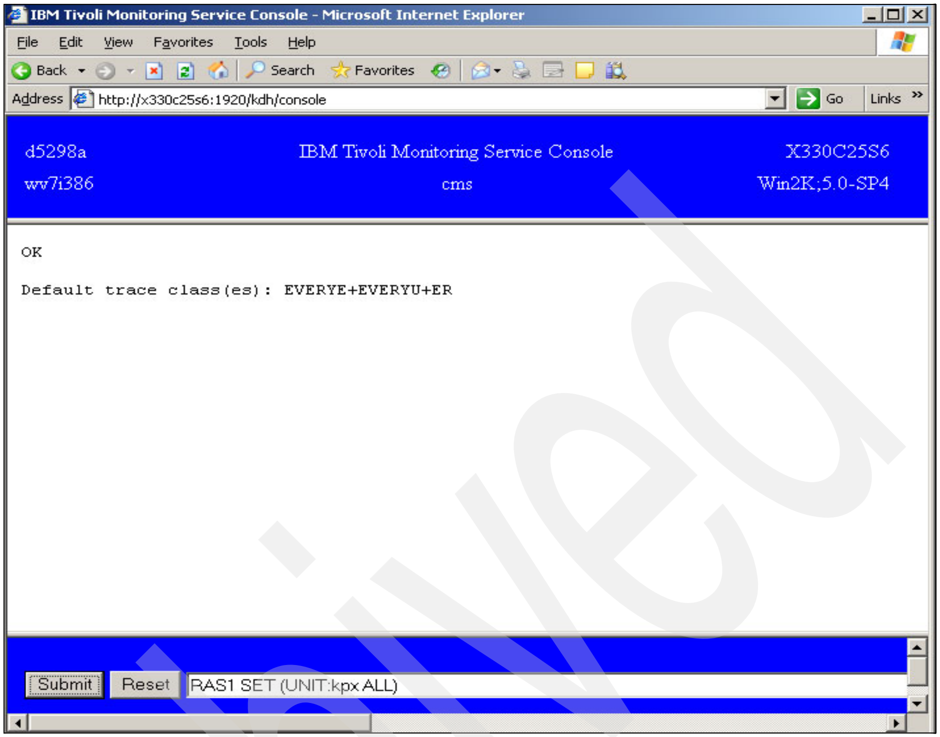


Figure 13-7 Confirmation

Use **LIST** to view currently enabled filters. If you want to see which trace levels are currently set, a **LIST** query will provide the answer. See Figure 13-8 on page 463.

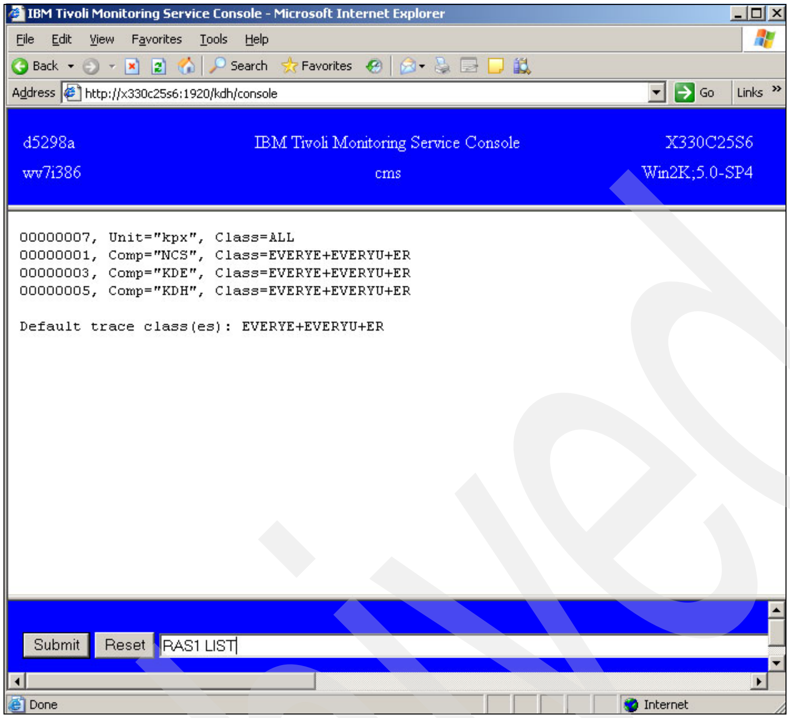


Figure 13-8 List command

13.6 Using DIGUP to retrieve UNIX log files

DIGUP is a tool (located in $ITMHOME/bin) that retrieves all log files from a targeted system and places them in a zipped tarred file under $ITMHOME/tmp directory. Use the following command to invoke DIGUP:

```
./digup –a
```

DIGUP tars the following directories under $ITMHOME:

- LOGS
- CONFIG
- REGISTRY

DIGUP also retrieves the following information:

- The listing of all files under $ITMHOME: itmhome.info
- The environment variables: env.info
- The system information: system.info
- The contents of hosts file: hosts.info

Example 13-2 shows sample DIGUP output.

Example 13-2 Sample output of DIGUP

```
sample output
bash-2.03# ./digup  -a
...Gathering directory information
...Gathering config directory
...Gathering registry directory
...Gathering log files
...Gathering /etc/hosts information
...Gathering environment information
...Gathering component information
...Creating zip file
********************************************************************
 /data/CNDL/tmp/DIG.tar.gz
This file has been created and contains diagnostic information.
Forward this binary file to your IBM Support representative.
********************************************************************
```

13.7 Using KINLOGS to format log file time stamps

The **kinlogs.exe** application formats time stamps into human-readable format for easier debugging. You can install KINLOGS with any Windows product, such as an OS agent. It is located at $ITMHOME/installITM/.

You do not need to have an IBM Tivoli Monitoring agent on your desktop in order to employ the KINLOGS application. The executable may be copied to any Windows system and used to read log files if the following DLLs are also copied and placed in the path:

- MFC71.DLL
- MSVCR71.DLL

Figure 13-9 on page 465 illustrates a sample KINLOGS session.

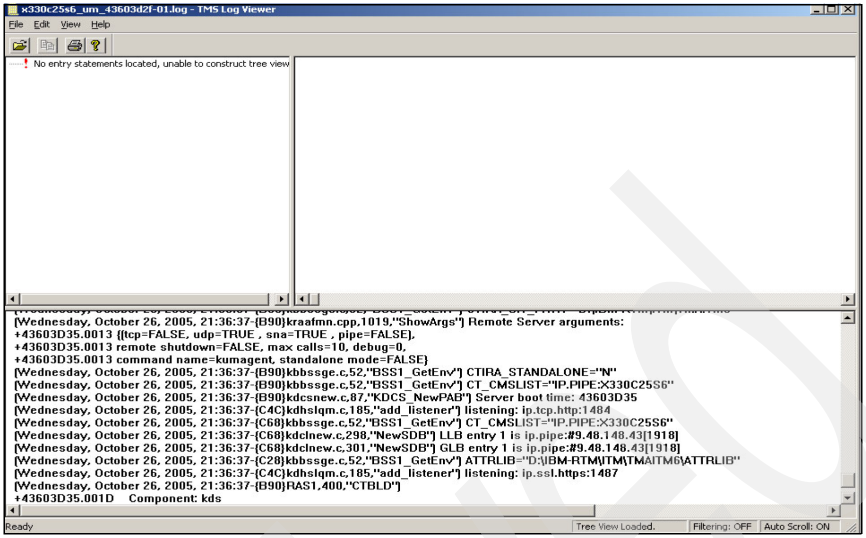


Figure 13-9 KINLOGS session

13.8 Common configuration problems

This section describes some of the more common configuration problems which are likely to be encountered by new customers doing initial deployments.

- Portal server logon failure
- Agent hostname length problem
- Agent/Server Communication Protocol Incompatibility
- Questions about the logging of Warehouse data

The section not only provides guidance about solving these particular problems, but also discusses a general problem-solving approach.

Portal server logon failure

The example shown in Figure 13-10 on page 466 shows a user logon failure.

Solution

The SYSADMIN user should have defined the user in the portal server using the Administer Users (Ctrl + U) panel. The user should be a system-defined user on the system hosting the monitoring server.

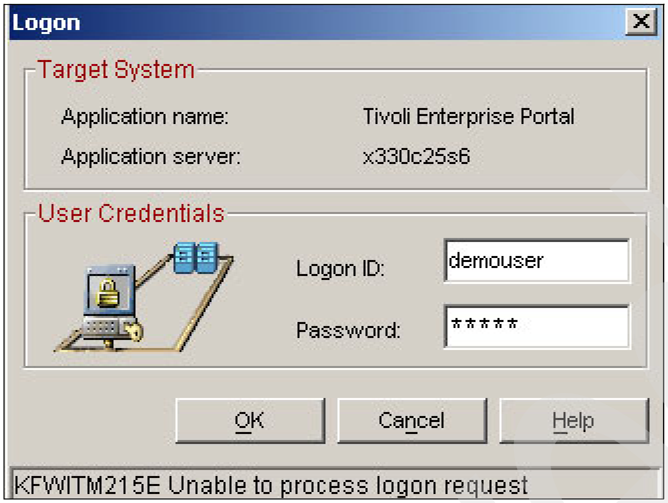


Figure 13-10 Portal server logon failure

If the user has not been defined on the monitoring server system, the portal server log will display an error message similar to Example 13-3.

Example 13-3 Portal server log

```
…Invalid Userid <demouser>
…
…EXCEPTION: CORBA User Exception has occurred
```

Hostname length problem

When agent systems are configured with a fully-qualified hostname, truncation of the agent name in the node status table can result. This is because the field is limited to 31 bits, and after adding the FQH, there may not be enough characters remaining to append the three-character identifier, such as kux for a UNIX OS agent.

In this scenario, only the k may be written, so when an additional agent is installed on that system (such as a Universal Agent), it too will only have the suffix k, and will not be successfully entered in the node status table.

The result will be an agent, which the user *thinks* was installed, not appearing in the portal server. Whichever agent was started first would appear in the portal server, and no others.

Solution

There is a simple manual workaround for this problem. Rather than asking users to reconfigure the machine to use a short hostname, you can simply add a CTIRA_HOSTNAME value to the configuration file for each of the agents on the system.

Example 13-3 on page 466 illustrates an agent name-length collision. This agent system has been configured with a fully-qualified hostname. The Universal Agent has been configured and started, but note that the value (which is highlighted) in the Name column contains only the letter K as a suffix.

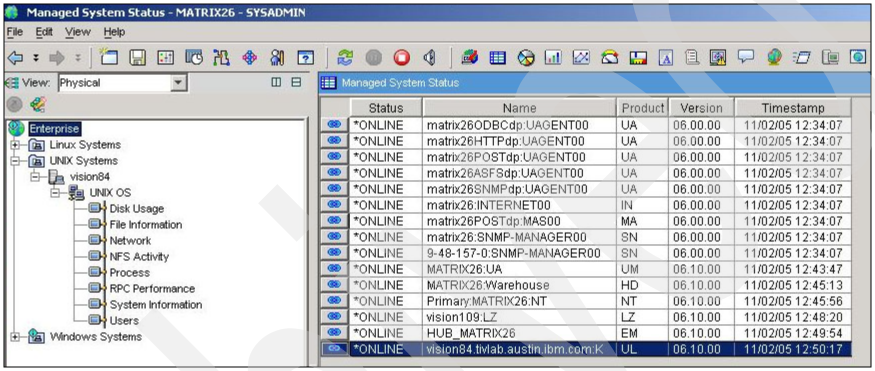


Figure 13-11 Example of agent name-length collision

To resolve the name-length collision problem, we edited the ux.config and ul.config files on the agent system, adding the following line to both:

```
CTIRA_HOSTNAME='short-hostname'
```

Then we restarted the agents with a `-c` flag (to preserve the configuration file. Subsequently both agents appeared in the status table, as shown in Figure 13-12.

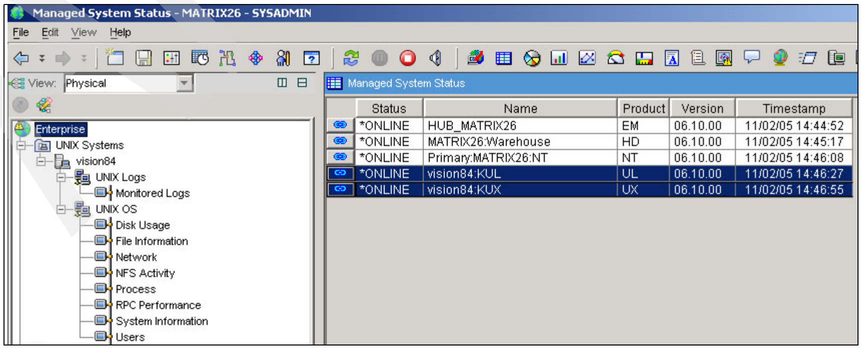


Figure 13-12 Both agents appear in the status table

Alternate solution

There is an alternate workaround for the hostname length issue. You can avoid the 32-bit hostname-length limitation by adding the following line to an agent's .ini file:

```
CTIRA_HOSTNAME=$RUNNINGHOSTNAME$
```

Some of the agents are shipped with this text already placed into their default .ini files. Having it in this file eliminates the need to always start the agent with the `-c` flag (to avoid regenerating the configuration file).

Agents/Server protocol incompatibility

Agents commonly fail to communicate with servers due to communication protocol configuration errors. Agents must use a protocol that the server has been configured to respond to.

In Example 13-4, the server has been configured to respond to IP.SPIPE communication, but the agent has been configured for IP.PIPE.

Example 13-4 Agents/Server protocol incompatibility

```
(4368F15B.0012-1:kbbssge.c,52,"BSS1_GetEnv")
CT_CMSLIST="ip.pipe:matrix26"
(4368F15B.0013-3:kbbssge.c,52,"BSS1_GetEnv")
CT_CMSLIST="ip.pipe:matrix26"
(4368F15B.0014-3:kdclnew.c,298,"NewSDB") LLB entry 1 is
ip.pipe:#9.48.133.109[1918]
(4368F15B.0015-3:kdclnew.c,301,"NewSDB") GLB entry 1 is
ip.pipe:#9.48.133.109[1918]
(4368F15B.0016-3:kdcc1sr.c,984,"rpc__sar") Endpoint unavailable:
"ip.pipe:#9.48.157.26[1918]", 0, 5(2), FFFF/0, 1.1.1.1, d5298a
(4368F15B.0017-3:kdcl0cl.c,116,"KDCL0_ClientLookup") status=1c020006,
"location server unavailable", ncs/KDC1_STC_SERVER_UNAVAILABLE
(4368F15B.0018-3:kraarreg.cpp,1241,"LookupProxy") Unable to connect to
broker at ip.pipe:matrix26.tivlab.austin.ibm.com: status=0, "success",
ncs/KDC1_STC_OK
(4368F15B.0019-3:kraarreg.cpp,1486,"FindProxyUsingLocalLookup") Unable
to find running CMS on CT_CMSLIST <ip.pipe:matrix26>
```

Solution

This problem can be resolved by reconfiguring the agent to use the IP.SPIPE protocol.

Confirming that warehouse data is being logged

To determine whether data is being logged for a particular agent, simply view the warehouse log file, which shows all of the rows of data loaded into the warehouse, as well as their source.

From the Navigator, right-click **Enterprise** and select the **Managed System Status** view. Then right-click the **Warehouse Proxy Agent** and select **Agent Operations Log** (see Figure 13-13).

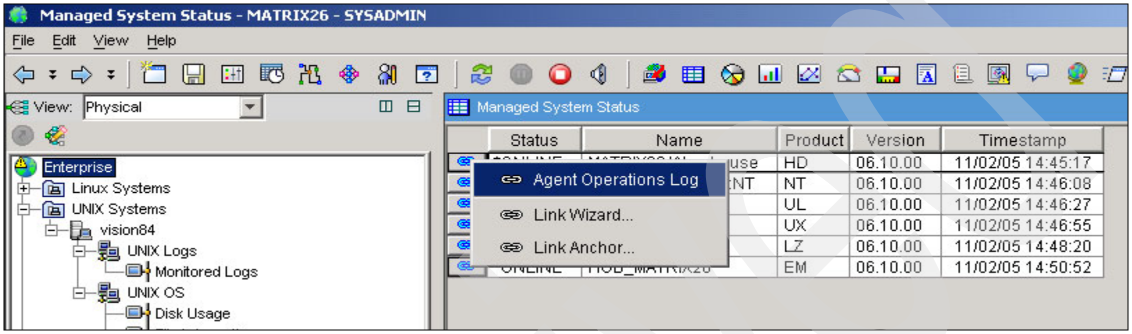


Figure 13-13 Agent operations log

You can view the status of agent data warehousing, as shown in Figure 13-14, and confirm that the warehouse data is being logged.

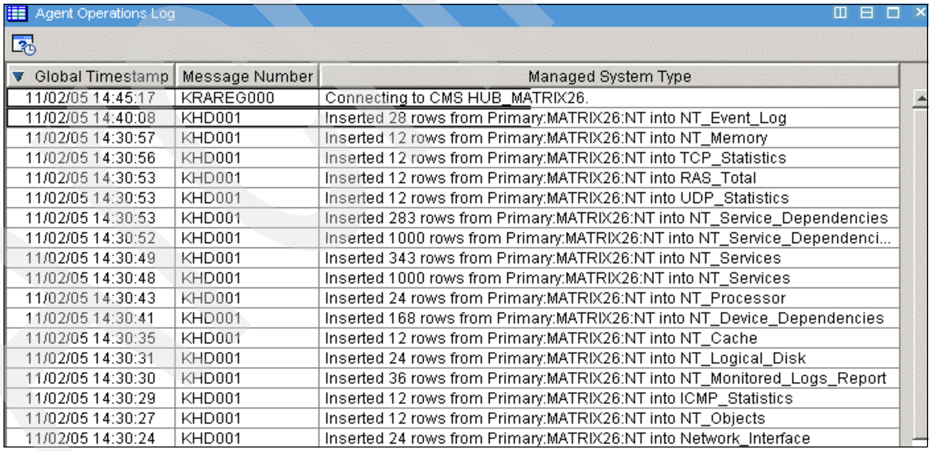


Figure 13-14 Status of the agent

Important: This discussion applies to IBM Tivoli Monitoring code level prior to IBM Tivoli Monitoring Version 6.1.0 Interim Fix 3 (6.1.0.5-TIV-ITM-IF0003).

As of Interim Fix 3, the Warehouse Proxy agent no longer writes a record for each export to the Agent Operations log. The only way to tell if records are being written is to query the WAREHOUSELOG table in the database.

A

Product codes

This appendix provides a comprehensive table of all the IBM Tivoli Monitoring components and their codes. These codes are also called AVA codes. Use these codes when running commands.

IBM Tivoli Monitoring product codes

Table A-1 lists the IBM Tivoli Monitoring product codes. A more recent list can be viewed in the file $INSTALLDIR/registry/proddsc.tbl.

Table A-1 Product codes

Product code	Component
3z	Monitoring Agent for Active Directory
a2	AF/Remote Alert Adapter
a4	Monitoring Agent for i5/OS
ah	System Automation for z/OS
am	OMEGACENTER® Gateway MVS Alert Adapter
au	CA-Unicenter Alert Emitter
ax	IBM Tivoli Monitoring Shared Libraries
bc	ITCAM System Edition for WebSphere DataPower®
bl	CASP Directory Server Monitoring Agent
br	CASP Exchange Connector Monitoring Agent
bs	Basic Services
c3	IBM Tivoli Monitoring for CICS
cf	Monitoring server Configurator
cg	IBM Tivoli Monitoring for Cryptographic Coprocessors
ci	IBM Tivoli Monitoring Product Installer
cj	Tivoli Enterprise Portal Desktop Client
co	Command and Control
cp	OMEGAMON XE for CICS on z/OS
cq	Tivoli Enterprise Portal Server
cw	Tivoli Enterprise Portal Browser Client
cz	IBM Tivoli Monitoring CommandPro
d3	IBM Tivoli Monitoring for DB2
d4	ITCAM for SOA

Product code	Component
dd	Distributed Database common code
dp	OMEGAMON XE for DB2 PE and PM on z/OS
e3	R/3 Clients (for ETEWatch®) Monitoring Agent
ea	Internet Monitoring Agent
el	Lotus Notes® Clients (for ETEWatch) Monitoring Agent
en	SNMP Gateway on Windows NT
er	Management Agent for Tivoli Enterprise Console Gateway
es	EIF to WS-Notification Converter
eu	Custom Clients (for ETEWatch) Monitoring Agent
ew	Web Browsers (for ETEWatch) Monitoring Agent
ex	Monitoring Agent for Microsoft Exchange Server
ez	OMA for eBA® Solutions
fw	Windows NT Tivoli Enterprise Portal
ga	SNMP Gateway on AIX
gb	IBM Tivoli Monitoring for Domino
gr	Graphics and Sound Library for TEP
gw	OMEGAMON XE for CICS TG on z/OS
hc	HMC Alert Adapter
hd	Warehouse Proxy
hl	OMEGAMON z/OS Management Console
ho	HP OpenView NNM Alert Adapter
ht	Monitoring Agent for Web Servers
ic	WebSphere InterChange Server Monitoring Agent
ie	WebSphere InterChange Server Data Source
ih	OpenView ITO Alert Emitter
ip	OMEGAMON XE for IMS on z/OS
it	TEC GUI Integration

Product code	Component
iw	IBM Tivoli Enterprise Portal Server Extensions
jr	Tivoli Enterprise-supplied JRE
ju	Monitoring Agent for JMX JSR-77
ka	Monitoring Agent for Tivoli Enterprise Console
kf	IBM Eclipse Help Server
ln	Lotus Notes Monitoring Agent
lz	Monitoring Agent for Linux OS
m3	IBM Tivoli Monitoring for OS/390®
m5	OMEGAMON XE on z/OS
ma	Remedy ARS Alert Adapter
mc	WebSphere MQ Configuration Agent
md	PQEdit®
mq	WebSphere MQ Monitoring Agent
ms	Tivoli Enterprise Monitoring Server
n3	OMEGAMON XE for Mainframe Networks
na	IBM Tivoli NetView for z/OS Enterprise Management Agent
nd	Monitoring Agent for Tivoli NetView Server
np	Tivoli Monitoring for IBM Tivoli Netcool/Precision
nt	Monitoring Agent for Windows OS
nv	NetView/AIX Alert Adapter
nw	Novell NetWare Monitoring Agent
oe	CCC for OS/390 Unix System Services
oq	Monitoring Agent for Microsoft SQL Server
or	Monitoring Agent for Oracle
os	IBM Tivoli Monitoring for Sysplex
ox	Informix® Monitoring Agent
oy	Monitoring Agent for Sybase Server

Product code	Component
p5	IBM Tivoli Monitoring for AIX Base Agent
pa	Performance Analytics for TEP
pc	DEC Polycenter Alert Adapter
ph	Base Monitoring Agent for HMC
pk	Base Monitoring Agent for CEC
pl	CandleLight Workstation
ps	PeopleSoft® Monitoring Agent
pt	Peregrine ServiceCenter Alert Adapter
pv	IBM Tivoli Monitoring for VIOS Base Agent
px	Premium Monitoring Agent for AIX
q5	Monitoring Agent for Microsoft Cluster Server
qi	WebSphere Message Broker Monitoring Agent
qr	Monitoring Agent for Microsoft Virtual Server
qv	Monitoring Agent for VMware ESX
qx	Monitoring Agent for Citrix Access Suite
r3	Monitoring Agent for Microsoft SQL Server
rc	IBM Tivoli Monitoring for Rational® Applications
s2	OS/2® Monitoring Agent
s3	OMEGAMON XE for Storage on z/OS
sa	Monitoring Agent for mySAP
sd	Status Data Manager
sh	Tivoli Enterprise Monitoring SOAP Server
sj	Best Practices for WebSphere
sp	SNMP Alert Adapter
ss	Windows NT SNA Server Monitoring Agent
sy	Summarization and Pruning Agent
t1	ITCAM File Transfer Enablement

Product code	Component
t2	ITCAM for Response Time Tracking
t3	ITCAM for End User Response Time Dashboard Agent
t4	ITCAM for Client Response Time Agent
t5	ITCAM for Web Response Time Agent
t6	ITCAM for Robotic Response Time Agent
tl	Omegamon XE for Message Transaction Tracker
tm	Monitoring Agent for IBM Tivoli Monitoring 5.x Endpoint
tn	Unicenter TNG Alert Emitter
tr	NetView Alert Emitter
tv	Tivoli Enterprise Console Alert Adapter
tx	Tuxedo Monitoring Agent
ua	CA-Unicenter Alert Adapter
ub	IBM Tivoli Monitoring for Applications: Siebel® Agent
ud	Monitoring Agent for DB2
uf	Universal Agent Framework
ui	Tivoli Enterprise Services User Interface
uj	Unison Maestro™ Alert Adapter
ul	Monitoring Agent for UNIX Logs
um	Universal Agent
ur	Unison RoadRunner Alert Adapter
ut	Unicenter TNG Alert Adapter
ux	Monitoring Agent for UNIX OS
va	Premium Monitoring Agent for VIOS
vi	HP OpenView Alert Emitter
vl	OMEGAMON XE on z/VM® and Linux
vt	Tivoli Enterprise Console Alert Emitter
vw	NetView for z/OS Agent Support

Product code	Component
we	WebSphere Application Server Monitoring Agent
wj	IBM Tivoli Composite Application Manager Common Components
wl	BEA Weblogic Server Monitoring Agent
wo	IBM Tivoli Monitoring for OMEGAVIEW® II® for the Enterprise
ww	WebSphere Application Server on OS/390
yb	IBM Tivoli Information Management for z/OS
yj	Monitoring Agent for J2EE
yn	Monitoring Agent for WebSphere

B

IBM Tivoli Monitoring component flow diagrams

This appendix presents several IBM Tivoli Monitoring component flow diagrams, and explains how different components and processes interact with each other. You may find these component flow diagrams to be useful when troubleshooting IBM Tivoli Monitoring problems or designing your IBM Tivoli Monitoring environment.

The following component flows are explained:

- Situation event processing component flow
- Portal client workspace query component flow
- Short-term history (Historydriver) collection instantiation component flow
- Warehouse Proxy Agent component flow

Figure B-1 on page 480 illustrates the situation event processing component flow.

Situation event processing component flow

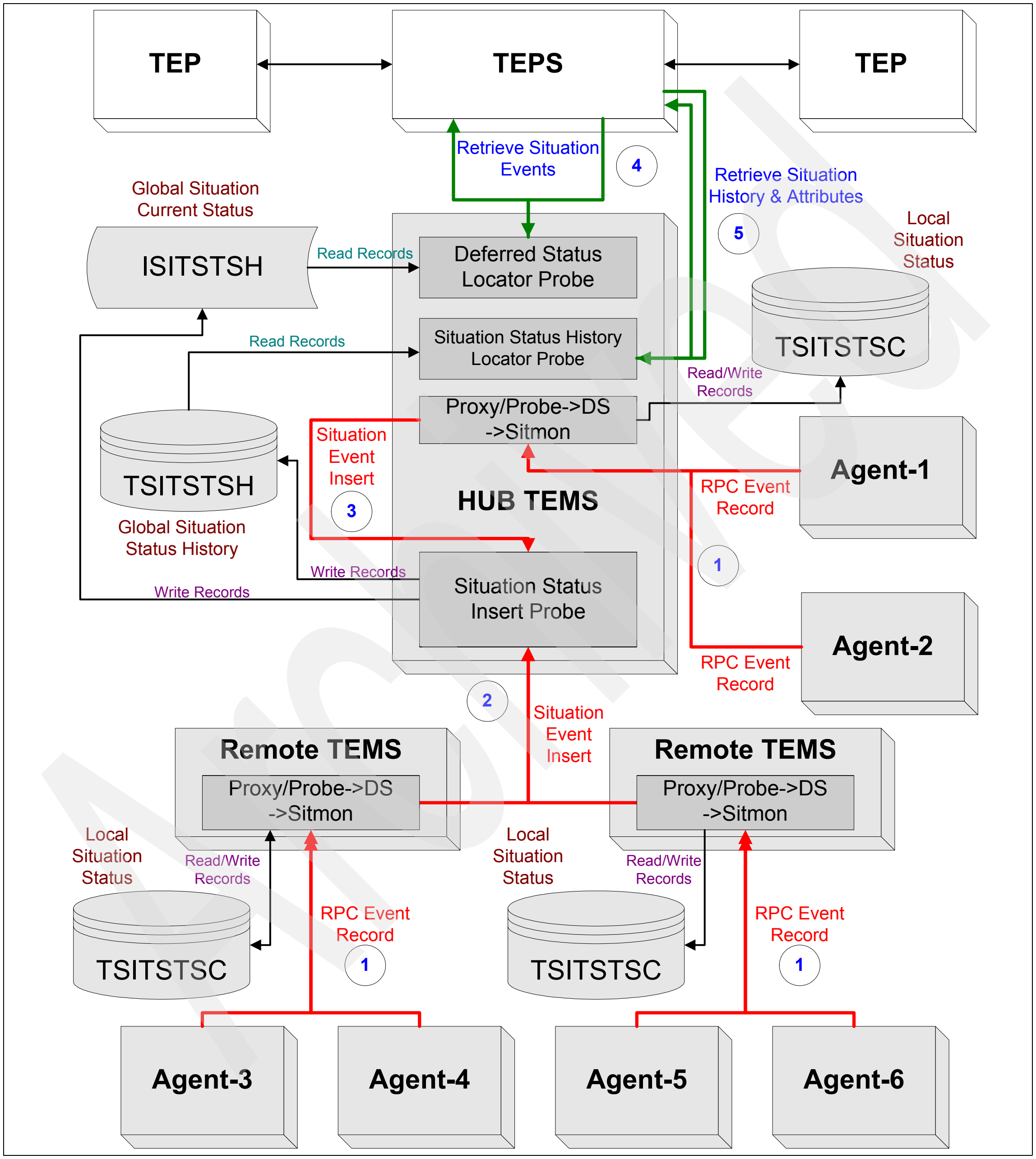


Figure B-1 Situation event processing

Description of the flow - situation event processing

1. The agent collector running in response to a received situation filter object satisfies the specified thresholds of the situation filter object. The collector attributes returned at the time of the threshold exceeded are transmitted from the agent endpoint or probe to the connected to monitoring server via RPC.
2. The received situation RPC results are returned to the monitoring server proxy, and then to the data server late filter processing probe. If all conditions of the thresholds of the situation are true, then an event record is emitted to the TADVISOR framework agent probe.

   The KSMOMS monitoring server subprocess that has been waiting on returned results for the situation next receives the situation record as a TADVISOR event row. In some cases, the KSMOMS further filters or delegates the event information forming one or more local situation status records (TSITSTSC). If the situation is indicated to return events to the hub, then one or more hub event records are also further generated. The hub event records are transmitted to the hub using INSERT SQL of TSITSTSH records, and are then received by the hub situation status inserter probe.
3. The hub situation status inserter probe then creates storage resident situation status records (ISITSTSH), and persists new received TSITSTSH records into a wrapping log file.
4. Clients interested in dynamic situation events use a deferred SELECT request against the ISITSTSH that has situation event records returned as they occur to the hub monitoring server.
5. Clients interested in situation event history or detailed information about the attributes originating from the agent or probe endpoint that produced the situation event use a TSITSTSH log SELECT request to retrieve selected records.

Figure B-2 on page 482 illustrates the portal client workspace query component flow.

Portal client workspace query component flow

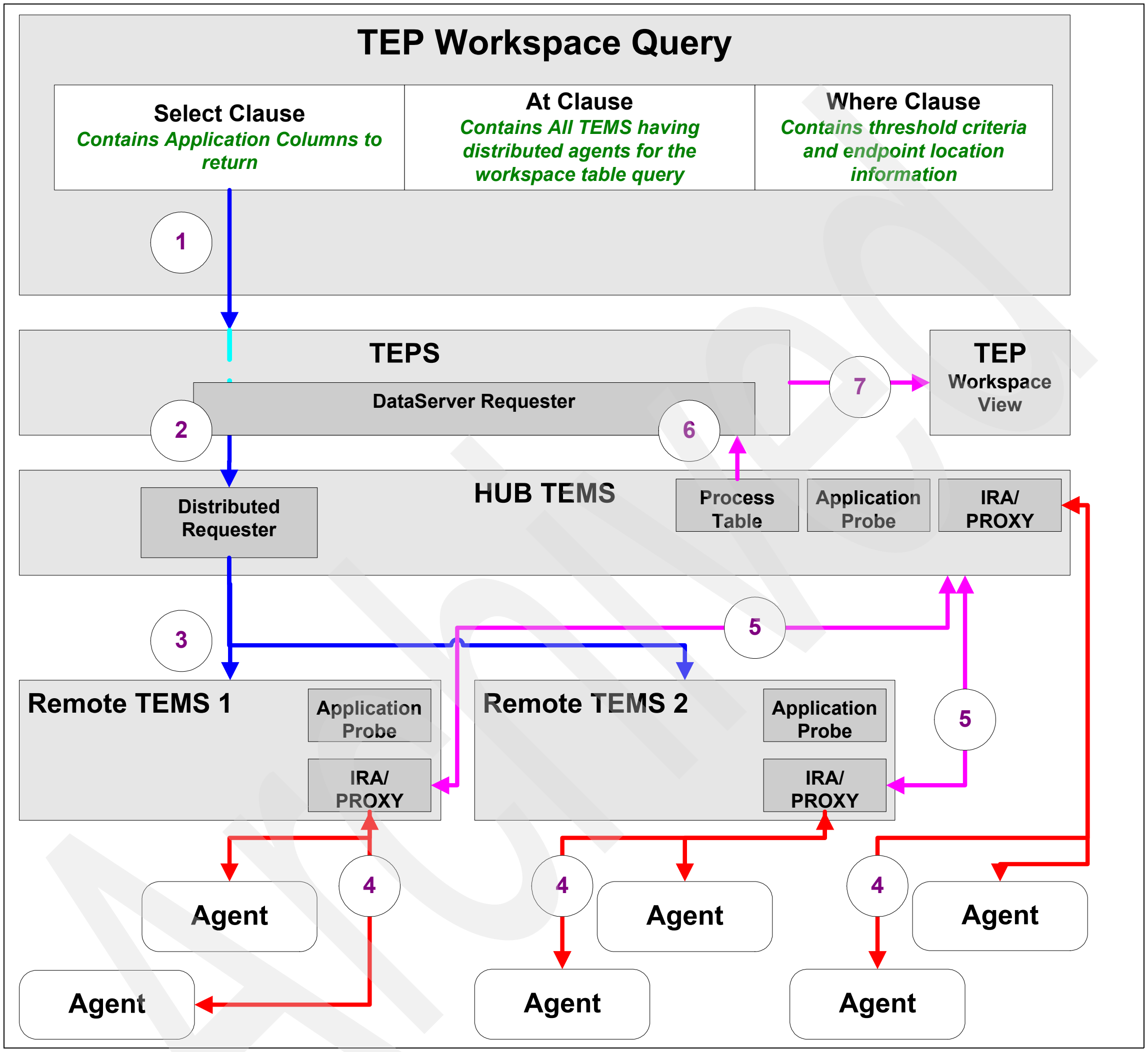


Figure B-2 Portal client workspace query

Description of the portal client workspace query flow

1. The client initiates a SELECT query for a given workspace view. The query will contain three major components, as described here:
   - SELECT clause - a list of column names that each record returned will contain.

– AT clause - a list of monitoring server NODEID locations that have agents or probes attached pertinent to where the query is returning information from.

– WHERE clause - contains threshold filtering criteria based on compare values for selected columns that belong to the application table name used for the query. This section also contains one or more ORIGINNODE filters or a special PARMA NODELIST filter.

  ORIGINNODE filters are used to specify individual endpoints that the query is to be directed.

  The PARMA NODELIST is used to specify a “Managed System List” which itself contains the desired individual endpoints the query is to be directed to.

2. Request formed as described by list item 1. is passed to the portal server and the Data Server Requester which has a connection maintained to the hub monitoring server. The request is transmitted through the SQL1 interface to the hub monitoring server distributed requester.
3. Depending on the AT clause monitoring server NODEID members specified in list item 1., the request is forwarded to each specified monitoring server found in the AT clause.
4. Each monitoring server proxy receives the request and creates a report filter object that is sent to either the specified ORIGINNODE targets or to all targets matching the SYSTEM.PARMA Managed system list specification.
5. Each proxy waits for all records to return from each discovered target before emitting results back to the hub monitoring server process table. Any records not returned within a special TIMEOUT (also given as a special WHERE clause PARMA) will cause the entire request to fail with a timed-out condition.
6. The hub data server process table engine returns the assembled record sets from each delegated AT clause differentiated portion of the query back to the portal server.
7. The query is presented at the portal client workspace view.

Figure B-3 on page 484 illustrates the short-term history (Historydriver) collection instantiation component flow.

Short term history (Historydriver) collection instantiation component flow

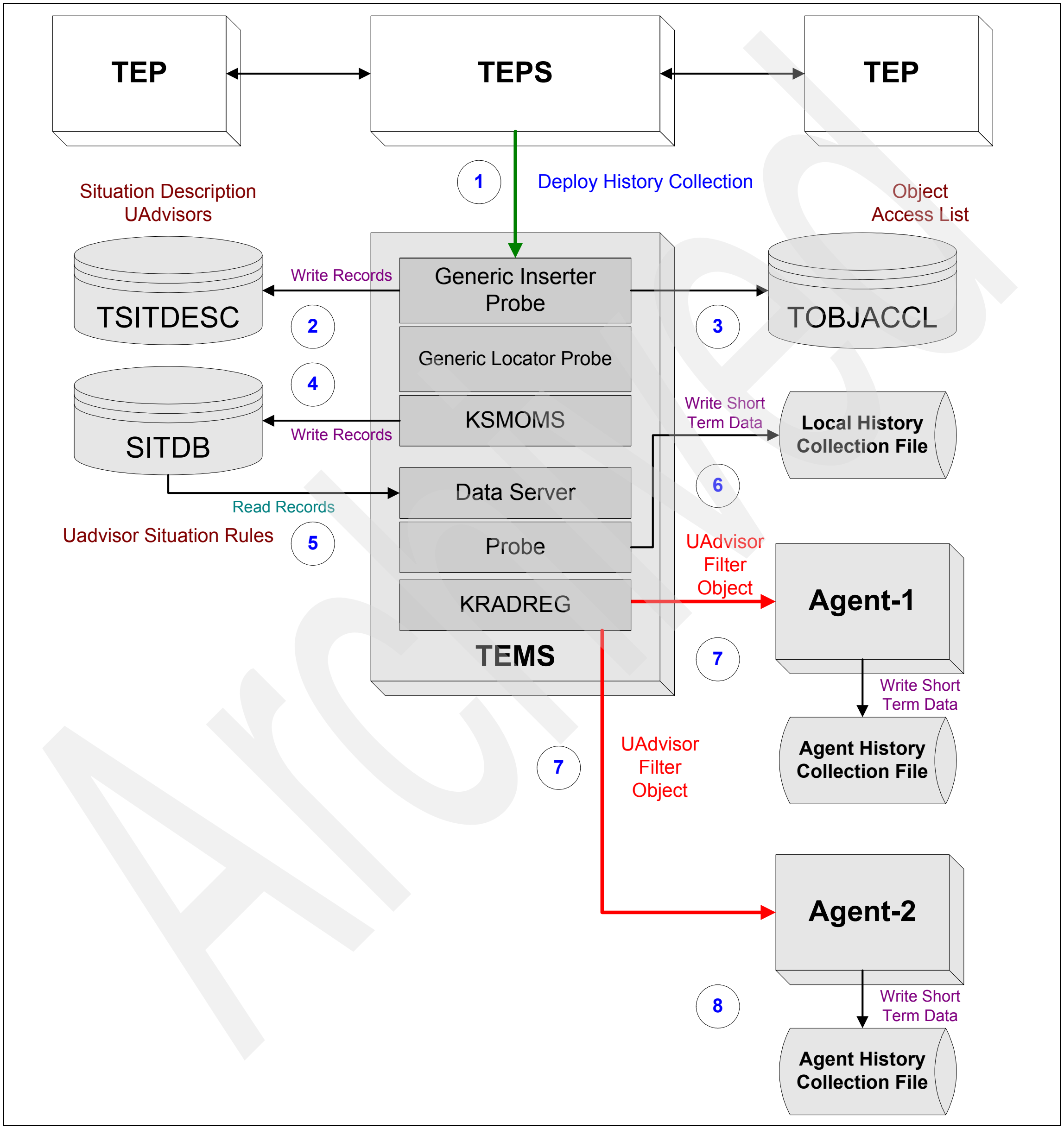


Figure B-3 Short-term history (Historydriver) collection instantiation

Warehouse Proxy Agent component flow

Figure B-4 illustrates the Warehouse Proxy Agent component flow, with explanations included in the figure.

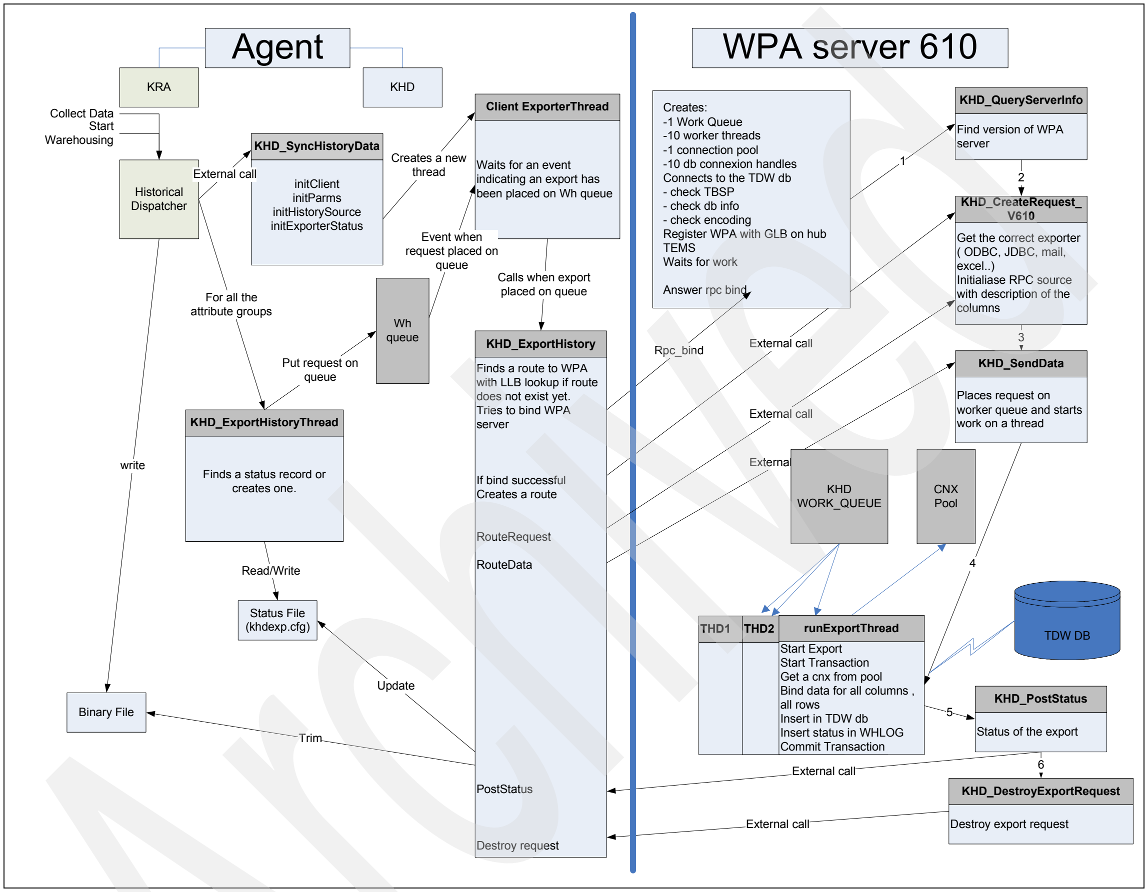


Figure B-4 Warehouse Proxy Agent component flow

C

Additional material

This book refers to additional material that can be downloaded from the Internet as described here.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks publications Web server. Point your Web browser at:

`ftp://www.redbooks.ibm.com/redbooks/SG247443`

Alternatively, you can go to the IBM Redbooks Web site at:

`ibm.com/redbooks`

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247443.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*
SG247443.zip	Zipped SPD files for IBM Tivoli Monitoring V6.1 FP5

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 20 MB minimum
Operating System: Windows/Linux/UNIX

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.

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 publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 490. Note that some of the documents referenced here may be available in softcopy only.

- *Tivoli Management Services Warehouse and Reporting*, SG24-7290
- *Deployment Guide Series: IBM Tivoli Provisioning Manager Version 5.1*, SG24-7261
- *Getting Started with IBM Tivoli Monitoring Version 6.1,* SG24-7143

Other publications

These publications are also relevant as further information sources:

- *IBM Tivoli Monitoring Version 6.1.0. Quick Start Guide*, SC32-1802
- *IBM Tivoli Monitoring Installation and Setup Guide Version 6.1.0*, GC32-9407
- *IBM Tivoli Monitoring Version 6.1.0 Administrator’s Guide*, SC32-9408
- *IBM Tivoli Monitoring Universal Agent User's Guide Version 6.1.0*, SC32-9459
- *IBM Tivoli Monitoring User's Guide Version 6.1.0*, SC32-9409
- *IBM Tivoli Monitoring Command Reference,* Version 6.1.0, SC23-6045
- *Program Directory for IBM Tivoli Monitoring Services,* GI11-4105

Online resources

These Web sites are also relevant as further information sources:

- IBM Tivoli Open Process Automation Library (OPAL) Web site:

http://catalog.lotus.com/wps/portal/topal

- Perl download Web site:

  http://www.cpan.org/modules/index.html

- DB2 JDBC plug-in download site:

  http://www14.software.ibm.com/webapp/download/preconfig.jsp?id=2004-09-20+10%3A09%3A21.003415R&cat=database&fam=&s=c&S_TACT=104CBW71&S_CMP=

- BIRT Web site:

  http://www.eclipse.org/birt/phoenix

- Microsoft Web site for downloads:

  http://www.microsoft.com/downloads

- Oracle Web site:

  http://www.oracle.com

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy IBM Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

X

Z

IBM Tivoli Monitoring: Implementation and Performance Optimization for Large Scale Environments

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM Tivoli Monitoring: Implementation and Performance Optimization for Large Scale Environments

Best practices - large scale deployment of IBM Tivoli Monitoring V6.1 and V6.2

Performance, scalability and high availability considerations

Includes CCMDB integration

The IBM Tivoli Monitoring solution is the next generation of the IBM Tivoli family of products that help monitor and manage critical hardware and software in distributed environments. This IBM Redbooks publication provides a practical guide to implementing, using and optimizing IBM Tivoli Monitoring, including best practices for performance tuning, sizing, high availability, scalability, reporting, IBM Change and Configuration Management Database integration and firewall considerations.

You will find a great deal of information about IBM Tivoli Universal Agents, including versioning, remote deployment and management, and meta servers.

We also delve into details of IBM Tivoli Monitoring of the components, such as how these components interact, what is the underlying technology, details of configuration files, and where to check in case of problems.

This book is a reference for IT professionals who implement and use the IBM Tivoli Monitoring solution in large scale environments.

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

SG24-7443-00 ISBN 0738488593